A scalable geometric multigrid solver for nonsymmetric elliptic systems with application to variable-density flows

M. Esmaily[a,b], L. Jofre[b], A. Mani[b], G. Iaccarino[b]

[a] *Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14850, USA*
[b] *Center for Turbulence Research, Stanford University, Stanford, CA 94305, USA*

Abstract

A geometric multigrid algorithm is introduced for solving nonsymmetric linear systems resulting from the discretization of the variable density Navier-Stokes equations on nonuniform structured rectilinear grids and high-Reynolds number flows. The restriction operation is defined such that the resulting system on the coarser grids is symmetric, thereby allowing for the use of efficient smoother algorithms. To achieve an optimal rate of convergence, the sequence of interpolation and restriction operations are determined through a dynamic procedure. A parallel partitioning strategy is introduced to minimize communication while maintaining the load balance between all processors. To test the proposed algorithm, we consider two cases: 1) homogeneous isotropic turbulence discretized on uniform grids and 2) turbulent duct flow discretized on stretched grids. Testing the algorithm on systems with up to a billion unknowns shows that the cost varies linearly with the number of unknowns. This $O(N)$ behavior confirms the robustness of the proposed multigrid method regarding ill-conditioning of large systems characteristic of multiscale high-Reynolds number turbulent flows. The robustness of our method to density variations is established by considering cases where density varies sharply in space by a factor of up to 10^4, showing its applicability to two-phase flow problems. Strong and weak scalability studies are carried out, employing up to 30,000 processors, to examine the parallel performance of our implementation. Excellent scalability of our solver is shown for a granularity as low as 10^4 to 10^5 unknowns per processor. At its tested peak throughput, it solves approximately 4 billion unknowns per second employing over 16,000 processors with a parallel efficiency higher than 50%.

1. Introduction

Elliptic partial differential equations (PDE) appear in the modeling and analysis of problems ranging from heat transfer to low-Mach number flows, where material properties and density typically vary in space and time. Solution to these PDE systems is generally obtained by reducing them into a system of algebraic linear equations through spatial and temporal discretization. Solving the resulting linear system, which varies in size and may contain thousands to billions of unknowns, accounts for a significant portion of the computational cost in an elliptic PDE solver.

Email addresses: me399@cornell.edu (M. Esmaily), jofre@stanford.edu (L. Jofre), alimani@stanford.edu (A. Mani), jops@stanford.edu (G. Iaccarino)

Therefore, efficient solution of such linear systems is of high importance given the continuously growing size of computational problems with scientific and engineering applications.

The cost of direct methods for solving linear systems, such as variants of LU factorization, is of $O(N^p)$, with N being the number of unknowns and $p > 1$. Thus, they are an unattractive option for large systems. They might be an attractive choice in case of systems with the constant coefficient matrix that needs to be solved successively at each time step, where the coefficient matrix is factorized and reused at each time step to reduce the overall cost. Recycling the factorized matrix is not, however, an option in the application targeted in this study, where the density variation changes the coefficient matrix in time.

Iterative methods are another option to offset the high cost associated with direct methods [1]. Despite their superior scaling properties, iterative methods suffer from the ill-conditioning of linear systems that deteriorates their convergence rate [2, 3, 4]. The ill-conditioning, which is a consequence of grid non-uniformity, inhomogeneity in the material properties, and separation of scales, is a common challenge in solving elliptic problems. The use of preconditioners to alleviate this problem is often hindered by the problem-specific tuning requirement and the introduction of additional computational cost [5, 6, 7].

The ill-conditioning of elliptic problems becomes a pressing issue as the size of the system grows. Since numerical representations of PDEs are based on local stencils, larger systems require more iterations to establish a correct coupling between physically distant points governed by an elliptic system. As a result, larger problems require more iterations, leading to a scaling worse than $O(N)$. One technique to tackle this issue is the multigrid approach [8, 9, 10]. The underlying idea behind multigrid methods is to accelerate the propagation of information to distant points by mapping the problem to a coarser grid. The solution on the coarser grid is then used as a surrogate to reduce the iterations needed for resolving large-scale structures of the solution at the fine grid. In general, multigrid methods can be divided into two classes: algebraic and geometric ([11, 12, 13, 14, 15]). Algebraic multigrid techniques, although applicable to a wider range of linear systems, are often less robust, more complex to implement, and less efficient than the geometric counterpart. Geometric multigrid methods, on the other hand, are well suited for applications with simple geometry that are large enough in size to justify the added complexity.

In this study, we are interested in the efficient solution of linear systems arising in incompressible flows with density variations, in which the boundary conditions are arbitrary and the grid is structured, rectilinear, and nonuniform. Fractional time-stepping methods are often used in this class of problems, where the conservation of mass is satisfied by solving an elliptic linear system [16]. This linear system, as a result of grid non-uniformity and density variation, is not symmetric and varies in time. Typically, the entries of the associated matrix vary significantly due to the grid refinement near the walls. The linear system may contain up to billions of unknowns, considering the intended application in direct numerical simulation of relatively high-Reynolds number flows. The geometric multigrid method is well suited for this class of problems, considering the simplicity of the cuboid geometry and the size of the linear system. Therefore, the objective of this study is to design an efficient and scalable geometric multigrid algorithm for large nonsymmetric linear systems resulting from discretization of elliptic operators on cuboid domains. Although we develop this algorithm and benchmark it for the class of problems described above, the outcome applies to any elliptic PDE that conforms to our formulation in the discrete setting.

The key components of a multigrid method that broadly affect its convergence rate are interpolation and restriction operations. A wide range of variants of these operations has been

introduced in the past with the intention of improving efficiency, stability, and generality of the underlying multigrid method [17]. In particular, methods have been introduced based on the local solution of the underlying PDE [9], decomposition of stencil points [18], agglomeration of neighboring elements [19, 20], retriangulation of a subset of nodes [21], energy minimization [22], and incomplete Gauss elimination [23] as candidates for interpolation and restriction operations. In this study, we propose a new approach that maps the nonuniform grid to a uniform coarse grid such that the two grids may not have overlapping nodes. The interpolation and restriction operations are constructed based on this mapping through a straightforward procedure. Apart from simplicity and robustness, this method achieves convergence rates that are independent of the number of unknowns and density ratio. The robustness of our algorithm to sharp variation in the density field allows for its utilization in two-phase flow problems, where density varies in space by several orders of magnitude.

The massive size of the linear systems under consideration mandates the use of parallel computing [24, 25, 26]. Maintaining the efficiency is the main challenge in the design of parallel algorithms since it is desirable to minimize the time-to-solution by employing a large number of processors. The design of a scalable algorithm is intertwined with that of the underlying iterative method as it relies on minimizing processor-to-processor communications and maintaining a load balance between all processors [27, 28, 29]. A naive design that relies on a static partitioning encounters load imbalance on nonuniform grids and loses efficiency on the coarser grids. To address these challenges, we will discuss a partitioning approach that is built on top of the proposed geometric multigrid technique.

This paper is organized as follows. In Section 2, we present a geometric multigrid technique for nonuniform grids that relies on a mapper to uniform grids and discuss a recursive implementation of this algorithm for optimal rate of convergence. In Section 3, we introduce our partitioning approach, designed for improved parallel scalability. In Section 4, after describing the adjustable parameters of the present multigrid solver, we discuss a reproducible test case for independent validation of our results. We then perform numerical experiments using (1) triply periodic isotropic turbulence on uniform grids, and (2) variable density duct simulations on nonuniform grids. These cases are selected, considering the application of the present multigrid in the study of particle-laden turbulent flows [30] and particle-based solar receivers [31, 32]. We present weak and strong scaling results to establish the parallel performance of our method. Two-phase flow problems are also considered to establish the robustness of our algorithm to large density variation. To benchmark our method, we compare it against a multigrid technique from the Trilinos package [33] and conventional Krylov-based iterative methods [34] in Appendix A.

2. A geometric multigrid method

In this section, we discuss our geometric multigrid method, which includes restriction (fine-to-coarse) and interpolation (coarse-to-fine) operations, a recursive implementation of the multigrid method, and the main algorithm. In what follows, roman superscripts are used to denote variable names, and italic subscripts are used as indices. Superscripts c and f are used to denote variables on coarse and fine grids, respectively. Indices i, j, and k are used for fine, and I, J, K are used for coarse grids.

The multigrid technique tackles the issue of ill-conditioning by reducing the number of cells in all directions, thus reducing the number of iterations required for the propagation of informa-

tion. Hence, given the linear system defined on a fine grid

$$\boldsymbol{A}^{\mathrm{f}}\boldsymbol{x}^{\mathrm{f}} = \boldsymbol{b}^{\mathrm{f}}, \tag{1}$$

the objective is to solve an equivalent system on a coarse grid as

$$\boldsymbol{A}^{\mathrm{c}}\boldsymbol{x}^{\mathrm{c}} = \boldsymbol{b}^{\mathrm{c}}, \tag{2}$$

such that the difference between $\boldsymbol{x}^{\mathrm{c}}$ and $\boldsymbol{x}^{\mathrm{f}}$ is minimal when mapped to the physical domain. To obtain Eq. (2) from Eq. (1), one needs to map $\boldsymbol{b}^{\mathrm{f}}$ to $\boldsymbol{b}^{\mathrm{c}}$ and define $\boldsymbol{A}^{\mathrm{c}}$ such that the solutions to both systems are as close as possible when mapped to the same grid. More specifically, we seek a linear system defined on the coarse grid that its solution is close enough to the solution of the original system on the fine grid such that the difference between the two can be resolved with minimal computational effort. To achieve this goal, we take the PDE that produces Eq. (1) and re-discretize it on a coarse grid. This process reduces to two core operations, which are mapping a field from the fine grid to the coarse grid and *vice versa*. We denote the former (restriction) by C and the latter (interpolation) by $\mathcal{F}$. Exploiting these two operators, the right-hand side (RHS) of Eq. (1) is mapped to the coarse grid using

$$\boldsymbol{b}^{\mathrm{c}} = C(\boldsymbol{b}^{\mathrm{f}}), \tag{3}$$

and the solution to Eq. (2) is mapped back to the fine grid using

$$\boldsymbol{x}^{\mathrm{f}} = \mathcal{F}(\boldsymbol{x}^{\mathrm{c}}). \tag{4}$$

Denoting the cardinality of a set or vector by $|\bullet|$, $\mathcal{F} : \mathbb{R}^{|x^{\mathrm{c}}|} \mapsto \mathbb{R}^{|x^{\mathrm{f}}|}$ and $C : \mathbb{R}^{|b^{\mathrm{f}}|} \mapsto \mathbb{R}^{|b^{\mathrm{c}}|}$.

The left-hand side matrices in Eqs. (1)-(2) are discrete nonsymmetric elliptic operators. In a continuous form, they represent a class of PDEs of the type

$$\nabla \cdot (\kappa \nabla T) = q, \tag{5}$$

which governs heat conduction in a solid with variable conductivity, $\kappa(\boldsymbol{\xi})$, under volumetric heating, $-q(\boldsymbol{\xi})$, and temperature, $T(\boldsymbol{\xi})$, with $\boldsymbol{\xi}$ the position vector. Here, the heat equation is chosen to provide a physical intuition; same equation governs the pressure field in a variable density flow by replacing T with pressure, κ with the inverse of density, and q with the divergence of predicted velocity plus the contribution from the energy equation [35].

