Math-for-industry
Education & Research Hub

ISSN 1881-4042

Joint Research Workshop of Institute of Mathematics for Industry (IMI), Kyushu University

"Propagation of Ultra-large-scale Computation by the Domain-decomposition-method for Industrial Problems (PUCDIP 2012)"

Editor : Hiroshi Kanayama (Faculty of Engineering, Kyushu University)

九州大学グローバルCOEプログラム

COE Lecture Note Vol.45 : Kyushu University

Joint Research Workshop of Institute of Mathematics for Industry (IMI),

Kyushu University

“Propagation of Ultra-large-scale Computation by the Domain-decomposition-method for Industrial Problems (PUCDIP 2012)”

Editor: Hiroshi Kanayama

(Faculty of Engineering, Kyushu University)

About MI Lecture Note Series

The Math-for-Industry (MI) Lecture Note Series is a successor to the COE Lecture Notes, published for the 21st COE Program "Development of Dynamic Mathematics with High Functionality", sponsored by Ministry of Education, Culture, Sports, Science and technology-Japan (MEXT) (From 2003 to 2007).

The MI series reports lectures given by scholars invited under the following two programs: "Training Program of Ph.D. and new Master's in Mathematics as Required by Industry", adopted as a Support Program for Improving Gradute School Education by MEXT (from 2007 to 2009); and Education-and-Research Hub for Mathematics-for-Industry", newly adopted as a Global COE Program by MEXT (from 2008 to 2012).

July 2008
Masato Wakayama
Global COE Program "Education-and-Research Hub for Mathematics-for-Industry"
Program Leader

Joint Research Workshop of Institute of Mathematics for Industry (IMI), Kyushu University

"Propagation of Ultra-large-scale Computation by the Domain-decomposition-method for Industrial Problems (PUCDIP 2012)"

COE Lecture Note Vol.45, Institute of Mathematics for Industry, Kyushu University
ISSN 1881-4042
Editor: Hiroshi Kanayama (Faculty of Engineering, Kyushu University)
Date of issue: 19 February 2013
Publisher:
Institute of Mathematics for Industry, Kyushu University
Faculty of Mathematics, Kyushu University
Global COE Program "Education-and-Research Hub for Mathematics-for-Industry"
Motooka 744, Nishi-ku, Fukuoka, 819-0395, JAPAN
Tel +81-(0)92-802-4404, Fax +81-(0)92-802-4405
URL http://gcoe-mi.jp/

Printed by
Kijima Printing, Inc.
Shirogane 2-9-6, Chuo-ku, Fukuoka, 810-0012, Japan
TEL +81-(0)92-531-7102 FAX +81-(0)92-524-4411

Contents

Preface

Joint Research Workshop of Institute of Mathematics for Industry (IMI), Kyushu University
"Propagation of Ultra-large-scale Computation by the
Domain-decomposition-method for Industrial Problems (PUCDIP 2012)"
was held at Nishijin Plaza, Kyushu University, Fukuoka, from 12th through 13th , October 2012. This activity was also organized as one of Collaborative Workshops of Mathematics and Mathematical Sciences with Various Sciences and Industrial Technologies supported by the Ministry of Education, Culture, Sports, Science & Technology in Japan (MEXT). This volume collects the papers of all the talks presented at the workshop.

In this research meeting, the whole view of the ADVanced ENgineering analysis Tool for Ultra large REal world (ADVENTURE) system and recent development in related technologies were shown. Also a kickoff meeting for international development of the system, especially technology transfer to Asia and Africa regions was held.
Recently, the K computer in Kobe has gathered much attention in computational world. Making use of the ADVENTURE system in the K computer is a hot topic. State of arts was shown in the meeting by related researches and users. Also researchers in foreign countries attended the meeting, because it was held after the JSME-CMD International Computational Mechanics Symposium (ICMS 2012) in Kobe during October 9-11. Since the system is also applied to real world industrial problems, related users from the industrial side also attended this meeting.

The presentations were invited from both fields of mathematics and engineering. This workshop was successful, collecting 30 participants. They sought issues jointly tackled by mathematics and engineering researchers, and also the way for collaboration between computational science and engineering, and mathematics. We are grateful to all the speakers and the participants who made fruitful discussions. The support by the staff of the MEXT and by Ms. Kyoko Sakaguchi of the IMI is also gratefully acknowledged.

Organizer

Hiroshi Kanayama (Faculty of Engineering, Kyushu University)


平成24年度 九州大学
マス・フォア・インダストリ研究所
共同利用研究集会
領域分割法による
超大規模計算の産業界への浸透
文部科学省　平成24年度　数学・数理科学と諸科学・産業との連携研究ワークショップ
2012年
10月12日金▸13日土
場所／九州大学西新プラザ
〒814-0002 福岡県福岡市早良区西新2丁目
主催：九州大学マス・フォア・インダストリ研究所
共催：文部科学省
運営責任者 金山　寛（九州大学大学院工学研究院）
in FUKUOKA
お問い合わせ先　九州大学マス・フォア・インダストリ研究所　TEL: 092-802-4402　E-mail: kyodo_riyou@imi.kyushu-u.ac.jp

PUCDIP 2012

Propagation of Ultra-large-scale Computation by the Domain-decomposition-method for Industrial Problems

Organized by Hiroshi Kanayama

October 12-13, 2012
Nishijin Plaza, Kyushu University

Friday, October 12, 2012

Chair : Hiroshi Kanayama

13:45-13:55 Opening Ceremony
1. Message from MEXT
2. Message from IMI
3. Message from Conference Secretary

14:00-15:30 Masahisa Tabata (Waseda University)
Some Remarks on Domain Decomposition Methods
Frederic Magoules (Ecole Centrale Paris)
Optimized Schwarz Domain Decomposition Methods

Chair : Masahisa Tabata

15:45-17:15 Daisuke Tagami (Kyushu University)
An Iterative Domain Decomposition Method with Mixed Formulations
Hiroshi Kanayama (Kyushu University)
Large Scale Magnetic Field Analysis for Induction Motors

18:00-20:00 Party at Baramon (波羅門)
in Hilton Fukuoka Seahawk (6th Floor)

Saturday, October 13, 2012

Chair : Ryuji Shioya

9:00-10:30 Mitsuteru Asai (Kyushu University)

Large Scale Tsunami Run-up Simulation by a Hybrid-parallel SPH

Akio Miyoshi (Insight, Inc)

Promotion of Open-Source ADVENTURE by Insight

Chair : Daisuke Tagami

10:45-12:15 Ryuji Shioya (Toyo University)

Development of a Numerical Library based on Hierarchical Domain Decomposition for Post Petascale Simulation

Hiroshi Kawai (Tokyo University of Science, Suwa)

Performance Evaluation of ADVENTURE on K Computer

Masao Ogino (Nagoya University)

A Hybrid CPU-GPU Implementation of Finite Element Methods based on the Domain Decomposition Method

A. M. M. Mukaddes (Toyo University)

ADVENTURE_Thermal- an Open Source Module for Heat Conduction Problems

Behavior of Finite Element Solutions Subject to Strong-Type Robin Interface Conditions

M. Tabata

Waseda University, 3-4-1, Ohkubo, Shinjuku, Tokyo, 169-8555 JAPAN
tabata@waseda.ac.jp

1 INTRODUCTION

Domain decomposition method is a powerful tool for solving large-scale scientific problems. The domain where the problem is considered is partitioned into several subdomains, where small problems are solved in parallel. Combined with the finite element method, domain decomposition method has been successfully developed to various directions, see, for examaple, [5, 4]. Among a lot of domain decomposition methods we focus our attention on an iterative substructuring method subject to Robin conditions at the interfaces, which was originally introduced by P. L. Lions [2]. Although the Robin boundary conditions are usually approximated by weak type formula, we consider the strong type approximation of the normal derivative and observe the behavior of finite element solutions.

2 AN ITERATIVE SUBSTRUCTURING METHOD

Let Ω be a bounded domain in $\Re^d$, $d = 1, 2, 3$, with Lipschitz boundary $\partial\Omega$. We consider the Poisson problem: Find $u : \Omega \to \Re$ such that

$$-\Delta u = f \quad \text{in } \Omega, \tag{1a}$$

$$u = g \quad \text{on } \partial\Omega, \tag{1b}$$

where

$$f : \Omega \to \Re, \quad g : \partial\Omega \to \Re$$

are given functions. The variational formulation of (1) is to find $u \in V(g)$ satisfying

$$a(u, v) = \langle f, v \rangle, \quad \forall v \in V, \tag{2}$$

where

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v \; dx, \quad \langle f, v \rangle = \int_\Omega f v \; dx,$$
$$V(g) = \{v \in H^1(\Omega); \; v = g \text{ on } \partial\Omega\}, \quad V = V(0).$$

Here we have assumed that

$$f \in L^2(\Omega), \quad g \in H^{1/2}(\partial\Omega).$$

The existence and uniqueness of the solution of problem (2) is established; see, for example, [3].

Now we partition Ω into two non-overlapping subdomains Ω_i, $i = 1, 2$, such that

$$\bar{\Omega} = \bar{\Omega}_1 \cup \bar{\Omega}_2, \quad \Omega_1 \cap \Omega_2 = \emptyset.$$

We denote the interface of the two domains by

$$\Gamma = \partial\Omega_1 \cap \partial\Omega_2$$

and two parts of $\partial\Omega$ by

$$\Gamma_i = \partial\Omega \cap \partial\Omega_i, \quad i = 1, 2.$$

Let n be the unit outward normal to Γ. We denote by n_{12} the unit normal to Γ from Ω_1 and by n_{21} the normal from Ω_2. An iterative substructuring method based on Robin interface conditions corresponding to (1) is described as follows [2]: Find $\{(u_1^{(k)}, u_2^{(k)})\}$, $k = 1, 2, \cdots$, such that

$$-\Delta u_1^{(k)} = f \qquad \text{in } \Omega_1, \tag{3a}$$
$$u_1^{(k)} = g \qquad \text{on } \Gamma_1, \tag{3b}$$
$$\frac{\partial u_1^{(k)}}{\partial n_{12}} + \alpha u_1^{(k)} = \frac{\partial u_2^{(k-1)}}{\partial n_{12}} + \alpha u_2^{(k-1)} \qquad \text{on } \Gamma \tag{3c}$$

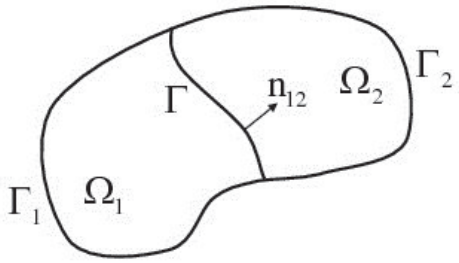


Figure 1: Subdomains Ω_1 and Ω_2, and the interface Γ.

and

$$-\Delta u_2^{(k)} = f \qquad \text{in } \Omega_2, \tag{4a}$$

$$u_2^{(k)} = g \qquad \text{on } \Gamma_2, \tag{4b}$$

$$\frac{\partial u_2^{(k)}}{\partial n_{21}} + \alpha u_2^{(k)} = \frac{\partial u_1^{(k-1)}}{\partial n_{21}} + \alpha u_1^{(k-1)} \qquad \text{on } \Gamma, \tag{4c}$$

where the initial functions, $u_1^{(0)}$ and $u_2^{(0)}$, are given and α is a positive constant. It is proved that for any $u_i^{(0)} \in H^1(\Omega_i)$, $i = 1, 2$, and $\alpha > 0$,

$$\lim_{k\to\infty} \|u_i^{(k)} - u_i\|_{H^1(\Omega_i)} = 0, \quad i = 1, 2,$$

where u_i is the restriction of u to Ω_i [2, 5].

3 A FINITE ELEMENT APPROXIMATION

We discretize the problems (3) and (4) by the finite element method. We consider the case where the domains Ω_i, $i = 1, 2$, are polygonal. Let $\mathcal{T}_{ih}$ be the finite element mesh of Ω_i. Let X_{ih} be a finite element space such that

$$X_{ih} \subset H^1(\Omega_i), \quad i = 1, 2.$$

Imposing the Dirichlet boundary conditions (1b), we define an affine finite element space

$$V_{ih}(g) \equiv \{v_h \in X_{ih};\ v_h(P_j) = v(P_j),\ \forall \text{node } P_j \in\ \Gamma_i\}, \quad i = 1, 2.$$

We denote $V_{ih}(0)$ by V_{ih}, which is a subspace of V_i, i.e.,

$$V_{ih} \subset V_i \equiv \{v \in H^1(\Omega_i);\ v = 0 \text{ on } \Gamma_i\}.$$

Our finite element approximaion of the problems (3) and (4) are as follows: Find $\{(u_{1h}^{(k)}, u_{2h}^{(k)}) \in V_{1h}(g) \times V_{2h}(g);\ k = 1, 2, \cdots\}$ such that

$$\begin{aligned}\int_{\Omega_1} \nabla u_{h1}^{(k)} \cdot \nabla v_h \, dx + \alpha \int_\Gamma u_{h1}^{(k)} \, v_h \, ds \\ = \int_{\Omega_1} f \, v_h \, dx + \int_\Gamma \left\{ \frac{\partial u_{h2}^{(k-1)}}{\partial n_{12}} + \alpha \, u_{h2}^{(k-1)} \right\} v_h \, ds, \quad \forall v_h \in V_{1h}\end{aligned} \tag{5}$$

and

$$\begin{aligned}\int_{\Omega_2} \nabla u_{h2}^{(k)} \cdot \nabla v_h \, dx + \alpha \int_\Gamma u_{h2}^{(k)} \, v_h \, ds \\ = \int_{\Omega_2} f \, v_h \, dx + \int_\Gamma \left\{ \frac{\partial u_{h1}^{(k-1)}}{\partial n_{21}} + \alpha \, u_{h1}^{(k-1)} \right\} v_h \, ds, \quad \forall v_h \in V_{2h},\end{aligned} \tag{6}$$

where initial functions $(u_{1h}^{(0)}, u_{2h}^{(0)})$ are given in $V_{1h}(g) \times V_{2h}(g)$. (5) and (6) are direct approximations to (3) and (4) subject to the Robin boundary conditions on Γ,

$$\frac{\partial u_i^{(k)}}{\partial n} + \alpha \, u_i^{(k)} = \beta_i^{(k)}, \quad i = 1, 2, \tag{7}$$

where $\beta_i^{(k)}$ are defined by

$$\beta_1^{(k)} = \frac{\partial u_2^{(k-1)}}{\partial n_{12}} + \alpha u_2^{(k-1)}, \quad \beta_2^{(k)} = \frac{\partial u_1^{(k-1)}}{\partial n_{21}} + \alpha u_1^{(k-1)}. \tag{8}$$

In (5) and (6) the normal derivatives in (8) are computed directly. We therefore call (5) and (6) the problems subject to strong-type Robin interface conditions.

4 BEHABIOR OF FINITE ELEMENT SOLUTIONS

Let Ω be the square in $\Re^2$,

$$\Omega = (-1, 1) \times (-1, 1).$$

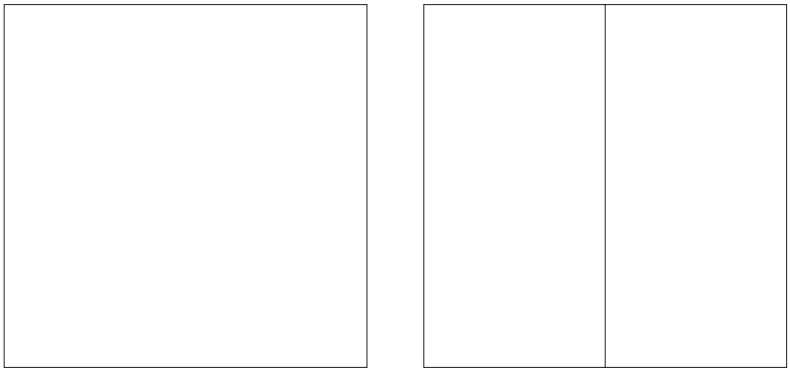


Figure 2: Domain Ω and subdomains Ω_1 and Ω_2

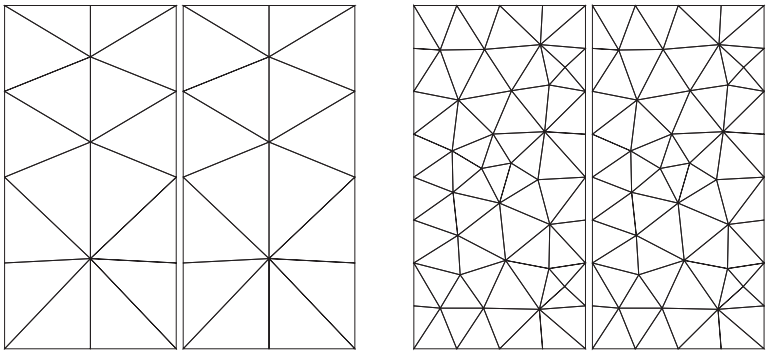


Figure 3: Meshes $\mathcal{T}_{1h}$ and $\mathcal{T}_{2h}$, $N = 4$(left), and $N = 8$(right).

The functions f and g in (1) are given so that the the exact solution of u is equal to

$$u(x_1, x_2) = \left(x_2 + \frac{1}{4}\right) \sin\{\pi(x_1^2 - x_1 + x_2)\}.$$

The domain Ω is partitioned into two domains,

$$\Omega_1 = (-1, 0) \times (-1, 1), \quad \Omega_2 = (0, 1) \times (-1, 1),$$

which are shown in Figure 2 (right). Let N be the parameter of mesh sub-

divisions of the domain Ω_i. The representative mesh size h is defined by

$$h = \frac{2}{N}.$$

N runs over $4, 8, \cdots, 128$ and we get finer meshes $(\mathcal{T}_{1h}, \mathcal{T}_{2h})$. Meshes for $N = 4$ and 8 are shown in Figure 3. Let X_{ih}, $i = 1, 2$, be P1-finite element spaces on mesh $\mathcal{T}_{ih}$. Let α be a given positive number. We solve the problems (5) and (6) subject to the initial functions,

$$u_{1h}^{(0)} = 0, \quad u_{2h}^{(0)} = 0$$

to get the sequence of

$$\left\{ \left(u_{1h}^{(k)}(\alpha), u_{2h}^{(k)}(\alpha) \right) \right\}, \quad k = 1, 2, \cdots.$$

When the sequence $\{u_{ih}^{(k)}\}$, $k = 1, 2, \cdots$, converges to a function, we denote it by u_{ih}, i.e.,

$$\lim_{k \to \infty} u_{ih}^{(k)}(\alpha) = u_{ih}(\alpha), \quad i = 1, 2.$$

We use the following notations,

$$\mathrm{Dif}(k, \alpha, N, X) = \frac{\|u_h^{(k)}(\alpha) - u_h^{(k-1)}(\alpha)\|_X}{\|u_h^{(k)}(\alpha)\|_X},$$

$$\mathrm{Er}(k, \alpha, N, X) = \frac{\|u_h^{(k)}(\alpha) - \Pi_h u\|_X}{\|\Pi_h u\|_X},$$

where X stands for $(L^\infty(\Omega_1), L^\infty(\Omega_2))$, $(L^2(\Omega_1), L^2(\Omega_2))$, or $(H_0^1(\Omega_1), H_0^1(\Omega_2))$, Π_h is the interpolation operator from $C(\bar{\Omega}_i)$ to X_{ih} defined by

$$(\Pi_h u)(P_j) = u(P_j), \quad \forall P_j \in \bar{\Omega}_i.$$

The iteration in k has been stopped when the criterion

$$\mathrm{Dif}(k, \alpha, N, L^\infty) < \epsilon \ (\equiv 10^{-6})$$

is satisfied.

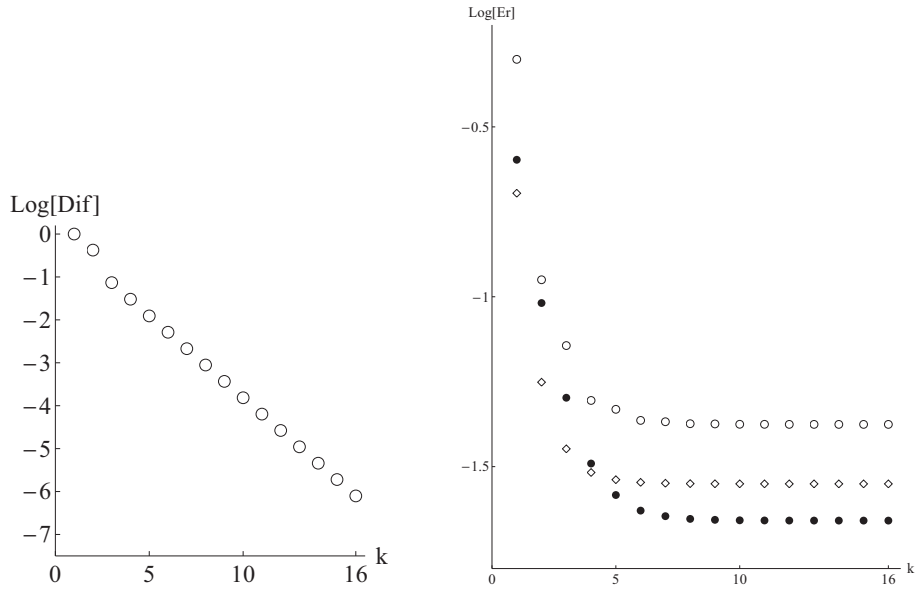


Figure 4: Graphs of $\mathrm{Dif}(k, 4, 32, L^\infty)$(left) and of $\mathrm{Er}(k, 4, 32, X)$(right)vs. k, $\circ : L^\infty$, $\bullet : L^2$, $\diamond : H_0^1$.

4.1 Convergence of the iteration

We select

$$\alpha = 4, \quad N = 32.$$

and observe the convergence of $(u_{1h}^{(k)}, u_{2h}^{(k)})$, $k = 1, 2, \cdots$. Figure 4 (left) shows the convergence history of $\mathrm{Dif}(k, 4, 32, L^\infty)$. $(u_{1h}^{(k)}, u_{2h}^{(k)})$ has converged at $k = 16$. Figure 4 (right) shows the graph of $\mathrm{Er}(k, 4, 32, X)$, $X = L^\infty$, L^2, and H_0^1. The errors decrease monotonically. Figure 5 shows the combined figures of $u_{1h}^{(k)}$ and $u_{2h}^{(k)}$ for $k = 1, 2, 3, 4, 16$ and the figure of $u_h \in V_h(g)$, the finite element solution solved in the whole domain Ω,

$$a(u_h, v_h) = \langle f, v_h \rangle, \quad \forall v_h \in V_h, \tag{9}$$

where V_h is the P1-finite element space on the mesh $\mathcal{T}_h = \mathcal{T}_{1h} \cup \mathcal{T}_{2h}$. In the figure at $k = 1$ there is a clear gap on the interface Γ. When k increases, the values and the derivatives of $u_{1h}^{(k)}$ and $u_{2h}^{(k)}$ become almost equal on the interface. The figure of converged solutions at $k = 16$ looks almost same as the one of u_h, the finite element solution solved in the whole domain.

