jh240080

Innovative multigrid III

Akihiro Fujii (Kogakuin University)

Abstract

Multigrid methods play a critical role in enabling large-scale simulations on modern supercomputers. 'Innovative Multigrid Methods III' was launched in FY2023 as a three-year project. In the second year of the project, we made significant progress in PinT-related studies and also advanced research in performance analysis of coarse grid solver in Multigrid methods.

1 Basic information

1.1 Collaborating JHPCN centers

- Tohoku University
- The University of Tokyo
- Nagoya University
- Kyoto University
- Kyushu University

1.2 Theme area

- Large-scale computational science area

1.3 Project members and their roles

Research items (1), (2) and (3) are described in Section 2.

Akihiro Fujii[3] : (1)
Kengo Nakajima[2,4] :(CoPI) (3),(1)
Matthias Bolten[8] : (1),(2)
Takeshi Iwashita[1] : (1),(2)
Akihiro Ida[10] : (2)
Masatoshi Kawai[6] : (2)
Satoshi Ohshima[5] : (1),(3)
Tetsuya Hoshino[6] : (2),(3)
Toshihiro Hanawa[2] : (2),(3)
Gerhard Wellein[9] : (3)
Kenji Ono[5] : (1)
Ryo Yoda[8] : (1),(3)
Yasuhito Takahashi[7] : (1)
Yen-Chen Chen[11] : (1)
Martin Schreiber[12] : (3)
Christie Alappat[9] : (3)
Teruo Tanaka[3] : (1)
Georg Hager[9] : (3)
Ayesha Afzal[9] : (3)
Bole Ma[9] : (3)

1: Kyoto U., 2: U. Tokyo, 3: Kogakuin U., 4: RIKEN R-CCS, 5: Kyushu U., 6: Nagoya U., 7: Doshisha U. 8: U. Wuppertal*, 9: FAU*, 10: JAMSTEC, 11: KIT* 12: the Université Grenoble Alpes, *:Germany

2 Purpose and Significance of the Research

We are planning to conduct research and development on the following three items:

(1) Research and development of fundamental algorithms in multigrid methods

(2) Parallel reordering methods

(3) Performance evaluation models for parallel multigrid procedures

(1) includes stabilization method for AMG solvers by specifying error component to be transferred to the coarse levels, and parallel time integration (PinT) algorithms. In (2), we study reordering technique on unstructured problem domain. (3) deals with performance evaluation models for multigrid solvers on supercomputers.

This year is the second year of a three-year plan. The goal is to increase the number of research papers and, at the same time, to concretize the development details for the release of the code.

3 Significance as JHPCN Joint Research Project

Multigrid method is scalable and used in many fields. It is known as one of the most efficient linear solvers. It can also be applied to parallel time integration problems, which exploits parallelism in time dimension. Our research project has original codes and algorithms. Therefore, research papers and codes from the project will enhance the efficiency of the multigrid solver, and will help many researchers exploit parallelism in time direction. Our research focuses on hierarchical algorithms and their performance on supercomputers. Thus, availability of supercomputers with different kinds of architectures helps us verify the codes we are developing. In addition, a JHPCN joint research project offers collaborative research opportunity with JHPCN members who have expertise knowledge in various application fields. Our project members include international experts in Germany, U.S., France and Japan on multigrid methods and PinT. We are sure that this JHPCN joint research project promotes the international collaborative activity with JHPCN members.

4 Outline of Research Achievements until FY2023 (Only for continuous projects)

We conducted our research project "Innovative Multigrid II" from FY2020 to FY2022. While the project yielded substantial progress, several key research challenges remained unresolved. To address these issues more systematically, we restructured the research themes into three focused areas as described in Section 2, and launched a new three-year initiative, "Innovative Multigrid III" starting in FY2023.

We were able to publish a number of research papers on topics such as MG, AMG, and PinT solvers. On the other hand, progress on performance model was limited and remains to be further addressed.