The differential operator in Eq. (5) can also be defined in the discrete form for a given grid $\mathcal{G}$. We denote the discrete form of the elliptic operator, which would be a function of κ, by $\mathcal{D}_{\kappa}\{\mathcal{G}\}$. With this definition, the left-hand-side matrix in Eq. (1) is the discrete elliptic operator on the fine grid $\mathcal{G}^{\mathrm{f}}$, and thus

$$\boldsymbol{A}^{\mathrm{f}} = \mathcal{D}_{\kappa^{\mathrm{f}}}\{\mathcal{G}^{\mathrm{f}}\}. \tag{6}$$

Through numerical experiments, in which we considered various combinations of calculating $\boldsymbol{A}^{\mathrm{c}}$ directly from $\boldsymbol{A}^{\mathrm{f}}$ and the underlying PDE itself, we found that

$$\boldsymbol{A}^{\mathrm{c}} = \mathcal{D}_{\kappa^{\mathrm{c}}}\{\mathcal{G}^{\mathrm{c}}\} \tag{7}$$

where $\kappa^{\mathrm{c}} = C(\kappa^{\mathrm{f}})$, provides a good estimate of an "optimal" $\boldsymbol{A}^{\mathrm{c}}$. An optimal $\boldsymbol{A}^{\mathrm{c}}$ in this context for given arbitrary non-singular $\boldsymbol{A}^{\mathrm{f}}$ and $\boldsymbol{b}^{\mathrm{f}}$ can be defined as $\mathrm{argmin}_{\boldsymbol{A}^{\mathrm{c}}} \left\| (\boldsymbol{A}^{\mathrm{f}})^{-1}\boldsymbol{b}^{\mathrm{f}} - \mathcal{F}\left((\boldsymbol{A}^{\mathrm{c}})^{-1}C(\boldsymbol{b}^{\mathrm{f}})\right)\right\|$.

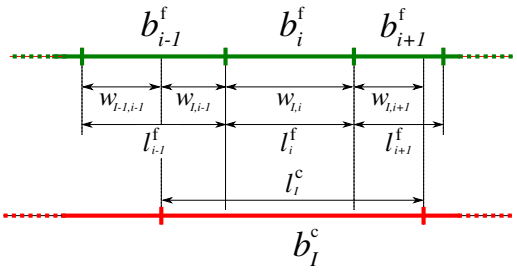

Figure 1: One-dimensional mapping of a field $\boldsymbol{b}$ between a fine (top/green) and a coarse (bottom/red) grid.

Before proceeding any further, it is necessary to specify operators C and $\mathcal{F}$. To satisfy conservation of energy (or mass), the integral of q in Eq. (5) over the entire computational domain must be preserved. In a discrete setting, this constraint translates to a similar condition on C pertaining to Eq. (3). To satisfy this constraint on any interval, we enforce it locally on all grid points, i.e., b^{c}_I with $I \in \{1, \cdots, |\boldsymbol{b}^{\mathrm{c}}|\}$. As shown schematically in Figure 1 for a one-dimensional grid, we accomplish this by relating $\boldsymbol{b}^{\mathrm{f}}$ to all the cells that overlap with b^{c}_I.

The definition of C relies on the exact relationship between Eqs. (5) and (1), which itself depends on the underlying spatial discretization method. To design a consistent method, one could compare the dimension of entries of $\boldsymbol{b}^{\mathrm{f}}$ relative to q for a given discretization approach. For a finite-volume technique, b^{f}_i is computed by integrating q over cell i and thus has a unit of heat (W), whereas, for the finite-difference discretization, its unit is identical to q that is heat per unit volume (W/m^3). In what follows, the restriction operation is defined assuming b^{f}_i has a unit of W/m^3 and is computed through a finite-difference scheme. To adopt this formulation for a finite-volume discretization, either entries of $\boldsymbol{b}^{\mathrm{f}}$ must be multiplied by cell volumes and $\boldsymbol{A}^{\mathrm{f}}$ adjusted accordingly or a volume ratio pre-factor be incorporated in the definition of C as outlined below.

To define C, we first define $w_{I,i}$ as the fraction of length of cell i that overlaps with cell I (Figure 1). Note that $\boldsymbol{w}$ has a unit of length, $w_{I,i} \geq 0$, and $\boldsymbol{w}\boldsymbol{e}^{\mathrm{f}} = \boldsymbol{l}^{\mathrm{c}}$, where $\boldsymbol{l}^{\mathrm{c}}$ is an array that contains the length of coarse grid cells and $\boldsymbol{e}^{\mathrm{f}}$ is a vector with one at all its entries defined on $\mathcal{G}^{\mathrm{f}}$. Also, $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{e}^{\mathrm{c}} = \boldsymbol{l}^{\mathrm{f}}$, in which $\boldsymbol{l}^{\mathrm{f}}$ is an array that contains length of coarse grid cells. These two properties are a direct consequence of the fact that $\boldsymbol{w}$ preserves the integral quantity.

The conservation of energy in one dimension requires $l^{\mathrm{c}}_I b^{\mathrm{c}}_I$ that is the heat released in cell I to be equal to the sum of heat released in the fine grid cells overlapping with cell I. As a result, $l^{\mathrm{c}}_I b^{\mathrm{c}}_I = \sum_i w_{I,i} b^{\mathrm{f}}_i$ condition must be satisfied, allowing us to define C operator as $C(\boldsymbol{b}^{\mathrm{f}}) = \boldsymbol{b}^{\mathrm{c}} = \boldsymbol{v}\boldsymbol{b}^{\mathrm{f}}$ with

$$v_{I,i} = \frac{w_{I,i}}{l^{\mathrm{c}}_I}. \tag{8}$$

In three dimensions, $\boldsymbol{v}$ is extended to three matrices, $\boldsymbol{v}^{\mathrm{x}}$, $\boldsymbol{v}^{\mathrm{y}}$, and $\boldsymbol{v}^{\mathrm{z}}$ for x, y, and z directions, respectively. Thus, C is expressed as

$$C(\boldsymbol{b}^{\mathrm{f}})_{IJK} = b^{\mathrm{c}}_{IJK} = \sum_i \sum_j \sum_k v^{\mathrm{x}}_{I,i} v^{\mathrm{y}}_{J,j} v^{\mathrm{z}}_{K,k} b^{\mathrm{f}}_{ijk}. \tag{9}$$

In Eq. (9), each summation must be carried out on a limited number of overlapping cells; hence, the computational cost remains as $O(|\boldsymbol{b}^{\mathrm{f}}|)$.

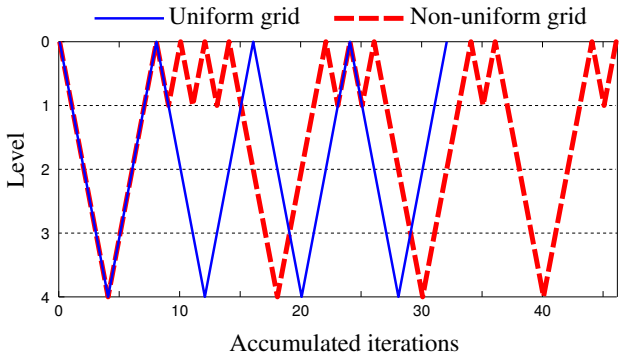

Figure 2: The dynamic pattern of multigrid restriction and interpolations for two representative cases: 1) a uniform 128^3 grid (blue solid) and 2) a nonuniform $120 \times 80 \times 80$ grid (red dashed).

Operator $\mathcal{F}$ is also defined and extended to three dimensions in a similar manner. The only difference is that $\boldsymbol{w}$ is normalized by $\boldsymbol{l}^{\mathrm{f}}$ in this case. Defining

$$u_{i,I} = \frac{w_{I,i}}{l_i^{\mathrm{f}}}, \tag{10}$$

in one dimension we have $\boldsymbol{b}^{\mathrm{f}} = \boldsymbol{u}\boldsymbol{b}^{\mathrm{c}}$, and in three dimensions

$$\mathcal{F}(\boldsymbol{b}^{\mathrm{c}})_{ijk} = b_{ijk}^{\mathrm{f}} = \sum_I \sum_J \sum_K u_{i,I}^{\mathrm{x}} u_{j,J}^{\mathrm{y}} u_{k,K}^{\mathrm{z}} b_{IJK}^{\mathrm{c}}, \tag{11}$$

in which $\boldsymbol{u}^{\mathrm{x}}$, $\boldsymbol{u}^{\mathrm{y}}$, and $\boldsymbol{u}^{\mathrm{z}}$ are the corresponding weights for x, y, and z directions, respectively. Note that $\boldsymbol{u}\boldsymbol{e}^{\mathrm{c}} = \boldsymbol{e}^{\mathrm{f}}$, viz. a field on $\mathcal{G}^{\mathrm{c}}$ with constant value is mapped to a constant value on $\mathcal{G}^{\mathrm{f}}$.

If $\boldsymbol{b}$ is computed by volumetric integration over the cells, as is the case with the finite-volume discretization, C definition changes by setting $v_{I,i} = w_{I,i}/l_i^{\mathrm{f}}$ in Eq. (8). The definition of $\mathcal{F}$, on the other hand, remains unchanged as long as the state variable is temperature T and $\boldsymbol{x}$ is a sample or volumetric average of T.

Conventionally, multigrid methods have a preset pattern of restriction and interpolation operations. The most common patterns are V cycle, i.e., m restrictions followed by m interpolations, and W cycle, i.e., m levels of restriction followed by $\tilde{m} \leq m$ interpolations, $\tilde{m}$ restrictions, and m interpolations. The performance of each of these methods depends on the quality of the solution on the coarser levels. For one class of problems, one cycle of restriction might be sufficient, while for another, several cycles might be necessary. Therefore, we do not preset a pattern, but rather allow it to be determined dynamically through a recursive implementation (see Figure 2). This recursive implementation is adopted to achieve a performance that is less sensitive to the underlying problem by dynamically emphasizing on levels that require more iterations to converge.

Apart from the cycle pattern, the overall cost depends on the efficiency of the smoothing operations, which are the few iterations performed at the intermediate levels of the multigrid scheme to remove high wavenumber errors. The Krylov-based iterative methods, namely the conjugate gradient (CG) for symmetric matrices and the bi-conjugate gradient (BCG) for nonsymmetric matrices, as well as other smoothing techniques such as Gauss-Seidel and relaxed-Jacobi [36]

are examined as candidates for smoother. Our results in Section 4.2 show that the choice of smoother moderately influences the overall performance of the multigrid solver. Therefore, we use the CG and BCG for symmetric and nonsymmetric matrices, respectively, throughout this study unless stated otherwise. These two iterative methods act as the smoother at the intermediate levels and solver at the coarsest level of the multigrid algorithm.

