

Overview of Overlapping Schwarz Methods in PETSc

Pierre Jolivet^[0009-0000-3410-0884],
Jordi Manyer^[0000-0002-0178-3890],
Raphaël Zanella^[0000-0003-3004-9625]

1 Introduction

The efficient numerical solution of large-scale linear systems remains a central challenge in scientific computing, particularly for simulations involving finely resolved partial differential equations (PDEs) on modern parallel architectures. Domain decomposition methods (DDMs) provide an effective strategy to address these challenges by exploiting locality, enabling scalable preconditioning, and facilitating the integration of heterogeneous discretizations and solvers. Among these methods, overlapping Schwarz techniques form a powerful and flexible family of preconditioners that improve convergence by combining local subdomain (and possibly global coarse) corrections with Krylov solvers.

PETSc [5] has deep historical roots in the domain decomposition community. Its early development was strongly influenced by the foundational ideas laid out by Smith, Bjørstad, and Gropp, in a seminal reference that helped shape modern parallel iterative solvers and preconditioner design [21]. Later on, PETSc was used to provide an efficient implementation of balancing domain decomposition by constraints [11] through PCBDDC [25].

This paper presents a comprehensive overview of overlapping Schwarz methods as implemented in PETSc, emphasizing both classical approaches and more recent algorithmic developments. We begin by examining the core preconditioners PCBJACOBI, PCASM, and PCGASM, which provide block Jacobi, additive Schwarz, and generalized additive Schwarz capabilities, respectively. These components constitute the backbone of many other preconditioning strategies in the library. We then discuss PCHPDDM, PETSc interface to the HPDDM library, which provides multilevel overlapping Schwarz preconditioners based on spectral coarse spaces. Finally, we il-

Pierre Jolivet
Sorbonne Université, CNRS, LIP6, France, e-mail: pierre@joliv.et

Jordi Manyer
Monash University, Australia, e-mail: jordi.manyer@monash.edu

Raphaël Zanella
Sorbonne Université, CNRS, LIP6 and Inria Paris, SERENA, France, e-mail:
raphael.zanella@inria.fr

illustrate the use of these techniques in challenging applications. We provide concrete examples in both Python (petsc4py [9]) and Julia (Gridap.jl [4]), demonstrating how high-level languages can drive sophisticated domain decomposition workflows in PETSc.

2 One-level overlapping Schwarz methods

One-level domain decomposition methods constitute the foundation of PETSc overlapping Schwarz capabilities. These methods rely exclusively on local subdomain solves, with no global coarse component, to accelerate convergence of Krylov methods. While their scalability is fundamentally limited by increasing global problem size, one-level methods remain highly effective for moderate-scale simulations, for problems with relatively benign conditioning, or as building blocks within more elaborate multilevel schemes. PETSc provides several one-level overlapping Schwarz preconditioners, with different implementation flexibility.

2.1 Block Jacobi (PCBJACOBI)

PCBJACOBI implements a classical block Jacobi preconditioner in which the global linear system is partitioned into N disjoint blocks, each solved independently. Algebraically speaking, let us introduce a set of N Boolean rectangular matrices $\{R_i\}_{i=1}^N$, such that if a user is solving a linear system $Ax = b \in \mathbb{K}^n$, then each disjoint block may be written as $R_i A R_i^T$, where R_i is of dimension $n_i \times n$. In parallel executions with P processes, the default configuration assigns one block per process, making block Jacobi the simplest form of domain decomposition and a natural baseline preconditioner. However, N may also be set to a value higher or lower than P , for example through the command-line option `-pc_bjacobi_blocks N`. In any case, one has $\sum_{i=1}^N n_i = n$. Each block can be equipped with its own local solver, configured via the subsolver prefix `-sub_`, which allows fine-grained composition of direct solvers, e.g., `-sub_pc_type lu`, incomplete factorizations, or even nested multilevel solvers at the subdomain level.

