

A Novel Domain Splitting Strategy for PDEs

Ken Trotti^[0000-0002-5496-9445]

1 Introduction

Modern supercomputers run at massive concurrency, so PDE solvers must expose ample parallelism. In the study of high-dimensional problems, common approaches include deep learning [6], Monte Carlo [4], sparse grids (SG) [1], and methods exploiting structure, symmetry, or regularity [9]. SG start from a uniform mesh of N^d points and prunes to $O(N \log^{d-1} N)$ points, reducing memory but degrading the L^2 error from $O(N^{-2})$ to $O(N^{-2} \log^{d-1} N)$ under standard regularity assumptions. Motivated by these ideas our aim is to retain the advantages of a reduced mesh without the SG accuracy penalty, while remaining highly parallel and memory-efficient. We propose the *Anisotropic Submeshes Domain Splitting Method* (ASDSM), which partitions a fine uniform mesh into strongly anisotropic submeshes, solves on each, and merges the partial solutions through a skeleton-based construction to approximate the solution. Unlike Schur-complement or Poincaré-Steklov approaches [10], ASDSM performs this reduction at the discretization level, without algebraic elimination or operator compression. In this proceeding, we present a concise, self-contained two-dimensional version of ASDSM. The construction extends naturally to d dimensions by using anisotropic meshes that are coarse along one coordinate direction and their intersections; this increases the number of subproblems but still reduces the effective computational cost by one dimension. An extended preprint with the full d -dimensional formulation and additional analysis is given in [13].

We focus on the following advection-diffusion model:

$$\begin{cases} -\sum_{i=1}^2 \alpha_i \frac{\partial^2 u}{\partial x_i^2}(\mathbf{x}) + \sum_{i=1}^2 \beta_i \frac{\partial u}{\partial x_i}(\mathbf{x}) = s(\mathbf{x}), & \mathbf{x} \in \Omega \subset \mathbb{R}^2, \\ u(\mathbf{x}) = \phi(\mathbf{x}), & \mathbf{x} \in \partial\Omega, \end{cases} \quad (1)$$

Ken Trotti
King Abdullah University Of Science And Technology, Thuwal 23955, Saudi Arabia, e-mail:
ken.trotti@kaust.edu.sa

where s is the source term, ϕ the boundary data, $\alpha_i > 0$ the diffusion coefficients, β_i the velocity field, and Ω the domain.

Section 2 introduces the meshes, projectors, and discretization. Section 3 presents the initial-guess construction. Section 4 reports the results and the conclusion.

2 Introduction to the ASDSM Algorithm

For simplicity, in this section we consider $\Omega = [0, 1]^2$. We first define the meshes and fix notation, then we discretize the PDE and introduce the projection operators used to transfer the discrete unknowns across meshes.

2.1 Meshes

Let $N_c^x, N_f^x, N_c^y, N_f^y \in \mathbb{N}$ denote the number of interior mesh points in the x and y directions for the coarse and fine meshes, subscripts c and f , respectively, s.t.

$$N_f^x + 1 = n_x(N_c^x + 1), \quad N_f^y + 1 = n_y(N_c^y + 1), \quad (2)$$

for some mesh refinement ratios $n_x, n_y \in \mathbb{N}$. Then consider the step lengths

$$h_x = \frac{1}{N_f^x + 1}, \quad h_y = \frac{1}{N_f^y + 1}, \quad H_x = \frac{1}{N_c^x + 1} = n_x h_x, \quad H_y = \frac{1}{N_c^y + 1} = n_y h_y,$$