In our numerical experiments, the CG outperforms the BCG. To exploit this superior performance, we ensure that all restriction operations lead to a symmetric matrix. This way, the BCG is only used to solve a given nonsymmetric matrix at the finest level and smooth the original system. To ensure a restriction operation leads to a symmetric matrix, we build the coarse grid such that it is uniform in any given direction, i.e., $l_i^{\mathrm{c}} = \overline{\boldsymbol{l}^{\mathrm{c}}}$, in which

$$\overline{\boldsymbol{l}^{\mathrm{c}}} \equiv |\boldsymbol{l}^{\mathrm{c}}|^{-1} \sum_i l_i^{\mathrm{c}} \tag{12}$$

is the average grid size. Note that $|\boldsymbol{l}^{\mathrm{c}}|$ and $\sum_i l_i^{\mathrm{c}}$ are equal to the number of cells and the length of the domain, respectively. Through numerical experiments, we found that the best restriction strategy is to increase the average grid size at maximum by a factor of two in each direction while simultaneously restricting to a more isotropic grid. On a highly stretched grid, this translates into significant coarsening to no coarsening (or even refining) on different segments of the grid and in different directions. This strategy is superior to an evenly-distributed coarsening of the grid, in which $\boldsymbol{l}^{\mathrm{c}}$ is nonuniform and has a similar profile to $\boldsymbol{l}^{\mathrm{f}}$. Mathematically, given the average grid size of the fine grid in x, y, and z directions as, $\overline{\boldsymbol{l}^{\mathrm{f,x}}}$, $\overline{\boldsymbol{l}^{\mathrm{f,y}}}$, and $\overline{\boldsymbol{l}^{\mathrm{f,z}}}$, respectively, we define a target coarse grid size as

$$\Delta \equiv 2 \min\left(\overline{\boldsymbol{l}^{\mathrm{f,x}}}, \overline{\boldsymbol{l}^{\mathrm{f,y}}}, \overline{\boldsymbol{l}^{\mathrm{f,z}}}\right). \tag{13}$$

Setting the size of the grid to Δ in all directions can lead to refinement in the direction with the largest $\overline{\boldsymbol{l}^{\mathrm{f}}}$. Hence, $\overline{\boldsymbol{l}^{\mathrm{c}}}$ in each direction is determined based on $\max\left(\Delta, \overline{\boldsymbol{l}^{\mathrm{f}}}\right)$. More specifically,

$$\overline{\boldsymbol{l}^{\mathrm{c,x}}} = \max\left(\Delta^{\mathrm{x}}, \overline{\boldsymbol{l}^{\mathrm{f,x}}}\right), \tag{14}$$

in which Δ^{x} is the nearest value to Δ with $\sum_i l_i^{\mathrm{f,x}}/\Delta^{\mathrm{x}} \in \mathbb{N}$. Similar expressions can also be written for $\overline{\boldsymbol{l}^{\mathrm{c,y}}}$ and $\overline{\boldsymbol{l}^{\mathrm{c,z}}}$.

We next put together all these components to build a multigrid algorithm for a problem defined as: given $\boldsymbol{A}$, $\boldsymbol{b}$, and ϵ, find $\boldsymbol{x}$ such that $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 / \|\boldsymbol{b}\|_2 \leq \epsilon$, where $\| \bullet \|_2$ is the 2-norm operator. First, let $\mathcal{A} = \{\boldsymbol{A}_0, \boldsymbol{A}_1, \cdots \boldsymbol{A}_m\}$ be a set of matrices in which $\boldsymbol{A}_0 = \boldsymbol{A}$, $\boldsymbol{A}_i$ relative to $\boldsymbol{A}_{i+1}$ is analogous to $\boldsymbol{A}^{\mathrm{f}}$ relative to $\boldsymbol{A}^{\mathrm{c}}$ with their relationship fully described above, and m is the maximum number of multigrid levels. Additionally, we define function $\mathcal{K}(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, n, \epsilon)$ to be represented by an iterative solver (the CG for a symmetric and the BCG for a nonsymmetric $\tilde{\boldsymbol{A}}$) that returns $\tilde{\boldsymbol{x}}$ as the solution to $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}$ with a tolerance ϵ and a maximum number of iterations n. Now, we define a recursive multigrid function $\mathcal{M}\left(\mathcal{A}, \hat{\boldsymbol{b}}, i\right)$ that returns $\hat{\boldsymbol{x}}$ as the solution to $\boldsymbol{A}_i\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$ with a tolerance ϵ. With this definition, it is clear that $\boldsymbol{x} = \mathcal{M}(\mathcal{A}, \boldsymbol{b}, 0)$ is the solution to the original problem. We define operator $\mathcal{M}$ as

Algorithm 1. Find $\boldsymbol{x} = \mathcal{M}(\mathcal{A}, \boldsymbol{b}, i)$

if $i = m$

```
    x ← K(A_i, b, n, ε)
    return x

x ← K(A_i, b, n^s, ε^s)
b^f ← b − A_i x
while ||b^f||_2 / ||b||_2 > ε
    b^c ← C(b^f)
    x^c ← M(A, b^c, i + 1)
    x^f ← F(x^c)
    b^f ← b^f − A_i x^f
    x ← x + x^f
    x^f ← K(A_i, b^f, n^s, ε^s)
    b^f ← b^f − A_i x^f
    x ← x + x^f

return x
```

In this algorithm, $\boldsymbol{b}$ is the given right-hand-side vector that is not modified throughout the algorithm whereas $\boldsymbol{b}^{\mathrm{f}}$ is the residual that is updated as the new solution candidates are computed and added to $\boldsymbol{x}$. Hence, due to the condition on the while loop, $\boldsymbol{x}$ is returned only when the solution is converged to the specified tolerance, i.e. $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 / \|\boldsymbol{b}\|_2 \leq \epsilon$ is always satisfied in case of a successful function return. The maximum number of iterations and tolerance of the iterative solver at the coarsest level, when $i = m$, are n and ϵ as opposed to n^{s} and ϵ^{s}, respectively, with $n \gg n^{\mathrm{s}}$ and $\epsilon \ll \epsilon^{\mathrm{s}}$. These differences are to ensure the solution to the coarsest problem fully converges before returning. In other words, two later calls to $\mathcal{K}$ are smoothers while the first call is for solving a linear system. Also, note the recursive call to the next level of the multigrid solver inside the while loop. To prevent an infinite loop, one may add a counter or an upper limit on $\|\boldsymbol{b}^{\mathrm{f}}\|_2$ that calls an exit command in case of excessive iterations. The singularity of the linear system or choosing ϵ to be smaller that machine precision are some of the reasons that prevent $\|\boldsymbol{b}^{\mathrm{f}}\|_2 / \|\boldsymbol{b}\|_2 < \epsilon$ to be satisfied.

3. Partitioning approach

The scalability of an iterative algorithm is a function of its communication overhead and load balance. The communication overhead is directly proportional to the partitioning granularity, which is the ratio between the number of cells shared between processors to the total number of cells, while the load balance relies on an equal number of cells assigned to each processor. Partitioning a Cartesian grid to equal-sized cubes with an equal number of cells in each direction is an optimal choice concerning both the communication overhead and load balance. However, this is not possible in general for an arbitrary grid $\mathcal{G}$ and an arbitrary number of partitions n^{p}. Hence, a compromise must be made between these two factors.

To improve flexibility and increase the number of partitioning combinations, we create a partition-grid, which is a rectilinear grid of partitions that is constructed by dividing the computational grid into a certain number of slices in each direction, with each of its blocks having different sizes. This added flexibility is to obtain an algorithm that remains optimal for a wider combination of $\mathcal{G}$ and n^{p}. As a one-dimensional illustration, consider a grid with 10 cells. A partitioning strategy that only allows for equal-sized partitions only accepts 1, 2, 5, or 10 as the number of processors. In three dimensions, mandating equal-size partitions limits the range

of possible partitioning options significantly. With this limitation removed, the average number of cells in a direction of a partition block, denoted by n_i, can be a positive real number, i.e., $n_i \in \mathbb{R}^+ \; \forall i \in \{1, 2, 3\}$, and not necessarily an integer.

To calculate n_i, we first decompose the total number of partitions to prime numbers

$$n^{\mathrm{p}} = \prod_i p_i^{k_i}, \tag{15}$$

in which $\boldsymbol{p} = \{2, 3, 5, 7, 11, 13, 17, 19\}$ is a truncated sequence of prime numbers, and $k_i \in \mathbb{N}$ is the repetition associated with prime number p_i, which must be calculated for a given n^{p}. Given $\boldsymbol{p}$, $\boldsymbol{k}$, and the total number of cells in each direction $\boldsymbol{N} = [N_1, N_2, N_3] = [|\boldsymbol{l}^{\mathrm{x}}|, |\boldsymbol{l}^{\mathrm{y}}|, |\boldsymbol{l}^{\mathrm{z}}|]$, $\boldsymbol{n}$ is calculated as

Algorithm 2. Find average number of partitioned cells $\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{p}, \boldsymbol{k}, \boldsymbol{N})$

```
n ← N
do i = |p|, ··· , 1
    while k_i > 0
        j ← argmax(n)
        n_j ← n_j / p_i
        k_i ← k_i − 1

return n
```

For example, a $300 \times 200 \times 100$ grid partitioned to $n^{\mathrm{p}} = 30$ will produce $\boldsymbol{n} = [60, 66.667, 50]$, viz. partitioning x-direction to 5, y-direction to 3, and z-direction to 2. In y-direction, this leads to slices with a thickness of 67, 67, and 66 cells, hence yielding a small load imbalance of 1.5%. Additionally, since it is the largest entry of $\boldsymbol{n}$ that is divided by p_i, the partitioned blocks are the closest possible cuboids to cubes.

To maximize parallel efficiency, we separately partition all the m grids employed in the multigrid algorithm according to the algorithm described above. These independent partitions can be significantly misaligned between adjacent levels (specifically between uniform and nonuniform grids). This misalignment has a direct implication on the implementation of C and $\mathcal{F}$ in terms of processor-to-processor communication. To simplify such implementation, we perform each of those operations in three steps. In the first step, considering C for example, values from the fine grid are copied into a buffer according to Eq. (9). Second, we communicate the buffer between processors, and third, assign it to the coarse grid. Using a buffer simplifies treatment of boundaries, specifically when multiple cells, each from a different partition, contribute to a single cell on the coarse grid.

Finally, we note that in the case of large m and n^{p}, there can be an excessive number of processors for solving a small system at the coarser grids. To prevent this issue, we impose a lower bound on n_i to ensure a minimal number of cells per processor. Inevitably, some processors will be idle when this lower bound is reached. Not employing the full capacity of the machine in these cases has minimal effect on the overall performance because the cost of calculations on these coarse grids constitute a negligible fraction of the entire cost. The optimal choice of this lower bound depends on the hardware latency among others, and thus, is machine dependent. However, our numerical experiments show that its choice has a negligible effect on the overall performance. Hence, the lower bound on n_i is set to 10 in all the calculations below.

4. Numerical test cases

The multigrid method described in the previous sections is implemented in an in-house code that is designed for solving the Navier-Stokes equations for variable-density flows. The coupling between the multigrid solver and the main fluid solver allows for substitution of the present multigrid solver with an external linear solver, an exploited feature in Appendix A. Manufactured solutions are employed to validate the solver by comparing numerical and analytical solutions and ensuring the error drops as the square of the grid size. The solution obtained from the multigrid solver is verified outside of the linear solver by examining the residual and ensuring that it is within the specified tolerance ϵ.