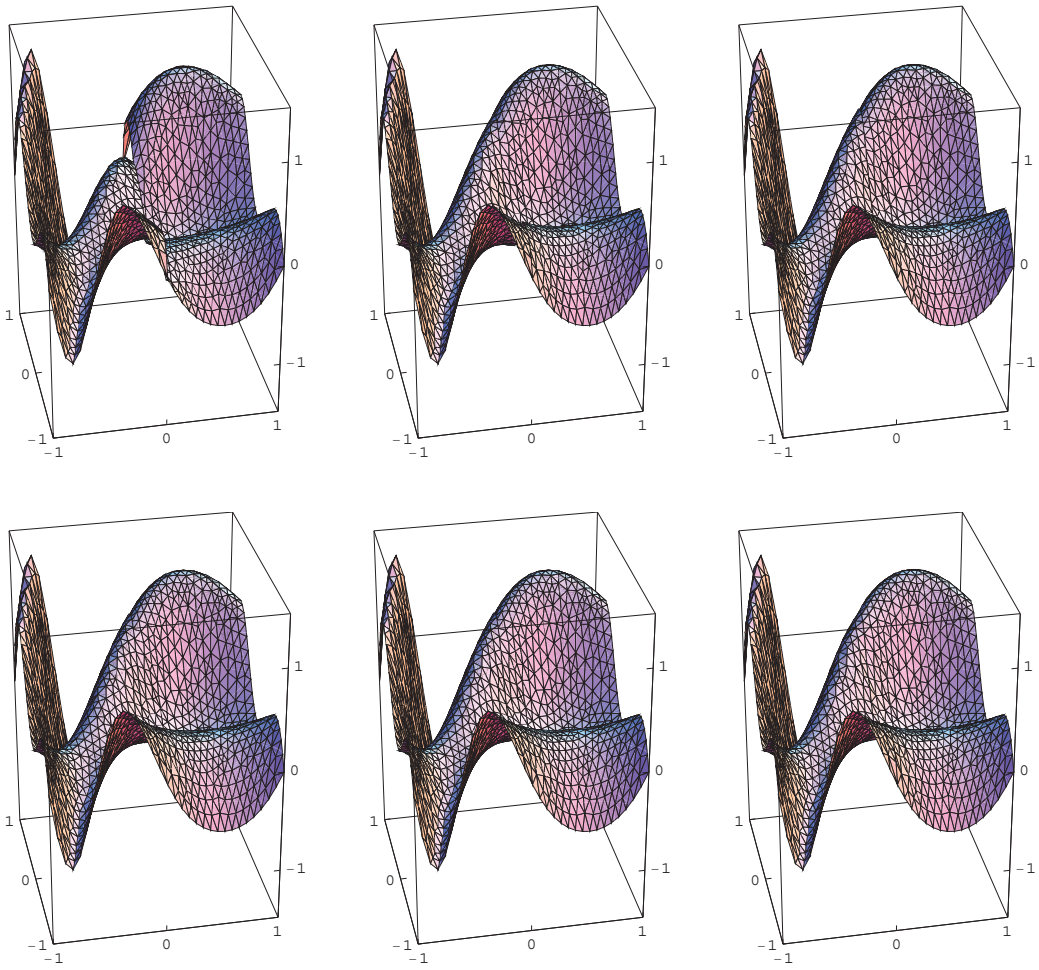


Figure 5: $(u_{1h}^{(k)}, u_{2h}^{(k)})$, $k = 1, 2, 3$(top), =4,16, and u_h(bottom), α=4

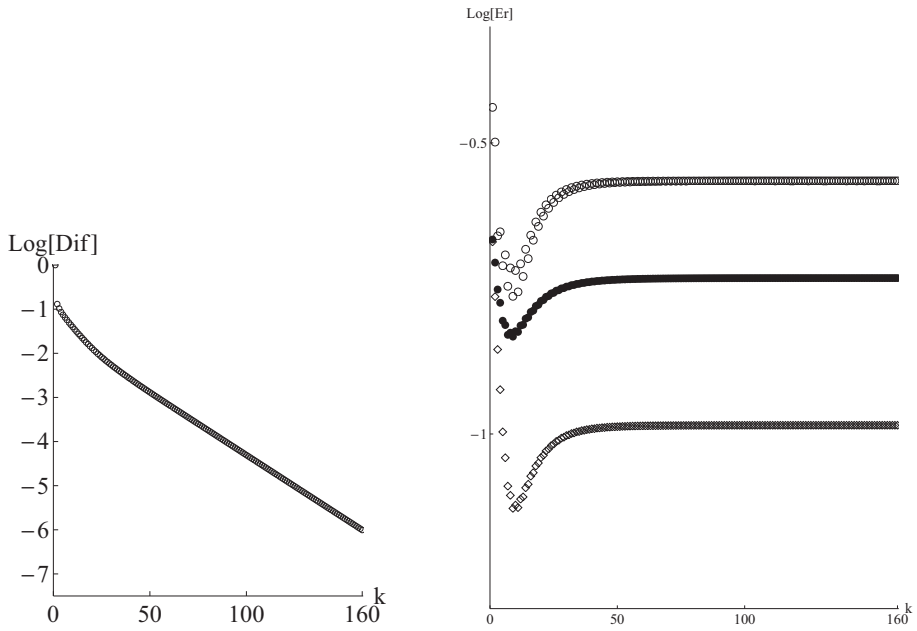


Figure 6: Graphs of $\mathrm{Dif}(k, 0.01, 32, L^\infty)$(left) and of $\mathrm{Er}(k, 0.01, 32, X)$(right) vs. k, $\circ : L^\infty$, $\bullet : L^2$, $\diamond : H_0^1$.

4.2 Dependence on α

Fixing the value of N, we select a small α,

$$\alpha = 0.01, \quad N = 32.$$

Figure 6 (left) shows the convergence history of $\mathrm{Dif}(k, 0.01, 32, L^\infty)$. $(u_{1h}^{(k)}, u_{2h}^{(k)})$ has converged at $k = 160$. Figure 7 shows the combined figures of $u_{1h}^{(k)}$ and $u_{2h}^{(k)}$ for $k = 1, 2, 3, 6, 7, 8, 9, 10, 160$. In the beginning, $k = 1, 2, 3$, the gap on the interface diminishes. At $k = 7$ the gap nearly vanishes, but after that the opposite gap appears and it augments. The converged solution $(u_{1h}^{(k)}, u_{2h}^{(k)})$, $k = 160$, has clear gap. This behabior is recognized in Figure 6 (right), where the graphs of $\mathrm{Er}(k, 0.01, 32, X)$, $X = L^\infty$, L^2, and H_0^1 are not monotone.

Next we select a large α,

$$\alpha = 10.0, \quad N = 32.$$

Figure 8 (left) shows the convergence history of $\mathrm{Dif}(k, 10, 32, L^\infty)$. $(u_{1h}^{(k)}, u_{2h}^{(k)})$ has converged at $k = 37$. Figure 9 shows the combined figures of $u_{1h}^{(k)}$ and

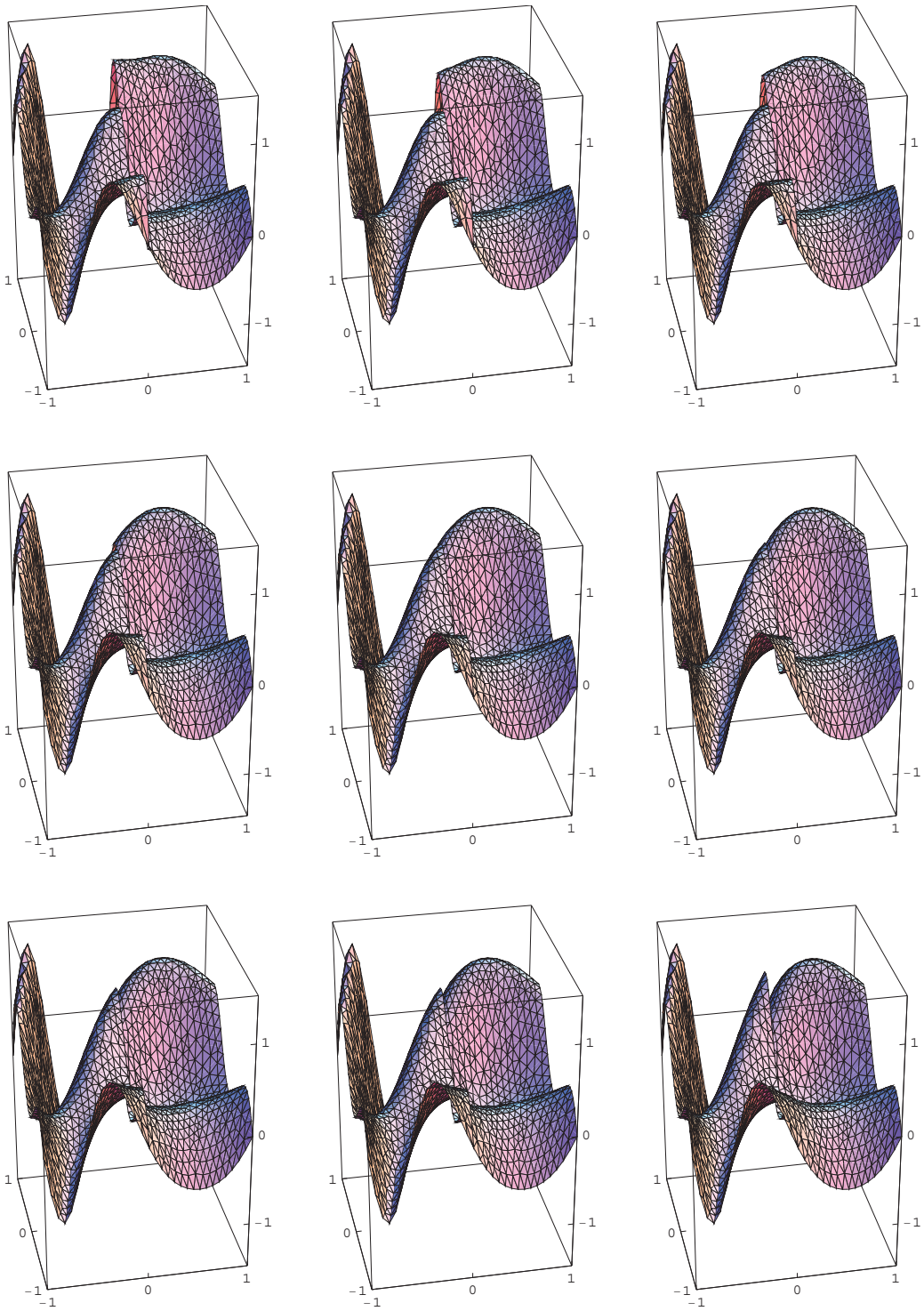


Figure 7: $(u_{1h}^{(k)}, u_{2h}^{(k)})$, $k = 1, 2, 3$(top), =6,7,8(middle), =9,10,160(bottom), α=0.01

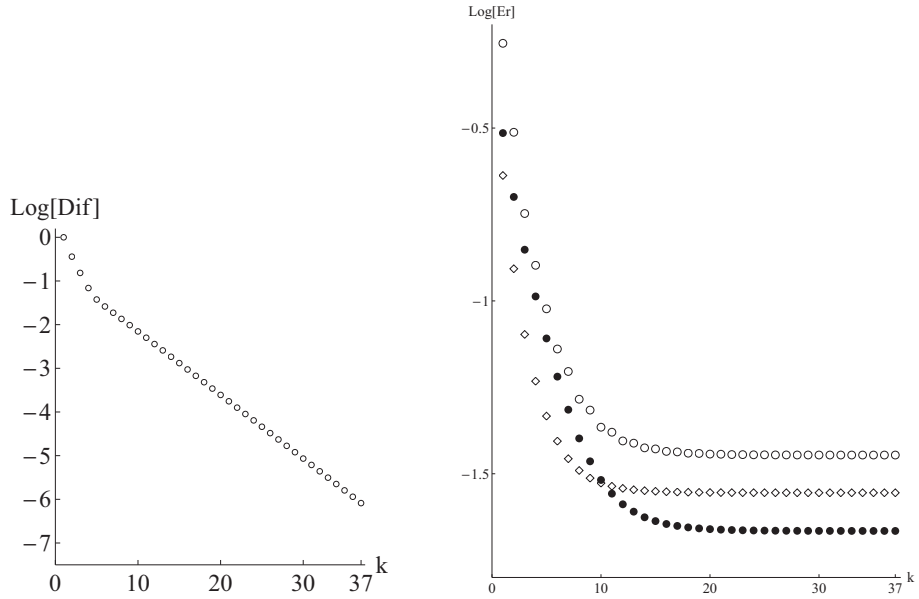


Figure 8: Graphs of $\mathrm{Dif}(k, 10, 32, L^\infty)$(left) and of $\mathrm{Er}(k, 10, 32, X)$(right) vs. k, $\circ : L^\infty$, $\bullet : L^2$, $\diamond : H_0^1$.

$u_{2h}^{(k)}$ for $k = 1, 2, 3, 4, 5, 37$. Initial gap on the interface diminishes, and the converged solution at $k = 37$ looks almost same as u_h, the finite element solution solved in the whole domain in Figure 5. Monotone convergence is also recognized in Figure 8 (right).

We denote by $(u_{1h}(\alpha), u_{2h}(\alpha))$ the converged solution. From the above observation $(u_{1h}(\alpha), u_{2h}(\alpha))$ differs from the solution u_h of (9) when α is small. Figure 10 (left) shows α-dependence of the errors of $(u_{1h}(\alpha), u_{2h}(\alpha)) - u_h$ in the norms of L^∞, L^2, and H_0^1. For α greater than or equal to 4 the errors are almost constant, whose values are considered to be dependent only on $N = 2/h$. When N increases, the mesh size h decreases. Figure 10 (right) shows h-dependence of the errors of $(u_{1h}, u_{2h}) - u_h$ in the norms of L^∞, L^2, and H_0^1 for $N = 4, 8, 16, 32, 64, 128$. The convergence orders in L^2 and H_0^1 are observed in order h, while it is proved that the convergence order in the whole domain using P1 finite element is $O(h^2)$ in L^2-norm and $O(h)$ in H_0^1-norm; see, for example, [1]. Figure 11 shows the converged finite element solutions of (u_{1h}, u_{2h}) and u_h solved in Ω when $N = 64$.

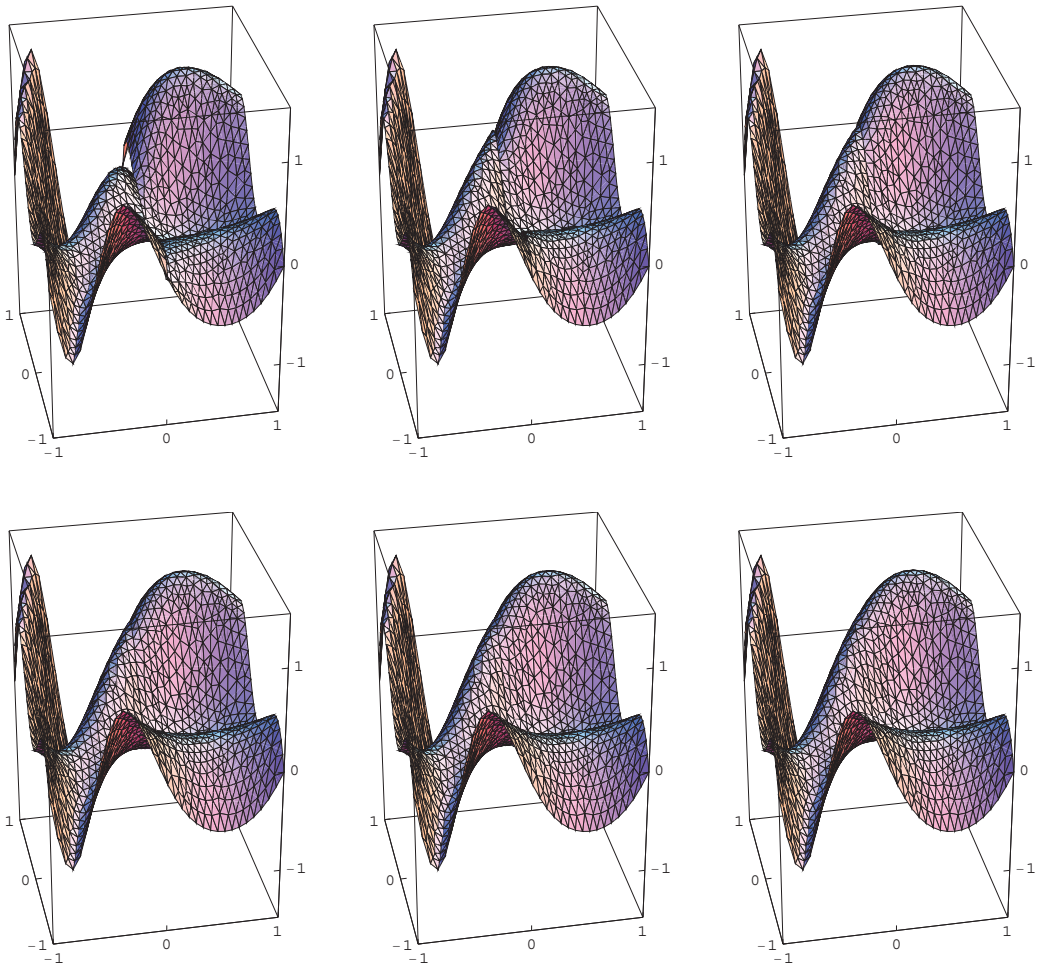


Figure 9: $(u_{1h}^{(k)}, u_{2h}^{(k)})$, $k = 1, 2, 3$(top), =4,5,37(bottom), α=10

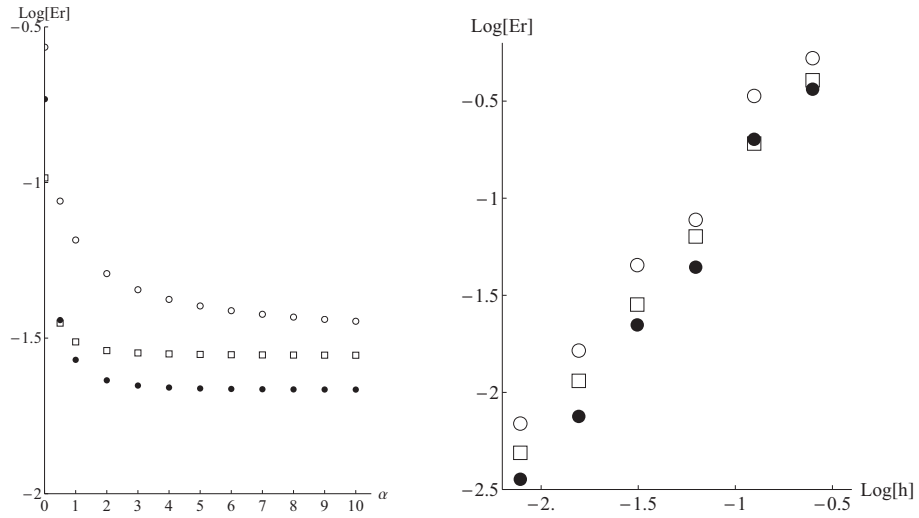


Figure 10: Graphs of $\mathrm{Er}(\cdot, \alpha, 32, X)$ vs. α and of $\mathrm{Er}(\cdot, \cdot, 2/h, X)$ vs. h; $\circ : L^\infty$, $\bullet : L^2$, $\diamond : H_0^1$.

5 CONCLUDING REMARKS

For an iterative substructuring method subject to Robin interface conditions, $\frac{\partial u}{\partial n} + \alpha\, u$, we have considered strong type approximation to the normal derivative and performed numerical experiments by the finite element method. It is observed that in the strong type approximation the converged solution depends on α and that the choice of too small α leads to discontinuous converged solutions, that is, no convergence result. In the weak type approximation of Robin conditions the converged solutions are independent of α.

Acknowledgment

This work was supported by the Japan Society for the Promotion of Science under Grants-in-Aid for Scientific Research (C), No.22540143 and Scientific Research (S), No.24224004.

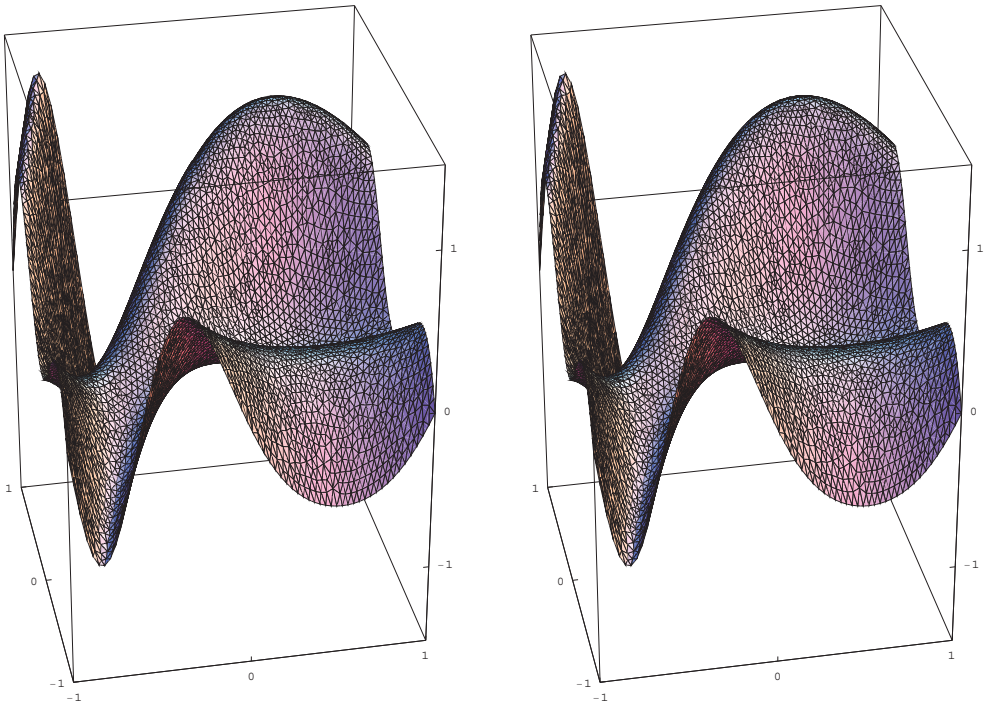


Figure 11: Graphs of (u_{1h}, u_{2h}) and u_h, $N = 64$.

References

[1] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. SIAM, New York, 2002.

[2] P. L. Lions. On the Schwarz alternating method III: a variant for non-overlapping subdomains. In T. F. Chan, R. Glowinski, J. Périaux, and O. Widlund, editors, *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pp. 202–231. SIAM, Philadelphia, 1990.

[3] J. L. Lions and E. Magenes. *Non Homogeneous Boundary Value Problems and Applications*, Vol. 1. Springer, Berlin, 1972.

[4] F. Magoulès and T. Kako, editors. *Domain Decomposition Methods: Theory and Applications*. Gakkotosho, Tokyo, 2006.

[5] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press, Oxford, 1999.

Optimized Schwarz Domain Decomposition Methods

Frédéric Magoulès**

**Ecole Centrale Paris

Applied Mathematics and Systems Laboratory, Grande Voie des Vignes

92295 Châtenay-Malabry Cedex, France

E-mail: frederic.magoules@hotmail.com

Summary:

In this paper, the general principles of asynchronous iterative algorithms and their extension to optimized Schwarz methods are presented. The mathematical framework needed for such algorithms is briefly introduced and the convergence of the asynchronous Schwarz method is given. The experimental results, performed on large scale engineering problems, demonstrate that the proposed asynchronous optimized Schwarz algorithms are as fast as their synchronous counterparts in all cases and become faster when the number of cores increases.

Key words: domain decomposition methods, Schwarz methods, parallel computing, asynchronous iterations

Abstract

Domain Decomposition methods are well suited for parallel computations. Indeed, the division of a problem into smaller subproblems, through artificial subdivisions of the domain, is a means for introducing parallelism. Domain Decomposition strategies include in one way or another the following ingredients: - a decomposer to split a mesh into subdomains ; - local solvers to find solutions for the subdomains for specific boundary conditions on the interface ; - interface conditions enforcing compatibility and equilibrium between overlapping or non-overlapping subdomains ; - a solution strategy for the interface problem. The differences between the methods lies in how those ingredients are actually put to work and how they are combined to form an efficient solution strategy for the problem at hand.

This paper focuses on the Schwarz domain decomposition method. It shows how these methods have efficiently evolved over the years to so called Optimized Schwarz methods by using specially designed boundary conditions on the interface. These optimized interface conditions specially designed to take into account the heterogeneity between the subdomains on each sides of the interfaces (in porous media), or the propagation of the wave through the interfaces (in acoustics), for instance, lead to robust and efficient algorithms.

This paper outlines the evolution of these optimized interfaces conditions over the years, and introduces how such methods can now be tuned for extremely large scale simulation on Petascale and Exascale computers. The proposed idea is based on the extension of optimized Schwarz methods to asynchronous context. The word asynchronous is here related to the iterations of the mathematical algorithm. Numerical experiments performed on large scale engineering problems illustrate the robustness and efficiency of the proposed method when solving engineering problems in fluid dynamics on parallel computers.

References

D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice-Hall, 1989.

P. Chevalier and F. Nataf. *Symmetrized method with optimized second-order conditions for the Helmholtz equation*. Contemporary Mathematics, Vol.218, pp.400–407, 1998.

M. Gander, F. Magoulès and F. Nataf. *Optimized Schwarz methods without overlap for the Helmholtz equation*. SIAM Journal on Scientific Computing, Vol.24, No.1, pp.38–60, 2002.

Y. Maday, F. Magoulès. *Absorbing interface conditions for domain decomposition methods: A general presentation.* Computer Methods in Applied Mechanics and Engineering, 195(29-32):3880–3900, 2006.

F. Magoulès and F.-X. Roux. *Lagrangian formulation of domain decomposition methods: a unified theory*. Applied Mathematical Modelling, 30(7):593-615, 2006.

F. Magoulès and T. Kako, editors. *Domain Decomposition Methods: Theory and Applications*, Volume 25 of Mathematical Sciences and Applications. Gakkotosho, Tokyo, Japan, 2006.

F. Nataf, F. Rogier, and E. de Sturler. *Optimal interface conditions for domain decomposition methods.* Technical Report 301, CMAP (Ecole Polytechnique),

1994.

