RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

Refactoring HPDDM for GPU-Enabled Domain Decomposition

HPDDM & PETSC 3.25

Erik Fabrizzi, Pierre Jolivet

01.06.2026

What is HPDDM?

- High-performance framework for *domain decomposition methods* [8]
- Provides robust and scalable preconditioners
- Supports advanced iterative solvers
- Integrated in several software packages: PETSc, FreeFEM, MEF++, code_aster, Gridap.jl, GmshDDM . . .
- Available in many more through PETSc (e.g. Feel++, TRUST . . .) [4]

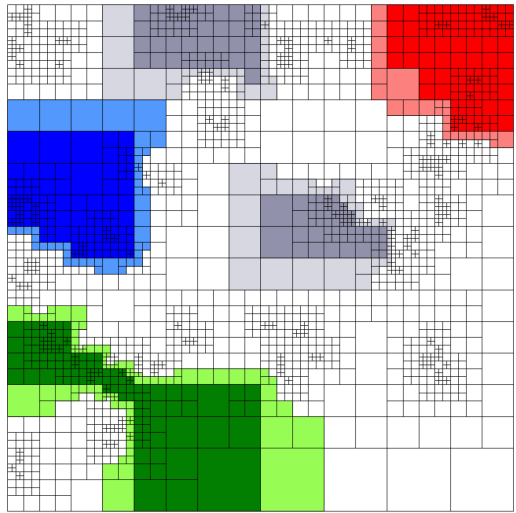
Introduction – Domain Decomposition

What is domain decomposition?

- Subdivision of a (large) global problem into (small) local subproblems [10]
- Global solution is approximated by contributions from local solutions

Why domain decomposition?

- Locality of subproblems allows for highly scalable algorithms [5]
- Yields attractive numerical properties, particularly in two-level setups



Introduction – DD Operators I

Consider a vector with three global degrees of freedom:

$$\mathbf{u} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}.$$

Choose the local restriction operators

$$R_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \qquad \widetilde{R}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix},$$

$$R_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \qquad \widetilde{R}_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Introduction – DD Operators II

The local subvectors extracted on each subdomain are

$$R_1\mathbf{u} = \begin{pmatrix} a \\ b \end{pmatrix}, \qquad R_2\mathbf{u} = \begin{pmatrix} c \\ b \end{pmatrix}.$$

Let D_i denote the diagonal partition-of-unity weights associated with the overlap. Then

$$\mathbf{u} = \sum_i R_i^T D_i R_i \mathbf{u}, \qquad \sum_i R_i^T D_i R_i = I.$$

This identity is the algebraic mechanism that makes it possible to combine local contributions into a globally consistent coarse correction.

GenEO Preconditioning

What is GenEO (Generalized Eigenvalue problem on the Overlap)?

- A spectral coarse space construction used in domain decomposition preconditioners [11]
- Built by solving local generalized eigenvalue problems on overlapping subdomains [7]
- Selects eigenvectors associated with low-energy (slow-to-converge) modes to form an adaptive coarse space that complements standard methods

GenEO – Main Ingredients I

- Overlapping decomposition operators: $\{R_i\}_{i=1}^{N}$
- Global operator: A
- Local assembled operators:

$$A_i = R_i A R_i^T$$

- Local Neumann operators, built without inter-subdomain assembly:

$$\widetilde{A}_i$$

GenEO – Main Ingredients II

Two-level preconditioner

$$M^{-1} = P(P^T A P)^{-1} P^T + \sum_{i=1}^{N} \widetilde{R}_i^T A_i^{-1} R_i.$$

Interpretation

- The second term is an additive Schwarz contribution, naturally connected to `PCASM`.
- The first term is the GenEO coarse correction, handled internally by HPDDM (GEVPs are solved using SLEPc [6]).

GenEO – Deflation Space I

The coarse space is built from local generalized eigenvectors.