The only performance related parameter that impacts the end-user is the computational cost, a parameter that is selected as the figure of merit in this study. Although the computational cost depends on the system architecture, the machine hardware, various software components, and the user-specific implementation of the algorithms, nevertheless, it provides a comparative measure of the overall performance of the multigrid solver. In contrast to the number of iterations, which is adopted as the figure of merit in some studies, it captures the cumulative cost associated with various components of the multigrid algorithm. Moreover, it provides a mean for examining the parallel efficiency of the algorithm described in Section 3. The dependence on the system architecture is expected to influence the absolute performance values uniformly for all the test cases, thereby not changing any conclusion drawn based on the relative performance values.

After each numerical experiment, we record the physical time to solution t^{s} as a basis for comparison of the cost. It is of critical importance to note that t^{s} is obtained based on the multigrid solver portion of the code only and the calculations outside of the multigrid solver, such as the construction of the linear system, do not contribute to the performance estimates. Using t^{s}, the performance results are reported in terms of computational throughput $\mathcal{T}$, defined as the number of unknowns solved per second

$$\mathcal{T} = N/t^{\mathrm{s}}. \tag{16}$$

$N = N_1 N_2 N_3$ refers to the number of unknowns of the original system that are solved to the tolerance of $\epsilon = 10^{-7}$. We also report parallel efficiency η, computed as the ratio of throughput per processor to the maximum throughput per processor

$$\eta = \frac{\mathcal{T}/n^{\mathrm{p}}}{\max(\mathcal{T}/n^{\mathrm{p}})}. \tag{17}$$

4.1. The adjustable parameters of the multigrid

The parameters of the multigrid method, described in Section 2, are chosen according to the following set of guidelines and are kept fixed throughout this study. The first parameter is the maximum number of iterations at the coarsest level n. The choice of n is straightforward because the linear system at the coarsest level must converge to the given tolerance. In other words, we design the solver at the coarsest level such that the exit decision is based on the tolerance only and not the number of iterations. In our computations, we found the choice of $n = 500$ is sufficiently large to ensure convergence at the coarsest level over a broad range of mesh and field conditions.

The best choice of parameters for the smoother, namely the tolerance ϵ^{s} and the maximum number of iterations n^{s}, lies in a finite range of values. For a moderate ϵ^{s} and sufficiently large n^{s}, the performance of the multigrid solver will be independent of n^{s}. Conversely, for a moderate n^{s} and sufficiently small ϵ^{s}, the results will be independent of the choice of ϵ^{s}. However, choosing

a large n^s or a small ϵ^s is not advised. Large n^s may lead to resolving the large features of the solution (low wavenumbers) by the smoother at fine levels, which are otherwise captured at much lower cost at the coarser grids. Therefore, we need to compromise between the higher cost of computation at the fine grid and the necessity of resolving high wavenumbers by the smoother. The correct balance is not always achieved at a particular n^s or ϵ^s. For instance, when the contribution of low wavenumbers is large, it is better to choose a smaller n^s and resolve the solution at the coarser grid. To ensure sufficient smoothing while preventing excessive iterations, we adopt a loose tolerance and a bounded maximum number of iterations. Through several numerical experiments with different cases, we found $\epsilon^s = 0.15$ and $n^s = 8$ are universally nearly optimal. The reported results, however, could be slightly improved if these parameters were to be optimized case by case.

Two remaining parameters are the maximum number of restriction levels m and the tolerance of the overall algorithm ϵ. In practice, ϵ is application-dependent, and its choice directly determines the cost. $\log(1/\epsilon)$ is linearly related to the number of required iterations. This linear dependence is shown in Figure 3, which is extracted from the simulation of turbulent flow in a duct on a 480×320×320 grid that is described in details later in Section 4.3. The absolute number of iterations – calculated as the number of matrix-vector products at the finest grid – might appear to be excessive in comparison to a typical multigrid solver. This higher number is primarily due to the ill-conditioning of the underlying system and the large number of restriction levels m. Regardless of m, a linear convergence is observed up to machine precision. Provided this linear convergence, the comparison between various test cases is not affected by the choice of ϵ, and hence is fixed at 10^{-7} for all the results reported. The number of multigrid levels m can have a large influence on the cost if it is too small. One can verify that for $m = 0$, the multigrid algorithm reduces to a standard iterative method, significantly reducing the overall performance in case of large linear systems. As m increases, the total number of iterations increases and the effect of the coarser grid on the error, which is highly oscillatory at larger m, becomes more apparent (Figure 3). Note that in this case, the lower number of iterations at the finest grid does not translate into a lower overall cost because the cost of the coarser grid solver dramatically increases for smaller m. For the problem sizes considered in this study the overall performance asymptotes to its minimum for $m \geq 3$ to 4. Therefore, $m = 4$ in all our computations. Although $m = 4$ is sufficiently large for all cases discussed here, larger m might produce a detectable improvement in performance for problems that are significantly larger than those considered here.

4.2. Heat conduction test cases

In this section, the effect of grid spacing and anisotropy and also various smoothers on the performance of the multigrid solver is examined. The test case is constructed as a benchmark to allow for the reproduction of our results in the future. Additionally, the discrete solution is compared against an asymptotic solution to demonstrate the physical relevance of the underlying linear system. Hence, we first describe the problem setup in detail and then present the performance evaluation of the multigrid solver.

A cuboid solid with a constant heat conductivity that is heated at its center is considered (Figure 4). The faces normal to the y-direction are isothermal walls. A periodic boundary condition is imposed on other faces. The grid is uniform in the periodic directions and nonuniform in the y-direction with

$$l_i^y = \frac{2}{\alpha - 1}\left[\frac{\alpha^{2i/N_2} - 1}{\alpha^{2i/N_2 - 1} + 1} - \frac{\alpha^{2(i-1)/N_2} - 1}{\alpha^{2(i-1)/N_2 - 1} + 1}\right], \qquad i = 1, \cdots, N_2. \tag{18}$$

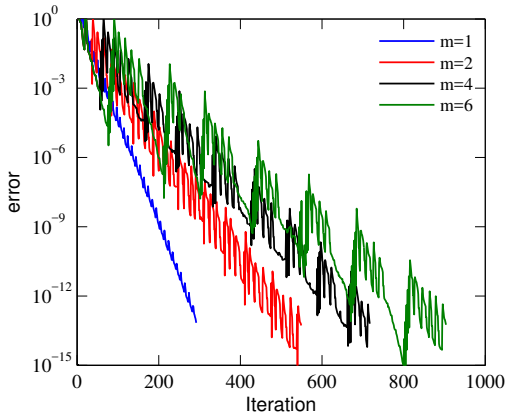

Figure 3: The error as a function of the number of iterations for the maximum number of restriction levels $m = 1$ (blue), 2 (red), 4 (black), and 6 (green). The number of iterations is computed as the number of matrix-vector products performed at the finest grid. The results are obtained from the duct simulation on a $480 \times 320 \times 320$ grid. Note the increase in the number of iterations as m increases does not translate to higher cost since the cost of the coarser grid calculations are significant at lower m.

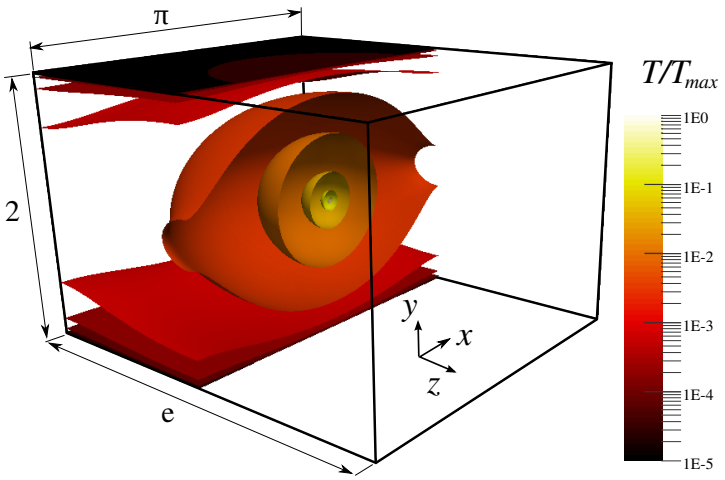

Figure 4: The temperature contours for a block of solid that is heated at its center, extracted from case T5 in Table 1. Top and bottom surfaces are kept at $T = 0$ while the other boundaries are periodic.

Case	$\mathcal{K}$	$\boldsymbol{N}$	α	AR	κ	N_{itr}	$\mathcal{T}$	CR
T0	BCG	$27 \times 35 \times 43$	43	10	20	91	0.46	1.00
T1	GS	$27 \times 35 \times 43$	43	10	20	108	0.45	1.02
T2	RJ	$27 \times 35 \times 43$	43	10	20	150	0.46	1.00
T3	BCG	$17 \times 19 \times 21$	47	10	5.2	77	0.48	0.96
T4	BCG	$53 \times 69 \times 85$	40	10	80	95	0.35	1.31
T5	BCG	$105 \times 137 \times 169$	39	10	324	119	0.27	1.70
T6	BCG	$27 \times 35 \times 43$	1	1	1.3	32	0.85	0.54
T7	BCG	$27 \times 35 \times 43$	20	5	7.4	71	0.54	0.85
T8	BCG	$27 \times 35 \times 43$	233	50	240	222	0.23	2.00
T9	BCG	$27 \times 35 \times 43$	480	100	760	308	0.17	2.71

Table 1: The effect of grid spacing and anisotropy and the choice of smoother on the performance of the present multigrid. Results correspond to a cuboid solid heated at its center (Figure 4). $\mathcal{K}$ denotes the employed smoother (BCG: Bi-conjugate gradient, GS: Gauss-Seidel, RJ: Relaxed Jacobi [36]). The grid has $\boldsymbol{N}$ points in each direction and is nonuniform in the y-direction according to Eq. (18) and α. AR denotes the grid aspect ratio in y-direction, i.e. $\max(\boldsymbol{l}^y)/\min(\boldsymbol{l}^y)$. κ is the condition number of the matrix, viz. the ratio of the largest eigenvalue of the matrix to the smallest, in thousands (estimated for T6). N_{itr} is the total number of matrix-vector products at the finest grid for the entire solve. $\mathcal{T}$ is the throughput of unknowns per second in millions and CR is the cost normalized by the cost of the reference case T0.

α in Eq. (18) determines stretching of the grid such that as α increases, the grid becomes more squeezed near the walls. For our second order discretization scheme, the resulting $\boldsymbol{A}$ is nonsymmetric with entries relating two adjacent cells j and $j+1$ being $2/[l^y_j(l^y_j + l^y_{j+1})]$ at row j and column $j+1$ and $2/[l^y_{j+1}(l^y_j + l^y_{j+1})]$ at row $j+1$ and column j. The heating at the center is translated to $b_i = 0$ except for a single cell at the center of the domain at which $\boldsymbol{b}$ has a non-zero entry. The size of the domain, which is $\pi \times 2 \times \exp(1)$, is selected to be irrational and the number of grid points to be arbitrary to test the generality of the coarsening strategy. In total, ten cases are tested where the smoothers, grid spacing, and grid stretching are varied (Table 1). To have a one-to-one comparison of various smoothers, we compute N_{itr} as the total number of matrix-vector products rather than the number of calls to the smoother. Provided the homogeneity of the conductivity and the linearity of the system, all the reported results are independent of the absolute value of the heat conductivity and the heat source at the center. These computations are performed using a single Intel® Xeon® processor E5630.