and the following uniform meshes

$$\begin{aligned} \Omega^{h_x, h_y} &= \left\{ (x_i, y_j) \mid x_i = i h_x, y_j = j h_y, i = 1, \dots, N_f^x, j = 1, \dots, N_f^y \right\} \\ \Omega^{h_x, H_y} &= \left\{ (x_i, y_j) \mid x_i = i h_x, y_j = j H_y, i = 1, \dots, N_f^x, j = 1, \dots, N_c^y \right\} \\ \Omega^{H_x, h_y} &= \left\{ (x_i, y_j) \mid x_i = i H_x, y_j = j h_y, i = 1, \dots, N_c^x, j = 1, \dots, N_f^y \right\} \\ \Omega^{H_x, H_y} &= \left\{ (x_i, y_j) \mid x_i = i H_x, y_j = j H_y, i = 1, \dots, N_c^x, j = 1, \dots, N_c^y \right\} \\ \Omega_{i,j}^{h_x, h_y} &= \left\{ (x_k, y_l) \mid x_k = i H_x + k h_x, k = 1, \dots, n_x - 1 \right. \\ &\quad \left. y_l = j H_y + l h_y, l = 1, \dots, n_y - 1 \right\}. \end{aligned} \quad (3)$$

We refer to the meshes in (3), shown in Figure 1, respectively as the fine mesh, the anisotropic meshes dense in the x - and y -directions, the coarse mesh, and the ‘‘holes’’. The union $\Omega^{h_x, H_y} \cup \Omega^{H_x, h_y}$ is called the *skeleton*, and we denote by $\Omega_{\text{holes}}^{h_x, h_y} := \bigcup_{i=0}^{N_c^x} \bigcup_{j=0}^{N_c^y} \Omega_{i,j}^{h_x, h_y}$ the union of all holes.

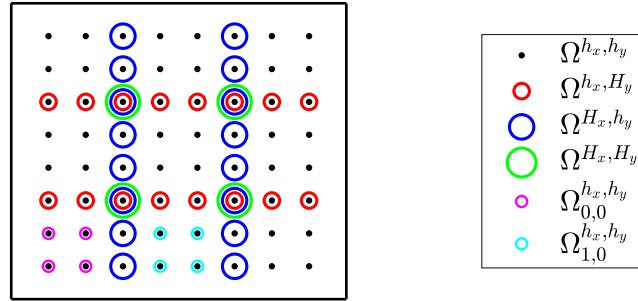


Fig. 1 Meshes in equation (3) with $N_c^x = N_c^y = 2$, $N_f^x = N_f^y = 8$, $n_x = n_y = 3$.

2.2 Projectors

The ASDSM algorithm computes the fine-grid solution using solves on coarser meshes. To transfer discrete functions between meshes we introduce projection operators. Consider the fine mesh Ω^{h_x, h_y} , in equation (3), and let $V^{h_x, h_y} = \mathbb{R}^{N_f^x \times N_f^y}$ with lexicographic order

$$v = [v_{1,1}, v_{2,1}, \dots, v_{N_f^x,1}, v_{1,2}, \dots, v_{N_f^x,2}, \dots, v_{N_f^x, N_f^y}] \in V^{h_x, h_y},$$

where $v_{i,j}$ is the value of the discrete function v at the grid point $(x_i, y_j) \in \Omega^{h_x, h_y}$. We also define the sets $V^{h_x, H_y}, V^{H_x, h_y}, V^{H_x, H_y}, V_{i,j}^{h_x, h_y}$ similarly.

For any target grid $g \in \{fc, cf, cc, holes\}$ we define the restriction operators

$$P_{ff}^g : V^{h_x, h_y} \rightarrow V^\star,$$

with \star being the mesh width linked to g . Each operator removes the components whose nodes do not lie in Ω^\star . Its canonical adjoint $P_g^{ff} = (P_{ff}^g)^T : V^\star \rightarrow V^{h_x, h_y}$ embeds a vector back to the fine grid by placing entries in V^\star and zeros elsewhere. By construction,

$$P_{ff}^g P_g^{ff} = I \quad \text{on } V^\star, \quad \|P_{ff}^g\|_2 = \|P_g^{ff}\|_2 = 1.$$