Deflation operator

$$P = \begin{bmatrix} R_1^T D_1 Z_1 & \cdots & R_i^T D_i Z_i & \cdots & R_N^T D_N Z_N \end{bmatrix},$$

$$P^T = \begin{bmatrix} Z_1^T D_1 R_1 \\ \vdots \\ Z_N^T D_N R_N \end{bmatrix}.$$

For each subdomain, the local basis Z_i is extracted from the generalized eigenvalue problem

$$\widetilde{A}_i y^{(i)} = \lambda^{(i)} D_i A_i D_i y^{(i)}.$$

GenEO – Deflation Space II

The coarse basis is then obtained by retaining the ν smallest eigenpairs:

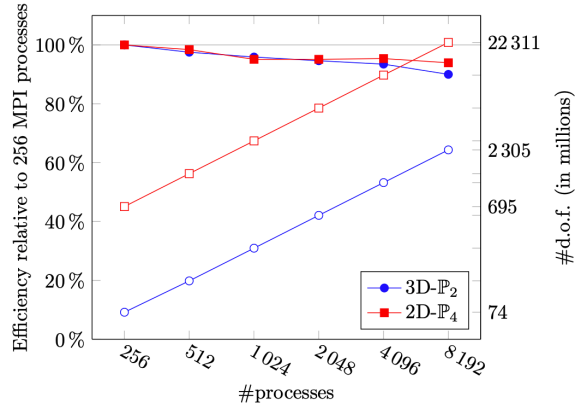
$$Z_i = \begin{bmatrix} y_1^{(i)} & \cdots & y_\nu^{(i)} \end{bmatrix}.$$

These vectors capture the troublesome low-energy components that are poorly reduced by purely local iterations.

GenEO – Motivation

Why use GenEO?

- Robustness: Handles highly complex problems where classical methods struggle
- Scalable convergence: convergence rates vary minimally with respect to the number of subdomains
- Minimal tuning required: Works reliably without hand-crafted coarse spaces or problem-specific adjustments



GenEO – What if I only have a Matrix?

Overview of algebraic alternatives

- **Block Splitting**: Derives an algebraic subdomain partition from the matrix adjacency graph. For HPD problems, efficiently constructs local HPSD splitting matrices used as auxiliary operators. Particularly effective for non-self-adjoint systems [3].
- **Harmonic Overlap**: Builds a fully algebraic spectral coarse space from dominant local harmonic extension modes identified through GEVPs or SVDs. Provides a robust two-level Schwarz preconditioner for both SPD and general sparse matrices [3, 2].
- **Multilevel Approach**: Extends the classical two-level framework when a direct factorization of the coarse operator becomes impractical. Recursively coarsens the coarse space using one of the aforementioned algebraic methods until a sufficiently small coarse problem is obtained [1].

How is it integrated in PETSc?

HPDDM Interfaces

- KSPHPDDM interface: adds advanced iterative solvers
- PCHPDDM interface: adds GenEO-based multilevel preconditioners [8]

PCHPDDM overview:

- `PCHPDDMSetAuxiliaryMat()`: LHS of local GEVP (Neumann matrices)
- `PCHPDDMSetRHSMat()`: RHS of local GEVP (usually computed automatically).
- `PCHPDDMSetDeflationMat()`: supply the local deflation matrices directly.
- `PCHPDDMSetCoarseCorrectionType()`, $Q = P(P^TAP)^{-1}P^T$

$$M^{-1}_{\text{balanced}} = Q + (I - AQ)^T M^{-1}_{\text{RAS}}(I - AQ),$$

$$M^{-1}_{\text{deflated}} = Q + M^{-1}_{\text{RAS}}(I - AQ).$$

Refactoring overview I

What changed?

The HPDDM–PETSc interface has evolved from a loose coupling to a tighter integration centered on PETSc-native data structures and kernels.

Problem statement

- PETSc primarily used for global problem setup, `PCHPDDM` acted as a bridge to HPDDM internals.
- HPDDM internals based on CPU-only kernels, hindering exploitation of accelerators.

Solution

- Core numerical hot spots in HPDDM have been rewritten around PETSc objects and operations.

Refactoring overview II

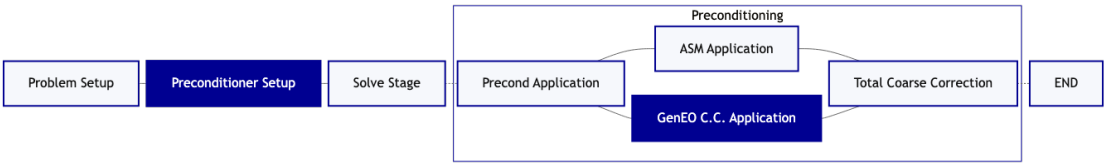
Hybrid computing implication

PETSc already provides mature device support for both sparse and dense linear algebra. Host device interactions are offloaded to the PETSc runtime [4, 9].

Other benefits

1. Avoids custom device kernels, memory management, and glue code: brittle and costly to maintain.
2. PETSc backend: robust, cleaner, and future-proof.