Cases T0, T1, and T2 show that the choice of smoother has minimal effect on the overall performance of the multigrid solver (Table 1). Despite lower N_{itr} of BCG, its overall cost remains similar to the other smoothers due to the additional operations in BCG algorithm such as vector-vector inner products. Increasing the number of grid points results in a significant increase in the condition number, a slight decrease in throughput, and an increase in N_{itr}. Comparing the cases T3 and T5, N is increased by a factor of 358 (corresponding to over 7-time grid refinement in each direction) whereas the performance only drops by a factor of 1.78. This disproportional drop in $\mathcal{T}$, where $\mathcal{T} \propto \kappa^{-1/7}$, shows the reasonable robustness of the present multigrid solver to the ill-conditioning of large systems, an observation that is reaffirmed by results in Section 4.3. The multigrid solver is less robust to the increase in grid aspect ratio as examined by cases T6, T1, T7, T8, and T9. $\mathcal{T}$ drops by a factor of five as the aspect ratio increases from 1 (uniform grid) to 100 and κ from 1.3×10^3 to 7.6×10^5. Although this drop is significant, it occurs at a lower rate than the rate at which the range of length scales widens from T6 to T9, i.e. $\mathcal{T} \propto \kappa^{-1/4}$ in this case. In overall, these results show the performance of the multigrid solver is insensitive to the

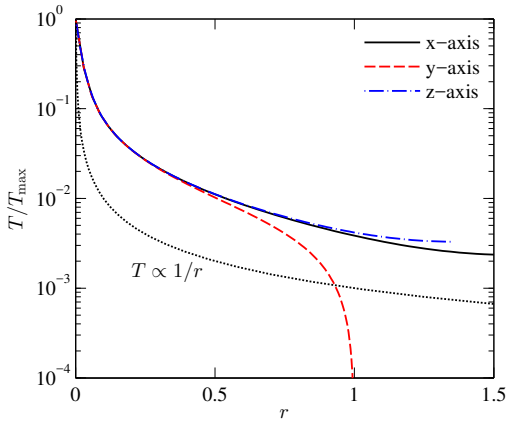

Figure 5: Temperature T variation as a function of distance from the center of the cube r in Figure 4, measured along the x-axis (solid black), y-axis (dashed red), and z-axis (dashed-dot blue). Dotted line shows $T \propto 1/r$ and is included to show the decay rate of $T(r)$.

choice of the smoother and is reasonably robust against variation in grid spacing and anisotropy.

The solution obtained from the multigrid solver is compared against an asymptotic solution in Figure 5. For the simple setup under consideration, where the heat is released in a small cell at the center of the domain, the temperature drops as $1/r$ with r being the distance from the center ($1/r$ is an analytical solution to the Poisson equation in the spherical coordinate system). Near the boundaries, where the solution is affected by the periodicity or isothermal conditions, temperature deviates from $1/r$ behavior. Nevertheless, there is an excellent collapse between $T(r)$ computed along various axes away from the boundaries.

4.3. Fluid dynamics test cases

We consider two sets of test cases in this section. The first set involves direct numerical simulations of isotropic homogeneous turbulence in a cubic box, where the grid is uniform in all directions, and density ($1/\kappa$ in Eq. (5)) is constant. These cases are considered for testing the components of the multigrid algorithm under simplified conditions, in which the linear system is symmetric. The second set involves direct numerical simulation of a radiated particle-laden flow in a duct with square cross section [32, 31]. Particles absorb radiative energy and transfer it to the fluid, hence density varies (i.e. $1/\kappa$ in Eq. (5)) in space by a factor of approximately 3 (Figure 6). In these cases, while the grid is homogeneous in the x-direction, which is the direction of the bulk flow, it is stretched near the walls in the y- and z-directions. The grid is stretched according to Eq. (18) and $\alpha = 58$ such that the cells near the walls have a much smaller l_i in comparison with those at the duct center. For $240 \times 160 \times 160$ grid, this choice of stretching parameter leads to $\max(\boldsymbol{l}^{\mathrm{y}})/\min(\boldsymbol{l}^{\mathrm{y}}) \approx 15$, $\overline{\boldsymbol{l}^{\mathrm{y}}}/\min(\boldsymbol{l}^{\mathrm{y}}) \approx 7$, and $\overline{\boldsymbol{l}^{\mathrm{x}}}/\min(\boldsymbol{l}^{\mathrm{y}}) \approx 28$. The domain is cuboid with an aspect ratio of $6 \times 1 \times 1$. The number of cells in the x-direction is approximately 1.5 times that of y- or z-direction. This set of test cases will evaluate the performance of the present multigrid algorithm for nonsymmetric matrices. For each of these two sets of cases, we ensure that the flow is sufficiently evolved in time such that all wavenumbers are present in the linear system. These

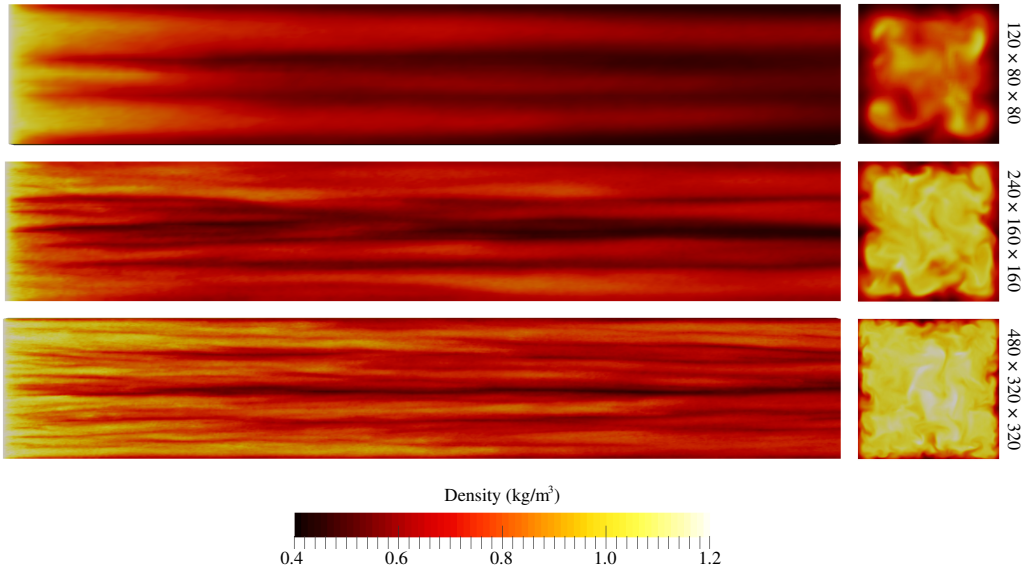

Figure 6: The snapshot of the density field in turbulent particle-laden duct flow simulations. For top to bottom: $120 \times 80 \times 80$, $240 \times 160 \times 160$, and $480 \times 320 \times 320$ stretched grids. The bulk Reynolds number of the flow simulated on these three grids are 5×10^3, 10^4, and 2×10^4, respectively. Fluid density at the side walls and outlets are shown in the left and right, respectively.

two cases are also employed in Appendix A to benchmark our solver by comparing it against the Trilinos's multigrid solver.

To show the ill-conditioning of the linear systems under consideration, we computed the condition number of a series of relatively small systems by varying N, $\max(\rho)/\min(\rho) = \max(\kappa)/\min(\kappa)$, and allowing for grid stretching (Figure 7). The results show that the condition number increases with the number of cells and density ratio. Comparing the stretched grid cases with density ratio of one and the uniform grid cases with an equivalent N shows approximately two orders of magnitude increase in the condition number. This increase in the condition number can be explained by $\overline{\boldsymbol{l}^x}/\min(\boldsymbol{l}^y) \approx 28$ for the stretched grids and the change of boundary conditions from periodic to physics based conditions consistent with no-slip adiabatic walls.

All of the results reported here are obtained from calculations on Titan supercomputer at the Oak Ridge Leadership Computing Facility (OLCF). Titan contains 18,688 compute nodes, each composed of a 16-core 2.2GHz AMD Opteron™ 6274 (Interlagos) processor and 32 GB of RAM. Two nodes share a Gemini™ high-speed interconnect router.

To test the strong scalability and $O(N)$ behavior of the present multigrid algorithm for the first set of test cases described above, three grids with 256^3, 512^3 and 1024^3 cells are considered. Simulations are performed at different n^p, ranging from 16 to over 3×10^4, and throughput and parallel efficiency are computed for each run (Figure 8). The results have a number of implications. The linear increase in throughput for a given grid shows parallel scalability of our implementation since an increase in the number of processors is followed by the same increase in the rate at which the system is solved. At peak performance, 5×10^9 unknowns are solved per second. The collapse of throughput curves at lower n^p in Figure 8-a shows that the cost is $O(N)$.

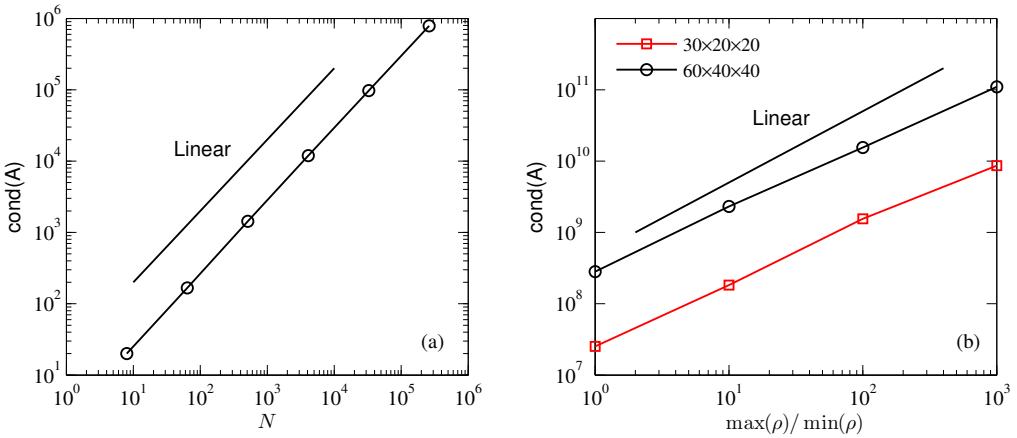

Figure 7: The effect of number of grid points N, mesh anisotropy, and density ratio on condition number: (a) the homogeneous turbulence with isotropic mesh $N_1 = N_2 = N_3 = N^{1/3}$ at a constant density, and (b) heated duct flow with stretched grids where condition number is computed at different level of heating (i.e. $\max(\kappa)/\min(\kappa)$ in Eq. (5)) for two grids $30 \times 20 \times 20$ (red squares) and $60 \times 40 \times 40$ (black circles). Lines with slope of 1 are shown for reference.