5 Details of FY2024 Research Achievements

This section highlights three research projects presented during the FY2024:

- Numerical Analysis of a Parallel-in-Time Method for Oseen Problems
- Fundamental Studies on Efficient Simulation of the Overdamped Langevin Equation

- Impact of Thread Count on the Coarse Grid Solver in MGCG Methods

Numerical Analysis of a Parallel-in-Time Method for Oseen Problems

Constructing stable coarse temporal levels in parallel-in-time (PinT) methods remains challenging, particularly for hyperbolic problems. However, a coarse-grid optimization method based on the spectral difference minimization has already yielded near-optimal coarse-grid operators and achieved good convergence for scalar advection problems. We expand this to coupled system equations: time-depenent Oseen problems, which is a linearized form of Navier-Stokes equations. The formulation assumes Runge-Kutta time integration within a projection framework and periodic boundaries; nevertheless, it extends naturally to Dirichlet boundaries, where it retains good convergence.

R. Yoda et. al. reports spectral analysis for convergence rates and parallel performance of proposed method [1]. Fig. 1 and 2 presents the estimated convergence rates $\|E_k\|$ obtained from the convergence analysis of MGRIT for coupled system equations. While the convergence rate of the conventional rediscretization method was 0.743, our proposed method achieved an good convergence rate of 0.098. Fig. 3 shows the results of the parallel test. We have demonstrated good strong scaling performance, achieving faster speeds than sequential time stepping with 64 processors. These results indicate that our method is highly effective for practical large-scale engineering simulations.

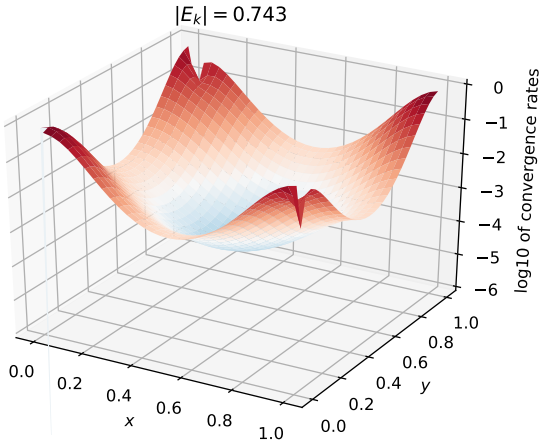

Fig. 1 Convergence analysis (conventional)

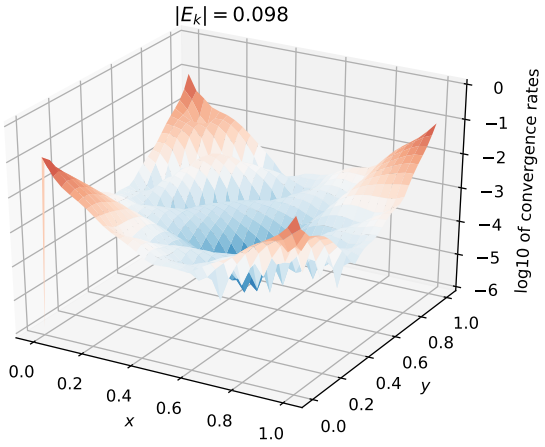

Fig. 2 Convergence analysis (proposed)

Fundamental Studies on Efficient Simulation of the Overdamped Langevin Equation

A.Fujii et.al. investigates the method for expanding the timestep width for overdamped Langevin equation.

The overdamped Langevin equation is widely used in molecular dynamics simulations, particularly for biomolecules such as proteins. Typically, an explicit integration

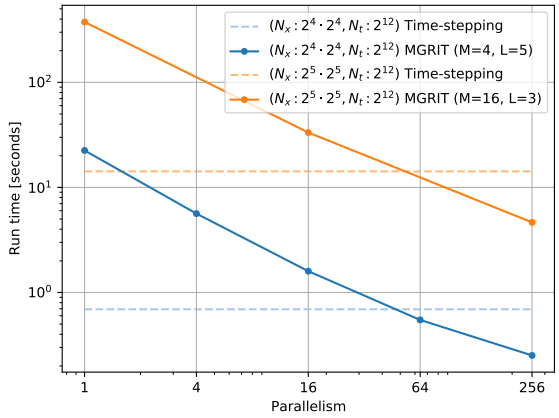

Fig. 3 Strong scaling experiments

scheme is employed with a time step of approximately 10^{-9} seconds, resulting in high computational costs. Therefore, accelerating these simulations remains a critical challenge.