2.2 (Generalized) additive Schwarz (PCASM and PCGASM)

PCASM extends block Jacobi by introducing overlap between subdomains [24, 8]. The Boolean rectangular matrices $\{R_i\}_{i=1}^N$ are now defined to take into account such overlap, and because of duplicated unknowns, one now has $\sum_{i=1}^N n_i > n$. Let us further introduce another set of Boolean rectangular matrices $\{\tilde{R}_i\}_{i=1}^N$ of the same dimensions as $\{R_i\}_{i=1}^N$ that defines a partition of unity, i.e., $\sum_{i=1}^N \tilde{R}_i^T \tilde{R}_i = I$. Overlapping

regions enable information to propagate more effectively across subdomains, thereby improving convergence of Krylov methods. PCASM supports a variety of ways to define subdomains, including sparsity pattern-based overlap specification, for example through the command-line option `-pc_asm_overlap δ` , and user-provided subdomain layouts, for example through the function `PCASMSetLocalSubdomains()`.

Just as for PCBJACOBI, a central feature of PCASM is the composability of subdomain solvers: each subdomain solver is itself a pair of a KSP and a PC that is fully configurable through the standard options database using the prefix `-sub_`. It also provides several variants of the additive Schwarz method through the command-line option `-pc_asm_type`, offering fine-grained control over how subdomain corrections are combined:

$$\begin{aligned} \text{restrict (default)} \quad M_{\text{restrict}}^{-1} &= \sum_{i=1}^N \tilde{R}_i^T (R_i A R_i^T)^{-1} R_i, \\ \text{basic} \quad M_{\text{basic}}^{-1} &= \sum_{i=1}^N R_i^T (R_i A R_i^T)^{-1} R_i, \\ \text{interpolate} \quad M_{\text{interpolate}}^{-1} &= \sum_{i=1}^N R_i^T (R_i A R_i^T)^{-1} \tilde{R}_i, \\ \text{none} \quad M_{\text{none}}^{-1} &= \sum_{i=1}^N \tilde{R}_i^T (R_i A R_i^T)^{-1} \tilde{R}_i. \end{aligned}$$

These options allow PCASM to reproduce the major one-level overlapping Schwarz variants found in classical domain decomposition references [22, 12]. Unlike PCBJACOBI, subdomains cannot be distributed on more than one MPI process, and the following inequality must hold: $P \leq N$. The value N may be adjusted for example through the command-line option `-pc_asm_blocks N` .

PCGASM generalizes PCASM such that subdomain solvers may be distributed on more than a single MPI process, which lifts the limitation that P should not be greater than N , enabling collective subdomain solves that can exploit parallel factorization. As of PETSc version 3.24, we notice the slight inconsistency in terms of option names, since the value N may now be set through the command-line option `-pc_gasm_total_subdomains N` . However, other important options such as `-pc_gasm_overlap` and `-pc_gasm_type` share the same characteristics as their PCASM counterparts.

3 High-performance unified framework for DDMs

While one-level Schwarz methods provide powerful preconditioners for moderate-sized problems, their convergence degrades as the global problem size increases, due to the lack of a mechanism for propagating global information. Multilevel extensions address this shortcoming by introducing coarse spaces that capture long-range interactions and reduce the effective condition number of the system.

PCHPDDM [18] serves as the primary interface between the high-performance unified framework for DDMs (HPDDM [16]) and PETSc. While HPDDM can in principle be used as a standalone library, most recent algorithmic developments, such as algebraic coarse operator construction or transpose applications, have been

implemented directly within PCHPDDM. As a result, these capabilities are currently accessible primarily to PETSc users, who benefit from HPDDM evolving features through PETSc infrastructure, options database, and language bindings, e.g., Julia, see section 4.3.