In other words, for fixed number of processors, the number of unknowns solved is proportional to the required time for the solution. The collapse of 512^3 and 1024^3 curves is observed, whereas $\mathcal{T}$ for 256^3 is slightly lower than 512^3 (Figure 8-a). The linear extrapolation of 512^3 curve to lower n^p coincides with 256^3 curve, suggesting that the lower performance of 256^3 is partly a result of lower parallel efficiency rather than the super-linear variation of the cost versus N.

To further investigate the variation of parallel efficiency, η is computed for each simulation and also shown in Figure 8-b. The absolute value of efficiency depends on the maximum performance, i.e., $\max(\mathcal{T}/n^p)$, that is only measured on a discrete set of n^p. The measured peak efficiency appears to be a good representative of the actual peak efficiency of the solver, since all curves plateau at large N/n^p. The overall trend of $\eta(N/n^p)$ shows the importance of communication latency and granularity. Parallel efficiency increases sharply at low N/n^p and asymptotes to its highest value as $N/n^p \to \infty$. These two behaviors are typical of the fine-grained (low N/n^p) and coarse-grained (high N/n^p) parallelism, where the communication overhead is significant and negligible in comparison to computations, respectively. The grain size, i.e., the number of unknowns per processor, at which the efficiency drops is of prime importance. It provides an estimate of the largest number of processors that can be efficiently used for a given problem size. The sharp drop in η varies slightly with the problem size. For a larger problem, a larger grain is required to achieve the same level of efficiency as a smaller problem. Considering $\eta = 50\%$ as the minimum acceptable efficiency, the 256^3 case can be run efficiently as long as there are $15,000$ unknowns per processor, whereas a minimum of $60,000$ unknowns is needed per processor for the 1024^3 case.

The reduction in efficiency versus problem size is further analyzed through the weak scaling study of the solver performed by a series of simulations at which the number of unknowns in each partition is held fixed as the number of partitions is increased. Two cases are considered with $N/n^p = 25,000$ (fine-grained) and 50,000 (coarse-grained). These results are also based on the simulations of homogeneous isotropic turbulence on uniform grids (Figure 9). The results show

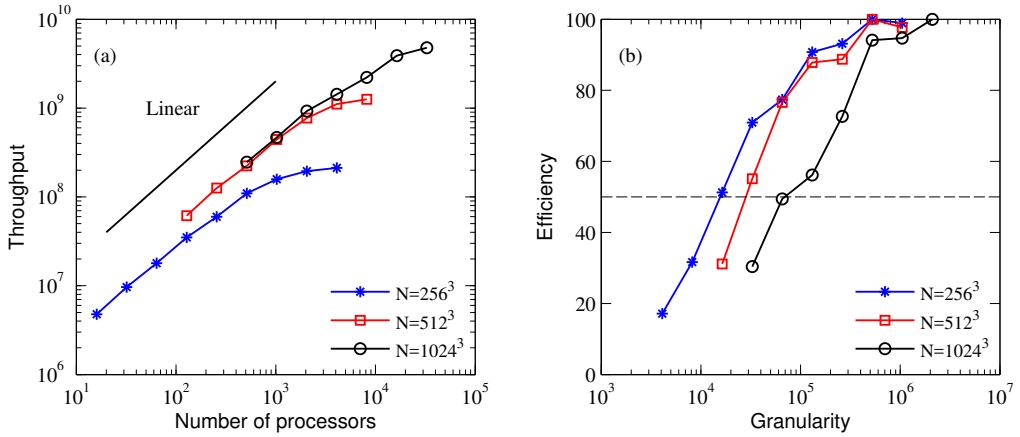

Figure 8: Throughput $\mathcal{T}$ as a function of number of processors n^{p} (a) and parallel efficiency η as a function of number of unknowns per processor N/n^{p} (b). Results, showing the strong scalability for a symmetric matrix, are related to the isotropic turbulence simulation on 256^3 (blue stars), 512^3 (red squares) and 1024^3 (black circles) grids.

an almost linear increase in throughput up to $n^{\mathrm{p}} \approx 10,000$. The throughput of the case $N/n^{\mathrm{p}} = 50,000$ at $n^{\mathrm{p}} = 8$ and $n^{\mathrm{p}} = 128$ indicates that the number of iterations is almost independent of N. The linear increase in throughput is observed in both cases, showing $O(N)$ scaling of the solver even at the finer grain size of $N/n^{\mathrm{p}} = 25,000$ (Figure 9-a). The $O(N)$ variation of the cost underscores the main advantage of the multigrid algorithm in comparison with the conventional iterative techniques, in which the cost increases super-linearly with N.

Parallel efficiency is also computed from the results of the weak scaling study (Figure 9-b). The efficiency remains above 60% for the large-grained case, while for the fine-grained case, it drops to approximately 40% at $n^{\mathrm{p}} = 4,048$. This drop in efficiency is mainly a result of the latency of the collective communications, which hinders scalability by introducing additional cost at large n^{p}. Increasing N/n^{p} from 25,000 to 50,000 improves efficiency as the computations account for a larger portion of the overall cost at larger N/n^{p}.

To test the performance of the solver for nonsymmetric linear systems, we next consider duct flow simulations (Figure 6). These cases allow for benchmarking the solver when the restriction operation C involves a nonuniform-to-uniform mapping and the smoother at the finest grid is based on the BCG. These are in contrast to the cases considered above, where the grid was uniform at all levels and the CG was employed as the smoother. We consider three nonuniform grids, i.e., $320 \times 180 \times 180$, $480 \times 320 \times 320$, and $1024 \times 768 \times 768$. The approximate number of unknowns in these three cases are 10^7, 5×10^7, and 6×10^8, respectively. The number of cells in each direction is dividable to small prime numbers, namely 2 and 3, hence allowing for a wider selection of n^{p} that leads to efficient partitioning. Computations are performed using $16 \leq n^{\mathrm{p}} \leq 18432$ and throughput is measured for each case and plotted in Figure 10-a.

The overall trends observed in Figure 10 are similar to Figure 8, showing that our solver behaves similarly for symmetric and nonsymmetric matrices. For all three problem sizes, throughput increases linearly at lower n^{p}, showing the linear parallel scalability of our solver. The collapse of the curves, as discussed before, indicates that the cost is of $O(N)$ and the number of iterations of the multigrid is independent of the problem size. In comparison to the uniform grid,

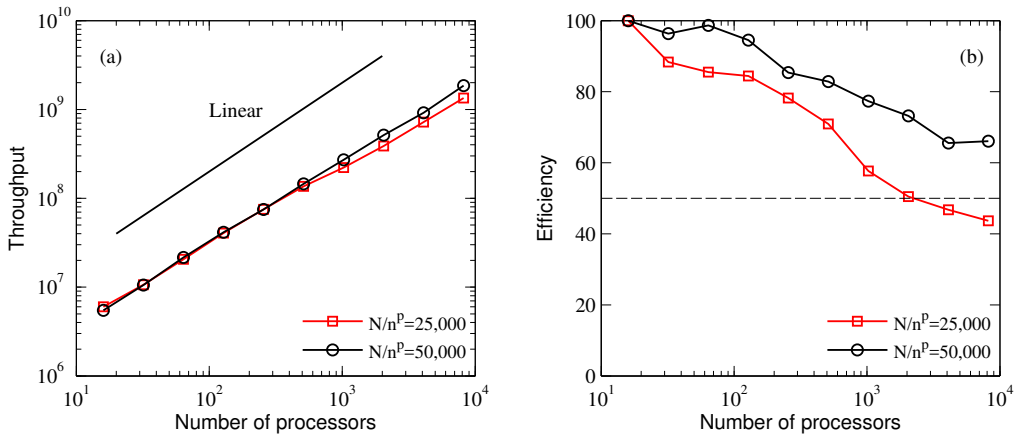

Figure 9: The weak scalability study of throughput $\mathcal{T}$ (a) and efficiency η (b) as a function of number of processors n^{p}. Results are obtained from the isotropic turbulence simulations on a uniform grid with 25,000 (red squares) and 50,000 (black circles) unknowns per processor.

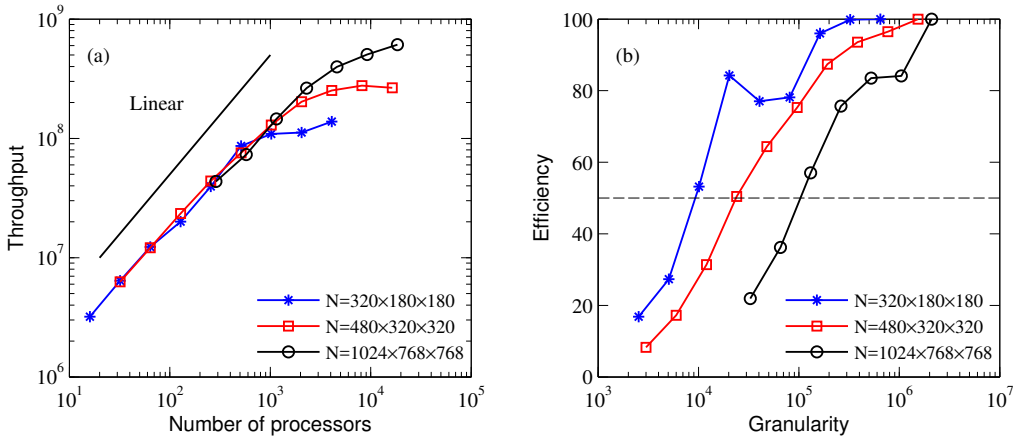

Figure 10: Throughput $\mathcal{T}$ as a function of number of processors n^{p} (a) and parallel efficiency η as a function of number of unknowns per processor N/n^{p} (b). Results, showing the strong scalability of the multigrid solver for nonsymmetric systems, are related to the duct simulation on $320 \times 180 \times 180$ (blue stars), $480 \times 320 \times 320$ (red squares), and $1024 \times 768 \times 768$ (black circles) nonuniform grids.

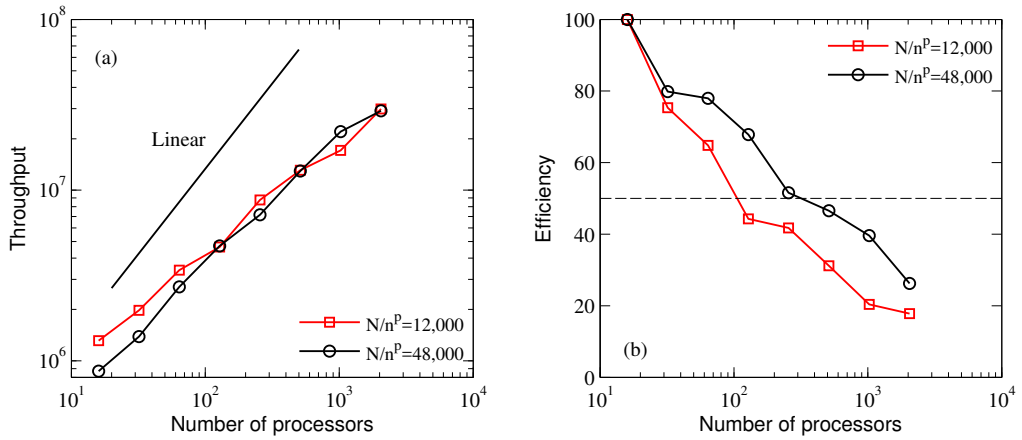

Figure 11: Throughput $\mathcal{T}$ (a) and efficiency η (b) as a function of number of processors n^p. These weak scaling results are obtained from the duct simulations on nonuniform grids with 12,000 (red squares) and 48,000 (black circles) unknowns per processor.

throughput at a given number of unknowns is slightly lower. Considering $n^p = 16$, throughput for the uniform grid is approximately 50% higher than that of nonuniform (Figures 8-a and 10-a). This difference is attributed to the use of the BCG, as opposed to the CG, in the computations involving nonsymmetric systems on nonuniform grids. The maximum throughput achieved in these computations is 6.1×10^8, which is approximately 8 times lower than that of the uniform computations. This larger gap is due to the smaller problem size (by 56%) and lower number of processors (by 56%) utilized in the case of computations with the nonuniform grid.