HPDDM I – Workflow Overview I



- **PETSc / PCHPDDM**: provides A_i, $\widetilde{A}_i$, user parameters, PCASM, and high-level control flow.
- **HPDDM internals**: constructs the coarse space and applies the deflation operator.

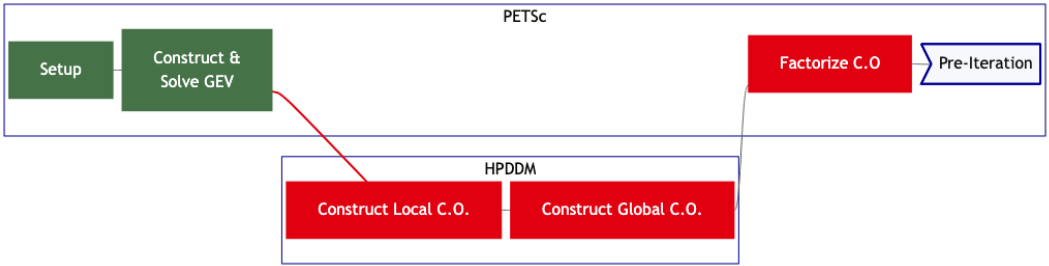
HPDDM II – Refactoring of Data Structures I

The internal types are now matched automatically to the VecType associated with A, A_i, and $\widetilde{A}_i$.

Data structure	Before	After
D_i	VECMPI	VECMPI or VECCUDA
Z_i	raw C++ array	MATSEQDENSE–MATSEQDENSECUDA
$\mathbf{u}_i^{(\cdot)}$	raw C++ array	MATSEQDENSE–MATSEQDENSECUDA

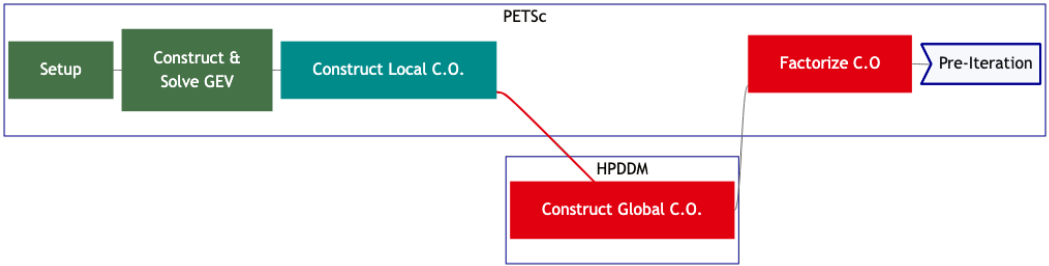
- This significantly improves consistency across CPU and GPU execution paths.
- It also reduces backend-specific code and simplifies future extensions.

HPDDM III – Changes to the Setup Phase I



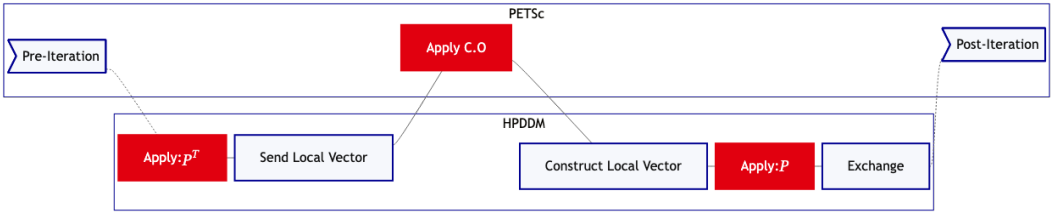
- The setup phase required only limited changes.
- A copy of Z_i back to host memory remains necessary because the construction of the coarse matrix $E = P^T AP$ is not yet fully device-resident and is under investigation.

HPDDM III – Changes to the Setup Phase II



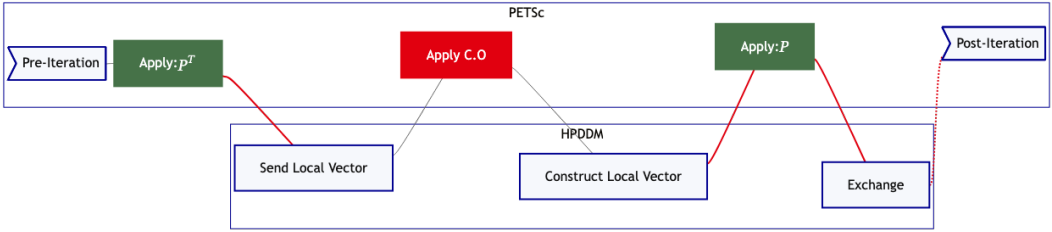
- The setup phase required only limited changes.
- A copy of Z_i back to host memory remains necessary because the construction of the coarse matrix $E = P^T AP$ is not yet fully device-resident and is under investigation.