2.3 Discretization

Using second-order centered FD on the meshes in (3) yields the linear systems

$$\begin{aligned}
A_{ff}u_{ff} &= b_{ff}, \quad u_{ff} \in \mathbb{R}^{N_f^x \times N_f^y} & A_{fc}u_{fc} &= b_{fc}, \quad u_{fc} \in \mathbb{R}^{N_f^x \times N_c^y} \\
A_{cf}u_{cf} &= b_{cf}, \quad u_{cf} \in \mathbb{R}^{N_c^x \times N_f^y} & A_{cc}u_{cc} &= b_{cc}, \quad u_{cc} \in \mathbb{R}^{N_c^x \times N_c^y} \\
A_{(ij)}u_{(ij)} &= b_{(ij)}, \quad u_{(ij)} \in \mathbb{R}^{(n_x-1) \times (n_y-1)}, \quad i = 1, \dots, n_x - 1, \quad j = 1, \dots, n_y - 1.
\end{aligned} \tag{4}$$

Here A_{ff} corresponds to the standard 5-point (2D) advection-diffusion stencil on the fine grid, and A_{fc}, A_{cf}, A_{cc} are the analogous operators on the anisotropic and coarse grids. The right-hand side terms are the usual load vectors induced by s and boundary data on the respective meshes. We employ centered differences for both diffusion and advection terms, which is appropriate for the diffusion-dominated test cases considered in Section 4. For advection-dominated problems, upwind or stabilized schemes could be incorporated within the same ASDSM framework.

We note that, since $\Omega_{i,j}^{h_x, h_y} \subset \Omega^{h_x, h_y}$ and the same discretization method is used, each $A_{(ij)}$ is a main-diagonal sub-block of A_{ff} .

3 The 2D ASDSM Algorithm

In this section we illustrate ASDSM step by step with figures, using a simple 2D Poisson test ($\alpha = 1, \beta = 0$) with a manufactured solution, discretized on a small grid. The MATLAB code is available on GitHub [12].

The ASDSM algorithm, outlined in Algorithm 1, broadly consists in computing or approximating the solutions u_{fc}, u_{cf}, u_{cc} of the anisotropic and coarse linear systems in equation (4), merging them together, thus forming the approximated *skeleton* of the fine solution u_{ff} , and finally filling in the empty regions (holes).

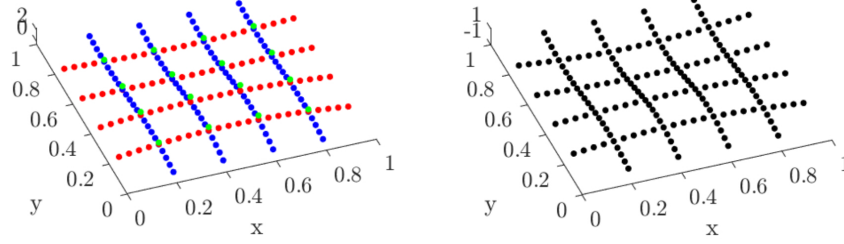
Algorithm 1 ASDSM Algorithm

```

function ASDSM( $b_{ff}, b_{fc}, b_{cf}, b_{cc}$ )
  1.1)  $u_{fc} = (A_{fc})^{-1}b_{fc}$                                 ▶ Solve the equation on  $\Omega^{h_x, H_y}$ 
  1.2)  $u_{cf} = (A_{cf})^{-1}b_{cf}$                                 ▶ Solve the equation on  $\Omega^{H_x, h_y}$ 
  1.3)  $u_{cc} = (A_{cc})^{-1}b_{cc}$                                 ▶ Solve the equation on  $\Omega^{H_x, H_y}$ 
  2)  $\tilde{u}_{ff} = \text{Skeleton}(u_{fc}, u_{cf}, u_{cc})$                 ▶ Builds the skeleton of the fine solution
  3)  $\tilde{u}_{ff} = \text{Filler}(\tilde{u}_{ff}, A_{ff}, b_{ff})$                 ▶ Fills the holes  $\Omega_{i,j}^{h_x, h_y}$ 
  return  $\tilde{u}_{ff}$ 
end function

```

In detail, step 1) computes or approximates the solutions u_{fc}, u_{cf}, u_{cc} through ad-hoc solvers. Since the three systems in steps 1.1)–1.3) are independent, they can be solved in parallel. For example, circulant preconditioners, incomplete LU preconditioners, and multigrid methods have already been shown to be valid preconditioners/solvers for structured anisotropic [8, 14, 3] and isotropic [7, 2, 5] linear systems. After computing u_{fc}, u_{cf}, u_{cc} , respectively shown in red, blue, and green in Figure 2 (left), step 2) merges them into a single unknown \tilde{u}_{ff} via the Skeleton Builder Algorithm 2.