Parallel efficiency is also computed (Figure 10-b), showing a similar trend to the results obtained from uniform grid computations. Consistent with our previous observation, parallel efficiency drops as grain size decreases. A minimum of 10^4, 2.5×10^4 and 13×10^4 unknowns per processor is required for the three studied problem sizes to achieve $\eta > 50\%$. Comparing these numbers with that of the uniform grid shows that the granularity requirement of the multigrid solver is not affected by the symmetric properties of the underlying system. The granularity requirement, however, is affected by the size of the problem, increasing as N increases.

To further investigate the behavior of the solver at larger problem sizes, we employed a range of nonuniform grids to perform a weak scaling study. The number of unknowns per processor is kept fixed at 12,000 (fine-grained) and 48,000 (coarse-grained) in these computations. The increase in throughput as a function of the number of processors is slower than linear, particularly for the fine-grained case (Figure 11-a). This slower than linear growth is in contrast to the simulations with the uniform grid, which showed $\mathcal{T} = O(n^p)$ (Figure 9-a). The sub-linear growth in the case of the nonuniform grid is partly a result of the grid and n^p selection that leads to an inefficient partitioning. To satisfy the $N = 1.2 \times 10^4 n^p$ or $N = 4.8 \times 10^4 n^p$ condition, the number of grid points in each direction must be selected according to the number of processors. The constructed grid, as a result, may not be dividable to cubes, hence producing an inefficient partitioning and reducing the overall throughput. In practice, analogous to our grid choice in the strong scaling study, it is better to select the number of grid cells from the product of small prime numbers to achieve higher efficiency. This choice of the number of grid cells is particularly

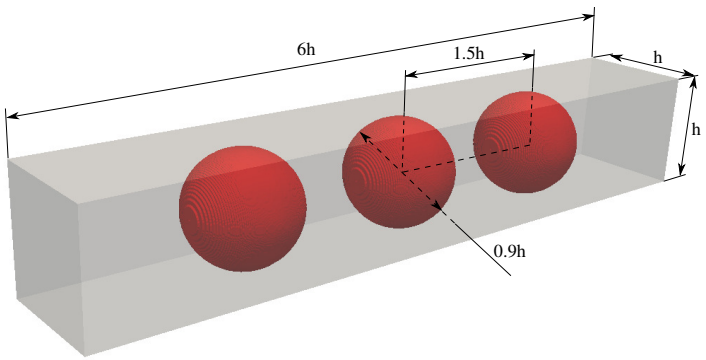

Figure 12: The schematic of a duct flow with three drops (red spheres). The fluid density ρ_r inside these drops is r times larger than the surrounding fluid.

important for fine-grained computations, where the communication latency is sufficiently large to impact the overall performance of the solver.

The sub-linear behavior of throughput for the weak scaling study is reaffirmed by the efficiency results (Figure 11-b). In comparison to the uniform grid computations that showed a plateau at a small number of processors (Figure 9-b), the efficiency drops sharply for the nonuniform computations, even at small n^{p}. Analogous to the uniform grid computations, the coarse-grained nonuniform computations show a higher efficiency in comparison to the fine-grained computations.

4.4. Two-phase flow test cases

To ensure robustness of the present multigrid method to large density variation, we consider the duct flow simulations on the nonuniform grids. The three cases shown in Figure 6 are considered. To obtain different levels of density variation, we generate additional artificial density fields, ρ_r, that have significantly higher or lower fluctuations, by post processing the available physical density data, according to the following mapping

$$\rho_r(\boldsymbol{x}) = \frac{(r-1)\langle\rho\rangle}{\max(\rho) - \langle\rho\rangle + r(\langle\rho\rangle - \min(\rho))}(\rho(\boldsymbol{x}) - \langle\rho\rangle) + \langle\rho\rangle, \tag{19}$$

where r is the desired density ratio, $\rho(\boldsymbol{x})$ is the density field extracted from the reference simulation (shown in Figure 6), and $\langle\bullet\rangle$ denotes spatial averaging. Equation (19) is formulated such that $r = \max(\rho_r)/\min(\rho_r)$ and also $\langle\rho_r\rangle = \langle\rho\rangle$. Furthermore $0 < \rho_r < \infty$, viz. the PDE coefficient (κ in Eq. (5)) remains physical, bounded, and nonzero. Additionally, to test the robustness of our multigrid algorithm to sharp changes in the density field, we consider a two-phase flow problem shown in Figure 12. In this case, three droplets are placed inside the duct with four side walls. ρ_r is uniform everywhere expect at the interface where it increases by a factor of r to the droplets density that is higher than the surrounding fluid. For a fair comparison with the previous case, where density varies continuously in space, we keep the grid and boundary conditions the same between two sets of cases. All these computations are performed on local computing resources (Certainty compute cluster at Stanford University), using 192 processors.

		$120 \times 80 \times 80$		$240 \times 160 \times 160$		$480 \times 320 \times 320$	
ρ_r	r	$\mathcal{T}$	CR	$\mathcal{T}$	CR	$\mathcal{T}$	CR
–	10^0	8.5	1	11.2	1	9.5	1
C	10^1	10.3	0.83	11.0	1.02	8.7	1.09
C	10^2	12.1	0.71	11.7	0.96	11.0	0.86
C	10^3	6.9	1.24	21.1	0.53	15.1	0.63
C	10^4	5.1	1.67	26.9	0.43	14.7	0.65
S	10^1	7.8	1.09	10.9	1.04	9.0	1.05
S	10^2	8.0	1.06	10.7	1.05	8.7	1.10
S	10^3	7.9	1.07	10.3	1.09	8.3	1.14
S	10^4	8.0	1.06	10.1	1.11	7.9	1.20

Table 2: The effect of density ratio on the performance of the present multigrid. ρ_r = C denotes continuous spatial variation of density extracted from the heated duct flow simulations (Figure 6). ρ_r = S denotes sharp spatial variation of density extracted from the two-phase-flow problem (Figure 12). r is the density ratio, $\mathcal{T}$ is the throughput of unknowns per second in millions, and CR is the cost normalized by the cost at the density ratio of 1. These computations are performed using 192 processors on $120 \times 80 \times 80$, $240 \times 160 \times 160$, and $480 \times 320 \times 320$ nonuniform grids.

The results are tabulated in Table 2. It is clear that the performance is not adversely affected by large density ratios. On the contrary, time-to-solution decreases at higher density ratios in some cases regardless of the size of the system. For continuous density field cases, as r grows, the throughput of 480×320×320 and 240×160×160 cases generally increases. The improved rate of convergence is achieved despite the higher condition number at higher r, as was demonstrated in Figure 7-b. Replacing the continuous (Figure 6) with the discontinuous (Figure 12) density field has little effect on the results reported in Table 2 since the cost ratio remains $O(1)$. These test cases show the robustness of the present multigrid to large density ratio, extending its potential use to two-phase flow problems where density ratios as large as 10000 and sharp changes in ρ are commonplace.

5. Conclusions

A scalable geometric multigrid method is introduced for efficiently solving nonsymmetric linear systems resulting from the discretization of elliptic PDE operators on rectilinear grids. This method is constructed based on the underlying PDE, exploiting its conservation properties. A restriction operation is defined such that the resulted grid is always uniform, producing a symmetric linear system at all levels except the finest grid. An optimal pattern of restriction and interpolations is obtained through a recursive implementation. To improve parallel scalability, we introduced a flexible partitioning approach that produces nearly optimal partitioning. Regardless of the multigrid level, all grids are partitioned such that the communication overhead is minimized and the load on all processors is balanced.

The symmetric and nonsymmetric test cases showed that the proposed algorithm is scalable in terms of both the problem size and the number of processors. Increasing the size of the problem led to a linear increase in cost, confirming $O(N)$ behavior of the solver and its robustness against ill-conditioning of the linear system due to the presence of wide range of wavenumbers. For a given problem size, increasing the number of processors led to a linear increase in throughput, confirming parallel scalability of the proposed implementation up to $N/n^{\mathrm{p}} = 10^4$ to

10^5. The excellent robustness of our algorithm to the large spatial variation of density was also demonstrated.

Only a few adjustable parameters appeared in our algorithm. General guidelines were provided for their adjustment. The overall performance of the solver was minimally affected by varying these parameters around the recommended values. The recommended values appear to be universally optimal as a near optimal performance was achieved in all the reported computations that were based on a fixed set of values. Application of our algorithm to a wider range of test cases, where the optimality of these adjustable parameters can be explored, remains a subject for future studies.

Acknowledgments

This work was funded by the United States Department of Energy's (DoE) National Nuclear Security Administration (NNSA) under the Predictive Science Academic Alliance Program II (PSAAP II) at Stanford University. This research used resources of the OAK Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. We acknowledge the use of computational hours on the Certainty cluster at Stanford University, where all these tools were initially developed.

Appendix A. Comparison against Trilinos

To benchmark our results, we repeated some of the calculations presented earlier using the Trilinos software [33]. For these cases, the same linear systems employed for testing our algorithm were loaded to Trilinos. The pre-existing template files, found in the library's documentation, are used to set up the problem. The pre-installed Trilinos library on Titan (version cray-trilinos/11.12.1.3) is employed with the multigrid and CG solvers from MueLu [37] and Belos [38] packages, respectively. The adjustable parameters are tuned based on the recommended criteria and our numerical experiments. In particular, the problem type is set to Poisson-3D for the multigrid solver and the Chebyshev polynomial is employed as the smoother with a V-cycle pattern, as recommended for Poisson linear systems. We selected the remaining parameters according to the default settings. We acknowledge the possibility of achieving higher performance with Trilinos if the different combination of parameters were to be examined to minimize the cost. We did not fully optimize all input parameters for practical reasons, as such an exercise entails an exploration of an over 20-dimensional parameter space.

The direct numerical simulation of homogeneous isotropic turbulence on a uniform 256^3 grid is performed using the present and Trilinos multigrid solver (Table A.3). To provide a reference point, we repeated the same computations using the Trilinos CG solver. From these results, the present multigrid outperforms the Trilinos's multigrid and the CG by a factor of approximately 3 and 10, respectively. As suggested by Figure 7-a, the increase in condition number will lead to higher cost ratio on larger grids. Regarding parallel efficiency, the present implementation is more scalable in comparison to the Trilinos. The better scalability is observed on coarse- and fine-grained computations with N/n^{p} ranging from millions to few thousands.