3.1 Setup phase

PCHPDDM constructs robust multilevel preconditioners by enriching one-level overlapping Schwarz methods with spectral coarse spaces. The core idea is to solve, concurrently, local sparse generalized eigenvalue problems (or singular value decompositions) on each subdomain, capturing low-energy modes that are poorly reduced by one-level preconditioners. The selected eigenvectors (or singular vectors) are then concatenated column-wise into a global deflation matrix, which is used to assemble a coarse operator through a Galerkin projection. Algebraically speaking, given a set of local matrix pencils $\{(A_i, B_i)\}_{i=1}^N$, each subdomain solves an eigenproblem equivalent to $A_i \Lambda_{i_j} = \lambda_{i_j} B_i \Lambda_{i_j}$, for a selected portion of the spectrum [15] (typically the smallest $\{\lambda_{i_j}\}_{j=1}^{\nu_i}$, where $\{\nu_i\}_{i=1}^N$ are either user-supplied or automatically determined). Then, HPDDM assembles $Z = \left[\tilde{R}_1^T \Lambda_{1_1} \cdots \tilde{R}_1^T \Lambda_{1_{\nu_1}} \cdots \tilde{R}_N^T \Lambda_{N_1} \cdots \tilde{R}_N^T \Lambda_{N_{\nu_N}} \right]$ and computes $A_C = Z^T A Z$, which is of dimension $n_C = \sum_{i=1}^N \nu_i \ll n$. The next section explains how this coarse operator A_C is used during the solution phase. We highlight here that HPDDM supports several definitions of local eigenproblems. In particular, in section 4, we will compare the numerical performance of preconditioners setup with the following eigenproblems:

- GenEO, which is not algebraic and requires an intrusive link with the discretization kernel but for which tight bound on the condition number of the preconditioned operator may be derived,
- `block_splitting` [3] and
- `harmonic_overlap` [2], which are both fully algebraic but which yield slightly looser condition number bounds.

For the sake of conciseness, we do not give the complete definitions of $\{(A_i, B_i)\}_{i=1}^N$ for all three methods, as they are explained in greater details in the corresponding citations.

3.2 Solution phase

Once $Z \in \mathbb{K}^{n \times n_C}$ and $A_C \in \mathbb{K}^{n_C \times n_C}$ have been determined, see previous section, they may be combined with any other preconditioner (typically a one-level overlapping Schwarz method), denoted in this section M^{-1} , which options are prefixed by `-pc_hpddm_levels_1_`. Let us further introduce $Q =$

$Z(ZA_C Z^T)^{-1} Z^T$, for which the action of $(ZA_C Z^T)^{-1}$ may be parametrized with options prefixed by `-pc_hpddm_coarse_`. Then, through the command-line option `-pc_hpddm_coarse_correction`, it is possible to adjust the action of PCHPDDM on any vector:

$$\begin{aligned} \text{deflated (default, nonsymmetric)} \quad & M'_{\text{deflated}}{}^{-1} = Q + M^{-1} (I - AQ), \\ \text{additive} \quad & M'_{\text{additive}}{}^{-1} = Q + M^{-1}, \\ \text{balanced} \quad & M'_{\text{balanced}}{}^{-1} = Q + (I - Q^T A^T) M^{-1} (I - AQ). \end{aligned}$$

It is worth mentioning that $M'_{\text{deflated}}{}^{-1}$ and $M'_{\text{additive}}{}^{-1}$ (resp. $M'_{\text{balanced}}{}^{-1}$) need to apply the coarse operator once (resp. twice) per application.

Solving a transposed system $A^T x = b$ is required by many applications, e.g., for adjoint systems in PDE-constrained optimization, sensitivity analysis, or data assimilation. PETSc provides dedicated high-level interfaces for solving transposed systems, namely `KSPSolveTranspose()` for systems with a single right-hand side and `KSPMatSolveTranspose()` for systems with multiple (dense) right-hand sides [19]. These routines mirror the standard `KSPSolve()` and `KSPMatSolve()` APIs but request the action of A^T (which may be known explicitly or implicitly) and, correspondingly, the transposed preconditioner. Inside Krylov methods, PETSc handles this by invoking `PCApplyTranspose()` or `PCMatApplyTranspose()`. We recently added in version 3.24 support for such operations in PCHPDDM. The coarse correction formulae from the previous paragraph now read:

$$\begin{aligned} M'_{\text{deflated}}{}^{-T} &= Q^T + (I - Q^T A^T) M^{-T}, \\ M'_{\text{additive}}{}^{-T} &= Q^T + M^{-T}, \\ M'_{\text{balanced}}{}^{-T} &= Q^T + (I - Q^T A^T) M^{-T} (I - AQ). \end{aligned}$$

$M'_{\text{deflated}}{}^{-T}$ now needs to apply the coarse operator twice, and $M'_{\text{balanced}}{}^{-T}$ needs to apply the coarse operator thrice when A is nonsymmetric and twice otherwise (since $Q^T = Q$).