P. Spiteri and M. Chau. *Parallel asynchronous Richardson method for the solution of obstacle problem*. In Parallel, High Performance Computing Systems and Applications, June 2002.

D. El Baz, A. Frommer, and P. Spiteri. *Asynchronous iterations with flexible communication: Contracting operators*. Journal of Computational and Applied 18 Mathematics, 176(1):91–103, 2005.

J.M. Bahi, S. Contassot-Vivier, and R. Couturier. *Parallel Iterative Algorithms: from Sequential to Grid Computing*, Chapman & Hall, CRC Press, Florida, US, 2007.

An Iterative Domain Decomposition Method with Mixed Formulations

TAGAMI, Daisuke[1]

1 Kyushu University, 744 Motooka, Nishi-ku, Fukuoka, 819-0395 JAPAN
tagami@imi.kyushu-u.ac.jp

Abstract

An iterative domain decomposition method is applied to eddy current problems. In our previous methods the gauge condition is neglected, then the magnetic vector potential is only one unknown function. On the other hand, in case of magnetostatic problems, it has been well-known that some theoretical results has been introduced, where a mixed formulation with the Lagrange multiplier is introduced in order to impose the gauge condition. Therefore, in this paper, we formulate again an iterative domain decomposition method based on a mixed formulation of eddy current problem, and discuss relations with the previous one.

Keywords: *eddy current problem, mixed formulation, iterative domain decomposition method*

1 Introduction

We have introduced an iterative domain decomposition method to solve quite large scale electromagnetic field problems; see, for example, Kanayama *et al*. [10]. In our previous methods the gauge condition is neglected, then the magnetic vector potential is only one unknown function. These previous results focus themselves on the engineering points of view: the previous formulation enables us to reduce computational consts in practical large scale simulations. However this formulation yields an indeterminate linear system, it is difficult to mathematically justify numerical results, for example unique solvability of the problems and convergency of the approximate solution.

On the other hand, in case of the magnetostatic problem, some theoretical results has been introduced by, for example, Kikuchi [8], [9], where a mixed formulation with the Lagrange multiplier is introduced in order to impose the gauge condition. These results focus themselves on the mathematical point of view: owing to the introduction of the Lagrange multiplier, their mixed formulation enable us to prove unique solvability of the problems and convergency of the approximate solution. However, this formulation yields an indefinite linear system, it is difficult to find an appropriate iterative solver, which is efficient enough to reduce computational costs for practical large scale problems.

At first in this paper, we formulate again an iterative domain decomposition method based on a mixed formulation of eddy current problem. Seconed, to reduce computational costs, we simplify our iterative domain decomposition method into another one, and we discuss relations between the reduced formulation and the previous one.

2 Formulation of eddy current problems

Let Ω be a convex polyhedoral domain with its boundary Γ. Assume that the domain Ω consists of two non-overlapping subdomains, a conducting part Ω_R and a non-conducting one Ω_S, with the interface Γ_{RS} between two subdomains. Let $\boldsymbol{n}$ be the outward unit normal of Ω_S. In this paper, for simplicity, assume that the conducting part Ω_R is also a convex polyhedral domain, and that the part Ω_R is strictly included in Ω.

Let $\boldsymbol{u}$ denote the magnetic vector potential, $\boldsymbol{f}$ an excitation current density, ν the magnetic reluctivity, σ the conductivity, ω the angular frequency, and i the imaginary unit. Then, we consider the three-dimensional eddy current problem with the Coulomb gauge condition:

$$\begin{cases} \operatorname{rot}(\nu \operatorname{rot} \boldsymbol{u}) - i\omega\sigma\boldsymbol{u} = \boldsymbol{f} & \text{in } \Omega, \quad (1a) \\ \operatorname{div} \boldsymbol{u} = 0 & \text{in } \Omega_S, \quad (1b) \\ \boldsymbol{u} \times \boldsymbol{n} = \boldsymbol{0} & \text{on } \Gamma, \quad (1c) \\ \displaystyle\int_{\Gamma_{RS}} \boldsymbol{u} \cdot n \, ds = 0; & (1d) \end{cases}$$

for some results of the related equations, for example, see Alonso and Valli [2]. Throughout this paper, assume that ν is a piecewise positive constant, that σ is a positive constant in Ω_R, while is equal to 0 in Ω_S, and that the divergence of $\boldsymbol{f}$ vanishes in Ω:

$$\operatorname{div} \boldsymbol{f} = 0 \text{ in } \Omega. \quad (2)$$

As usual, let $L^2(\Omega)$ be the space of complex functions defined in Ω and 2nd power summable in Ω, and let $(\,.\,,\,.\,)$ be its inner product; let $H^1(\Omega)$ be the space of functions in $L^2(\Omega)$ with derivatives up to the 1st order and set functional spaces X, M, V, and Q by

$$X := \{\boldsymbol{v} \in (L^2(\Omega))^3;\ \operatorname{rot} \boldsymbol{v} \in (L^2(\Omega))^3\}, \quad V := \{\boldsymbol{v} \in X;\ \boldsymbol{v} \times \boldsymbol{n} = \boldsymbol{0} \text{ on } \Gamma\},$$
$$M := H^1(\Omega), \qquad Q := \{q \in M;\ q = 0 \text{ on } \Gamma,\ \exists c \in \mathbf{C} \text{ s.t. } q = c \text{ in } \Omega_R\},$$

respectively; set bilinear forms $a(\,.\,,\,.\,)$ and $b(\,.\,,\,.\,)$ by

$$a(\boldsymbol{u}, \boldsymbol{v}) := (\nu \operatorname{rot} \boldsymbol{u}, \operatorname{rot} \boldsymbol{v}) - i\,(\omega\sigma\,\boldsymbol{u}, \boldsymbol{v}) \qquad \forall (\boldsymbol{u}, \boldsymbol{v}) \in X \times X,$$
$$b(\boldsymbol{v}, q) := (\boldsymbol{v}, \operatorname{grad} q) \qquad \forall (\boldsymbol{v}, q) \in (L^2(\Omega))^3 \times M,$$

respectively.

Now, by introducing the Lagrange multiplier p, we obtain a mixed weak formulation of (1) as follows: given $\boldsymbol{f} \in (L^2(\Omega))^3$, find $(\boldsymbol{u}, p) \in V \times Q$ such that

$$\begin{cases} a(\boldsymbol{u}, \boldsymbol{v}) + b(\boldsymbol{v}, p) = (\boldsymbol{f}, \boldsymbol{v}), & (3a) \\ b(\boldsymbol{u}, q) = 0, \qquad \forall (\boldsymbol{v}, q) \in V \times Q. & (3b) \end{cases}$$

Remark 1 *As in case of the magnetstatic problem in Kikuchi [8], if* $\boldsymbol{f}$ *satisfies that* $\mathrm{div}\,\boldsymbol{f} = 0$ *in* Ω*, then* $p = 0$*. This property plays a key role in the forthcoming section.*

3 Domain decomposition method

For simplicity, the domain Ω is assumed to be decomposed into two non-overlapping subdomains $\Omega^{(1)}$ and $\Omega^{(2)}$ with their boundaries $\partial\Omega^{(1)}$ and $\partial\Omega^{(2)}$, respectively:

$$\Omega^{(i)} \neq \emptyset \quad (i = 1, 2), \qquad \overline{\Omega} = \overline{\Omega}^{(1)} \cup \overline{\Omega}^{(2)}, \qquad \Omega^{(1)} \cap \Omega^{(2)} = \emptyset;$$

and let γ_{12} be the interface between $\Omega^{(1)}$ and $\Omega^{(2)}$ defined by $\gamma_{12} := \overline{\Omega}^{(1)} \cap \overline{\Omega}^{(2)}$; see Fig. 1. For $i = 1, 2$, the outward unit normal of $\Omega^{(i)}$ is denoted by $\boldsymbol{n}^{(i)}$, and set $\boldsymbol{n} = \boldsymbol{n}^{(1)} (= -\boldsymbol{n}^{(2)})$ on the interface γ_{12}. Moreover, the subdomain Ω_R is assumed to be decomposed into two non-

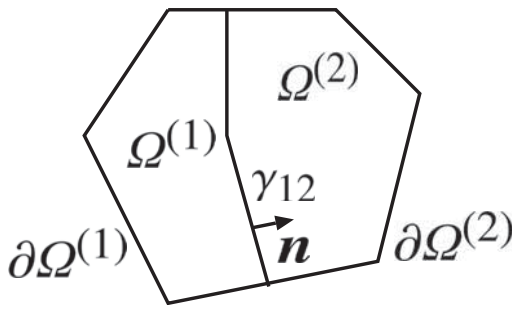


Fig. 1: Two non-overlapping subdomains of Ω.

overlapping subdomains $\Omega_R^{(1)}$ and $\Omega_R^{(2)}$ with the same assumptions as in the domain Ω.

Instead of the complex functions defined in Ω, we associate this decomposition to function spaces, bilinear forms, and inner product: let $L^2(\Omega^{(i)})$ and $H^1(\Omega^{(i)})$ be the space of real functions defined in $\Omega^{(i)}$, which are corresponding to $L^2(\Omega)$ and $H^1(\Omega)$; set function spaces $X^{(i)}$, $M^{(i)}$, $V^{(i)}_{\gamma_{12}}$, $Q^{(i)}_{\gamma_{12}}$, $V^{(i)}$, and $Q^{(i)}$ by

$$\begin{aligned}
X^{(i)} &:= \{\boldsymbol{v} \in (L^2(\Omega^{(i)}))^3;\ \mathrm{rot}\,\boldsymbol{v} \in (L^2(\Omega^{(i)}))^3\},\\
M^{(i)} &:= H^1(\Omega^{(i)}),\\
V^{(i)}_{\gamma_{12}} &:= \left\{\boldsymbol{v} \in X^{(i)};\ \ \boldsymbol{v} \times \boldsymbol{n} = \boldsymbol{0} \ \text{ on } \partial\Omega^{(i)} \backslash \gamma_{12}\right\},\\
Q^{(i)}_{\gamma_{12}} &:= \left\{q \in M^{(i)};\ \ q = 0 \ \text{ on } \partial\Omega^{(i)} \backslash \gamma_{12}, \exists c \in \mathbf{C} \text{ s.t. } q = c \ \text{ in } \Omega_R^{(i)}\right\},\\
V^{(i)} &:= \left\{\boldsymbol{v} \in X^{(i)};\ \ \boldsymbol{v} \times \boldsymbol{n} = \boldsymbol{0} \ \text{ on } \partial\Omega^{(i)}\right\},\\
Q^{(i)} &:= \left\{q \in M^{(i)};\ \ q = 0 \ \text{ on } \partial\Omega^{(i)}\right\},
\end{aligned}$$

respectively; and set bilinear forms and inner product $a^{(i)}(\,.\,,\,.\,)$, $b^{(i)}(\,.\,,\,.\,)$, and $(\,.\,,\,.\,)_{\Omega^{(i)}}$ by the integrals over $\Omega^{(i)}$, respectively. Moreover, set function spaces Λ and Ξ by

$$\Lambda := \left\{ \boldsymbol{\lambda} : \gamma_{12} \to \mathbb{R}^3;\ \boldsymbol{\lambda} = (\boldsymbol{v} \times \boldsymbol{n})|_{\gamma_{12}},\ \boldsymbol{v} \in V \right\}, \qquad \Xi := \left\{ \xi : \gamma_{12} \to \mathbb{R};\ \xi = q|_{\gamma_{12}},\ q \in Q \right\};$$

and set $\overline{\boldsymbol{u}}^{(i)}(\boldsymbol{\eta})$ by any extension operator from Λ to $V^{(i)}_{\gamma_{12}}$ such that $\boldsymbol{\eta} = (\overline{\boldsymbol{u}}^{(i)}(\boldsymbol{\eta}) \times \boldsymbol{n})|_{\gamma_{12}}$, and $\bar{p}^{(i)}(\zeta)$ by any extension operator from Ξ to $Q^{(i)}_{\gamma_{12}}$ such that $\zeta = p(\zeta)|_{\gamma_{12}}$. A characterization of *tangential trace* spaces Λ and an *tangential extension* operator on $\overline{\boldsymbol{u}}^{(i)}(\boldsymbol{\eta})$ has been given in Alonso–Valli [1], Buffa–Ciarlet [3], [4], Buffa, *et al.* [5], and Quarteroni–Valli [11].

Now, a two-subdomain problem is introduced by the followings: for $i = 1, 2$, find $(\boldsymbol{u}^{(i)}, p^{(i)}) \in V^{(i)}_{\gamma_{12}} \times Q^{(i)}_{\gamma_{12}}$ such that

$$\begin{cases} a^{(i)}(\boldsymbol{u}^{(i)},\ \boldsymbol{v}^{(i)}) + b^{(i)}(\boldsymbol{v}^{(i)},\ p^{(i)}) = (\boldsymbol{f}^{(i)},\ \boldsymbol{v}^{(i)})_{\Omega^{(i)}}, & \text{(4a)} \\ b^{(i)}(\boldsymbol{u}^{(i)},\ q^{(i)}) = 0, \qquad \forall (\boldsymbol{v}^{(i)},\ q^{(i)}) \in V^{(i)} \times Q^{(i)} & \text{(4b)} \\ \boldsymbol{u}^{(1)} \times \boldsymbol{n} = \boldsymbol{u}^{(2)} \times \boldsymbol{n} \quad \text{on } \gamma_{12}, & \text{(4c)} \\ p^{(1)} = p^{(2)} \quad \text{on } \gamma_{12}, & \text{(4d)} \\ a^{(2)}(\boldsymbol{u}^{(2)},\ \overline{\boldsymbol{u}}^{(2)}(\boldsymbol{\eta})) + b^{(2)}(\overline{\boldsymbol{u}}^{(2)}(\boldsymbol{\eta}),\ p^{(2)}) & \\ \qquad = (\boldsymbol{f}^{(1)},\ \overline{\boldsymbol{u}}^{(1)}(\boldsymbol{\eta}))_{\Omega^{(1)}} + (\boldsymbol{f}^{(2)},\ \overline{\boldsymbol{u}}^{(2)}(\boldsymbol{\eta}))_{\Omega^{(2)}} - a^{(1)}(\boldsymbol{u}^{(1)},\ \overline{\boldsymbol{u}}^{(1)}(\boldsymbol{\eta})) - b^{(1)}(\overline{\boldsymbol{u}}^{(1)}(\boldsymbol{\eta}),\ p^{(1)}), & \text{(4e)} \\ b^{(2)}(\boldsymbol{u}^{(2)},\ \bar{p}^{(2)}(\zeta)) = b^{(1)}(\boldsymbol{u}^{(1)},\ \bar{p}^{(1)}(\zeta)), \qquad \forall (\boldsymbol{\eta},\ \zeta) \in \Lambda \times \Xi. & \text{(4f)} \end{cases}$$

If $\{(\boldsymbol{u}^{(1)},\ p^{(1)}), (\boldsymbol{u}^{(2)},\ p^{(2)})\}$ is a pair of the solutions of two-subdomain problem (4), then the solution of the one-domain problem (3) could be constructed by

$$(\boldsymbol{u}, p) := \begin{cases} (\boldsymbol{u}^{(1)},\ p^{(1)}) \quad \text{in } \Omega^{(1)}, & \text{(5a)} \\ (\boldsymbol{u}^{(2)},\ p^{(2)}) \quad \text{in } \Omega^{(2)}. & \text{(5b)} \end{cases}$$

On the other hand, if $(\boldsymbol{u}, p)$ is a solution of the one-domain problem (3), then a pair of the solutions $\{(\boldsymbol{u}^{(1)},\ p^{(1)}), (\boldsymbol{u}^{(2)},\ p^{(2)})\}$ of the two-subdomain problem (4) could be constructed by

$$(\boldsymbol{u}^{(i)}, p^{(i)}) := (\boldsymbol{u}|_{\Omega^{(i)}}, p|_{\Omega^{(i)}}) \quad \text{in } \Omega^{(i)}. \tag{6}$$

Therefore, the equivalency between both formulations could be obtained as follows:

Theorem 1 *The one-domain problem* (3) *and the two-subdomain problem* (4) *are equivalent.*

For $i = 1, 2$, let $\mathscr{E}^{(i)}(\boldsymbol{f}, \boldsymbol{\lambda}, \xi)$ an extention operator from $(L^2(\Omega))^3 \times \Lambda \times \Xi$ to $V^{(i)}_{\gamma_{12}} \times Q^{(i)}_{\gamma_{12}}$ defined by $\mathscr{E}^{(i)}(\boldsymbol{f}, \boldsymbol{\lambda}, \xi) := (\boldsymbol{u}^{(i)}, p^{(i)})$, where $(\boldsymbol{u}^{(i)}, p^{(i)})$ is the solution of the following eddy current problem:

$$\begin{cases} a^{(i)}(\boldsymbol{u}^{(i)},\ \boldsymbol{v}^{(i)}) + b^{(i)}(\boldsymbol{v}^{(i)},\ p^{(i)}) = (\boldsymbol{f}^{(i)},\ \boldsymbol{v}^{(i)})_{\Omega^{(i)}}, & \text{(7a)} \\ b^{(i)}(\boldsymbol{u}^{(i)},\ q^{(i)}) = 0, \qquad \forall (\boldsymbol{v}^{(i)}, q^{(i)}) \in V^{(i)} \times Q^{(i)}, & \text{(7b)} \\ \boldsymbol{u}^{(i)} \times \boldsymbol{n} = \boldsymbol{\lambda} \quad \text{on } \gamma_{12}, & \text{(7c)} \\ p^{(i)} = \xi \quad \text{on } \gamma_{12}. & \text{(7d)} \end{cases}$$

Then, a *Steklov–Poincaré* operator $\mathscr{A}$ from $\Lambda \times \Xi$ to $(\Lambda \times \Xi)'$ is set by

$$\langle \mathscr{A}(\boldsymbol{\lambda}, \xi), (\boldsymbol{\eta}, \zeta) \rangle_{\gamma_{12}} := \sum_{i=1}^{2} \{ a^{(i)}(\overline{\boldsymbol{u}}^{(i)}, \overline{\boldsymbol{v}}^{(i)}) + b^{(i)}(\overline{\boldsymbol{v}}^{(i)}, \bar{p}^{(i)}) + b^{(i)}(\overline{\boldsymbol{u}}^{(i)}, \bar{q}^{(i)}) \}, \quad \forall \boldsymbol{\lambda}, \boldsymbol{\eta} \in \Lambda, \ \forall \xi, \zeta \in \Xi \tag{8}$$

where $(\overline{\boldsymbol{u}}^{(i)}, \bar{p}^{(i)}) := \mathscr{E}^{(i)}(0, \boldsymbol{\lambda}, \xi)$ and $(\overline{\boldsymbol{v}}^{(i)}, \bar{q}^{(i)}) := \mathscr{E}^{(i)}(0, \boldsymbol{\eta}, \zeta)$; and an *interface source* $\boldsymbol{\chi} \in (\Lambda \times \Xi)'$ is set by

$$\langle \boldsymbol{\chi}, (\boldsymbol{\eta}, \zeta) \rangle_{\gamma_{12}} := \sum_{i=1}^{2} \{ (\boldsymbol{f}^{(i)}, \overline{\boldsymbol{v}}^{(i)})_{\Omega^{(i)}} - a^{(i)}(\widehat{\boldsymbol{u}}^{(i)}, \overline{\boldsymbol{v}}^{(i)}) - b^{(i)}(\overline{\boldsymbol{v}}^{(i)}, \widehat{p}^{(i)}) - b^{(i)}(\widehat{\boldsymbol{u}}^{(i)}, \bar{q}^{(i)}) \}, \quad \forall \xi, \zeta \in \Xi \tag{9}$$

where $(\widehat{\boldsymbol{u}}^{(i)}, \widehat{p}^{(i)}) := \mathscr{E}^{(i)}(\boldsymbol{f}^{(i)}, \mathbf{0}, 0)$ and $(\overline{\boldsymbol{v}}^{(i)}, \bar{q}^{(i)}) := \mathscr{E}^{(i)}(\mathbf{0}, \boldsymbol{\eta}, \zeta)$. Now we introduce the following *interface problem* on γ_{12}:

$$\langle \mathscr{A}(\boldsymbol{\lambda}, \xi), (\boldsymbol{\eta}, \zeta) \rangle_{\gamma_{12}} = \langle \boldsymbol{\chi}, (\boldsymbol{\eta}, \zeta) \rangle_{\gamma_{12}}, \qquad \forall (\boldsymbol{\eta}, \zeta) \in \Lambda \times \Xi. \tag{10}$$

By using the solution $(\boldsymbol{u}^{(i)}, p^{(i)})$ of two-subdomain problem (4), let us set $(\boldsymbol{\lambda}, \xi)$ by $\boldsymbol{\lambda} := \boldsymbol{u}^{(1)} \times \boldsymbol{n}$ $(= \boldsymbol{u}^{(2)} \times \boldsymbol{n})$ and $\xi := p^{(1)} (= p^{(2)})$. Then, because of (4c)–(4f), $(\boldsymbol{\lambda}, \xi)$ satisfies the *interface problem* (10). On the other hand, once the solution $(\boldsymbol{\lambda}, \xi)$ is obtained by solving the *interface problem* (10), for $i = 1, 2$, each pair $(\boldsymbol{u}^{(i)}, p^{(i)}) \in V^{(i)}_{\gamma_{12}} \times Q^{(i)}_{\gamma_{12}}$ could be found from the problem (4a) and (4b) in the corresponding subdomain $\Omega^{(i)}$, where the solution $(\boldsymbol{\lambda}, \xi)$ is regarded as the Dirichlet boundary on the interface: $\boldsymbol{u}^{(i)} \times \boldsymbol{n} = \boldsymbol{\lambda}$ and $p^{(i)} = \xi$ on γ_{12}. Finally, from (5), we can obtain the solution $(\boldsymbol{u}, p)$ of the one-domain problem (3).

The *interface problem* (10) complex-symmetric. Therefore, as the solver, the BiConjugate Gradient method (BiCG) is used; see Freund [6]. Then, by choosing an appropriate dual initial residual, BiCG is formally the same as the conjugate gradient method for real valued matrices; see Van der Vorst and Melissen [12]. Using these facts, we can now describe the following *biconjugate gradient* method of the linear system derived from the interface problem (10) as in Glowinski *et al* [7] (at least formally):

Choose $(\boldsymbol{\lambda}_0, \xi_0)$;

Compute $(\boldsymbol{g}_0, \delta_0)$ **by** (11);

$(\boldsymbol{w}_0, \omega_0) := (\boldsymbol{g}_0, \delta_0)$;

for $k = 0, 1, \ldots$;

  Compute $\mathscr{A}(\boldsymbol{w}_k, \omega_k)$ **by** (12);

  $\alpha_k := ((\boldsymbol{g}_k, \delta_k), (\boldsymbol{g}_k, \delta_k))/(\mathscr{A}(\boldsymbol{w}_k, \omega_k), (\boldsymbol{w}_k, \omega_k))$;

  $(\boldsymbol{\lambda}_{k+1}, \xi_{k+1}) := (\boldsymbol{\lambda}_k, \xi_k) - \alpha_k (\boldsymbol{w}_k, \omega_k)$;

  $(\boldsymbol{g}_{k+1}, \delta_{k+1}) := (\boldsymbol{g}_k, \delta_k) - \alpha_k \mathscr{A}(\boldsymbol{w}_k, \omega_k)$;

$\beta_k := ((\boldsymbol{g}_{k+1}, \delta_{k+1}), (\boldsymbol{g}_{k+1}, \delta_{k+1}))/((\boldsymbol{g}_k, \delta_k), (\boldsymbol{g}_k, \delta_k))$;

If $((\boldsymbol{g}_{k+1}, \delta_{k+1}), (\boldsymbol{g}_{k+1}, \delta_{k+1}))/((\boldsymbol{g}_0, \delta_0), (\boldsymbol{g}_0, \delta_0)) < \varepsilon$, **break**;

$(\boldsymbol{w}_{k+1}, \omega_{k+1}) := (\boldsymbol{g}_{k+1}, \delta_{k+1}) + \beta_k (\boldsymbol{w}_k, \omega_k)$;

end;

where $(\,.\,,\,.\,)$ is a just multiplication of complex numbers, and ε is a positive constant for the criterion of the convergence. In the above biconjugate gradient algorythm, $(\boldsymbol{g}_0, \delta_0)$ could be computed by the extentions $(\widetilde{\boldsymbol{u}}_0^{(i)}, \widetilde{p}_0^{(i)})$ and $(\bar{\boldsymbol{v}}^{(i)}, \bar{q}^{(i)})$ as follow:

$$\begin{aligned}&\langle(\boldsymbol{g}_0, \delta_0),\ (\boldsymbol{\eta}, \zeta)\rangle_{\gamma_{12}}\\ &= \sum_{i=1}^{2}\{a^{(i)}(\widetilde{\boldsymbol{u}}_0^{(i)},\ \bar{\boldsymbol{v}}^{(i)}) + b^{(i)}(\bar{\boldsymbol{v}}^{(i)},\ \widetilde{p}_0^{(i)}) - (\boldsymbol{f}^{(i)},\ \bar{\boldsymbol{v}}^{(i)})_{\Omega^{(i)}} + b^{(i)}(\widetilde{\boldsymbol{u}}_0^{(i)},\ \bar{q}^{(i)})\}, \qquad \forall(\boldsymbol{\eta},\ \zeta) \in \Lambda \times \Xi, \quad (11)\end{aligned}$$

where $(\widetilde{\boldsymbol{u}}_0^{(i)}, \widetilde{p}_0^{(i)}) := \mathscr{E}^{(i)}(\boldsymbol{f}^{(i)}, \boldsymbol{\lambda}_0, \xi_0)$; and $\mathscr{A}(\boldsymbol{w}_k, \omega_k)$ could be computed by the extentions $(\widehat{\boldsymbol{u}}_0^{(i)}, \widehat{p}_0^{(i)})$ and $(\bar{\boldsymbol{v}}^{(i)}, \bar{q}^{(i)})$ as follow:

$$\langle\mathscr{A}(\boldsymbol{w}_k, \omega_k),\ (\boldsymbol{\eta}, \zeta)\rangle_{\gamma_{12}} = \sum_{i=1}^{2}\{a^{(i)}(\widehat{\boldsymbol{u}}_k^{(i)},\ \bar{\boldsymbol{v}}^{(i)}) + b^{(i)}(\bar{\boldsymbol{v}}^{(i)}, \widehat{p}_k^{(i)}) + b^{(i)}(\widehat{\boldsymbol{u}}_k^{(i)},\ \bar{q}^{(i)})\}, \qquad \forall(\boldsymbol{\eta},\ \zeta) \in \Lambda \times \Xi, \tag{12}$$

where $(\widehat{\boldsymbol{u}}_k^{(i)}, \widehat{p}_k^{(i)}) := \mathscr{E}^{(i)}(\boldsymbol{0}, \boldsymbol{w}_k, \omega_k)$. The extentions $(\widetilde{\boldsymbol{u}}_0^{(i)}, \widetilde{p}_0^{(i)})$, $(\widehat{\boldsymbol{u}}_0^{(i)}, \widehat{p}_0^{(i)})$, and $(\bar{\boldsymbol{v}}^{(i)}, \bar{q}^{(i)})$ in (11) and (12) could be computed in $\Omega^{(1)}$ and $\Omega^{(2)}$ independently. Therefore, the above biconjugate gradient algorythm is familiar with parallel computations.