HPDDM IV – Changes to Deflation I



- This phase required a much deeper refactoring of the main computational kernel.
- The new implementation introduces four small memory transfers per iteration.
- These host-device copies are delegated to PETSc rather than managed manually inside HPDDM.

HPDDM IV – Changes to Deflation II



- This phase required a much deeper refactoring of the main computational kernel.
- The new implementation introduces four small memory transfers per iteration.
- These host-device copies are delegated to PETSc rather than managed manually inside HPDDM.

Related Contributions I

PETSc API improvements

- Implementation of a device-agnostic CUPM version of `MatDiagonalScale`
- New memory-type-aware APIs:
  - `MatCreateDenseWithMemType()`
  - `MatDenseReplaceArrayWithMemType()`

Improvements to KSPHPDDM

- Removal of obsolete code paths
- Compatibility fixes with PCHPDDM when device-enabled objects are passed through the interface

Conclusion

Changes available in PETSc 3.25

- New PETSc backend for `PCHPDDM`.
- Improved Generics and GPU kernel for the PETSc API

Goals achieved:

- Improved GPU usage of `PCHPDDM`
- Reduced number of explicit mentions of device-related keywords in HPDDM
- Increased maintainability and extensibility of `PCHPDDM`

In development

1. Performance study of the new interface both in CPU-only and hybrid scenarios
2. Optimization of memory transactions in hybrid scenarios
3. Multiple processes per subdomain: allow parallel direct solvers per subdomain
4. Multiple subdomains per process: batching of dense operations (particularly on GPU)
5. Porting to PETSc of coarse operator factorization (under scrutiny)
6. Extension to Kokkos

References I

[1] Hussam Al Daas, Laura Grigori, Pierre Jolivet, and Pierre-Henri Tournier.
A multilevel Schwarz preconditioner based on a hierarchy of robust coarse spaces.
SIAM Journal on Scientific Computing, 43(3):A1907–A1928, 2021.

[2] Hussam Al Daas, Pierre Jolivet, Frédéric Nataf, and Pierre-Henri Tournier.
A robust algebraic two-level Schwarz preconditioner for sparse matrices.
SIAM Journal on Scientific Computing, 47(4):A2378–A2402, 2025.

[3] Hussam Al Daas, Pierre Jolivet, and Tamsin Rees.
Efficient algebraic two-level Schwarz preconditioner for sparse matrices.
SIAM Journal on Scientific Computing, 45(3):A1199–A1213, 2023.

References II

[4] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil M. Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, Jacob Faibussowitsch, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Hannah Suh, Stefano Zampini, Hong Zhang, and Junchao Zhang.
PETSc: Portable, extensible toolkit for scientific computation.
https://petsc.org/, 2026.
Version 3.25.

References III

[5] Xiao-Chuan Cai and Marcus Sarkis.
A restricted additive schwarz preconditioner for general sparse linear systems.
SIAM Journal on Scientific Computing, 21(2):792–797, 1999.

[6] Vicente Hernandez, Jose E. Roman, and Vicente Vidal.
SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems.
ACM Transactions on Mathematical Software, 31(3):351–362, 2005.

References IV

[7] Pierre Jolivet, Frédéric Hecht, Frédé ric Nataf, and Christophe Prud'homme.
Scalable domain decomposition preconditioners for heterogeneous elliptic problems.
In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13. ACM, 2013.

[8] Pierre Jolivet, Jose E. Roman, and Stefano Zampini.
KSPHPDDM and PCHPDDM: Extending PETSc with advanced Krylov methods and robust multilevel overlapping Schwarz preconditioners.
Computers & Mathematics with Applications, 84:277–295, 2021.

[9] NVIDIA Corporation.
CUDA C++ Programming Guide.
NVIDIA Corporation, 2026.

References V

[10] Hermann Amandus Schwarz.
Über einen Grenzübergang durch alternierendes Verfahren.
Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, 15:272–286, 1870.

[11] Nicole Spillane, Victorita Dolean, Patrice Hauret, Frédéric Nataf, Clemens Pechstein, and Robert Scheichl.
Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps.
Numerische Mathematik, 126(4):741–770, 2014.

Thank you for your attention

Questions?