4 Use cases and interoperability of PCHPDDM

Building on the multilevel and spectral capabilities described above, PCHPDDM can be deployed across a wide range of challenging applications that demand robustness, scalability, and seamless integration with high-level programming environments. This section highlights three representative use cases that illustrate the flexibility and interoperability of the proposed framework. In the first two subsections, we examine problems arising from discrete fracture networks, where strong coefficient jumps and geometric anisotropies make spectral coarse spaces particularly effective. We then discuss extensions of PCHPDDM to saddle-point and nonsymmetric systems, demonstrating how it is used beyond the classical SPD settings. In the third and final

subsection, we provide Python (petsc4py) and Julia (GridapPETSc.jl) examples that showcase how these methods can be accessed through concise, user-friendly interfaces suitable for rapid prototyping and production workflows alike.

4.1 Discrete fracture networks

Discrete fracture networks (DFNs) arise in subsurface flow, geothermal simulations, and structural mechanics, where thin but highly conductive (or highly stiff) fractures intersect within a lower-conductivity matrix. The resulting PDE systems often involve: strong coefficient jumps across fracture–matrix interfaces, anisotropic meshes with large aspect ratios, multiscale heterogeneities. . . These features lead to ill-conditioned systems (with condition numbers typically greater than 10^{12}) that challenge classical one-level Schwarz methods, as local subdomain problems do not adequately capture the global hydraulic connectivity induced by the fractures, as well as off-the-shelf algebraic multigrid methods such as PCGAMG [1] or BoomerAMG [14], available in PETSc through PCHYPRE. In contrast, spectral coarse spaces and particularly GenEO have proven extremely robust in this setting. We provide in this section a comparison of most of the aforementioned PC for a particular test case from a recent work with the SERENA team at Inria Paris [17], see [L130geo-heter] from table 8 in the previous reference. The problem is discretized on 3,985 subdomains. Since special transfer operators are needed for multigrid preconditioners to handle such a discretization, see, e.g., [10], one can notice that off-the-shelves AMG preconditioners struggle to converge, even after careful tuning of their parameters. Similarly, one-level Schwarz methods cannot handle such large numbers of subdomains and problem size (141M unknowns). In the end, both GenEO and `block_splitting` [3] are able to solve the linear system in a reasonable number of iterations, which allow to obtain a solution in a reduced wall-clock time.

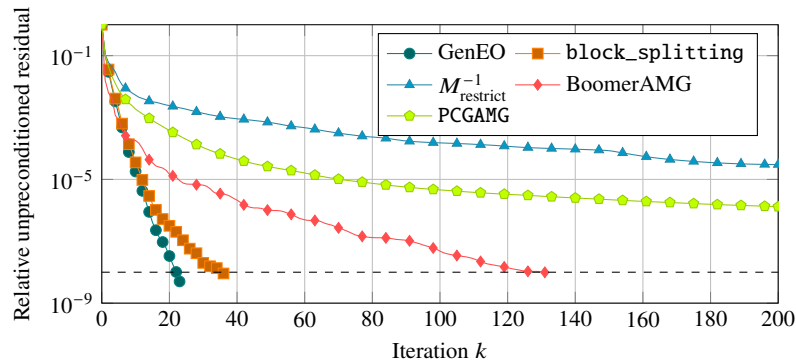


Fig. 1 Convergence histograms for solving the [L130geo-heter] problem from [17].