Moreover, as mentioned in Remark 1, if $\boldsymbol{f}^{(i)}$ satisfies that $\operatorname{div} \boldsymbol{f}^{(i)} = 0$ in $\Omega^{(i)}$, then $p^{(i)}$ vanishes. This implies that we can neglect the components corresponding to the Lagrange multiplier in the biconjugate gradient algorythm. Therefore we can get the reduced biconjugate gradient algorythm as follows:

Choose $\boldsymbol{\lambda}_0$;

Compute $\boldsymbol{g}_0$ **by** (13);

$\boldsymbol{w}_0 := \boldsymbol{g}_0$;

for $k = 0, 1, \ldots$;

Compute $\mathscr{A}_1(\boldsymbol{w}_k, 0)$ **by** (14);

$\alpha_k := (\boldsymbol{g}_k, \boldsymbol{g}_k)/(\mathscr{A}_1(\boldsymbol{w}_k, 0), \boldsymbol{w}_k,)$;

$\boldsymbol{\lambda}_{k+1} := \boldsymbol{\lambda}_k - \alpha_k \boldsymbol{w}_k$;

$\boldsymbol{g}_{k+1} := \boldsymbol{g}_k - \alpha_k \mathscr{A}_1(\boldsymbol{w}_k, 0)$;

$\beta_k := (\boldsymbol{g}_{k+1}, \boldsymbol{g}_{k+1})/(\boldsymbol{g}_k, \boldsymbol{g}_k)$;

If $(\boldsymbol{g}_{k+1}, \boldsymbol{g}_{k+1})/(\boldsymbol{g}_0, \boldsymbol{g}_0) < \varepsilon$, **break**;

$\boldsymbol{w}_{k+1} := \boldsymbol{g}_{k+1} + \beta_k \boldsymbol{w}_k;$

end;

In the reduced biconjugate gradient algorythm, $\boldsymbol{g}_0$ could be computed by the first component of the following equation:

$$\begin{aligned}&\langle(\boldsymbol{g}_0, \delta_0),\ (\boldsymbol{\eta}, \zeta)\rangle_{\gamma_{12}} \\ &= \sum_{i=1}^{2}\{a^{(i)}(\widetilde{\boldsymbol{u}}_0^{(i)},\ \bar{\boldsymbol{v}}^{(i)}) + b^{(i)}(\bar{\boldsymbol{v}}^{(i)},\ \widetilde{p}_0^{(i)}) - (\boldsymbol{f}^{(i)},\ \bar{\boldsymbol{v}}^{(i)})_{\Omega^{(i)}} + b^{(i)}(\widetilde{\boldsymbol{u}}_0^{(i)},\ \bar{q}^{(i)})\}, \qquad \forall(\boldsymbol{\eta},\ \zeta) \in \Lambda \times \Xi, \quad (13)\end{aligned}$$

where $(\widetilde{\boldsymbol{u}}_0^{(i)}, \widetilde{p}_0^{(i)}) := \mathscr{E}^{(i)}(\boldsymbol{f}^{(i)}, \boldsymbol{\lambda}_0, 0)$; and $\mathscr{A}_1(\boldsymbol{w}_k, 0)$ could be computed by the first component of the following equation:

$$\langle\mathscr{A}(\boldsymbol{w}_k, 0),\ (\boldsymbol{\eta}, \zeta)\rangle_{\gamma_{12}} = \sum_{i=1}^{2}\{a^{(i)}(\widehat{\boldsymbol{u}}_k^{(i)},\ \bar{\boldsymbol{v}}^{(i)}) + b^{(i)}(\bar{\boldsymbol{v}}^{(i)},\ \widehat{p}_k^{(i)}) + b^{(i)}(\widehat{\boldsymbol{u}}_k^{(i)},\ \bar{q}^{(i)})\}, \qquad \forall(\boldsymbol{\eta},\ \zeta) \in \Lambda \times \Xi, \quad (14)$$

where $(\widehat{\boldsymbol{u}}_k^{(i)}, \widehat{p}_k^{(i)}) := \mathscr{E}^{(i)}(\boldsymbol{0}, \boldsymbol{w}_k, 0)$.

References

[1] Alonso Rodríguez, A. and Valli, A., Some remarks on the characterization of the space of tangential traces of $H(\mathrm{rot}; \Omega)$ and the construction of an extension operator, *Manuscripta Math.*, 89 (1996), pp.159–178.

[2] Alonso Rodríguez, A. and Valli, A., *Eddy Current Approximation of Maxwell equations, Theory, algorithms and aplications*, MS&A Vol.4, Springer, 2010.

[3] Buffa, A. and Ciarlet, P.-G., On traces for functional spaces related to Maxwell's equations. I, An integration by parts formula in Lipschitz polyhedra, *Math. Methods Appl. Sci.*, 24 (2001), pp.9–30.

[4] Buffa, A. and Ciarlet, P.-G., On traces for functional spaces related to Maxwell's equations. II, Hodge decompositions on the boundary of Lipschitz polyhedra and applications, *Math. Methods Appl. Sci.*, 24 (2001), pp.31–48.

[5] Buffa, A., Costabel, M., and Sheen, D., On traces for $\mathbf{H}(\mathbf{curl}, \Omega)$ in Lipschitz domains, *J. Math. Anal. Appl.*, 276 (2002), pp.845–867.

[6] Freund, R. W., Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices, *SIAM J. Sci. Statist. Comput.*, 13 (1992), pp.425–448.

[7] Glowinski, R., Dinh, Q.V., and Periaux, J., Domain decomposition methods for nonlinear problems in fluid dynamics, *Compt. Meths. Appl. Mech. Engrg.*, 40 (1983), pp.27–109.

[8] Kikuchi, F., Mixed formulations for finite element analysis of magnetostatic and electrostatic problems, *Japan J. Appl. Math.*, 6 (1989), pp.209–221.

[9] Kikuchi, F., On a discrete compactness property for the Nedelec finite elements, *J. Fac. Sci. Univ. Tokyo, Sect. IA Math.*, 36 (1989), pp.479–490.

[10] Kanayama, H., Shioya, R., Tagami, D., and Matsumoto, S., 3-D Eddy Current Computation for a Transformer Tank, *COMPEL*, 21 (2002), pp.554–562.

[11] Quarteroni, A. and Valli, A., Domain Decomposition Methods for Partial Differential Equations, Oxford, 1999.

[12] Van der Vorst, H. A., and Melissen, J. B. M., A Petrov–Galerkin type method for solving $Ax = b$, where A is symmetric complex, *IEEE Trans. Magn.*, 26 (1990), pp.706–709.

Application of Transient Eddy Current Analysis to Rotating Machines

Hiroshi Kanayama*(Kyushu University), Shin-ichiro Sugimoto (The University of Tokyo),
Kouhei Murotani (The University of Tokyo), Seigo Terada (Kyushu University), Seiya Kuramoto (Kyushu University)

Transient eddy current analysis of rotating machines is considered. NEXST_Magnetic is mainly used in this study. Cases without and with rotor rotation are considered. The accuracy of a simple model is confirmed and a three-phase induction motor is analyzed.

Keywords: Induction motor, Eddy current, Transient analysis, NEXST_Magnetic, ADVENTURE_Magnetic

1. Introduction

Various methods have been considered for three-dimensional time-harmonic eddy current problems and effective numerical calculations have been performed to obtain many results. The A method in which the magnetic vector potential A is the unknown function and the $A-\phi$ method in which both A and the electric scalar potential ϕ are unknown functions have both been demonstrated to be effective for these types of problems. In a previous study, we applied hierarchical domain decomposition to these methods and demonstrated that accurate numerical results for large-scale problems could be obtained by parallel computations [1,2]. However, it is necessary to treat the problem as non-stationary for many real-world phenomena. For time-harmonic three-dimensional eddy current problems, we used the $A-\phi$ method, which exhibits favorable performance from a computation-time perspective. Using the same technique for the space approximation, we now introduce a backward Euler method for time stepping to study transient three-dimensional eddy-current problems [3]. For this purpose, we employed and improved the NEXST_Magnetic software [4]. This software provides a single-processor module for finite-element analysis and allows analysis of nonlinear magnetostatic fields and non-stationary eddy-current analysis for up to tens of millions of degrees of freedom. In this study, we extend this software by adding functionality to consider the *B–H* characteristics of materials [5] and we present computational results obtained using this new functionality. Finally, we discuss a method for addressing the rotation of rotors [6].

2. Eddy Current Problems

2.1 Basic equations and boundary conditions

We consider a polyhedral domain Ω with boundary $\partial\Omega$. The domain Ω consists of two non-overlapping polyhedral subregions: a conducting region R and a non-conducting region S. Let Γ denote the common portion of the boundaries of R and S and let n be the outward unit normal vector to the boundary surfaces. The magnetic vector potential A [Wb/m] and the electric scalar potential ϕ [V] are unknown functions. From Maxwell's equations, we can derive the following equations for the magnetic field in three-dimensional non-stationary eddy current problems; here,

ν is the magnetic reluctivity and σ is the conductivity.

$$\mathrm{rot}\left(\nu\,\mathrm{rot}\,A\right)+\sigma\frac{\partial A}{\partial t}+\sigma\,\mathrm{grad}\,\phi=J \ \text{ in } \Omega\times(0,\mathrm{T}), \tag{1a}$$

$$-\mathrm{div}\left(\sigma\frac{\partial A}{\partial t}+\sigma\,\mathrm{grad}\,\phi\right)=0 \ \text{in}\ R\times(0,\mathrm{T}), \tag{1b}$$

$$-\left(\sigma\frac{\partial A}{\partial t}+\sigma\,\mathrm{grad}\,\phi\right)\cdot n=0 \ \text{on}\ \Gamma\times(0,\mathrm{T}), \tag{1c}$$

$$A\times n=0 \ \text{on}\ \partial\Omega\times(0,\mathrm{T}), \tag{1d}$$

$$A\big|_{t=0}=A^0 \ \text{in}\ \Omega. \tag{1e}$$

The excitation current density $J\left[\mathrm{A/m^2}\right]$ satisfies the following continuity equation:

$$\mathrm{div}\,J=0 \ \text{ in } \Omega\times(0,\mathrm{T}). \tag{2}$$

2.2 The weak form and finite element approximation

Let $L^2(\Omega)$ be a space of functions defined in Ω and square summable in Ω with its inner product (. , .) and let $H^1(\Omega)$ be a space of functions in $L^2(\Omega)$ with derivatives up to the first order. We define V,U and W by

$$V\equiv\left\{v\in\left(L^2(\Omega)\right)^3;\mathrm{rot}\,v\in\left(L^2(\Omega)\right)^3,v\times n=0 \ \text{ on } \ \partial\Omega\right\},$$

$$U\equiv\left\{u\in H^1(R)\right\},$$

$$W\equiv\left\{w\in H^1(\Omega);w=0 \ \text{ on } \partial\Omega\right\}.$$

The weak form of equation (1) may be expressed as follows. Here A,ϕ are unknown functions $\left(\text{with}\,(A,\phi)\in V\times U\right)$ and A^*,ϕ^* are arbitrary test functions (again with

$(A^*, \phi^*) \in V \times U)$.

$$(\nu \operatorname{rot} A, \operatorname{rot} A^*) + \left(\sigma \frac{\partial A}{\partial t}, A^*\right) + (\sigma \operatorname{grad} \phi, A^*) = (J, A^*), \tag{3a}$$

$$(\sigma \operatorname{grad} \phi, \operatorname{grad} \phi^*) + \left(\sigma \frac{\partial A}{\partial t}, \operatorname{grad} \phi^*\right) = 0. \tag{3b}$$

We next divide the domain Ω into union of tetrahedra and discretize space. We approximate the magnetic vector potential using first-order Nedelec elements and we approximate the electric scalar potential and other quantities using the conventional first-order tetrahedral elements. Let V_h, U_h, W_h respectively denote the finite element spaces corresponding to V, U, W. Introducing the backward Euler method for time discretization (Δt : a time increment) allows us to write the finite element equation in the following form:

$$(\nu \operatorname{rot} A_h^{n+1}, \operatorname{rot} A_h^*) + \left(\frac{\sigma}{\Delta t} A_h^{n+1}, A_h^*\right) + (\sigma \operatorname{grad} \phi_h^{n+1}, A_h^*) = (\tilde{J}_h^{n+1}, A_h^*) + \left(\frac{\sigma}{\Delta t} A_h^n, A_h^*\right), \tag{4a}$$

$$\left(\frac{\sigma}{\Delta t} A_h^{n+1}, \operatorname{grad} \phi_h^*\right) + (\sigma \operatorname{grad} \phi_h^{n+1}, \operatorname{grad} \phi_h^*) = \left(\frac{\sigma}{\Delta t} A_h^n, \operatorname{grad} \phi_h^*\right). \tag{4b}$$

Here, for actual calculations, we use the conventional first-order tetrahedral element for $\tilde{J}_h^{n+1}$. For a correction obtained by considering Eq. (2), we approximate J^{n+1} and perform a pre-calculation as follows:

$$(\operatorname{grad} I_h^{n+1}, \operatorname{grad} I_h^*) = (J_h^{n+1}, \operatorname{grad} I_h^*). \tag{5}$$

After obtaining I_h^{n+1} by Eq. (5), we find the corrected approximate current density $\tilde{J}_h^{n+1}$ from the relation :

$$\tilde{J}_h^{n+1} = J_h^{n+1} - \operatorname{grad} I_h^{n+1}. \tag{6}$$

Note that $I_h^{n+1}, I_h^* \in W_h$.

2.3 Transient analysis considering the B–H characteristics

The magnetic reluctivity ν exhibits a material-dependent nonlinearity. In transient analysis, the values of ν are obtained at each time step from the *H–B* curve (Fig. 1)

obtained from measured data [5]. Denoting the magnetic reluctivity ν at the n-th time step by ν_h^{n} and substituting into Eq. (4a) yields

$$\left(\nu_h^{\mathrm{n}}\ \mathrm{rot}\ A_h^{\mathrm{n+1}}, \mathrm{rot}\ A_h^*\right)+\left(\frac{\sigma}{\Delta t}A_h^{\mathrm{n+1}}, A_h^*\right)+\left(\sigma\ \mathrm{grad}\,\phi_h^{\mathrm{n+1}}, A_h^*\right)=\left(\tilde{J}_h^{\mathrm{n+1}}, A_h^*\right)+\left(\frac{\sigma}{\Delta t}A_h^{\mathrm{n}}, A_h^*\right). \quad (7)$$

Thus, the value of the magnetic reluctivity ν_h^{n} obtained from the calculation for time step n is used in the computation for time step $\mathrm{n}+1$. We have

$$\begin{aligned}|H| &= \nu_i\left(|B|-|B_i|\right)+|H_i| \\ &= \nu_i|B|+\left(|H_i|-\nu_i|B_i|\right) \\ &= \left\{\nu_i+\frac{\left(|H_i|-\nu_i|B_i|\right)}{|B|}\right\}|B|,\end{aligned}$$

from which it follows that

$$\nu\left(|B|\right)=\nu_i+\frac{|H_i|-\nu_i|B_i|}{|B|}. \quad (8)$$

In these equations, $|B_i|\le|B|\le|B_{i+1}|$. If, in addition, $|B_0|\le|B|\le|B_1|$, then the following result holds:

$$\nu\left(|B|\right)=\nu_0\ . \quad (9)$$

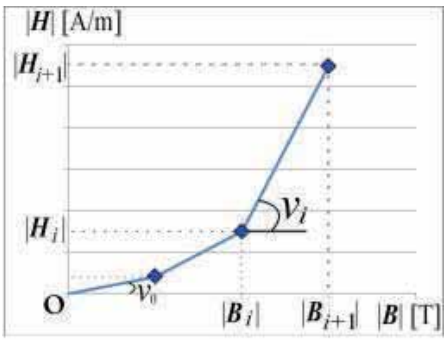


Fig.1. *H-B* curve

If we simply use the magnetic reluctivity at the time step for Eq. (8), it is not possible to obtain stable results. However, by introducing a weighting factor $w\left(0<w\le1\right)$, it is

possible to obtain stable values for the magnetic reluctivity [5]. The values of $\left|B_g^{\mathrm{n}}\right|$ at the centroids of the elements are obtained from the solution A_h^{n} at time step n and v_h^{n} is determined by the following relations:

$$v_{HB}^{\mathrm{n}} = v_i^{\mathrm{n}} + \frac{\left|H_i^{\mathrm{n}}\right| - v_i^{\mathrm{n}}\left|B_i^{\mathrm{n}}\right|}{\left|B_g^{\mathrm{n}}\right|},$$

$$v_h^{\mathrm{n}} = w\, v_{HB}^{\mathrm{n}} + (1-w) v_h^{\mathrm{n}-1}. \tag{10}$$

The value of v_h^{n} obtained from Eq. (10) is used in the computation for time step $\mathrm{n}+1$.

2.4 Transient analysis in the presence of rotation

If we work in a moving coordinate system fixed within a rotating body, then the basic equations (1) may still be valid. Thus, in the following treatment of rotating bodies, we consider Eq. (1) as holding within a moving coordinate system fixed within the rotating body and we subsequently transform this equation system back to a coordinate system fixed in space to apply Eq. (4) in the context of the finite element method. In this case, the time derivatives $\frac{\partial A}{\partial t}$ in Eq. (1) become Lagrange derivatives $\frac{DA}{Dt}$. On discretizing space into finite elements [6], we use the following approximate relation for the Lagrange derivative term at a point p in the rotating body at time $(\mathrm{n}+1)\Delta t$:

$$\frac{DA_h(x,(\mathrm{n}+1)\Delta t)}{Dt} \cong \frac{A_h(x,(\mathrm{n}+1)\Delta t) - A_h(x - v\Delta t, \mathrm{n}\Delta t)}{\Delta t}, \tag{11}$$

here, x denotes the coordinates of the point p and v is a given velocity vector (assumed to be constant here for simplicity). The quantity $A_h(x - v\Delta t, \mathrm{n}\Delta t)$ is known and $A_h(x,(\mathrm{n}+1)\Delta t)$ is the unknown quantity for which we wish to solve. Using superscripts n and $\mathrm{n}+1$ to denote time steps, the approximate values are written respectively as A_h^{n} and $A_h^{\mathrm{n}+1}$. Ultimately, the finite element equations simply recover Eqs. (4a) and (4b), but some ingenuity is needed to obtain A_h^{n}.

Thus, we now consider the determination of A_h^{n}. As shown in Fig. 2, due to the

rotation of the body, edge a–b is – at a time Δt s previously – at the position of edge a'–b'. As indicated in the figure, in most cases edge a'–b' is not defined at the edge of the finite element mesh. Consequently, we must use an interpolation scheme to estimate values of A_h^{n}. In accordance with the definition of the Nedelec element, we evaluate line integrals over all elements containing edge a'–b' as follows:

$$\begin{aligned} A_{\mathrm{a'-b'}}^{\mathrm{n}} l_{\mathrm{a'-b'}} &= \int_{\mathrm{a'}}^{\mathrm{b'}} A_h^{\mathrm{n}} \cdot s \,\mathrm{d}\, l \\ &= \int_{\mathrm{a'}}^{c'} A_h^{\mathrm{n}} \cdot s \,\mathrm{d}\, l + \int_{c'}^{d'} A_h^{\mathrm{n}} \cdot s \,\mathrm{d}\, l + \int_{d'}^{\mathrm{b'}} A_h^{\mathrm{n}} \cdot s \,\mathrm{d}\, l . \end{aligned} \tag{12}$$

Here, $A_{\mathrm{a'-b'}}^{\mathrm{n}}$ corresponds to the component of A_h^{n} on edge a'–b' and in the direction of that edge, $l_{\mathrm{a'–b'}}$ denotes the length of edge a'–b', and s denotes the unit tangent vector to each edge. By evaluating line integrals for each element in this way, we obtain a value for A_h^{n}.

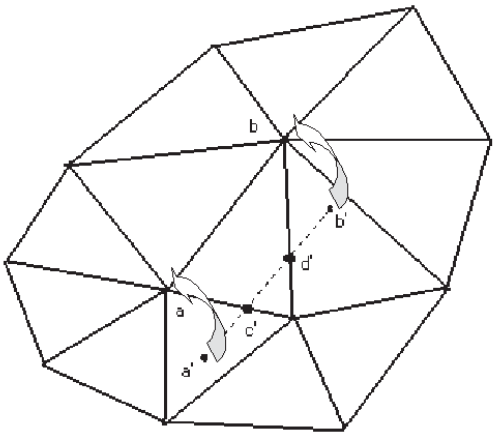


Fig. 2. Movement of an edge during Δt

3. Numerical Examples

3.1 Comparison with ADVENTURE_Magnetic for frequency response analysis

We confirm the validity of our procedure by comparing the results of a non-stationary eddy current analysis conducted using NEXST_Magnetic with the results of a frequency

response analysis conducted using ADVENTURE_Magnetic. By comparing with ADVENTURE_Magnetic, which has a proven track record for frequency response analysis, we can assess the accuracy of NEXST_Magnetic and obtain a thorough understanding of the real and imaginary parts of the frequency response analysis results obtained from ADVENTURE_Magnetic.

Frequency response analyses conducted using ADVENTURE_Magnetic return two types of results: real and imaginary parts. These agree with the results obtained using NEXST_Magnetic when the current density is maximum and 0, respectively.

Our computational model obtained eddy current analysis results for the infinite-length solenoidal coil in Fig. 3 [3]. The radius of the conductor was taken to be 0.1 m. The magnetic reluctivity v was taken to be $1/(4\pi)\times 10^7$ [m/H], while the conductivity σ of the conductor was taken to be 7.7×10^6 [S/m]. The frequency ω was $2\pi\times 60$ [rad/s]. The excitation current density J flowing through the coil was an AC current in the form $J_0 \cos \omega t$ [A/m^2] with $|J_0| = 50$ [A/m^2]. The NEXST_Magnetic analysis was conducted with all initial values A_h^0 set equal to 0 and Δt taken to be 1/(60×40). A total of 250 time steps were used, corresponding to 6.25 full cycles. The results of NEXST_Magnetic agree with the real parts of the ADVENTURE_Magnetic results at time steps 40, 80, 120, 160, 200, and 240 and they agree with the imaginary parts of the ADVENTURE_Magnetic results at time steps 50, 90, 130, 170, 210, 250. In this cake model, the magnetic flux density arises solely in the direction of the z axis, so we consider only the z component.