(a) u_{fc}, u_{cf}, u_{cc} in red, blue and green, respectively, over the meshes in Figure 1. (b) Skeleton of the surface over the mesh $\Omega^{h_x, H_y} \cup \Omega^{H_x, h_y}$ after the correction.

Fig. 2 Skeleton before (a) and after (b) the merging process.

Algorithm 2 Skeleton Builder

```

function SKELETON( $u_{fc}, u_{cf}, u_{cc}$ )
  1.1)  $\tilde{u}_{cc}^{(1)} = P_{fc}^{cc} u_{fc}$ 
  1.2)  $\tilde{u}_{cc}^{(2)} = P_{cf}^{cc} u_{cf}$  ▷ Project the anisotropic solutions over the coarse mesh
  2)  $\hat{u}_{cc} = \text{RE}(\tilde{u}_{cc}^{(1)}, \tilde{u}_{cc}^{(2)}, u_{cc})$  ▷ Exploit Richardson Extrapolation (RE)
  4.1)  $e_{cc}^{(1)} = \hat{u}_{cc} - \tilde{u}_{cc}^{(1)}$ 
  4.2)  $e_{cc}^{(2)} = \hat{u}_{cc} - \tilde{u}_{cc}^{(2)}$  ▷ Compute the error with respect to  $\hat{u}_{cc}$ 
  5.1)  $\tilde{u}_{fc} = u_{fc} + \text{Spline}(e_{cc}^{(1)}, \Omega^{h_x, H_y})$ 
  5.2)  $\tilde{u}_{cf} = u_{cf} + \text{Spline}(e_{cc}^{(2)}, \Omega^{H_x, h_y})$  ▷ Correct the two solutions
  6)  $\tilde{u}_{ff} = P_{fc}^{ff} \tilde{u}_{fc} + P_{cf}^{ff} \tilde{u}_{cf} - P_{cc}^{ff} \hat{u}_{cc}$  ▷ nodal exactness ensures  $P_{fc}^{cc} \tilde{u}_{fc} = P_{cf}^{cc} \tilde{u}_{cf} = \hat{u}_{cc}$ 
  return  $\tilde{u}_{ff}$ 
end function

```

In Figure 2 (left) we observe that, due to the different mesh geometries, the discretization introduces mesh-dependent truncation errors, causing the three solutions to exhibit discrepancies at cross-points. For instance, at coordinates $(0.4, 0.4)$, the coarse mesh solution (green) lies visibly above the anisotropic solution u_{fc} (red), whereas at $(0.2, 0.8)$ the discrepancy is less pronounced. Algorithm 2 addresses these inconsistencies by projecting u_{fc}, u_{cf} onto Ω^{H_x, H_y} , computing improved cross-point values via Richardson extrapolation [11]. Here u_{cc} is essential because it provides an additional approximation at the same cross-points, allowing the extrapolation to cancel the leading coarse-mesh truncation error contribution. The algorithm then corrects u_{fc}, u_{cf} through interpolation of their coarse-grid discrepancies, and merges them to the fine grid while avoiding double-counting at shared nodes. The resulting skeleton \tilde{u}_{ff} is zero in the holes and exhibits smoothness on $\Omega^{h_x, H_y} \cup \Omega^{H_x, h_y}$, as shown in Figure 2 (right), thereby enabling the efficient filling step given by Algorithm 3.