4.2 Saddle-point and nonsymmetric systems

In the case of saddle-point systems, PCHPDDM builds upon the recent work of Nataf and Tournier [20]. In PETSc, we reformulate their strategy using PCFIELDSPLIT [7], which allows each block of the saddle-point system to be associated with its own solver configuration: $A = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix}$ and $S = A_{11} - A_{10}A_{00}^{-1}A_{01}$. The difficulty then lies on how to either appropriately approximate S or how to solve systems involving S efficiently. Nataf and Tournier proposed to approximate the action of A_{00}^{-1} by an overlapping Schwarz method, in particular $M_{\text{additive}}'^{-1}$ coupled with M_{basic}^{-1} . This yields a first operator \tilde{S} spectrally equivalent to S :

$$\tilde{S} = A_{11} - A_{10}M_{\text{additive}}'^{-1}A_{01} = A_{11} - \sum_{i=1}^N A_{10}R_i^T (R_i A R_i^T)^{-1} R_i A_{01} - A_{10}Q A_{01}.$$

Using a GenEO-like decomposition of A_{11} into local symmetric positive semidefinite matrices, a coarse operator may also be built for the Schur complement block, and by using an inner-outer scheme, they are able to precondition the exact Schur complement S without forming it explicitly. Although the method in [20] was used to solve the system of nearly incompressible linear elasticity and Stokes equations, its implementation there is tightly integrated into FreeFEM [13], and more precisely ffdm [23], in a largely monolithic manner, which makes it challenging to explore alternative formulations or extend the approach to other classes of saddle-point systems. In contrast, using PETSc enables a more modular composition of block structures through PCFIELDSPLIT, allowing us to experiment, for example, with a broader range of coarse spaces for the A_{00} block which may be nonsymmetric and for which GenEO may not be appropriate. In particular, in the following example, we leverage the coarse space introduced by Al Daas et al. [2]. While this extension forfeits the strict theoretical guarantees provided by the analysis from [20], it significantly broadens the practical applicability of their work and demonstrates that robust overlapping Schwarz techniques can be built for nonsymmetric saddle-point systems. Indeed, the Oseen equations are being solved in a 2D geometry Ω inspired by the Ph.D. thesis [6], in which much more in-depth testing is carried out. Given an advection velocity w , FreeFEM is used to discretize with lowest-order Taylor–Hood finite elements and $N = 10$ subdomains the following system of PDEs, for unknown velocities $u = (u_x, u_y)$ and pressure p :

$$\nu \Delta u + w \cdot \nabla u = \nabla p \quad \text{and} \quad \nabla \cdot u = 0 \text{ in } \Omega. \quad (1)$$

A parabolic inlet velocity profile is enforced, alongside a no-slip condition on perforations and free flow at the outlet. As ν tends to 0, the system becomes advection-dominated, as shown by comparing velocity profiles in fig. 2, and the efficiency of GenEO on the A_{00} block worsens, as show in table 1.

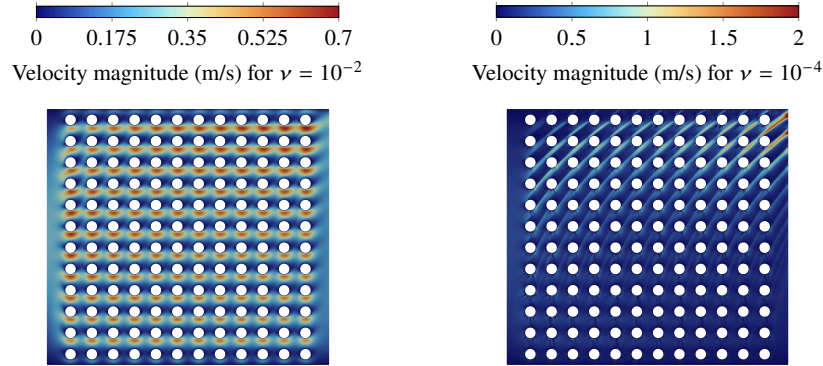


Fig. 2 Velocity profiles for two values of ν from eq. (1) on a perforated 2D domain.

Coarse operator	Outer it.	A_{00}^{-1} it.	S^{-1} it.
GenEO	4	9	4
harmonic_overlap	4	4	4
$\nu = 10^{-2}$			
Coarse operator	Outer it.	A_{00}^{-1} it.	S^{-1} it.
GenEO	5	17	12
harmonic_overlap	5	5	8
$\nu = 10^{-4}$			

Table 1 Comparison of GenEO and harmonic_overlap [2] coarse spaces for different values of ν from eq. (1). Number of iterations for the A_{00}^{-1} and S^{-1} solves are average over all outer iterations.