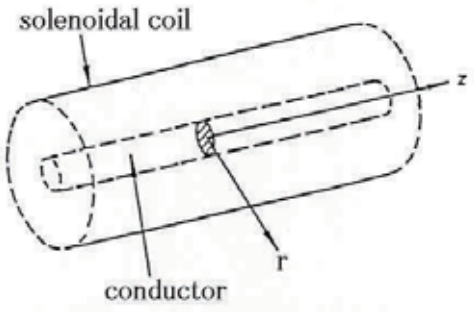


Fig.3. Solenoidal coil with unlimited length

Figure 4 compares the magnitude and direction of the magnetic flux density for points at which the excitation current density applied to the coil is a maximum (corresponding to the real part of the solution). To compute the relative error in each of these values, we compute the actual value using ADVENTURE_Magnetic and compare it with the results of NEXST_Magnetic.

Figure 4 shows the relative error for the magnetic flux density throughout the entire model. A relative error of between 1% and 2% is apparent at each time step.

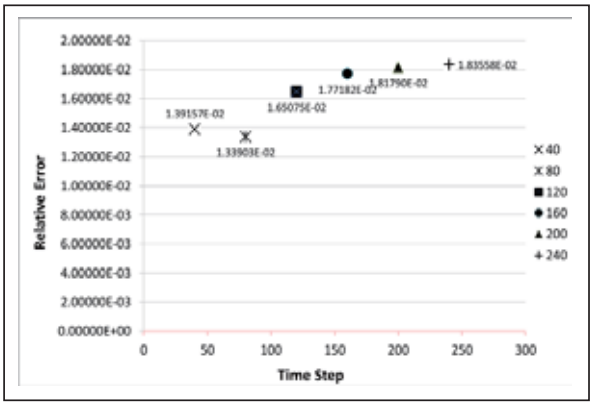


Fig.4. Magnetic flux densities in *z* direction
(Comparison of real part)

Figure 5 compares the magnitude and direction of the magnetic flux density for points at which the excitation current density applied to the coil is zero (corresponding to the imaginary part of the solution). As with the results shown in Fig. 4, there is a relative error of between 6% and 7% at each time step.

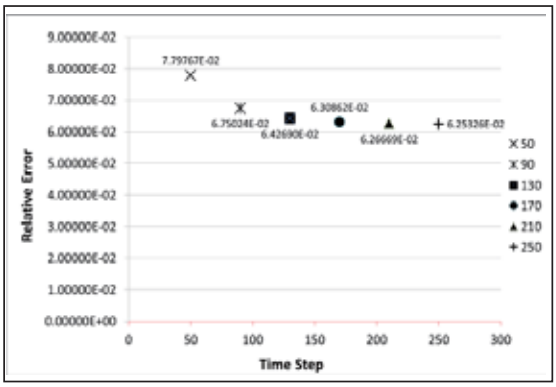


Fig. 5. Magnetic flux densities in *z* direction
(Comparison of imaginary part)

These results demonstrate that the results of transient analysis conducted using NEXST_Magnetic may be considered equivalent to the results of frequency response analysis.

3.2 A first analysis neglecting the B–H characteristics

As a numerical example, we consider the results of an eddy current analysis conducted for a rotating machine of the type shown in Fig. 6. This machine is composed of one fixed component and one rotating component and consists of 36 primary conductors and 44 secondary conductors. The model considered in this section does not account for the air gap between the fixed and rotating components. The magnetic reluctivity ν is $1/(4\pi)\times10^4\,[\mathrm{m/H}]$ for both the fixed and rotating components and $1/(4\pi)\times10^7\,[\mathrm{m/H}]$ for both the primary and secondary conductors. The conductivity σ of the conductor regions is $3.0\times10^7\,[\Omega/\mathrm{m}]$ and the frequency is $\omega=2\pi\times50[\mathrm{rad/s}]$. The excitation current density J flowing in the coil is an AC signal with the form $J_0\cos\omega t$ [A/m^2] where $|J_0|=6.61\times10^6$ [A/m^2]. It goes without saying that phase change of the excitation current density is suitably considered in our model.

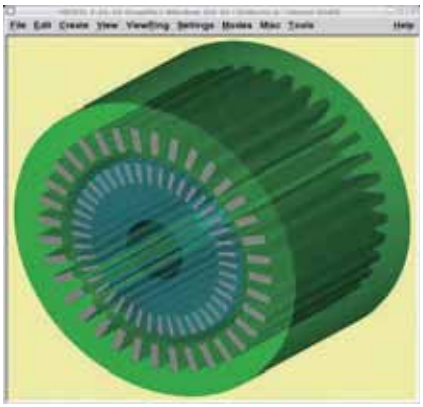


Fig. 6. A rotating machine

We considered computational models with 152,336 elements. The initial values A_h^0 were all set to zero. We chose a time step Δt equal to 1/(50×24) s and ran the computation for 72 time steps (three cycles as in Fig.7). The computation was executed on a desktop workstation having an Intel Core i7 920 CPU (2.66 GHz) and 24 GB of RAM.

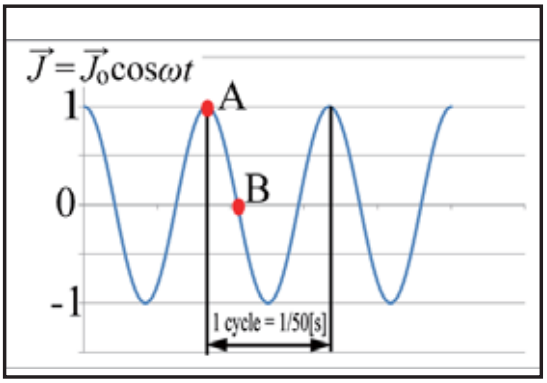


Fig. 7. Excitation current density

Figure 8 presents results for the magnetic flux density distribution on a cross-sectional slice through the center of the rotating machine at a time corresponding to point A in Fig. 7. Figure 9 presents similar results at a time corresponding to point B in Fig. 7. Comparison of these two plots reveals that the regions near the fixed primary conductor

in which the magnetic flux density is high are moving.

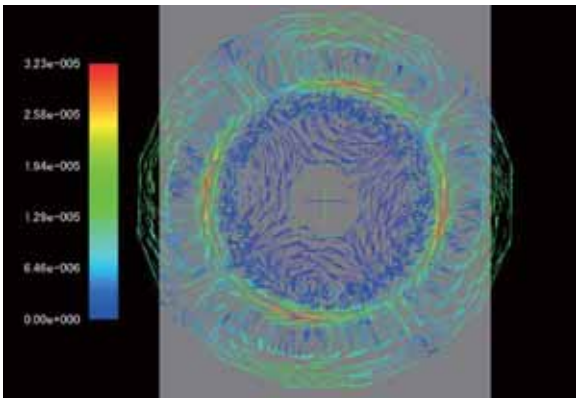


Fig. 8. Magnetic flux density for point A in Fig. 7

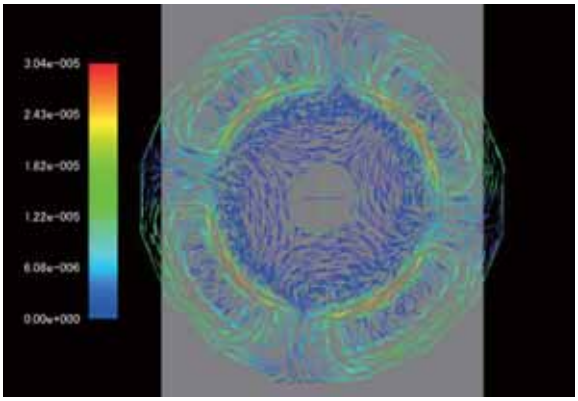


Fig. 9. Magnetic flux density for point B in Fig. 7

3.3 A second analysis accounting for the B–H characteristics

As in Section 3.2, we consider the results of an eddy current analysis conducted on a rotating machine. However, we now enhance our analysis by considering the *B–H* characteristics of the fixed and rotating components of the machine. Here, we use a weighting constant of value $w = 0.112$ and set the initial magnetic reluctivity v equal to the same values used in Section 3.2, namely, $1/(4\pi)\times 10^4\,[\mathrm{m/H}]$ for both the fixed and rotating components. The remaining analysis conditions are the same as those in Section 3.2.

In comparison with the computational results obtained without considering the *B–H* characteristics, Figs. 10 and 11 exhibit lower values in the magnetic flux density. In both figures, the maximum value becomes about 1/5. The color bars do not match those of Figs. 8 and 9.

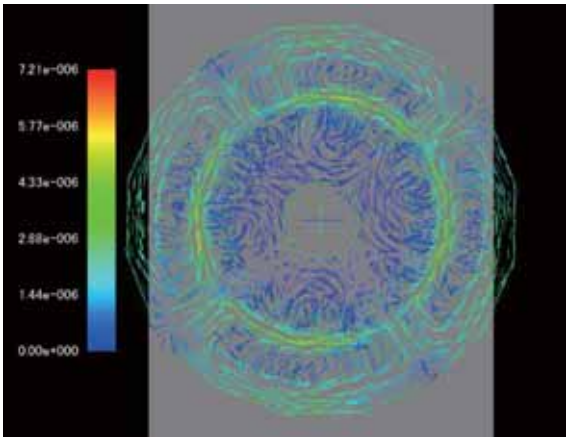


Fig. 10. Magnetic flux density for point A in Fig. 7

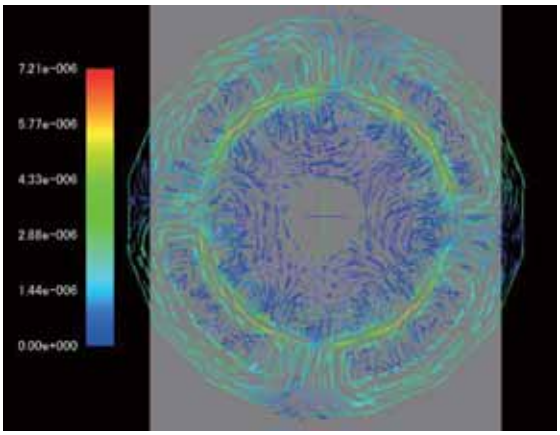


Fig. 11. Magnetic flux density for point B in Fig. 7

3.4 A computational model of a rotating machine with an air gap

The previous rotating machine model discussed in Section 3.2 does not account for the air gap. In addition, the shape of the coil is approximated as rectangular. We next consider a frequency response analysis (neglecting the *B–H* characteristics) conducted

using ADVENTURE_Magnetic on a new model, indicated in Fig. 12, based on the two points considered above.

Fig. 12. A rotating machine model with an air gap

We used the same computational conditions as in Section 3.2 and analyzed a model with 4,850,700 elements using a domain decomposition method. With a total of 80 parts, the computation time was approximately 12.25 h.

Figures 13 and 14 show the results. Comparison of these plots reveals that the regions of large magnetic flux density around the fixed primary conductor are moving, as in Section 3.2. It is also noted that there are regions with high magnetic flux densities near the air gap.

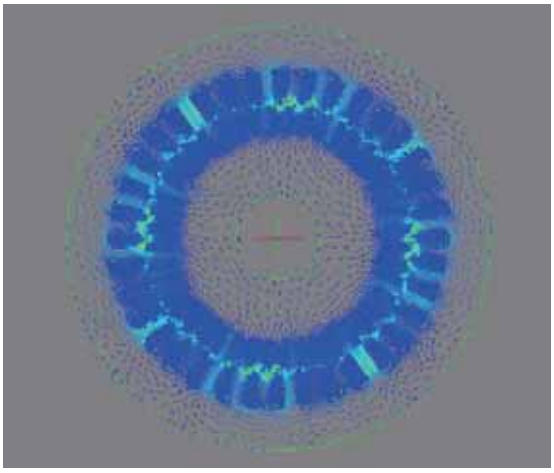


Fig. 13. Result of ADVENTURE_Magnetic (real part)

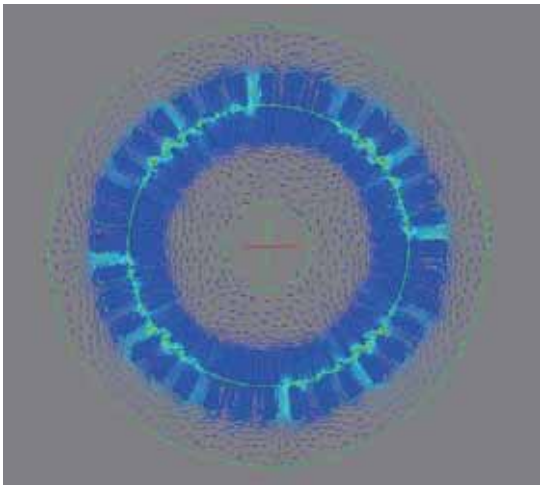


Fig. 14. Result of ADVENTURE_Magnetic
(imaginary part)

3.5 Consideration of rotation for a simplified rotating machine with an air gap

The previous rotating machine model discussed in Section 3.4 does not account for the movement of rotation. We finally consider a transient analysis considering the rotation (neglecting the *B–H* characteristics) conducted using NEXST_Magnetic on a simplified model, indicated in Fig. 15. The fixed component has 4 primary conductors and the rotating component also has 4 secondary conductors. The air gap is considered between

the fixed component and the rotating component. 20 kinds of meshes have been prepared for one cycle rotation and we are using different meshes for each time step.

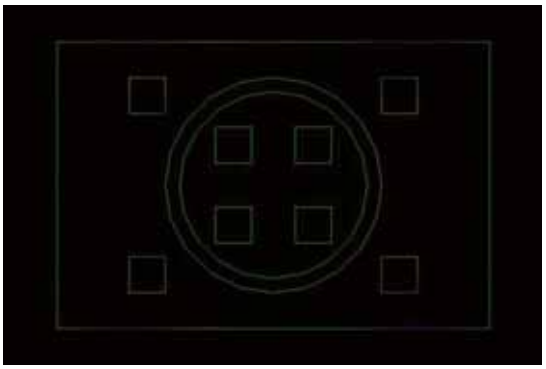


Fig. 15. A simplified rotating machine

Figure 16 shows the magnetic flux densities from the time step 1081 to 1084. As mentioned before, the previous position of the edge within the rotating component is searched for each time step. The conjugate gradient solver uses the previous step result for the initial value of each time step iteration.

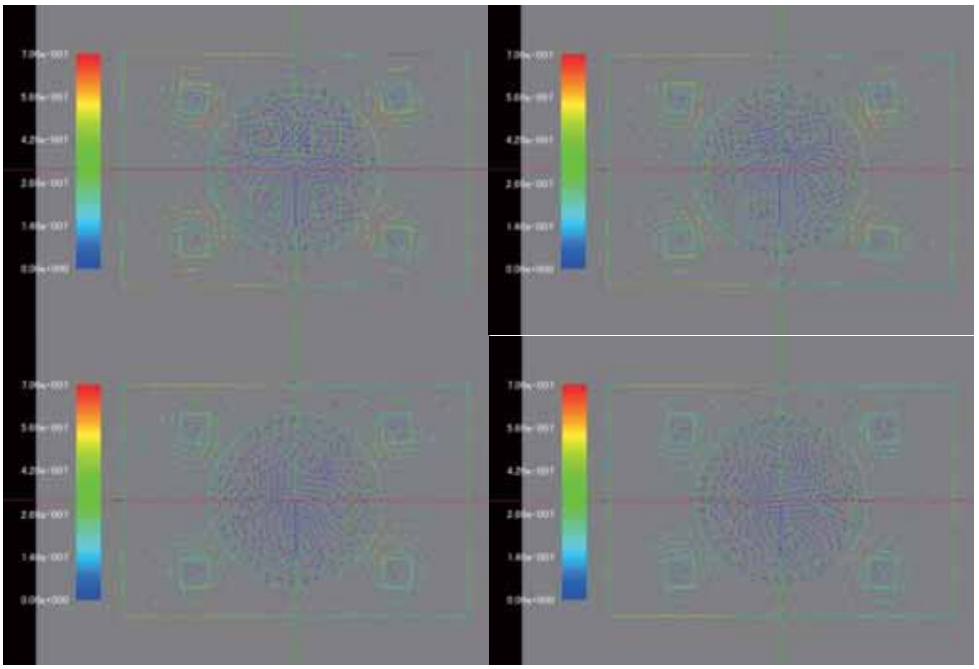


Fig. 16. The 4 step results of magnetic flux densities

4. Conclusions

We have used the $A-\phi$ method to analyze non-stationary eddy current problems. In near future, we plan to extend our computational models to larger-scale problems.

References

(1) Hiroshi Kanayama and Shin-ichiro Sugimoto : "Effectiveness of $A-\phi$ method in a parallel computing with an iterative domain decomposition method", IEEE TRANSACTIONS ON MAGNETICS, Vol.42, No.4, pp.539-542 (2006)

(2) ADVENTURE Project home page "http://adventure.sys.t.u-tokyo.ac.jp/jp/"

(3) Hiroshi Kanayama, Daisuke Tagami and Shin-ichiro Sugimoto : "A finite element computation of unsteady eddy current problems using the $A-\phi$ method", Proceedings of the 8th JSCES Conference, Vol.8, pp.687-690 (2003) in Japanese

(4) NEXST_Magnetic, Version 1.0 (2005) "http://www.ciss.iis.u-tokyo.ac.jp/fsis/index.html"

(5) Hiroshi Kanayama, Ryuji Shioya, Daisuke Tagami and Hongjie Zheng : "A numerical procedure for 3-D nonlinear magnetostatic problems using the magnetic vector potential", THEORETICAL AND APPLIED MECHANICS, Vol.50, pp.411-418 (2001)

(6) Shin-ichiro Sugimoto and Hiroshi Kanayama: "Eddy current computation with moving conductors", IEEJ SA-05-9, RM-05-9, pp.47-50 (2003) in Japanese

Large Scale Tsunami Run-up Simulation by a Hybrid-parallel SPH

M. Asai[1], K. Fujimoto[1], and M. Isshiki[2]

1 Kyushu University, 744 Motooka, Nishi-ku, Fukuoka, 819-0395 JAPAN
{asai, fujimoto}@doc.kyushu-u.ac.jp

2 Ehime University, 3 Bunkyo, Matsuyama, 790-0826, JAPAN
isshiki@cs.ehime-u.ac.jp

1 Introduction

The The huge tsunami caused by the Great Tohoku Earthquake devastated the coastal area on March 11, 2011. After the Tohoku Earthquake, the Japanese government and each local government has been discussing about the next generation of disaster prevention and mitigation method against millennium Tsunami. In order to construct safe and secure coastal structures, it is necessary to generate an accurate simulator, which can predict not only the flood area but also the structural damage. Thus, numerical simulation of free surface fluid flows will be essential to predict the flood damage and to aid the design of effective flood defense structures. Therefore, we are developing a 3-D tsunami simulator based on the Incompressible Smoothed Particle Hydrodynamics (ISPH)[1]. The features of our proposed simulator are stabilization of ISPH with a modified source term in the pressure Poisson equation[2]. The source term in pressure Poisson equation (PPE) for ISPH is not unique, it have several formulations in the literature as Lee et al [3], Khayyer et al[4] and Shao et. al[5]. The source term is derived as a function of density variation and velocity divergence condition. The former formulation with density variation can keep an uniform particle distribution, although evaluated pressure include high unrealistic fluctuation. On the other hand, the formulation of the divergence-free condition evaluates much

smoother pressure distribution, but density errors may occur due to particle clustering. Then, modified schemes have been proposed to satisfy the above two conditions; density invariant and divergence free condition and the relaxation coefficient is multiply in term of density invariant for smoothing the resultant pressure. Recently, in the framework of MPS[6], there is a trend to introduce a higher order source term in the PPE. Kondo and Koshizuka[7] proposed a new formulation with a source term composed by three parts; one is main part and another two terms related to error-compensating parts. Tanaka and Masunaga[8] introduced a similar high order source term with two components incorporated with quasi-compressibility. In this paper, the similar approach is implemented with the ISPH for their stabilization and for smoothed pressure evaluation. The main target of our simulation is to represent the collapse behavior of the sea-wall by the tsunami. Tsunami run-up simulation by using a hybrid parallel SPH is the first step for this purpose to tsunami inundated zone with velocity distribution.

2 Stabilized ISPH

In this section, a stabilized ISPH in previous work by the author[2], which include a modified source term in the pressure Poisson equation, for incompressible flow is summarized. In addition, a conventional turbulence model ‘ Smagorinsky model ’ is introduced into ISPH.

2.1 Governing equation

The mass and momentum equations of the flows are given as:

$$\frac{\mathrm{D}\rho}{\mathrm{Dt}} + \rho\nabla\cdot\boldsymbol{u} = 0 \tag{1}$$

$$\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{Dt}} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\boldsymbol{u} + \frac{1}{\rho}\nabla\cdot\boldsymbol{\tau} + \boldsymbol{F} \tag{2}$$

where ρ and ν are density and kinematic viscosity of fluid, $\boldsymbol{u}$ and p are the velocity and pressure vectors of fluid respectively. $\boldsymbol{F}$ is external force, and t indicates time. In the most general incompressible flow approach, the density is assumed by a constant value with its initial value.

2.2 Pressure Poisson equation

The incompressible SPH method is solving a discretized pressure Poisson equation at every time step to evaluate the current pressure value. In the sense of physical observation, physical density should keep its initial value for incompressible flow. However, during numerical simulation, the ‘ particle ’ density may change slightly from the initial value because the particle density is strongly dependent on particle locations in the SPH method. If the particle distribution can keep almost uniformity, the difference between ‘ physical ’ and ‘ particle ’ density may be vanishingly small. In other words, accurate SPH results in incompressible flow need to keep the uniform particle distribution. For this purpose, the different source term in pressure Poisson equation can be derived using the ‘ particle ’ density. The SPH interpolations are introduced into the original mass conservation law before the perfect compressibility condition is applied.

$$\left\langle \nabla^2 p_i^{n+1} \right\rangle = \frac{\rho^0}{\Delta t} \left\langle \nabla \cdot \boldsymbol{u}_i^* \right\rangle \tag{3}$$

Alternatively, the different source term of the pressure Poisson equation can be re-formulated with an weakly compressible condition as follows;

$$\left\langle \nabla^2 p_i^{n+1} \right\rangle = \frac{\rho^0}{\Delta t} \left\langle \nabla \cdot \boldsymbol{u}_i^* \right\rangle + \alpha \frac{1}{\rho^0} \frac{\left\langle \rho_i^{n+1} \right\rangle - \left\langle \rho_i^* \right\rangle}{\Delta t} \tag{4}$$

where α is relaxation coefficient, $\boldsymbol{u}_i^*$ is temporal velocity and triangle bracket $< \cdot >$ means SPH approximation with the kernel function. Figure 1 and 2 show the main flowchart of the original ISPH and the stabilized ISPH with the relaxation coefficient α, respectively. Note that this relaxation coefficient is strongly dependent on the time increment and the particle resolution. Then, the reasonable value can be estimated by the simple hydrostatic pressure test using the same settings on its time increment and the resolution.