In this study, we focus on a semi-implicit scheme with pseudo-random forces[2], which enables larger time step sizes for the overdamped Langevin equation, and we analyze its numerical characteristics. From the perspective of time parallelization, establishing a method that allows stable and accurate simulation with larger time steps is essential, and our evaluation provides a foundation for such approaches.

The method under consideration corrects the stochastic forces from the solvent by adding pseudo-random forces, allowing time evolution with larger steps. This approach corresponds to applying Newton' s method once per time step and requires solving a linear system of equations. To compute the pseudo-random forces, a Cholesky decomposition of the coefficient matrix is used.

We first examined how changing the ordering of particles affects the Cholesky decomposition and the generated pseudo-random forces. Our analysis showed that the accuracy of the solution is not affected by the ordering, confirming its robustness.

Furthermore, since the cost of Cholesky decomposition increases significantly with matrix size, we explored the feasibility of using incomplete Cholesky decompositions. In particular, we analyzed which eigenvalue components are essential for computing pseudo-random forces. The results indicate that for simulations with larger time steps, accurately capturing the low eigenvalue components is sufficient.

Future work will focus on developing cost-effective incomplete Cholesky decomposition techniques that can faithfully reproduce the influence of low eigenvalue components, thereby improving the efficiency of large time step simulations.

Impact of Thread Count on the Coarse Grid Solver in MGCG Methods

In the previous works by authors using Wisteria/BDEC-01 (Odyssey) [3], the same number of threads (=12) was applied at each level of the parallel multigrid method, including the coarse grid solver. However, this study also examined scenarios where the number of threads varied by level. In the multigrid method, the levels are defined such that the finest grid is at level-1, and the level number increases as the grid becomes coarser. When dealing with large-scale problem sizes per node, the benefits of parallelization using OpenMP and MPI are significant. However, for smaller-scale ones, parallelization can sometimes lead to

a decrease in computational performance. In such cases, strategies to maintain computational efficiency include reducing the number of threads, or even disabling parallelization for OpenMP [4]. In the present work, we evaluate the effects of thread number for the smoother in parallel multigrid procedure, and the coarse grid solver, as follows:

T-1: In the smoother for parallel multigrid procedure, the number of threads dedicated to computation was set to 1 or half of the original thread number at some levels.

T-2: The number of threads for the coarse grid solver was set to 1 at some levels.

We considered the best cases for manual-scheduling (xy-man-a, and xy-man-b) for these evaluations for T-1 and T-2. Figure 4 and Table 1 show configurations of each case in T-1. Each case is defined as xy-z-k, where k is the number of idle cores in each case. Because we consider only manual scheduling (man-a, man-b) as the scheduling policy in T-1 and T-2, one core is dedicated to halo communication including copy. Therefore, thread number of computation for SpMV and forward/backward substitutions (only for man-b) is 2 for x03-z, 5 for x06-z, and 11 for x12-z, respectively. In each Process/Thread Allocation (HB 3 × 16, HB 6 × 8, and HB 12 × 4), 2 types of configurations are considered in the smoother of the parallel multigrid procedures at each level. The first type is that half the cores are available for SpMV, and forward/backward substitutions compared to the original configuration, and the other is that only a single core is available for such computations. x06-z-2 and x12-z-5 are in the 1st type, while x06-z-4 and x12-z-10 are in the 2nd type. x03-z-1 can be considered as both types. Allocations of threads at each level are defined as shown in Table 2. lev-j+ means that, xy-z with original number of threads is applied to level-(j-1) or lower (finer), while xy-z-k is applied to level-j or higher (coarser). If we apply xy-z-k, same number of k is applied to each level. In T-1, 128 nodes of Odyssey have been applied.