To establish consistency of the results with respect to the symmetric properties of the linear system, we considered the duct flow simulations on the nonuniform grids (Figure 6). The density variation in the entire domain (i.e., $\max(\rho)/\min(\rho)$) is approximately 3. Computations are performed on three grids $120 \times 80 \times 80$, $240 \times 160 \times 160$, and $480 \times 320 \times 320$ with the number

				present MG			Trilinos MG			BCG		
Case	N	n^{p}	N/n^{p}	$\mathcal{T}$	η	CR	$\mathcal{T}$	η	CR	$\mathcal{T}$	η	CR
U1	17	16	1048	1.2	100	1	0.75	100	1.6	0.11	100	11
U2	17	32	512	2.4	100	1	1.3	84	1.9	0.20	93	12
U3	17	64	256	4.6	97	1	1.9	65	2.4	0.35	82	13
U4	17	128	128	8.7	90	1	2.9	49	3.0	0.59	68	15
U5	17	256	64	16	82	1	4.3	36	3.7	0.88	51	18
U6	17	512	32	29	76	1	8.6	36	3.4	1.8	51	16
U7	17	1024	16	45	59	1	12	26	3.7	3.1	45	14
U8	17	2048	8	54	35	1	15	16	3.6	5.3	38	10

Table A.3: Strong scaling of the present multigrid method and its comparison to the Trilinos's multigrid and the standard CG algorithm. Results correspond to homogeneous isotropic simulations on a 256^3 uniform grid. n^{p} is number of processors, N is number of unknowns in millions, N/n^{p} is the number of unknowns per processor in thousands, $\mathcal{T}$ is throughput of unknowns per second in millions, η is parallel efficiency with respect to the most efficient case of the corresponding algorithm, and CR is the cost normalized by the cost of the present multigrid.

of processors proportional to the problem size. The same computations are also carried out using the multigrid method from the Trilinos package and the standard BCG (Table A.4).

From Table A.4, the present multigrid outperforms the Trilinos's multigrid and the BCG by approximately one and two orders of magnitude, respectively. This difference is much larger than the corresponding difference in the uniform grid computations. Using a standard iterative solver is not a viable option for these computations as a solve that takes a second with the present multigrid takes minutes with the BCG. The extremely low performance of the BCG, which becomes even worst on finer grids, is due to the ill-conditioning of the linear system as discussed in Section 1.

The difference between the performance of the present multigrid and the Trilinos' multigrid becomes larger as N increases. The throughput of the Trilinos's multigrid is 3 times lower than the present multigrid on the coarser grid, whereas, on the finer grid, it is over 10 times lower. The difference between the two solvers is partly due to the better scalability of the present solver in this particular configuration. Consistent with the foregoing results (Figure 11), the present multigrid scales well when the number of unknowns per processor drops from 48,000 to 24,000. The efficiency of the Trilinos's multigrid solver, on the other hand, drops significantly as n^{p} increases (e.g. η drops by 50% from case N5 to N6), leading to its lower overall performance.

References

[1] Y. Saad, Iterative methods for sparse linear systems, SIAM, 2003.

[2] Y. Saad, M. Sosonkina, J. Zhang, Domain decomposition and multi-level type techniques for general sparse linear systems, Contemporary Mathematics 218 (1998) 174–190.

[3] M. Esmaily, Y. Bazilevs, A. Marsden, A bi-partitioned iterative algorithm for solving linear systems arising from incompressible flow problems, Computer Methods in Applied Mechanics and Engineering 286 (2015) 40–62.

[4] F. Shakib, T. Hughes, Z. Johan, A multi-element group preconditioned GMRES algorithm for nonsymmetric systems arising in finite element analysis, Computer Methods in Applied Mechanics and Engineering 75 (1989) 415–456.

[5] P. Fischer, Projection techniques for iterative solution of Ax=b with successive right-hand sides, Computer Methods in Applied Mechanics and Engineering 163 (1998) 193–204.

[6] G. Carey, B.-N. Jiang, Nonlinear preconditioned conjugate gradient and least-squares finite elements, Computer Methods in Applied Mechanics and Engineering 62 (1987) 145–154.

				present MG			Trilinos MG			BCG		
Case	N	n^{p}	N/n^{p}	$\mathcal{T}$	η	CR	$\mathcal{T}$	η	CR	$\mathcal{T}$	η	CR
N1	0.77	16	48	0.81	100	1	0.27	100	3.0	0.01	100	85
N2	0.77	32	24	1.6	100	1	0.46	86	3.5	0.012	63	135
N3	6.1	128	48	4.9	75	1	0.76	35	6.5	0.025	32	200
N4	6.1	256	24	9.8	76	1	1.2	28	8.1	0.043	28	231
N5	49	1024	48	36	69	1	3.2	19	11	–	–	–
N6	49	2048	24	61	58	1	4.2	12	14	–	–	–

Table A.4: Weak scaling of the present multigrid method and its comparison to the Trilinos's multigrid and an in-house implementation of the BCG. Results are related to the duct flow simulations on nonuniform grids. The grid for cases N1 and N2 is $120 \times 80 \times 80$, for N3 and N4 is $240 \times 160 \times 160$, and for N5 and N6 is $480 \times 320 \times 320$. n^{p} is number of processors, N is number of unknowns in millions, N/n^{p} is the number of unknowns per processor in thousands, $\mathcal{T}$ is throughput of unknowns per second in millions, η is parallel efficiency with respect to the most efficient case of the corresponding algorithm, and CR is the cost normalized by the cost of the present multigrid. The BCG did not converge on the large grids within 20 minutes, and thus the corresponding results are not reported.

[7] M. Esmaily, Y. Bazilevs, A. L. Marsden, A new preconditioning technique for implicitly coupled multidomain simulations with applications to hemodynamics, Computational Mechanics 52 (2013) 1141–1152.

[8] N. S. Bakhvalov, On the convergence of a relaxation method with natural constraints on the elliptic operator, USSR Computational Mathematics and Mathematical Physics 6 (1966) 101–135.

[9] W. Hackbusch, Multi-grid methods and applications, volume 4, Springer Science & Business Media, 2013.

[10] A. Brandt, Multi-level adaptive solutions to boundary-value problems, Mathematics of computation 31 (1977) 333–390.

[11] P. Wesseling, Introduction to multigrid methods, Technical Report ICASE-95-11, DTIC Document, 1995.

[12] P. Vaněk, J. Mandel, M. Brezina, Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems, Computing 56 (1996) 179–196.

[13] U. Ghia, K. N. Ghia, C. Shin, High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, Journal of Computational Physics 48 (1982) 387–411.

[14] A. Brandt, Algebraic multigrid theory: The symmetric case, Applied mathematics and computation 19 (1986) 23–56.

[15] P. F. Antonietti, M. Sarti, M. Verani, Multigrid algorithms for hp-discontinuous Galerkin discretizations of elliptic problems, SIAM Journal on Numerical Analysis 53 (2015) 598–618.

[16] L. Jofre, O. Lehmkuhl, J. Ventosa, F. X. Trias, A. Oliva, Conservation properties of unstructured finite-volume mesh schemes for the Navier-Stokes equations, Numerical Heat Transfer, Part B: Fundamentals 65 (2014) 53–79.

[17] T. F. Chan, W. Wan, Robust multigrid methods for nonsmooth coefficient elliptic linear systems, Journal of computational and applied mathematics 123 (2000) 323–352.

[18] P. M. De Zeeuw, Matrix-dependent prolongations and restrictions in a blackbox multigrid solver, Journal of computational and applied mathematics 33 (1990) 1–27.

[19] T. F. Chan, J. Xu, L. Zikatanov, An agglomeration multigrid method for unstructured grids, Contemporary Mathematics 218 (1998) 67–81.

[20] M.-H. Lallemand, H. Steve, A. Dervieux, Unstructured multigridding by volume agglomeration: current status, Computers & Fluids 21 (1992) 397–433.

[21] T. F. Chan, S. Go, J. Zou, Multilevel domain decomposition and multigrid methods for unstructured meshes: algorithms and theory, Technical Report CAM 95-24, Department of Mathematics, UCLA, May 1995.

[22] W. L. Wan, T. F. Chan, B. Smith, An energy-minimizing interpolation for robust multigrid methods, SIAM Journal on Scientific Computing 21 (1999) 1632–1649.

[23] A. Reusken, A multigrid method based on incomplete gaussian elimination, Numerical linear algebra with applications 3 (1996) 369–390.

[24] C. Richter, S. Schoeps, M. Clemens, GPU acceleration of algebraic multigrid preconditioners for discrete elliptic field problems, IEEE Transactions on Magnetics 50 (2014) 461–464.

[25] B. Smith, P. Bjorstad, W. Gropp, Domain decomposition: parallel multilevel methods for elliptic partial differential equations, Cambridge university press, 2004.

[26] U. M. Yang, et al., Boomeramg: a parallel algebraic multigrid solver and preconditioner, Applied Numerical

Mathematics 41 (2002) 155–177.
[27] M. Esmaily, Y. Bazilevs, A. Marsden, Impact of data distribution on the parallel performance of iterative linear solvers with emphasis on CFD of incompressible flows, Computational Mechanics 55 (2015) 93–103.
[28] E. Polizzi, A. H. Sameh, A parallel hybrid banded system solver: the SPIKE algorithm, Parallel computing 32 (2006) 177–194.
[29] D. J. Kuck, E. S. Davidson, D. H. Lawrie, A. H. Sameh, Parallel supercomputing today and the cedar approach, Science 231 (1986) 967–974.
[30] M. Esmaily, A. Mani, Analysis of the clustering of inertial particles in turbulent flows, Physical Review Fluids 1 (2016) 084202.
[31] H. Pouransari, A. Mani, Effects of preferential concentration on heat transfer in particle-based solar receivers, Journal of Solar Energy Engineering 139 (2017) 021008.
[32] E. Farbar, I. D. Boyd, M. Esmaily, Monte carlo modeling of radiative heat transfer in particle-laden flow, Journal of Quantitative Spectroscopy and Radiative Transfer 184 (2016) 146–160.
[33] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, K. S. Stanley, An overview of the Trilinos project, ACM Transactions on Mathematical Software 31 (2005) 397–423.
[34] M. R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, Journal of Research of the National Bureau of Standards 49 (1952) 409–436.
[35] H. Pouransari, M. Mortazavi, A. Mani, Parallel variable-density particle-laden turbulence simulation, arXiv preprint arXiv:1601.05448 (2016).
[36] X. Yang, R. Mittal, Efficient relaxed-Jacobi smoothers for multigrid on parallel computers, Journal of Computational Physics (2016).
[37] A. Prokopenko, J. J. Hu, T. A. Wiesner, C. M. Siefert, R. S. Tuminaro, MueLu User?s Guide 1.0, Technical Report SAND2014-18874, Sandia National Labs, 2014.
[38] E. Bavier, M. Hoemmen, S. Rajamanickam, H. Thornquist, Amesos2 and Belos: Direct and iterative solvers for large sparse linear systems, Technical Report 3, 2012.