2.3 Large Eddy Simulation by the Smagorinsky model

The SPS stress in the particle simulation needs to be modeled for introducing an effect of the eddy viscosity. In this thesis, a large eddy simulation approach [10] is used for modeling the SPS stress in incompressible fluid as:

$$\frac{\tau^{IJ}}{\rho^0} = 2\nu_T S^{IJ} - \frac{2}{3} K \delta^{IJ}, \tag{5}$$

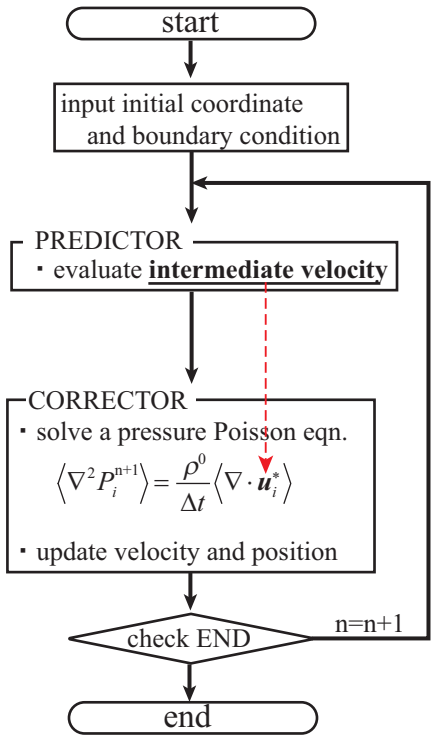


Figure 1: Original ISPH flowchart

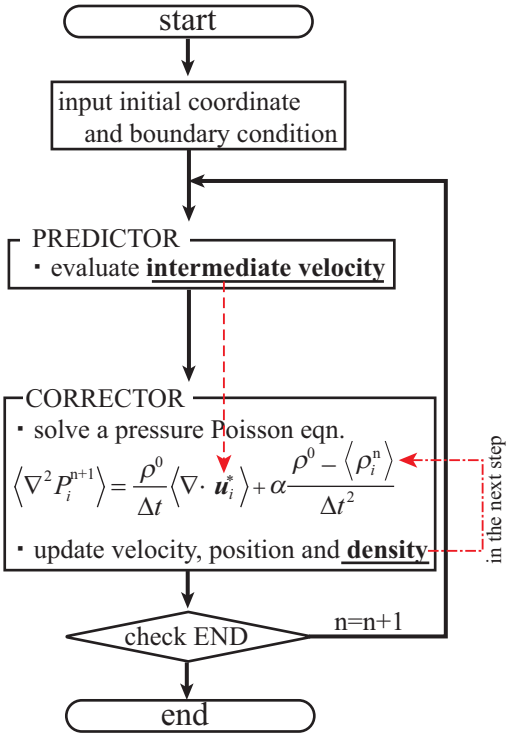


Figure 2: Stabilized ISPH flowchart

where, ν_T and K are are the eddy viscosity and the kinetic energy, respectively (in contrast to indices i and j referring to particles, letters I and J here denote spatial coordinates). S^{IJ} indicates the strain rate of the mean flow. It is assumed that, the eddy viscosity is modeled by the static Smagorinsky model as:

$$\nu_T = (C_s\Delta)^2|\mathbf{S}|, \tag{6}$$

where, $C_s = 0.2$ is the Smagorinsky constant (taken as the analytical value in this thesis), Δ is constant and it taken as smoothing compact support in this thesis. And, the local strain rate is calculated as, $|\mathbf{S}| = (2S^{IJ}S^{IJ})^{1/2}$. Where, the components of the mean rate of strain tensor:

$$S^{IJ} = \frac{1}{2}\Big(\frac{\partial u^I}{\partial x^J} + \frac{\partial u^J}{\partial x^I}\Big), \tag{7}$$

The local strain rate can be calculated in the SPH formulation corresponding to [**?**] as follows:

$$S_i^2 = \frac{1}{2}\sum_j m_j \frac{\rho_i + \rho_j}{\rho_i\rho_j}\frac{|\mathbf{u}_{ij}^2|}{r_{ij}^2}\mathbf{r}_{ij}\cdot\nabla W_{ij}. \tag{8}$$

The turbulent kinetic energy K incorporated in the pressure term by defining $P_E = P + \frac{2}{3}\rho K$. As there is no difference in the numerical procedures between P_E and P since physically ($K << P$), from here P will be used instead of P_E. Then, by introducing Eq. (5) into the momentum equation, yields a momentum equation as[9]:

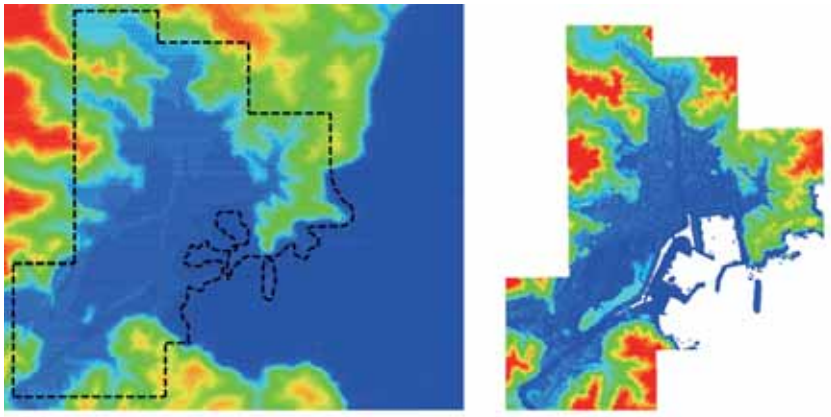
$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho^o}\nabla P + \nu_e\nabla^2\mathbf{u} + \mathbf{g}. \tag{9}$$

with

$$< \nu_e\nabla^2\mathbf{u} >= \sum_j m_j\Big(\frac{\rho_i\nu_{e,i} + \rho_j\nu_{e,i}}{\rho_i\rho_j}\frac{\mathbf{r}_{ij}\cdot\nabla W(|r_i - r_j|, h)}{r_{ij}^2 + \eta^2}\Big)\mathbf{u}_{ij} \tag{10}$$

3 Geometrical modeling using Aerial Survey and Bathymetry

Here, a multi-resolution geometrical modeling is performed for a highly resolved geometrical map for tsunami simulation. As a first step, Digital Elevation Map (DEM) with 5m structure grid information is utilized for a low



(a) 5m resolution by DEM (b) 1m resolution by Aerial Survey data

Figure 3: Multi-resolution geometrical modeling

resolution geometrical modeling as shown in Fig.3(a). The DEM doesn ' t include the information of structures including sea-walls, houses and building. The deficient information of the geometry has recovered by aerial survey data as points with 1m resolution. The domain bounded by a dashed line in Fig.3(a) is replaced by the aerial survey data shown in Fig.3(b). These multi-resolution point data is converted to the Standard Triangulated Language (STL), which is one of the standard 3D-CAD formats. After defining the external surfaces by STL CAD format, inside region is occupied by particles. In this phase, we have used commercial software ' AVS-Express ', which is a general pre- and post- processer using meshes and particles for numerical simulation. These sequential operations can be easily conducted only with a few commands even if geography and shape of structure is complicated. This is one of the advantages to use particle type simulation. Finally, the particles are located with 4m resolution, and the total number of particles is about 12 millions.

4 Tsunami Simulation at Taro city

Tsunami run-up behavior is simulated with the real geometrical map generated from a multi-resolution aerial survey data. The tsunami is inputted

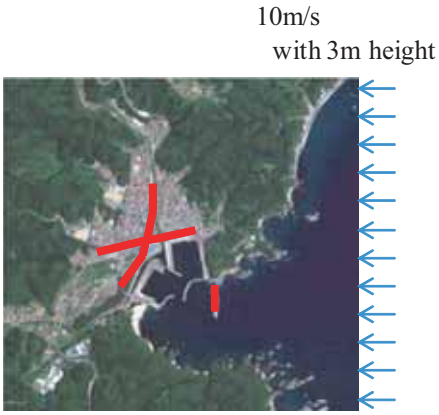


Figure 4: Tsunami simulation model at Taro

on the right boundary edge in our model shown in Fig.4, and the input of tsunami is assumed from the investigation of Tohoku earthquake. The height is 3m, and its velocity is 10m/s. The constant tsunami is given during our tsunami simulation for 3 minutes in the real time. Fig 5 shows the snapshot of tsunami simulation after 1m50s and 2m25s. In the figures, contour color indicates the velocity magnitude in the horizontal direction of the figure. Fig. 6 shows a particle rendering images with the Gaussian filter to represent the foaming waves. These run-up behaviors show a good agreement with the damage investigation reports of the Great Tohoku Earthquake. The computation was carried out using the facilities at Research Institute for Information Technology at Kyushu University. It spends about 3days for the 3minutes behaviors using 32 nodes which have 2 CPU with 8 cores. Our developed tool has been utilized the hybrid parallel computation with OpenMP and MPI.

5 Conclusions

A stabilized incompressible smoothed particle hydrodynamics is proposed to simulate free surface flow. The modification is appeared in the source term of pressure Poisson equation, and the idea is similar to the recent development in Moving Particles Semi-implicit method (MPS). For tsunami simulation in coastal area, we have developed a SPH analysis tools with semi-automatic

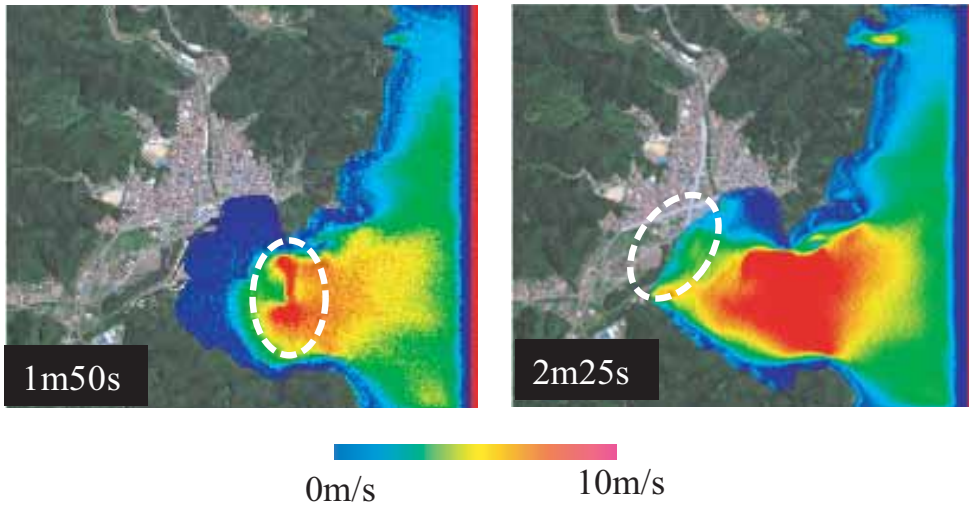


Figure 5: Tsunami simulation results with horizontal velocity contour

Figure 6: Flowchart of the stabilized ISPH

geometrical modeling for particle type simulation. The tool can easily generate the particle data from numerical maps through the 3D CAD format. In our future work, we strongly desire a parallel computational efficiency for the large scale computations. In addition, we are going to simulate coastal structures which receive impulsive fluid load due to Tsunami or storm wave. Therefore, the global tsunami run-up simulation should be reasonably connected to a local zooming analysis with structures.

Acknowledgment

This work is partially supported by "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures" in Japan.

References

[1] S.J. Cummins and M. Rudman, An SPH projection method, *Journal Computational Physics*, Vol.152(2), pp.584-607, 1999

[2] M. Asai, AM. Aly, Y. Sonoda and Y. Sakai, A stabilized incompressible SPH method by relaxing the density invariance condition, *Int.l J. for Applied Mathematics*, Volume 2012 (2012), Article ID 139583

[3] E. S. Lee, C. Moulinec, R. Xu, D. Violeau, D. Laurence and P. Stansby, Comparisons of weakly compressible and truly incompressible algorithms for the SPH mesh free particle method, *Journal of Computational Physics*, Vol.227(18), pp.8417-8436, 2008

[4] A. Khayyer, H. Gotoh and S. Shao, Corrected incompressible SPH method for urate water-surface tracking in breaking waves, *Coastal Engineering*, Vol.55, pp. 236-250, 2008

[5] S. Shao, E.Y.M. Lo, Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface, *Advances in Water Resources*, Vol.26, pp.787-800, 2003

[6] S. Koshizuka and Y. Oka, Moving-Particle Semi-implicit method for fragmentation of incompressible fluid,*Nucl. Sci. Phys. Comm.*, Vol. 48, pp.421-434, 1996

[7] M. Kondo and S. Koshizuka, Improvement of stability in moving particle semi-implicit method, *Int. J. Numer. Meth. Fluids*, Vol. 65, pp.638-654, 2011

[8] M. Tanaka and T. Masunaga, Stabilization and smoothing of pressure in MPS method by Quasi-Compressibility, *Journal of Computational Physics*, Vol. 229, pp. 4279-4290, 2010

[9] J.P. Morris, P.J. Fox and Y. Zhu, Modeling Low Reynolds Number Incompressible Flows Using SPH. *Journal of Computational Physics*, Vol. 136, pp. 214-226, 1997

[10] H. Gotoh, S. Shao, and T. Memita, SPH—LES model for numerical investigation of wave interaction with partially immersed breakwater. *Coast. Eng. J. JSCE*, Vol. 46(1), pp.39-63, 2001

Promotion of Open-source ADVENTURE by Insight

Miyoshi, A.[1]

1 Insight, Inc., 5-29-12-407 Hongo, Bunkyo, Tokyo, 113-0033, JAPAN
amiyoshi@meshman.jp

1 Introduction

The official project name of the ADVENTURE [1] is "Development of Computational Mechanics System for Large Scale Analysis and Design". It is a free and open source finite element analysis system and is a distributed parallel processing system by the domain decomposition method for realizing a super large-scale computation. ADVENTURE is an integrated system and its functions include a CAD interface, automatic mesh generator, boundary condition attachment, designation of material properties, domain decomposition, obtainment of solutions, and result visualization. Since ADVENTURE is trying to contribute to design work in manufacturing industry, its objective is to become a tool for analyzing problems the manufacturing industry has.

Insight, inc. is involved in development of ADVENTURE and promotion of its utilisation since its foundation in 1999. This paper introduces the ADVENTURE and the activities of Insight.

2 Overview of ADVENTURE

ADVENTURE's complete name is "ADVanced ENgineering analysis Tool for Ultra large REal world". It is effective to increase the size of models and the speed of computation dramatically to obtain an accurate solution of complicated analysis problems in design of artifacts.

From this viewpoint, a project for "Development of Computational Mechanics System for Large Scale Analysis and Design", so-called ADVENTURE was executed for the purpose of becoming a de-facto standard in twenty-first century's CAE software for four and a half year from August, 1997 to March, 2002 by a research group led by members of universities such as the University of Tokyo. The project is one of five projects in the "Computational Science and Engineering" field selected by the "Research for the Future (RFTF)" program that is sponsored by the Japan Society for the Promotion of Science (JSPS). In the project, research and development of the

general-purpose parallel computational mechanics system, ADVENTURE was performed. Its capability includes modelling of the whole and detailed artefacts by means of ten million to one hundred million-class large-scale mesh, which could not be handled so far. Mechanical analysis such as deformation of solid materials, thermal analysis and fluid analysis and visualisation of this large-scale mesh are possible under various parallel and distributed computer environment. Design optimisation is also included in its capability.

ADVENTURE system's initial release was beta and its source code became downloadable free of charge on Dec. 1, 2000 at the project's home page. The initial release consisted of seven basic modules. On Mar. 1, 2002, the seven basic modules were updated to version 1 and beta version source codes of eleven new modules were released for download at the project's home page. Moreover, in May, 2001, ADVENTURECluster system, a first commercial version of the ADVENTURE system was released and was introduced into laboratories, universities, automobile manufacturers etc., which means ADVNTURE rapidly prevailed in the industrial and academic community.

In April of 2002, ADVENTURE was repositioned as an independent open-source software development project, and efforts have been made to continue the system's maintenance, function enhancement and promotion to industry. Also, because of its leading edge features, it was used from 2002 to 2005 in the "Frontier Simulation Software for Industrial Science", which is an industry and academia collaboration project led by the Institute of Industrial Science, the University of Tokyo.

From Apr. 2002, to Mar. 2006, JAEA's (Japan Atomic Energy Agency) Centre for Computational Science and e-Systems has performed a collaborative research with the ADVENTURE project, i.e. "To Make ADVENTURE System a Community Software of ITBL (IT Based Laboratory)." was executed targeting application of ADVENTURE to grid computing.

From Dec. 2002 to Sep. 2008, in "Development of Next Generation Computational Solid Mechanics Simulator for Virtual Verification Testing" project lead by Kyushu University, ADVENTURE was applied to the Earth Simulator, and a whole nuclear reactor seismic response simulation was performed.

From Apr. 2009 to present, "Development of Next Generation Computational Fracture Mechanics Simulator for a Safe and Secure Sustainable Society" project has been led by Toyo University and crack extension analyses were made on the so-called

Earth Simulator 2.

From 2004 to 2007, an industry and academic collaborative project named "Research and Development of an Integrated and Optimised Maintenance of Actual Piping by Probabilistic Fracture Mechanics" was funded by the Innovative and Practical Nuclear Research and Development Grant Program of the Agency for Natural Resources and Energy, Ministry of Economy, Trade and Industry (METI). ADVENTURE was used as an engine for stress analyses in this project led by a private manufacturer, the University of Tokyo and a private research laboratory. Also, a “Decision Making Support System Based on the Partitioned Type Multi Dimensional Visualization Method”, which had been developed during project execution, was released as ADVENTURE_DecisionMaker.

From Jul. 2005 to Mar. 2008, in the Revolutionary Simulation Software Project of the Ministry of Education, Culture, Sports, Science and Technology (MEXT), ADVENTURE was used in “Innovative General-Purpose Coupled Analysis System”.

From Oct. 2008 to Mar. 2013, in the Research and Development of Innovative Simulation Software Project of the MEXT, ADVENTURE was evolved in “Large Scale Assembly, Structural Correspondence, Multi Dynamics Simulator”.

From Oct. 2008 to Mar. 2013, ADVENTURE is being used and developed as a dynamic structural analysis tool in the “Simulation for Predicting Quake-Proof Capability of Nuclear Power Plants” project in High Performance Computing for Multi-Scale and Multi-Physics Phenomena Project of CREST (Core Research for Evolutional Science and Technology) of the JST (Japan Science and Technology Agency). It achieved a 40 percent peak performance ratio in the K computer that was ranked as number one in performance in 2011.

From Apr. 2002 to present, application research has been performed by means of ADVENTURE on practical problems in various industry areas (automobile, power, steal, heavy industry and information equipment).

From Apr. 2003 to present, ADVENTURE has been used as software for the auxiliary training of the Certification Program of Computational Mechanics Engineers held by the Japan Society of Mechanical Engineers.

The features of ADVENTURE include the following:

i) Whole model analyses with a few M to a few hundred M DOF meshes are possible.

ii) Can achieve a high parallel efficiency of more than 90 percent even on a massively parallel computer environment of one thousand processors.

iii) Highly portable to environments such as single PCs, PC clusters, massively parallel computers like the Earth Simulator and heterogeneous environments like JAEA'S ITBL.

iv) License-free and open-source.

As for a commercial version, Allied Engineering's ADVENTURECluster was selected as a Gordon Bell Award finalist in IEEE/ACM SC2006. In 2008, it won the Award of the Japan Society of Mechanical Engineers in Technical Division. In 2009, it won the Science and Technology Prize of the Award of the Minister of MEXT in the Science and Technology Area. ADVENTURE's mesh generation technology was utilized in Kubota's domestic pre-post processor KSWAD. Also, ADVENTURE's parallel solver's BDD (Balancing Domain Decomposition) technology was introduced to the domestic commercial code FINASTR by the CTC (ITOCHU Techno-Solutions Corporation).

v) Expandability and maintainability: A module structure was adopted. IO was standardized. Commodity technology is used.

Figure 1 shows the module structure of the ADVENTURE.

The ADVENTURE system consists of mesh generating modules, a module for IO of dedicated binary format, a module for setting boundary conditions and material properties, a module for domain decompose mesh to generate distributed data, analysis solver modules, visualization modules, utility modules and other modules.

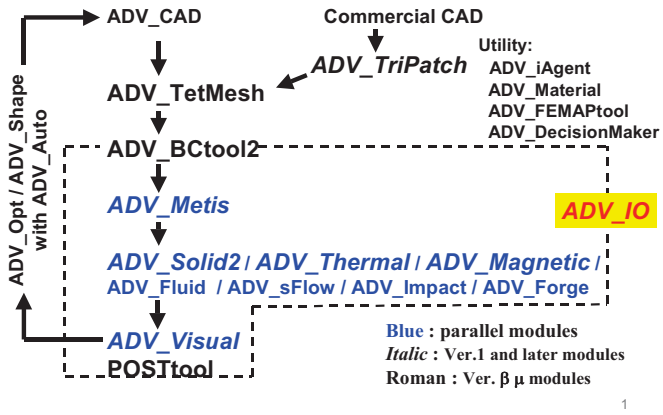


Fig.1: Currently released twenty-two modules (as of Oct. 11, 2012)

2.1 Mesh generating modules

Mesh generating modules consist of three modules, i.e. ADVENTURE_CAD, ADVENTURE_TriPatch (hereafter called as TriPatch, other modules will be called similarly), and TetMesh. The CAD is a module for generating linear triangle surface mesh from geometry description files with an extension of "gm3d". Its functions include extruding two dimensional sketch figures, making bodies of revolution and performing Boolean operations of bodies.

TriPatch has a capability to generate linear triangle surface mesh and face information from IGES format CAD files. The IGES data TriPatch can handle include the entities that are generated by I-DEAS etc. But the entity types TriPatch can handle are limited. It also has a function of handling multiple parts by merging two triangle surface meshes to one.

Neither CAD nor TriPatch uses the IO library described in section 2.2.

2.2 A module for IO of dedicated binary format

The IO module defines the binary IO format (hereafter called as ADVENTURE file format) used by ADVENTURE's main modules except for mesh generating modules and utility modules and provides an IO library.

2.3 A module for setting boundary conditions and material properties

The basic functions of BCtool is to make boundary condition definition files, to make material properties definition files and by integrating these files and mesh related files to generate a binary file that completely describes an analysis model. BCtool also has a three dimensional GUI tool for interactively defining boundary conditions. The analysis solver modules that BCtool supports are Solid and Thermal. For Solid module, it also has a capability to designate time history data such as loads of dynamic analysis and load history data of non-linear analysis.

BCtool has several other functions of generating MPC (Multiple Point Constraints) data although the Solid solver does not handle MPC currently. It has a tool capable of converting a text data file of a dedicated format to a binary data file of the ADVENTURE_IO format.

2.4 A module for domain decomposition

Metis is a tool of domain decomposition for ADVENTURE. Its features are as follows:

i) It requires a mesh data written in the ADVENTURE file format as input and it outputs single stage or hierarchically domain decomposed input data for solvers. The output data are also written in the ADVENTURE file format.
ii) It uses METIS and ParMETIS, which are the graph partitioning libraries developed by the University of Minnesota.
iii) It is capable of parallel processing.

2.5 Analysis solver modules

2.5.1 Solid2

Solid 2, i.e. Solid version 2 is a solver module for static and dynamic stress analysis. As for static analysis, elasto-plastic analysis is possible and as for dynamic analysis, linear elastic transient analysis by direct integration is possible. It uses the IO library.

2.5.2 Thermal

Thermal is a solver module for static and transient heat transfer analysis. Four types of boundary conditions, i.e. temperature, heat flux, heat transfer and radiation can be designated. It supports linear and quadratic tetrahedral elements. The BDD (Balancing Domain Decomposition) can be used as a linear equation solver. It uses the IO library.

2.5.3 Magnetic

Magnetic is a solver module for non-linear magnetostatic problems and eddy current problems. The hierarchical domain decomposition method makes it possible to perform load distributed parallel processing. It uses the IO library.

2.5.4 sFlow

sFlow's development started as a static fluid analysis solver module, but currently, it also supports transient fluid analysis.

It can solve static and transient thermal convection problems with linear tetrahedral elements. The BiCGSTAB, GP-BiCG and BiCGSTAB2 are provided as iterative solvers for linear equations. The hierarchical domain decomposition method makes it possible

to perform load distributed parallel processing. It uses the IO library.

2.5.5 Fluid

Incompressible transient thermal convection and fluid analysis is possible with linear hexahedral elements and incompressible transient fluid analysis is possible with linear tetrahedral elements. The BiCGSTAB, GP-BiCG, BiCGSTAB2 and GMRES (m) are provided as iterative solvers for linear equations. A non-hierarchical domain decomposition method is used and distributed parallel processing is possible. It uses the IO library.

2.5.6 Other solver modules

Although details are omitted, Impact for impact analyses and Forge for metal forging analyses are provided. Both of them use the IO library.

2.6 Visualization modules

POSTtool has a capability to visualize analysis results of the Solid. It has the following functions:

i) Visualize surface geometry.
ii) Change of viewpoint (translation, rotation and zoom).
iii) Visualize a deformed shape of a model.
iv) Visualize computation results as contour plot. The range of values can be changed.
v) A surface node can be picked and its computation results can be shown.
vi) Visualize a model cut away by a designated plane.
vii) Visualized figure can be saved as a png file.

It uses the IO library.

2.7 Utility modules

2.7.1 iAgent

iAgent is a software module that leads a user to a series of analysis processing operations and is equipped with a GUI. It can be used on a distributed parallel environment. It coordinates with the TriPatch, TetMesh, BCtool Version 1.02, Solid Version 1.21, Thermal and Visual Version 1.0. It does not use the IO library.

2.7.2 on_Windows

The on_Windows consists of the iAgent and almost all the coordinating modules ported to Windows except for the Thermal's all functions and the Solid's non-linear functions. Also, the function to set boundary conditions and the visualization function are developed separately from the Linux equivalents, and their implementations are different from the original iAgent to a large degree. It does not support parallel computation. It uses the IO library through the coordinating modules.

2.7.3 FEMAPtool

FEMAPtool is a converter from/to input/output files of a general purpose pre-post processor FEMAP. It extracts a mesh from a Nastran's bulk data formatted file and output as a text file. Also, it converts an analysis result text file to a FEMAP neutral file. It does not use the IO library.