Fig. 4 configurations of Idle Cores for T-1

Table 1 Cases in T-1 (xy-z-k)

	(x)	(y)	Scheduling Policy(z)	Idle Core # (k)	Comp. Core #
x03-z	Large (l) Medium (m) Small (s)	HB 3×16	man-a *or* man-b	0	2
x03-z-1				1	1
x06-z		HB 6×8		0	5
x06-z-2				2	3
x06-z-4				4	1
x12-z		HB 12×4		0	11
x12-z-5				5	6
x12-z-10				10	1

(x): Problem Size, (y): Process/Thread Allocation

Table 3 shows configurations of each case

Table 2 Thread Allocation at Each Level of Parallel Multigrid Procedures in T-1

	level-1	level-2	level-3	level-4	level-5 or coarser
lev-1+	*xy-z-k*	*xy-z-k*	*xy-z-k*	*xy-z-k*	*xy-z-k*
lev-2+	*xy-z*	*xy-z-k*	*xy-z-k*	*xy-z-k*	*xy-z-k*
lev-3+	*xy-z*	*xy-z*	*xy-z-k*	*xy-z-k*	*xy-z-k*
lev-4+	*xy-z*	*xy-z*	*xy-z*	*xy-z-k*	*xy-z-k*
lev-5+	*xy-z*	*xy-z*	*xy-z*	*xy-z*	*xy-z-k*

in T-2. In T-2, 128 nodes and 512 nodes are applied, and only large (l) and small (s) cases are considered. Allocations of threads at each level of coarse grid solver are defined as shown in Table 4 Each level of the coarse grid solver is defined as levelC. levC-j+ in Table 4 means that, multi-threaded parallel coarse grid solver with original number of threads is applied to levelC-(j-1) or lower (finer), while single thread solver is applied to levelC-j or higher (coarser).

Table 3 Cases in T-2(xy-z-p)

	(x)	(y)	Scheduling Policy(z)	Node # (p)
x03-z-p	Large (l) Small (s)	HB 3×16	man-a *or* man-b	128 *or* 512
x06-z-p		HB 6×8		
x12-z-p		HB 12×4		

(x): Problem Size, (y): Process/Thread Allocation

Table 4 Number of Threads for Coarse Grid Solver at Each Level (LevelC) in T-2

	levelC					
	1	2	3	4	5	6 or higher
levC-1+	1	1	1	1	1	1
levC-2+	3/6/12	1	1	1	1	1
levC-3+	3/6/12	3/6/12	1	1	1	1
levC-4+	3/6/12	3/6/12	3/6/12	1	1	1
levC-5+	3/6/12	3/6/12	3/6/12	3/6/12	1	1
levC-6+	3/6/12	3/6/12	3/6/12	3/6/12	3/6/12	1

Each of Fig. 5 (a,b,c) shows effects of thread counts for smoother in MGCG at each level (T-1). Elapsed computation time for MGCG solver is normalized by that of the cases without idle cores, such as x03-z, x06-z, and x12-z in Table 1. The impact of varying the number of threads is generally minimal. Except for the finest grid level, changes in the number of threads resulted in only slight variations in computation time. For smaller-scale problems (sy-man-a, sy-man-b), reducing the number of threads at finer grid levels does not significantly affect computation time.

Figure 6 shows effects of thread number for coarse grid solver at each level (T-2). Elapsed computation time for MGCG solver is normalized by that for the original MGCG solvers with fully multi-threaded parallel coarse grid solver at all levels, such as xy-man-a, and xy-man-b. The impact of varying the number of threads is generally minimal. For problem sizes for coarse grid solver below 104, setting the number of threads to one across all levels did not result in significant differences in computation time, as shown in Fig. 6 and Fig. 7. For problem sizes exceeding 104, thread parallelism was necessary at the finest grid level (levelC=1), but for coarser levels, setting the number of threads to one had no impact on computation time. The problem size for the coarse grid solver is determined by the total number of MPI processes, but it can also increase significantly depending on the level at which the switch to the coarse grid solver occurs in the CGA.