In Algorithm 3 we use the skeleton \tilde{u}_{ff} as boundary conditions to solve on the holes domain and update the fine solution. As noted in Section 2.3, the matrix $P_{ff}^{holes} A_{ff} P_{holes}^{ff}$ in step 2) is block-diagonal, with each block corresponding to a

Algorithm 3 Filler

```

function FILLER( $\tilde{u}_{fff}, A_{fff}, b_{fff}$ )
  1)  $c = -P_{fff}^{holes} A_{fff} \tilde{u}_{fff}$  ▷ Use  $\tilde{u}_{fff}$  as boundary conditions
  2)  $v = (P_{fff}^{holes} A_{fff} P_{holes}^{fff})^{-1} (P_{fff}^{holes} b_{fff} + c)$ 
  3)  $\tilde{u}_{fff} = \tilde{u}_{fff} + P_{holes}^{fff} v$ 
  return  $\tilde{u}_{fff}$ 
end function

```

disjoint hole $\Omega_{i,j}^{h_x, h_y}$. Therefore, the solve in step 2) decomposes into independent subproblems that can be executed in parallel. Finally, the update in step 3) yields the accurate solution in Figure 3 (left). We note that by construction, the residual is localized on the skeleton. Consequently, any extension of ASDSM to an iterative framework would require smoothing the error along the skeleton.

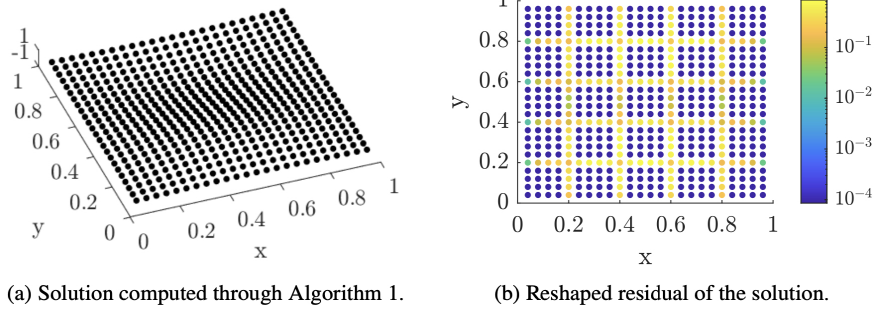


Fig. 3 Solution (a) and respective residual (b).

4 Numerical results

In this section, we test ASDSM [12] by varying the mesh for two problem settings. The implementation is in Matlab 2020b and the tests are run on a machine with an AMD Ryzen 5950X processor and 64GB of RAM.

In the following examples, we set $N_c^x = N_c^y = N_c$ and $N_f^x = N_f^y = N_f$, and denote $N_{\min} = N_c + 1$ and $N_{\max} = N_f + 1$ so that the refinement factors n_x, n_y in (2) are explicit. We consider two scenarios:

- 1) $\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = 1$ with exact solution $u(x, y) = \sin(\pi x + \pi y)$;
- 2) $\alpha_1 = 1 + x^2, \alpha_2 = 2 + xy, \beta_1 = 2 - x, \beta_2 = 1 + y$ with exact solution $u(x, y) = \sin(4\pi x + 4\pi y)$.

In both cases, the source term s and boundary data ϕ are obtained by inserting the exact solution into equation (1). For each scenario we compute the initial guess \tilde{u}_{fff} using Algorithm 1 and Figure 4 (left) shows the absolute residual $r_0 = |b_{fff} -$

$A_{fff} \tilde{u}_{fff}$ reshaped on the fine grid for scenario 1. The residual vanishes on the holes because the filler solves exactly there, but only relative to the (approximate) skeleton boundary values. The residual is concentrated on the skeleton $\Omega^{h_x, H_y} \cup \Omega^{H_x, h_y}$, with oscillations at the coarse cross-points. In Figure 4 (right), we note that by increasing the fine mesh points count, the residual slightly decreases while the error is constant. This is due to the highly condensed residual along the skeleton. Moreover, more oscillatory settings generally lead to higher residuals due to the increased complexity of the problem being solved.

In both scenarios, both the residual and the error decrease when increasing N_{min} . This reflects that the skeleton cross-points accuracy is determined by the coarse grid, so increasing N_c directly improves cross-point consistency, whereas increasing N_f mainly refines within coarse cells yielding essentially no improvement in the error. In most cases, the error is sufficiently small to be considered acceptable as a standalone approximation for exploratory analyses, and it provides a high-quality starting point for any subsequent solver if higher accuracy is desired.