2.7.4 Other utility modules

Material is a module for identifying parameters of inelastic material models. Currently, there is no interface to ADVENTURE'S other modules.

DecisionMaker is a GUI module in which optimization results and other result are multi-dimensionally visualized and a user's decision making is supported.

2.8 Other modules

Auto is a module that consists of commands for automatic analyses. Since each command is written in plain C language, researchers can modify its source code for oneself to extend it to an original tool for use of ADVENTURE. The Auto commands include those with a capability of 3D visualization and they are especially called as AutoGL.

Opt is an optimization module. SQP (Sequential Quadratic Programming), Sub-modules of RGA (Real-code Genetic Algorithm), and the Center Neighborhood Crossover are provided. The Auto and the Solid must be used at the same time.

Shape is an optimization module that has a non-parametric geometry optimization function and a topology optimization function. The Solid and the Metis should be used at the same time.

2.9 Data flow

Figure 2 shows a data flow diagram when the ADVENTURE_CAD is used for geometry definition and the Solid is used as a solver.

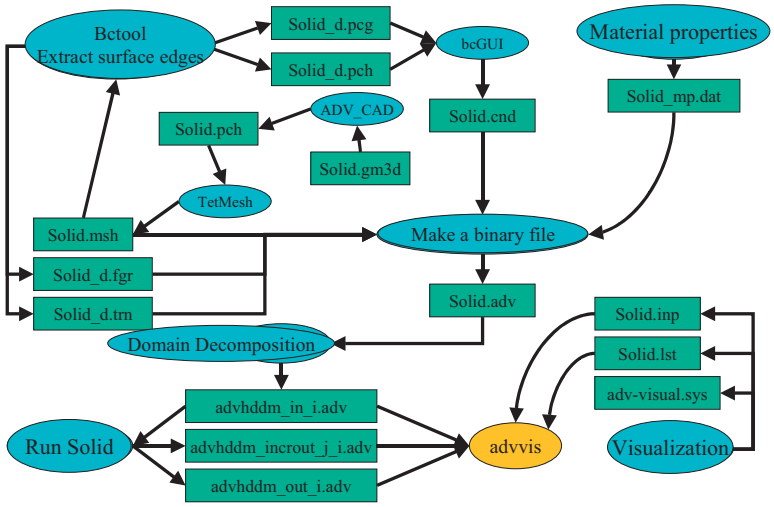


Fig. 2 Data flow with the Solid and the CAD

3 Calculation example of ADVENTURE

A calculation example in the project of "Simulation for Predicting Quake-Proof Capability of Nuclear Power Plants" which was addressed in Chapter 2 is presented. Professor Yoshimura of the Graduate School of Engineering, the University of Tokyo performed an analysis of seismic response of soil, building, pressure vessels piping, and other components with interaction of coolant water with members of ADVENTURE development [2].

The following functions and performance required for the above analysis was added to the Solid.

i) Non-linear material modeling for steel, concrete and soil.
ii) A whole model of assembled structures with MPC (Multiple-Point Constraints)
iii) Fluid-structure interaction by partitioned coupling technique (ADVENTURE_Coupler).
iv) Performance tuning towards peta-FLOPS computer and the K-computer.

As a result, one peta FLOPS calculation performance was achieved in a sustained manner on K-computer with a 100 million DOF model to a ten billion DOF model.

The analysis was performed in the following order. Step one is a wave propagation analysis in crust, soil and nuclear power plant building model by the MMA (Micro-Macro Analysis). Step two is to subtract time histories of displacements at nodes on building surface from MMA results. Step three is to perform seismic response analysis of the nuclear power plant with the displacement histories provided as Dirichlet boundary conditions.

The geometry model is outlined in Fig. 3 and Fig. 4. It includes the reactor building, the containment vessel, the reactor pressure vessel and the internals. The model is made through collaboration with TEPCO, Allied Eng., and Hitachi GE.

The stress contour of the BWR (Boiling Water Reactor) building caused by an inland earthquake wave is shown in Fig. 5.

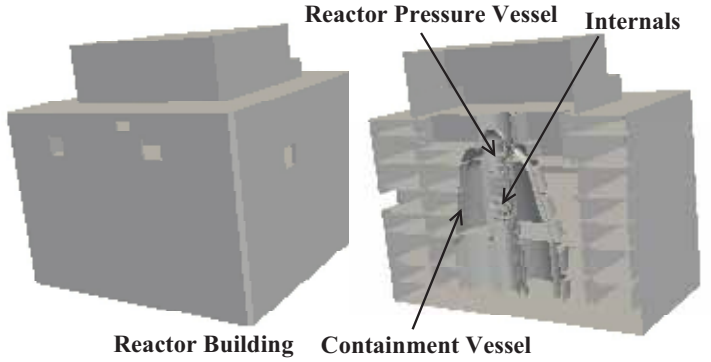


Fig. 3 Geometry model of a BWR (Boiling Water Reactor) building

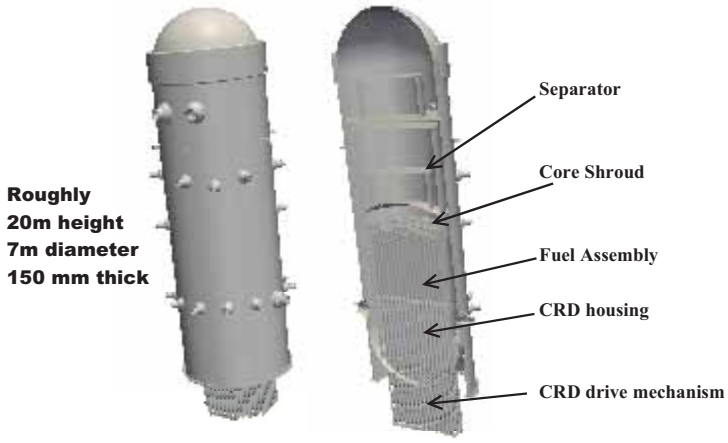


Fig. 4 Geometry model of a BWR pressure vessel and internals

4 Latest status of ADVENTURE

On Aug. 10, 2012, Solid Ver. 2.0, BCtool Ver. 2.0 and POSTtool Ver. 1.0 were released. The Solid Ver.2's new feature is transient analysis by direct integration. BCtool's BcGUI was totally revised. The functions added to the BCtool include setting of boundary conditions for the Thermal, an interactive setting and saving of material properties, and a few MPC related ones. The POSTtool is newly released.

As of Jan. 10, 2013, the number of registered users is 7,455, and the total download instances are 37,335.

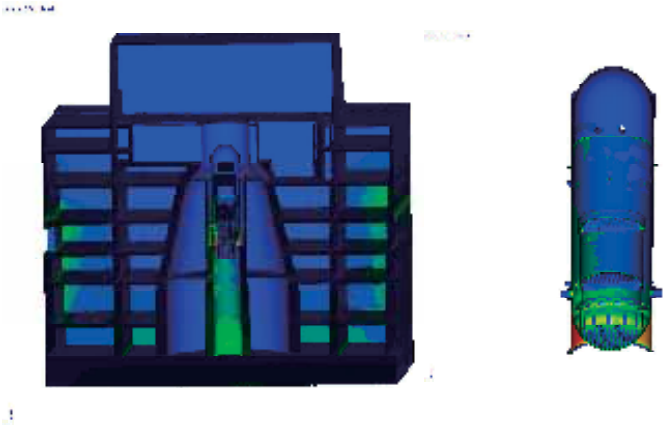


Fig. 5 Dynamic stress response of a BWR to an earthquake in Japan

5 Promotion of ADVENTURE by Insight

5.1 Limited functions of ADVENTURE

ADVENTURE is not equipped with many functions as commercial codes. For example, the capability to set boundary conditions is not adequate. Nonlinear capability of the Solid is also very limited. No contact analysis is supported. But, it can be put to practical use as told later.

5.2 Target uses

Although ADVENTURE has the above mentioned function limitations, the following uses are possible because it's free of charge and open-source.

i) Self-education/homework of students.

ii) Education in universities, technical colleges, and vocational training schools.
iii) Research in graduate schools, and technical colleges.
iv) Self-education of working people.
v) Education of design engineers in companies.
vi) Research in companies.
vii) Operations in companies (development or design).

5.3 Why ADVENTURE is worth using?

ADVENTURE's value of use is presented next.

At first, it can solve software license shortages. To increase the number of commercial CAE software licenses is very costly. ADVENTURE solvers' functions are limited, but costs per PC can be very low. For example, if 10 PCs are equipped with ADVENTURE, load sharing can be realized in which straightforward analyses are made by ADVENTURE and high-end analyses are performed by commercial software.

Secondly, ADVENTURE enables large-scale analyses. For example, the Solid gives a 12M DOF problem solution on a 5-PC cluster with 8GB main memory each. A PC-cluster can resolve memory shortage for a large-scale analysis.

Thirdly, mid-size problem can be solved in a short time. For example, the Solid can solve a 500 K DOF cantilever bending problem in less than a half minutes if 3 PC's of typical latest specs are used in parallel. Particularly, Solid Ver. 2 make it possible to run parallel computation on a multi-core PC since it can use the OpenMP. Therefore, it is useful at times of solving structure defects, or in early stages of product design study when quick simulation results are needed.

Fourthly, software can be deployed to each design engineer. Commercial software is versatile, but too expensive to distribute to each design engineer. ADVENTURE solvers are much less expensive even if installation is ordered outside. Therefore, it is possible to build environment in which every design engineer can use the software at a time even if there are many.

5.4 Key to promotion of use

Three keys to promotion of use will be introduced next.

Firstly, a key person is necessary who understands the advantages of the ADVENTURE system and who has the following skills and qualifications.

i) Can handle Linux OS and software compilation.

ii) Have sufficient computer literacy to prepare environment and use ADVENTURE.

iii) Have enough knowledge of FEA and design operations.

iv) Can create and solve practical problems that make managers to allocate budget for introduction of ADVENTURE to design operations.

v) Passion.

vi) Ability to educate design engineers to make FEA.

Secondly, CAD Interface to ADVENTURE must be available. There are three options. First option is use of the ADVENTURE_CAD. The ADVENTURE_CAD is nothing more than polygon-based, but surprisingly, can create a relatively big variety of surface mesh. Commercial CAD is not necessarily a must. The grammar of the ADVENTURE_CAD's geometry definition script is not that difficult. ADV_CAD is included in the Windows version. So, you can create surface mesh either on Windows or on Linux. The second option is use of the FEMAPtool. In this case, actually, no CAD interface is necessary. You prepare any Nastran Bulk data of tetrahedral or hexahedral mesh and they are converted to ADVENTURE's msh formatted mesh. Unfortunately, FEMAPtool's capability is very limited. No boundary conditions are converted. The third option is use of Meshman_Nastran_I/F, an Insight's commercial product. Boundary conditions and material properties as well as mesh data are converted to an ADVENTURE formatted binary file. The product is equipped with a GUI. It has no limitation that only data generated by the FEMAP can be processed.

Thirdly, easy installation and easy handling of ADVENTURE software must be realized. On option is use of the ADVENTURE Grid Stations that is a commercial product of Insight. A full turnkey PC-cluster equipped with ADVENTURE software can be provided. A second option is use of the Windows version. This is an official module and easy to use but only single solver is supported. It has disadvantages of slow visualization for models greater than 100 K DOFs and that only linear-elastic static stress analysis is supported etc.

6 Insight's activities

In fact, Insight performs the following activities:

i) Free seminar.

ii) A free viewer Meshman_Viewer is distributed.

iii) A high performance viewer, Meshman_Viewer HPC is distributed.

iv) Sales of ADVENTURE Grid Stations.

v) Sales of Meshman_Nastran_I/F for ADVENTURE.

vi) Tutorials are available.

vii) Paid seminars with ADVENTURE project were held.

viii) Meshman for ADVENTURE Custom version is sold.

ix) Development of POSTtool and BCtool2.

x) Consulting about ADVENTURE.

Free seminars have been held since Oct. 2003. Hands-on experience of ADVENTURE is the objective of the seminar. Solid, Fluid and Magnetic courses are available. Ad-hoc consulting for introduction of ADVENTURE is also provided.

6.1 Paid seminar

Seven paid seminars have been held as follows. Developers presented contents of modules and a few guests introduced use examples.

i) Auto version 0.1b.

ii) Thermal version 1.0 and Magnetic version 1.1.

iii) Solid version 1.1 and Fluid version 0.41b.

iv) Solid's non-linear capability and Solid's C functions.

v) on_Windows version 0.2b and sFlow version 0.11b.

vi) Magnetic version1.3 and Solid version 1.2.

vii) Auto GL version 0.11b.

6.2 Tutorials

Tutorials fully describe a series of procedure from making surface mesh of any geometry to visualizing results with a few exceptions. A total of ten tutorials are available. Only the Solid is provided with three types of tutorials, i.e. one in which the Solid is used along with the iAgent, another in which the Solid is used by commands and the other that handles the non-linear capability. Other tutorials include the Thermal, the Auto, the Fluid, the Magnetic, the CAD, the Shape and the sFlow.

6.3 Introduction to a company

6.3.1 Case 1

A company that manufactures seals and fuses of electronic devices introduces ADVENTURE. A key person exists in the design section. This case was a bottom up approach. Other design engineers start using ADVENTURE. The Solid and the Thermal

are used as a design tool. Analyses are performed on Linux OS. Virtually no support was given from Insight. It should be regarded as a success instance of ADVENTURE implementation.

6.3.2 Case 2

A key person in a company that manufactures rubber related products has just started study of ADVENTURE use. The Windows version is being tried. ADV_CAD is used to make mesh.

Insight has started supporting this company. But currently no budget is allocated yet. This case is also bottom up approach. Accumulation of useful analysis results should be needed before ADVENTURE is adopted.

7 Concluding remarks

Overview, analysis examples, and promotion of ADVENTURE have been addresses. From here on, more vigorous ADVENTURE promotion activities are planned. As official ADVENTURE activities, on_Windows Ver. 0.41b will be released soon. Since ADVENTURE Solid Ver. 2 etc. was released, another paid seminar will be held. Also, TriPatch will be modified to incorporate a new CAD format interface.

As Insight's activities, marketing for analysis consulting with ADVENTURE and customization of ADVENTURE has started. Insight's original pre-post processor product will be released for sale.

References

[1] http://adventure.sys.t.u-tokyo.ac.jp/

[2] S. Yoshimura, H. Kawai, S. Sugimoto and K. Murotani: High Performance Computation and Walkthrough Visualization for Assessing Seismic Safety of Nuclear Power Plants, 21st International Conference on Structural Mechanics in Reactor Technology (SMiRT21), New Delhi, India, Nov. 6-11, (2011). Author, A. and Author, B., Pre-conference paper title, in Editor, A. and Editor, B., eds., Proceedings of the Conference Name, Place, Vol.6, No.1 (2012), pp.1–4.

Development of a Numerical Library based on Hierarchical Domain Decomposition for Post Petascale Simulation

Ryuji SHIOYA

Toyo University, 2100 Kujirai Kawagoe Saitama, 8508585 JAPAN
shioya@toyo.jp

1 Introduction

In November 2012, TOP500 [1] List of the world's top supercomputers reported that the world's fastest supercomputer is Titan, DOE/SC/Oak Ridge National Laboratory, United States with 17.59 Peta Flops (PFlop/s) of maximal LINPACK performance and in June 1993 when the first list of TOP500 was released, it was CM-5/1024, Los Alamos National Laboratory, United States with 59.7 Giga Flops (GFlop/s). Therefore, it is about 300,000 times faster in these 20 years, that is, about 1.9 times faster per year. According to the Moore's law [2], it is 2 times per year. With such speed-up, in 2018 or 2019, it will achieve over 1 Exa Flops (EFlop/s), so called post petascale speed.

To use such high performance computers in the post generations of petascale supercomputer, developing system software technologies as well as related systems is very important. More concretely, research and development of system software enable us to exploit maximum efficiency and reliability from supercomputers which will be composed of general purpose many-core processors as well as special purpose processors, so called GPGPU. In addition to the system software such as programming languages, compilers, runtime systems, operation systems, communication middleware, and file systems, application development support systems and ultra-large data processing systems are important targets for research and development.

In such situation, we have been developing open source system software, ADVENTURE [3], which is a general-purpose parallel finite element analysis system and can simulate a large scale analysis model with supercomputer like the Earth Simulator or K-computer. In the system, HDDM (hierarchical domain decomposition method), which is a very effective technique to large-scale analysis, was developed. The

aim of our project is to develop a numerical library based on HDDM that is extended to pre and post processing parts, including mesh generation and visualization of large scale data, for the post petascale, that is, exa scale simulation (Fig. 1).

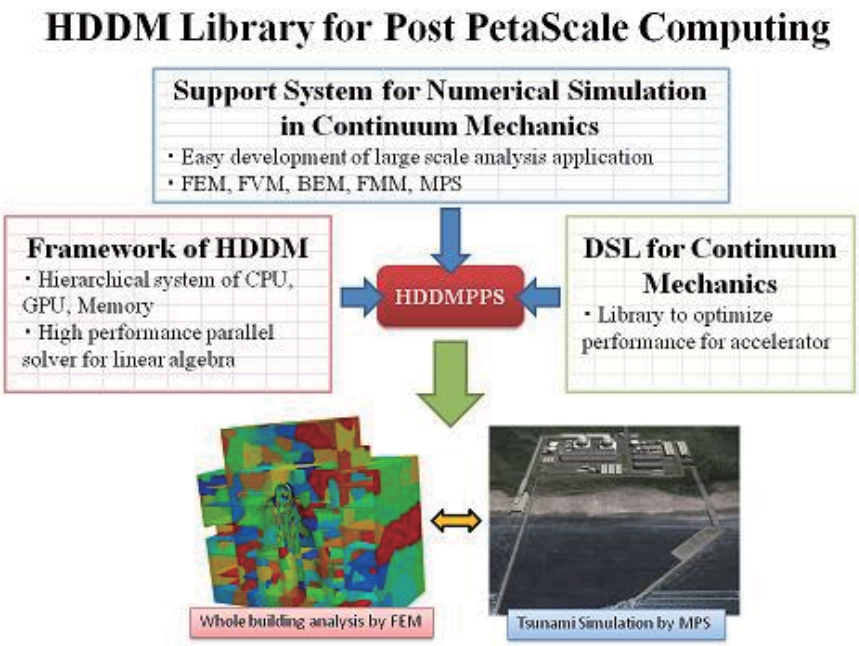


Fig.1: HDDM Library for Post Petascale Computing.

2 Overview of ADVENTURE SYSTEM

Various general-purpose computational mechanics systems have been developed in the last three decades to quantitatively evaluate mechanical / physical phenomena such as deformation of solid, heat transfer, fluid flow and electromagnetics. Nowadays such systems are regarded as infrastructural tools for the present industrialized society. The existing systems, however, cannot be used with massively parallel processors (MPPs) with the order of 100-10,000 processing elements (PEs), which are about to dominate the high-performance computing market in this century, as they were developed for single-processor computers, which took leadership an age ago. Neither can the current systems be used in heterogeneous parallel and distributed computing environments such as the Grid. Owing to the fact, they can deal with only medium scale problems with

millions degrees of freedom (DOFs) at most.

The ADVENTURE project [3, 4] was one of the research projects in the "Computational Science & Engineering" field selected for the "Research for the Future (RFTF)" Program sponsored by the Japan Society for the Promotion of Science (JSPS) [5] during 1997-2002. The project is continuously going on as an open source software development project. In the project we have been developing an advanced general-purpose computational mechanics system named ADVENTURE since August 1997. The system was designed to be able to analyze a three-dimensional (3D) finite element model of arbitrary shape over 100 million DOF mesh, and additionally to enable parametric and non-parametric shape optimization. The first version of the ADVENTURE system has been released from the project website as open source software since March 2002 [3]. About 1,320 registered users in academia and industries are now using the programs, while one private company has developed and released its commercial version named ADVENTUREcluster [6, 7].

Domain-decomposition-based parallel algorithms are implemented in pre-processes (domain decomposition), main processes (system matrix assembling and solutions) and post-process (visualization), respectively. Especially the hierarchical domain decomposition method with a preconditioned iterative solver (HDDM) [8-10] is adopted in two of the main modules for solid analysis and thermal conduction analysis, named ADVENTURE_Solid and ADVENTURE_Thermal. The employed pre-conditioner is the Balancing Domain Decomposition (BDD) type method [11-18]. To efficiently solve a coarse space problem derived from equilibrium conditions for singular problems associated with a number of subdomains appeared in the BDD formulation, an incomplete factorization based parallel direct solver is employed. The ADVENTURE_Solid has been successfully implemented on a single PC, PC clusters and supercomputer such as the Earth Simulator or K-computer.

The ADVENTURE system consists of pre-, main- and post-processing modules and design modules that can be used in various kinds of parallel and distributed environments. The sys-tem employs HDDM based massively parallel algorithm as one of the major solution algorithms in order to handle a huge-scale finite element model over 100 million DOFs efficiently. The system employs module-based architecture and consists of 19 modules. The pre-process modules include the surface patch generator which converts geometry model data into a collection of triangular surface patch data, named ADVENTURE_TriPatch, a tetrahedral mesh generator [19, 20], i.e. ADVENTURE_TetMesh, an attachment tool of boundary conditions and material

properties onto the mesh, i.e. ADVENTURE_BCtool, and a domain decomposer of a finite element model, i.e. ADVENTURE_Metis. The kernels of the ADVENTURE_Metis are a graph partitioning tool METIS and its parallel version ParMETIS developed in the University of Minnesota [21, 22]. The main process modules, i.e. solvers include an implicit elastic-plastic analysis module named ADVENTURE_Solid [9-10, 18] which enables large-deformation and implicit dynamic analyses, a thermal conductive analysis module named ADVENTURE_Thermal, a thermal-fluid analysis module named ADVENTURE_Fluid, a magnetic analysis module named ADVENTURE_Magnetic [23], an explicit impact analysis module named ADVENTURE_Impact, and a rigid plastic analysis module named ADVENTURE_Forge. The post process module named ADVENTURE_Visual is for parallel visualization of analysis results [24]. Common functions related to finite elements are programmed as class libraries named libFEM. Figure 2 shows the configuration of the ADVENTURE modules.

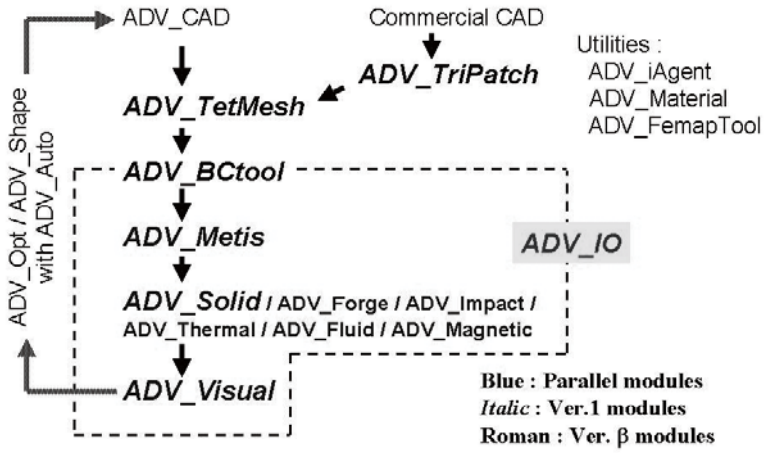


Fig.2: Configuration of ADVENTURE modules.

3 Parallel Algorithms Implemented in ADVENTURE System

One of the key technologies of the ADVENTURE System is the HDDM, which enables parallel finite element calculations on various kinds of computing environments. Basically in the HDDM, force equivalence and continuity conditions among subdomains are satisfied through iterative calculations such as the Conjugate Gradient (CG) method. Therefore it is indispensable to reduce the number of iterations by

adopting some appropriate preconditioning technique especially for solving large-scale ill-conditioned problems. The Neumann-Neumann algorithm (N-N) [11] is known as efficient domain decomposition preconditioner for unstructured subdomains. However, its convergence deteriorates with the increasing number of subdomains due to lack of a coarse space problem which takes care of global propagation of error. The Balancing Domain Decomposition (BDD) based N-N algorithm proposed by Mandel [12] shows that the equilibrium conditions for the singular problems on subdomains result in simple and natural construction of a coarse space problem and that its construction is purely algebraic. The BDD has been applied to solve various phenomena [13-15]. There are also several researches on parallelization of the BDD and also the FETI (Finite Element Tearing and Interconnecting) [26-32]. However, most problems solved there are still medium scale ones such as sub-millions to one million DOFs. As the DOFs of the coarse space problem is directly related to the number of subdomains, it is indispensable to consider the parallelization of the solution process of the coarse space problem as well when solving large-scale problems. The Salinas system [33], which employed the FETI-DP method [32], is succeeded in solving large-scale problems such as over 100 million DOF mesh of optical shutter model [34]. It shows good performance but does not seem to include load-balancing techniques. In the present study, an incomplete parallel direct method and the HDDM are adopted.