4.3 Python and Julia snippets

HPDDM is interfaced with PETSc so that, as long as PETSc is configured and compiled with the dedicated option (and the required solver inputs are provided at runtime), HPDDM methods can be used in any C or Fortran code based on PETSc. Examples of such codes are available in the PETSc source tree¹. Recent efforts have been made to facilitate the use of PETSc and HPDDM through higher-level languages: Python and Julia. The following Python example illustrates how to configure and use PCHPDDM through petsc4py on a small-scale but representative heterogeneous linear elasticity problem. The demonstration is intentionally limited to four MPI processes, since all matrices are loaded directly from disk rather than assembled from a discretization kernel. This design choice keeps the script entirely self-contained and reproducible on any machine without requiring a meshing or PDE toolbox. A complete, maintained version of this example is available in the PETSc source tree², alongside the associated data³. These resources provide a convenient and portable environment for experimenting with GenEO coarse spaces and other algebraic preconditioners available in PETSc.

¹ <https://gitlab.com/petsc/petsc/-/blob/main/src/ksp/ksp/tutorials/ex76.c>

² <https://gitlab.com/petsc/petsc/-/blob/main/src/binding/petsc4py/demo/hpddm/hpddm.py>

³ <https://gitlab.com/petsc/datafiles/-/tree/main/matrices/hpddm/GENEO>

```

30 # Create a PETSc matrix object
31 A = PETSc.Mat().create()
32 # Set the local and global sizes of the matrix
33 A.setSizes([[idx[0], idx[2]], [idx[1], idx[3]]])
34 # Configure matrix using runtime options
35 A.setFromOptions()
36 # Load matrix A from binary file
37 A = A.load(PETSc.Viewer().createBinary(f'{load_dir}/A.dat', 'r', comm = PETSc.COMM_WORLD))

38 # Load an index set (IS) from binary file
39 aux_IS = PETSc.IS().load(PETSc.Viewer().createBinary(f'{load_dir}/is_{rank}_4.dat', 'r', comm = PETSc.COMM_SELF))
40 # Set the block size of the index set
41 aux_IS.setBlockSize(A.getBlockSize())
42 # Load the Neumann matrix of the current process
43 aux_Mat = PETSc.Mat().load(PETSc.Viewer().createBinary(f'{load_dir}/Neumann_{rank}_4.dat', 'r', comm = PETSc.COMM_SELF))

44 # Create and configure the linear solver (KSP) and preconditioner (PC)
45 ksp = PETSc.KSP(PETSc.COMM_WORLD).create()
46 pc = ksp.getPC()
47 # Use HPDDM as the preconditioner
48 pc.setType(PETSc.PC.Type.HPDDM)
49 # Set the index set (local-to-global numbering) and auxiliary matrix
50 pc.setHPDDMAuxiliaryMat(aux_IS, aux_Mat)
51 # Inform HPDDM that the auxiliary matrix is the local Neumann matrix
52 pc.setHPDDMHasNeumannMat(True)
53 # Apply any command-line options (which may override options from the source file)
54 ksp.setFromOptions()
55 # Set the system matrix (Amat = Pmat)
56 ksp.setOperators(A)

58 # Create RHS (b) and solution (x) vectors, and solve the system Ax = b
59 b, x = A.createVecs()
60 b.setRandom()
61 ksp.solve(b, x)

```

The Julia ecosystem offers a modern, composable workflow for PDE discretization and scalable linear algebra through Gridap.jl. It provides a high-level finite element framework with expressive syntax for defining weak forms, spaces, and meshes, while GridapPETSc.jl bridges these discretizations with the linear and nonlinear solvers of PETSc, enabling users to seamlessly employ its Krylov methods and preconditioners. Building on this foundation, GridapSolvers.jl extends the toolkit with higher-level solver abstractions and convenient access to advanced preconditioning strategies. Very shortly after the 29th International Conference on Domain Decomposition Methods, GridapSolvers.jl was extended to support PCHPDDM, combining the flexible environment of Gridap.jl with advanced domain decomposition methods, such as GenEO. The following code, which is also maintained in the GridapSolvers.jl source tree⁴, illustrates this integration in practice for solving the 2D Poisson equation, by assembling the associated (distributed/global) stiffness and (concurrent/local) Neumann matrices using concise yet elegant formulations. It may be run on any user-defined number of MPI processes.