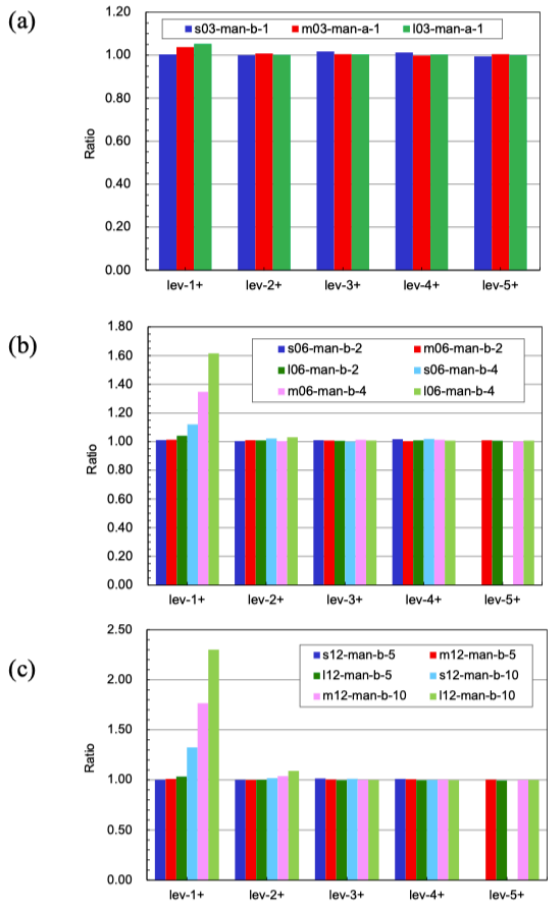

Fig. 5 Effects of Thread Counts for Smoothers in Parallel Multigrid Procedures (T-1): Elapsed Computation Time for MGCG (a) HB 3 × 16 (normalized by elapsed time for MGCG in s03-man-b, m03-man-a, l03-man-a), (b) HB 6 × 8 (s06-man-b, m06-man-b, l06-man-b), (c) HB 12 × 4 (s12-man-b, m12-man-b, l12-man-b)

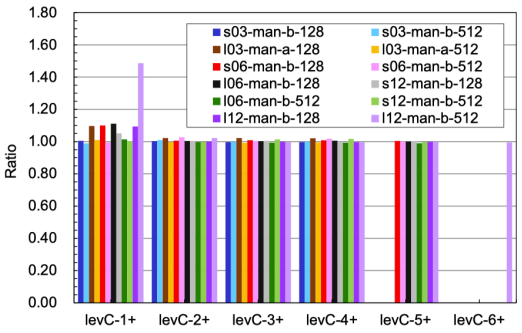

Fig. 6 Effects of Thread Counts for Coarse Grid Solver in Parallel Multigrid Procedures (T-2): Normalized Elapsed Computation Time for MGCG

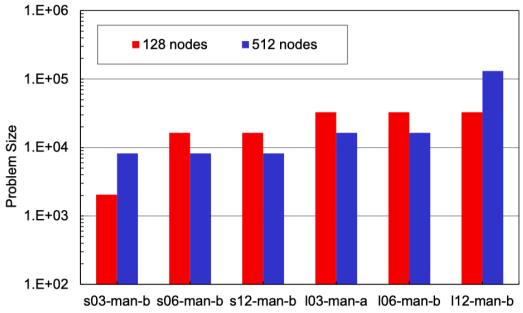

Fig. 7 Problem Size for Coarse Grid Solver

References

[1] Ryo Yoda, Matthias Bolten, Kengo Nakajima, Akihiro Fujii, 2024, Coarse-grid operator optimization in multigrid reduction in time for time-dependent Stokes and Oseen problems, Japan Journal of Industrial and Applied Mathematics, 41 (3), 1315-1339

[2] Takumi Washio, Akihiro Fujii, Toshiaki Hisada. On random force correction for large time steps in semi-implicitly discretized overdamped Langevin equations[J]. AIMS Mathematics, 2024, 9(8): 20793-20810. doi: 10.3934/math.20241011

[3] Nakajima, K., Communication-computation Overlapping for Parallel Multigrid Methods, 2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 751-760, 2024, DOI

10.1109/IPDPSW63119.2024.00139

[4] Nakajima, K., New strategy for coarse grid solvers in parallel multigrid methods using openMP/MPI hybrid programming models, ACM Proceedings of he 2012 International Workshop on Programming Models & pplications for Multi/Manycores, 2012

6 Self-review of Current Progress and Future Prospects

While significant progress was made on PinT and performance analysis, advancements in performance modeling were limited. With some adjustments to our research direction, we plan to move forward this fiscal year with a renewed focus, also taking GPU environments into consideration.