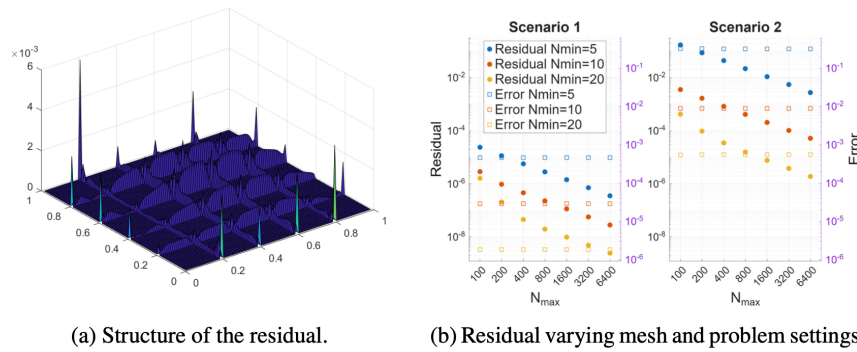


Fig. 4 Reshaped absolute residual of the solution and residual varying the mesh and problem settings.

In conclusion, we have shown that by decomposing the problem into disjoint, smaller sub-problems and merging their solutions, we can obtain a high-quality initial approximation to the fine-grid solution at a reduced computational cost, effectively lowering the dimensionality by one. The resulting solution can be used as an initial guess for other methods, or it may serve as a fast approximation for shooting methods, which do not always require high accuracy. The future introduction of a smoother may further improve the quality of the initial guess.

Acknowledgements This project has received funding from the Federal Ministry of Education and Research and the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701, Time-X. The JU receives support from the European Union’s Horizon 2020 research and innovation programme and Belgium, France, Germany, Switzerland.

Competing Interests The author has no conflicts of interest to declare that are relevant to the content of this chapter.

References

1. Bungartz, H., Griebel, M.: Sparse grids. *Acta Numer.* **13**, 147–269 (2004)
2. Doi, S.: On parallelism and convergence of incomplete LU factorizations. *Appl. Numer. Math.* **7**(5), 417–436 (1991)
3. Donatelli, M., Krause, R., Mazza, M., Trotti, K.: Multigrid preconditioners for anisotropic space-fractional diffusion equations. *Adv. Comput. Math.* **46**, 49 (2020)
4. E, W., Han, J., Jentzen, A.: Algorithms for solving high dimensional PDEs: from nonlinear Monte Carlo to machine learning. *Nonlinearity* **35**(1), 278–309 (2021)
5. Hackbusch, W.: *Multi-grid Methods and Applications*. Springer, Berlin (2013)
6. Huré, C., Pham, H., Warin, X.: Deep backward schemes for high-dimensional nonlinear PDEs. *Math. Comp.* **89**(324), 1547–1579 (2020)
7. Lirkov, I.D., Margenov, S.D., Vassilevski, P.S.: Circulant block-factorization preconditioners for elliptic problems. *Computing* **53**, 59–74 (1994)
8. Lirkov, I.D., Margenov, S.D., Zikatanov, L.T.: Circulant block-factorization preconditioning of anisotropic elliptic problems. *Computing* **58**, 245–258 (1997)
9. Lisle, I.G., Huang, S.L.T., Reid, G.J.: Structure of symmetry of PDE: exploiting partially integrated systems. In: *Proc. 2014 Symposium on Symbolic-Numeric Computation* (2014)
10. Quarteroni, A., Valli, A.: *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press (1999)
11. Richards, S.A.: Completed richardson extrapolation in space and time. *Commun. Numer. Methods Eng.* **13**(7), 573–582 (1997)
12. Trotti, K.: ASDSM algorithm. GitHub repository (2023). <https://github.com/ken13889/ASDSM>
13. Trotti, K.: A domain splitting strategy for solving PDEs (2023). ArXiv preprint arXiv:2303.01163 [math.NA]
14. Vinogradova, S.A., Krukier, L.A.: The use of incomplete LU decomposition in modeling convection–diffusion processes in an anisotropic medium. *Math. Models Comput. Simul.* **5**(2), 190–197 (2013)