⁴ <https://github.com/gridap/GridapSolvers.jl/blob/main/test/ExtLibraries/drivers/HPDDMTTests.jl>

```

16  model = CartesianDiscreteModel(ranks,np,(0,1,0,1),(16,16))
18  order = 1
19  reffe = ReferenceFE(lagrangian,Float64,order)
20  V = FESpace(model,reffe;dirichlet_tags="boundary")
21  U = TrialFESpace(V,u)
22
23  # Global assembled problem
24  qdegree = 2*(order-1)
25  Ω = Triangulation(model)
26  dΩ = Measure(Ω,qdegree)
27
28  f(x) = -Δ(u)(x)
29  a(u,v) = ∫(∇(u)·∇(v))dΩ
30  l(v) = ∫(f·v)dΩ
31
32  assem = SparseMatrixAssembler(
33      SparseMatrixCSR{0,PetscScalar,PetscInt},Vector{PetscScalar},U,V
34  )
35  op = AffineFEOperator(a,l,U,V,assem)
36
37  # Overlapping Neumann problems require ghost cells in the measure
38  Ωg = Triangulation(with_ghost,model)
39  dΩg = Measure(Ωg,qdegree)
40  a_g(u,v) = ∫(∇(u)·∇(v))dΩg
41
42  options = "-ksp_error_if_not_converged true -ksp_converged_reason -ksp_monitor -
43      pc_hpddm_levels_1_eps_nev 10 -pc_hpddm_levels_1_sub_pc_type cholesky -
44      pc_hpddm_define_subdomains"
45  GridapPETS.with(args=split(options)) do
46      solver = HPDDMLinearSolver(V,a_g)
47      uh = solve(solver,op)
48
49      eh = u - uh
50      err_l2 = sqrt(sum(f(eh·eh)dΩ))
51      if i_am_main(ranks)
52          @info "L2 error: $err_l2"
53          @test err_l2 < 1e-6
54      end
55  end

```

5 Conclusion and perspectives

This paper has presented an overview of overlapping Schwarz methods as available in PETSc, emphasizing both their algorithmic foundations and their practical realization within a flexible, composable solver framework. Through a range of examples, from symmetric positive definite problems to nonsymmetric saddle-point systems, we illustrated how PETSc enables robust, scalable solvers while remaining accessible from high-level languages such as Python and Julia.

Several promising directions for future work naturally emerge from this study. First, porting PCHPDDM to GPU architectures is a major opportunity by leveraging PETSc low-level infrastructure. Second, mixed-precision strategies offer significant potential: lower precisions could be employed selectively in subdomain solvers, in the solution of local eigenvalue problems, or in the application and assembly of coarse operators, while preserving global convergence. Third, further advances are

expected from more flexible subdomain-to-process mappings, including support for multiple subdomains per MPI process.

Acknowledgements As part of the “France 2030” initiative, this work has benefited from a national grant managed by the French National Research Agency (Agence Nationale de la Recherche) attributed to the Exa-MA project of the NumPEX PEPR program, under the reference ANR-22-EXNU-0002. The authors would like to thank in no particular order: the SERENA team at Inria Paris, the PETSc development team, and the Gridap.jl development team.

References

1. Adams, M.F., Bayraktar, H.H., Keaveny, T.M., Papadopoulos, P.: Ultrascale implicit finite element analyses in solid mechanics with over half a billion degrees of freedom. In: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, SC04, pp. 34:1–34:15. IEEE Computer Society (2004)
2. Al Daas, H., Jolivet, P., Nataf, F., Tournier, P.H.: A robust two-level Schwarz preconditioner for sparse matrices. *SIAM J. Sci. Comput.* **47**(4), A2378–A2402 (2025)
3. Al Daas, H., Jolivet, P., Rees, T.: Efficient algebraic two-level Schwarz preconditioner for sparse matrices. *SIAM J. Sci. Comput.* **45**(3), A1199–A1213 (2023)
4. Badia, S., Verdugo, F.: Gridap: An extensible finite element toolbox in Julia. *Journal of Open Source Software* **5**(52), 2520 (2020)
5. Balay, S., et al.: PETSc/TAO users manual. Tech. Rep. ANL-21/39 - Revision 3.24, Argonne National Laboratory (2025). DOI 10.2172/2998643
6. Balazi, L.: Multi-scale finite element method for incompressible flows in heterogeneous media: Implementation and convergence analysis. Ph.D. thesis, Institut Polytechnique de Paris (2024)
7. Brown, J., Knepley, M.G., May, D.A., McInnes, L.C., Smith, B.F.: Composable linear solvers for multiphysics. In: Proceedings of the 11th International Symposium on Parallel and Distributed Computing, pp. 55–62 (2012)
8. Cai, X.C., Sarkis, M.: A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM J. Sci. Comput.* **21**(2), 792–797 (1999)
9. Dalcin, L.D., Paz, R.R., Kler, P.A., Cosimo, A.: Parallel distributed computing using Python. *Adv. Water Resour.* **34**(9), 1124–1139 (2011)
10. Di Pietro, D.A., Dong, Z., Kanschat, G., Matalon, P., Rupp, A.: Homogeneous multigrid for hybrid discretizations: Application to HHO methods. *Numer. Methods Partial Differ. Equ.* **41**(5), e70023 (2025)
11. Dohrmann, C.R.: An approximate BDDC preconditioner. *Numer. Linear Algebra Appl.* **14**(2), 149–168 (2007)
12. Dolean, V., Jolivet, P., Nataf, F.: An Introduction to Domain Decomposition Methods. SIAM (2015)
13. Hecht, F.: New development in FreeFem++. *J. Numer. Math.* **20**(3-4), 251–266 (2012)
14. Henson, V.E., Yang, U.M.: BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Appl. Numer. Math.* **41**(1), 155–177 (2002)
15. Hernandez, V., Roman, J.E., Vidal, V.: SLEPC: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Softw.* **31**(3), 351–362 (2005)
16. Jolivet, P., Hecht, F., Nataf, F., Prud’homme, C.: Scalable domain decomposition preconditioners for heterogeneous elliptic problems. In: Proceedings of the 2013 International Conference on High Performance Computing, Networking, Storage and Analysis, SC13, pp. 80:1–80:11. ACM (2013)
17. Jolivet, P., Kern, M., Nataf, F., Pichot, G., Zegarra Vasquez, D.: Domain decomposition preconditioners for efficient parallel simulations of single-phase flow in three-dimensional

- fractured porous media with a very large number of fractures (2025). Working paper available at <https://inria.hal.science/hal-05029676>
18. Jolivet, P., Roman, J.E., Zampini, S.: KSPHPDDM and PCHPDDM: Extending PETSc with advanced Krylov methods and robust multilevel overlapping Schwarz preconditioners. *Comput. Math. Appl.* **84**, 277–295 (2021)
 19. Jolivet, P., Tournier, P.H.: Block iterative methods and recycling for improved scalability of linear solvers. In: *Proceedings of the 2016 International Conference for High Performance Computing, Networking, Storage and Analysis, SC16*. IEEE (2016)
 20. Nataf, F., Tournier, P.H.: Recent advances in domain decomposition methods for large-scale saddle point problems. *Comptes Rendus. Mécanique* **350**(S1), 59–73 (2022)
 21. Smith, B.F., Bjørstad, P., Gropp, W.D.: *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press (2004)
 22. Toselli, A., Widlund, O.B.: *Domain Decomposition Methods - Algorithms and Theory*. Springer Berlin Heidelberg (2004)
 23. Tournier, P.H., Jolivet, P., Nataf, F.: *ffdm: FreeFEM domain decomposition methods*. <https://doc.freefem.org/documentation/ffdm/index.html> (2019)
 24. Widlund, O.B., Dryja, M.: An additive variant of the Schwarz alternating method for the case of many subregions. Tech. Rep. 339, Department of Computer Science, Courant Institute, New York University (1987)
 25. Zampini, S.: PCBDDC: A class of robust dual-primal methods in PETSc. *SIAM J. Sci. Comput.* **38**(5), S282–S306 (2016)