3.1 Hierarchical Domain Decomposition Method (HDDM)

In Domain Decomposition Methods (DDM), an analysis model, i.e. a finite element mesh with boundary conditions and material properties, is subdivided into a number of subdomains. The HDDM employs a hierarchical technique to implement the DDM on various parallel computers. In the HDDM, a group of processing elements (PEs) are subdivided into the following three subgroups: one Grand Parent PE (Grand), several Parent PEs (Parent or Parents), and many Child PEs (Child or Children). At the same time, the analysis model is subdivided into some 'parts' whose number is the same as the number of the Parents. Each part is further subdivided into a number of subdomains, the number of which can be much larger than that of the Children. Figure 3 shows a 35 million DOFs mesh for an Advanced Boiling Water Reactor (ABWR) model, generated by the ADVENTURE_TriPatch and the ADVENTURE_TetMesh. Figure 4 illustrates an example of the hierarchically decomposed mesh generated by the ADVENTURE_Metis.

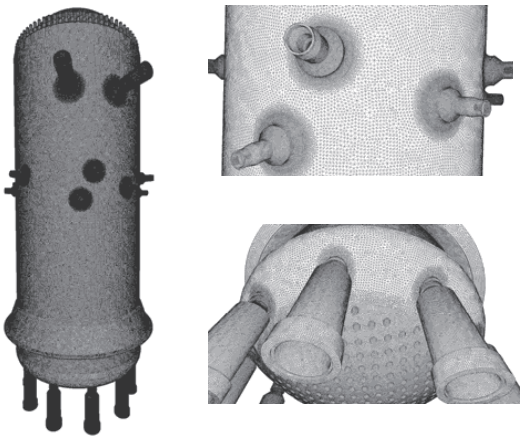


Fig.3: 35 million DOF mesh of ABWR model.

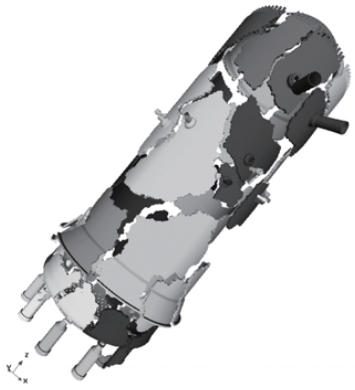


Fig.4: Part decomposition of ABWR vessel model.

Owing to the HDDM algorithm, large-scale analysis data can be easily handled by increasing the number of the Parents. The main roles of the three kinds of processors are summarized as follows. The Grand manages all PEs, i.e. synchronization and calculation of the sum of vectors spread over a number of Children. Each Parent stores mesh data and material properties of subdomains, sends / receives subdomains data to / from Child, and iterates loops of the CG method. Each Child performs finite element calculations of the subdomains received from the Parent, and sends analyzed data back to the Parent. Figure 5 shows the schematic data flow among PEs.

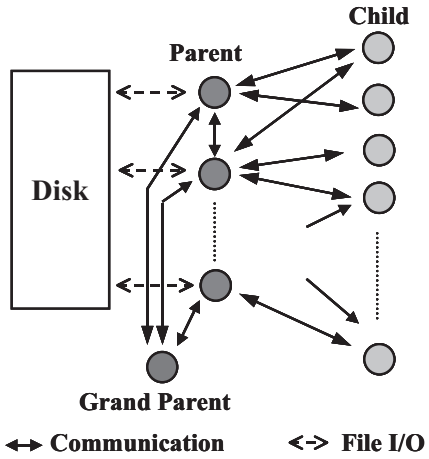


Fig.5: Schematic data flow in h-mode.

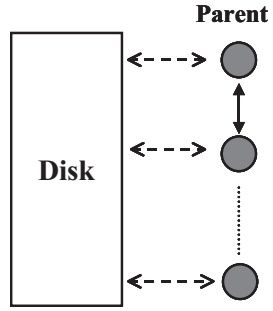


Fig.6: Schematic data flow in p-mode.

According to the design concept of the HDDM, most computation is assigned to the Children, while most communication occurs in between Parents and Children. Varying the number of Parents and Children for different kinds of parallel computers, the present HDDM-based system can easily achieve high performance. In the HDDM architecture, thanks to the dynamic load balancing technique among Child processors, high parallel performance can be achieved even in heterogeneous computer environments. However in this mode, an amount of data communication between Child and Parent tends to be large. To reduce such data communication among Children and Parents, it is useful to assign all balance becomes static. This analysis mode shown in Figure 6 is called in parallel processors mode (p-mode), while the original analysis mode as shown in Figure 5 is named hierarchical processors mode (h-mode).

3.2 Balancing Domain Decomposition (BDD)

The BDD algorithm is based on the DDM with a preconditioned iterative solver. After eliminating interior DOFs of local subdomain matrices, the problem to be solved is reduced onto the interface DOFs of subdomains. The reduced matrix is so-called Schur complement. The reduced problem is also called the interface problem, and is to be solved by a preconditioned iterative method. There are two main methods as such preconditioner, i.e. local subdomain correction and coarse grid correction in a coarse space.

3.3 Performance Optimization of ADVENTURE System

As optimization approaches of structural analysis code, ADVENTURE Solid, one example for The Earth Simulator 2 is described. In this case, we have chosen two approaches. One is to optimize the ADVENTURE Solid code within the range of the existing design, by varying a few selected performance-sensitive parameters. The other is to apply more drastic design changes, such as the local Schur complement approach.

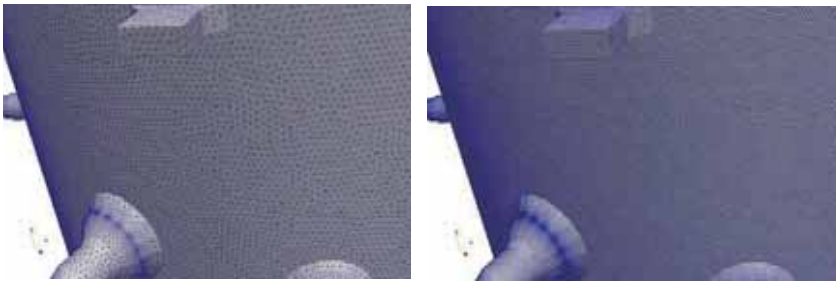
Here, the performance design issues of ADVENTURE Solid are briefly explained. ADVENTURE Solid is based on the hierarchical domain decomposition method (HDDM). In HDDM, a whole analysis domain is subdivided into many small subdomains. The parallelization of ADVENTURE Solid code is primarily based on subdomain-wise FEM calculation. On the FE analysis of each subdomain, a linear system of the subdomain stiffness matrix is solved. A skyline solver is employed for the solution of the relatively small system. In the current version of ADVENTURE Solid, this subdomain-wise skyline solver is identified as a hot spot. The inner-most loop of the hot spot is the double loop in forward and back substitution of the skyline solver. Its loop length, which means the band width of the skyline matrix, is not so large. It is usually about several hundreds.

As for the former approach, we selected average subdomain size of the domain decomposition method as the most performance-sensitive parameter. This parameter controls the effective vector length of inner-most loops of ADVENTURE Solid. By varying the subdomain size parameter in the input data files through the analysis of 245 million DOF Pantheon model, about 15 % of peak performance was achieved. In this case, vector operation ratio was 98.57 %. Using 512 processors, 6.5 T flops was obtained. The analysis of this Pantheon model took 18.6 minutes, using 4.1 TB memory. The number of DDM iterations was 277. Each DDM iteration took 2.48 seconds.

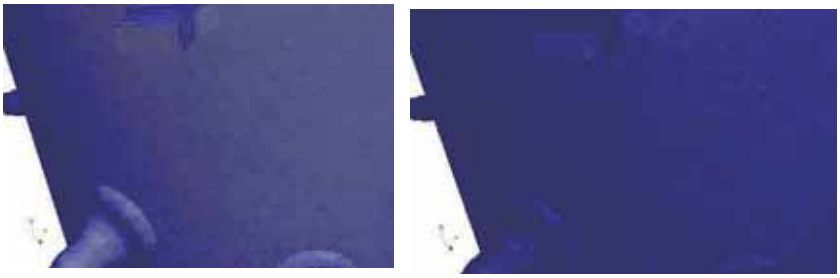
As the latter approach, the local Schur complement (LSC) of each DDM subdomain is explicitly formed. An LSC matrix is a symmetric full matrix. In each DDM iteration, simply a matrix vector product using the LSC is performed for each subdomain. The last year, the performance of this approach itself has already been investigated fully using the extracted hot spot code from ADVENTURE Solid. About 40 % of the peak performance was obtained through the hot spot code. This year, the new LSC-based performance design was verified within the single process code of ADVENTURE Solid. Scalar version of this new LSC implementation achieved about twice faster than the existing implementation using skyline solver for the DDM subdomain local solver on PC. This means the total number of floating point operations, or the number of I/O requests to memory system can be halved by this approach.

3.4 Developing mesh refinement function for ADVENTURE Metis

A mesh refinement function for ADVENTURE_Metis was implemented, since the huge size models must be generated because of the huge analyses in post petascale simulation. We have succeeded to generate the refined mesh models of more than tens-of-billions DOF scales from the coarse mesh models in parallel computers at short times (Fig.7).



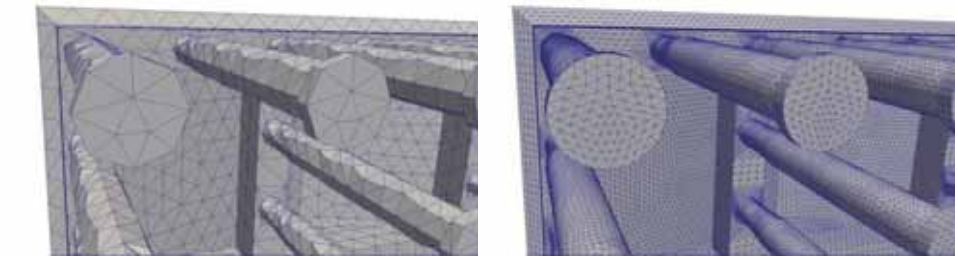
(a) Original mesh of 10 million nodes. (b) First refined mesh of 68 million nodes.



(c) 2nd refined mesh of 480 million nodes. (d) 3rd refined mesh of 3,600 million nodes.

Fig. 7: Refined mesh generating system.

The generated elements by the mesh refinement function can be geometrically fitted on surfaces of CAD models by shape functions of finite element method, as shown in Figure 8. Additionally, to generate mesh model of the same scale size by memory-saving mode using only one CPU, this system can be flexible for computer environment.



(a) Initial mesh (b) Twice refined mesh by Geometry fitting

Fig. 8: Geometry fitting function.

4 Numerical Analysis of Pressure Vessel Model

The ADVENTURE_Solid is implemented on the Earth Simulator (ES) consisting of 256 nodes, i.e. 2,048 PEs with 4TB of main memory, whose theoretical peak performance is 16 TFLOPS. The second problem is an elastostatic stress analysis of a simplified pressure vessel model with 100 million DOFs unstructured mesh. Its mesh size is listed in Table 1. As a boundary condition, the bottom surface of the vessel is fixed, and a static gravitational force is applied to the vessel in the horizontal direction, being similar to the previous problem.

Table 1: Mesh size for a simplified vessel model.

Number of elements	25,084,456
Number of nodes	34,772,634
Total degrees of freedom	104,195,500

Although we do not show convergence histories of relative residual, (1) HDDM with BDD and N-N preconditioner (denoted as BDD) and (2) HDDM with BDD and diagonal-scaling preconditioner (denoted as BDD-DIAG) demonstrate excellent performance in convergence. By considering the performance results, it is concluded

here that the diagonal scaling is sufficient as local subdomain correction in the BDD method. The analysis model is divided into 34,816 subdomains and then the number of DOFs of its coarse space is 208,896. The LU factorization of the coarse grid operator is calculated in only 20 seconds. As the result, the present system successfully achieved 5.1TFLOPS, which is 31.8% of the peak performance. The calculation time is only 8.5 minutes. Parallel ratio over 99.9% is achieved, and then parallel efficiency exceeds 80% not only for computation time per iteration but also for total computation time.

5 Conclusions

We have been developing an advanced general-purpose finite element analysis system, named ADENTURE, which is designed to be able to analyze a model of arbitrary shape over 100 million DOF mesh. The ADVENTURE_Solid has been successfully implemented on a single PC, PC clusters and supercomputers such as K computer.

To perform our systems on the post petascale supercomputers with high efficiencies, we are developing a numerical library based on HDDM [35, 36] that is extended to pre and post processing parts, including mesh generation and visualization of large scale data, for the post petascale simulation.

Acknowledgment

This research was supported by CREST, JST. This work is performed as a part of the ADVENTURE project and the authors also would like to thank all the members of the ADVENTURE project.

References

[1] The Top500 List: http://www.top500.org

[2] Gordon E. Moore, Cramming more components onto integrated circuits, Electronics Magazine, Vol. 38, No. 8 (1965).

[3] ADVENUTRE System: http://adventure.sys.t.u-tokyo.ac.jp

[4] S. Yoshimura, R. Shioya, H. Noguchi and T. Miyamura, Advanced general-purpose computational mechanics system for large-scale analysis and design, Journal of Computational and Applied Mathe-matics, Vol. 49 (2002) 279-296.

[5] Report on Computational Science and Engineering, JSPS-RFTF Program 2001-2002, (2002).

[6] M. Suzuki, H. Akiba, S. Yoshimura and G. Yagawa, Analysis of stress intensity factor of piping using large scale analysis code AD-VentureCluster, Trans. 16th SMiRT, G05/5, Washington D.C., CD-ROM, (2001).

[7] M. Suzuki, T. Ohyama, H. Akiba, H. Noguchi and S. Yoshimura, Development of fast and robust parallel coarse-grid based CG solver for large scale finite element analyses, Trans. JSME, 68A-671 (2002) 1010-1017 (in Japanese).

[8] G.Yagawa and R.Shioya, Parallel finite elements on a massively parallel computer with domain decomposition, Computing Systems in Engineering, 4 (1994) 495-503.

[9] R.Shioya and G.Yagawa, Parallel finite elements of ten-million DOFs based on domain decomposition method, WCCM IV Computa-tional Mechanics -New Trends and Applications- IV 11 (1998) 1-12.

[10] T. Miyamura, H. Noguchi, R. Shioya, S. Yoshimura and G. Ya-gawa, Elastic-plastic analysis of nuclear structures with millions of DOFs using the hierarchical domain decomposition method, Nuclear Engineering & Design, 212 (2002) 335-355.

[11] Y. H. DeRoeck and P. LeTallec, Analysis and test of a local domain decomposition preconditioner, 4th International Symposium on Do-main Decomposition Methods (1991) 112-128.

[12] J. Mandel, Balancing domain decomposition, Communications on Numerical Methods in Engineering, 9 (1993) 233-241.

[13] P. LeTallec and M. Vidrascu, Generalized Neumann-Neumann preconditioners for iterative substructuring, 9th International Sympo-sium on Domain Decomposition Methods, (1996) 413-425.

[14] P. LeTallec, J. Mandel and M. Vidrascu, A Neumann-Neumann domain decomposition algorithm for solving plate and shell problems, SIAM J. Number.

Math., 35 (1997) 836-867.

[15] J. Mandel and C. R. Dohrmann, Convergence of a balancing do-main decomposition by constraints and energy minimization, Numer. Lin. Alg..

[16] R. Shioya, H. Kanayama, D. Tagami and M. Ogino, 3D large scale structural analysis using a balancing domain decomposition method, Trans. JSCES, 2 (2000) 139-144 (in Japanese).

[17] R. Shioya, M. Ogino, H. Kanayama and D.Tagami, Large scale finite element analysis with a balancing domain decomposition method, Key Engineering Materials, 243-244 (2003) 21-26.

[18] T. Miyamura and S. Yoshimura, Parallel stress analyses of ancient architecture Pantheon on PC cluster, Trans. Architectural Institute of Japan, 55 (2001) 95-102 (in Japanese).

[19] G. Yagawa, S. Yoshimura and K. Nakao, Automatic mesh generation of complex geometries based on fuzzy knowledge processing and computational geometry, Integrated Computer-Aided Engineering 2 (1995) 265-280.

[20] S. Yoshimura, H. Nitta, G. Yagawa and H. Akiba, Parallel auto-matic mesh generation of nuclear structures with ten-million nodes, Trans. 15th SMiRT, Seoul, II (1999) 21-28.

[21] G. Karypis and V. Kumar, Multilevel k-way partitioning scheme for irregular graphs, Technical Report TR 95-064, Department of Computer Science, University of Minnesota, (1995).

[22] G. Karypis and V. Kumar, Parallel multilevel k-way partitioning scheme for irregular graphs, Technical Report TR 96-036, Department of Computer Science, University of Minnesota, (1996).

[23] H. Kanayama, R. Shioya, D. Tagami and M. Saito, Numerical analysis of 3D eddy current problems by the hierarchical domain de-composition method, Trans. JSCES, 3 (2001) 151-156 (in Japanese).

[24] S. Shoui, S. Yoshimura, H. Akiba, T. Ohyama and G. Yagawa, Parallel visualization of finite element solutions with ten million DOFs using PC cluster, Proceedings of European Congress on Computational Methods in Science and Engineering (ECCOMAS2000), Balcelona, CD-ROM, (2000)

[25] http://www.es.jamstec.go.jp/

[26] M. Vidrascu, Remarks on the implementation of the generalized neumann-neumann algorithm, 11th International Conference on Do-main Decomposition Methods, (1998) 485-493.

[27] P. R. Amestoy, I. S. Duff, J. -Y. L'Excellent and P. Plechac, PARASOL. An Integrated programming environment for parallel sparse matrix solvers, High-Performance Computing, (1999) 79-90.

[28] J.-M. Cross, A preconditioner for the Shur complement domain decomposition method, 14th International Conference on Domain Decomposition Methods (2002)

[29] P. Goldfeld, Balancing Neumann-Neumann for (in)compressible linear elasticity and (generalized) stokes-parallel implementation, 14th International Conference on Domain Decomposition Methods, (2002)

[30] F.-X. Roux and C. Farhat, Parallel implementation of the two-level FETI method, 9th International Conference on Domain Decomposition Methods, (1997)

[31] J. Mandel, R. Tezaur and C. Farhat, A scalable substructuring method by lagrange multipliers for plate bending problems, SIAM Journal of Numerical Analysis, 36 (1999) 1370-1391.

[32] M. Lesoinne and K. Pierson, FETI-DP : An efficient, scalable and unified dual-primal FETI method, 12th International Conference on Domain Decomposition Methods, (1999) 421-428.

[33] http://endo.sandia.gov/9234/salinas

[34] M. Bhardwaj, K. Pierson, G. Reese, T. Walsh, D. Day, K. Alvin, J. Peery, C. Farhat and M. Lesoinne, Salinas : A scalable software for high-performance structural and solid mechanics simulations, Technical Papers of SC2002, (2002).

[35] H. Kawai, M. Ogino, R. Shioya and S. Yoshimura, Large Scale Elasto-Plastic Analysis Using Domain Decomposition Method Optimized for Multi-core CPU Architecture, Key Eng. Materials, 462-463 (2011) 605-610.

[36] M. Ogino, R. Shioya, A Scalable and High Performance Implementation of the Domain Decomposition Method, The 4th International Conference on Computational Methods (ICCM2012), 160.pdf, (2012) 1-8.

Performance Evaluation of ADVENTURE on K Computer

Hiroshi Kawai[1], Masao Ogino[2], Ryuji Shioya[3], Shinobu Yoshimura[4]

1 Tokyo University of Science-Suwa, 5000-1 Toyohira, Chino-shi, Nagano, 391-0292 JAPAN
kawai@rs.tus.ac.jp
2 Nagoya University, Furo-cho, Chikusa-ku, Nagoya-shi, Aichi, 464-8601 JAPAN
masao.ogino@cc.nagoya-u.ac.jp
3 Toyo University, 2100 Kujirai, Kawagoe-shi, Saitama, 350-8585 JAPAN
shioya@toyo.jp
4 The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656 JAPAN
yoshi@sys.t.u-tokyo.ac.jp

1 Introduction

Recently, rapid adoption of multi-core scalar CPU in the supercomputing industry is forcing researchers, engineers, scientists and application developers running a large-scale simulation to optimize the performance of their simulation codes for a multi-core PC cluster. Structural analysis codes based on the finite element method are no exception. Furthermore, because of the three challenging issues in the semi-conductor industry, namely, ILP (instruction level parallelism), memory and power walls, CPU architecture is gradually shifting from multi-core to many core.

For example, K Computer at Kobe, Japan has already achieved 10 peta FLOPS in LINPACK benchmark. It has more than 80 thousands processors, connected each other through TOFU high speed 6-D torus-type interconnect. Each processor, Fujitsuu SPARC64 VIII fx, has 8 cores. Total number of cores reaches more than half millions. In case of Fujitsuu PRIMEHPC FX10, the commercial version of K Computer, 16 cores are integrated on a single processor chip.

To obtain high intra-node performance on a multi-core PC cluster or a supercomputer, efficient utilization of processor cache memory and SIMD instruction set should be considered. Therefore, the traditional memory access-intensive approach,

which prefers less computing and more storage on memory, may not be effective for supercomputers in the next post-peta FLOPS era.

The Domain Decomposition Method (DDM) is one of the effective parallel finite element schemes. We have been developing an FE-based parallel structural analysis code, ADVENTURE_Solid [1] [2] [3], based on DDM, with the Balancing Domain Decomposition (BDD) pre-conditioners [4]. It was successfully applied on a vector-type supercomputer, The Earth Simulator (ES) [5]. However, the performance of scalar-type supercomputers has already surpassed those of vector-type machines. To switch the hardware platform from high Byte / FLOPS environments like ES to relatively low B/F one like multi-core PC clusters and K Computer, we were forced to re-designing some performance sensitive kernels in DDM code, such as subdomain-level local FE solver and coarse solver.

In this paper, various types of performance tuning approaches for the DDM-based structural analysis code, ADVENTURE_Solid, on peta-scale massively parallel supercomputers, such as K Computer and Fujitsuu PRIMEHPC FX10, are presented.

There are many points to be considered in design and implementation of the subdomain local solver, such as the choice of a linear algebraic solver (direct or iterative), matrix storage-free or not, use of element-by-element techniques, and also pre-conditioner types if it is iterative solver-based. Several types of subdomain local solvers are investigated [6]. They are roughly classified into four categories, DS (Direct solver-based matrix Storage), DSF (Direct Storage-Free), IS (Iterative solver-based matrix Storage) and ISF (Iterative Storage-Free). Of them, except the first one, DS, other three approaches consume less memory and can be used for solving a much larger scale problem. Also, they can utilize processor cache memory more efficiently. It is identified that, in terms of execution time, they perform comparably well against DS approach. On some low B/F environments, they even run faster than DS. Also, some new local solvers, IS-ICT using Incomplete Cholesky factorization with Threshold pre-conditioner, and DS-LSC, using explicit evaluation of local Schur complement, are introduced.

In addition, to accelerate the coarse grid correction stage of the BDD pre-conditioners, we employ a parallel skyline solver, tuned for solving a repeated RHS problem on distributed memory parallel computers.

2 Design and Implementation of DDM code : Survey

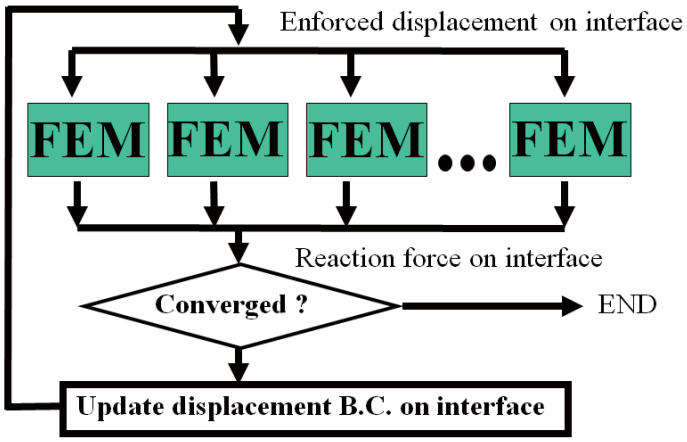
To design and implement a parallel finite element code based on the domain decomposition method (DDM), there are several points to be considered in terms of performance, mainly computational cost, efficiency and memory consumption. Assuming that inter-node parallelization on a distributed memory supercomputer is going well and communication cost is acceptable, obviously, the next important issue is the design and the implementation strategy of the subdomain-level local FE solver. In addition, when a global solver or a coarse grid solver is required for the global data transfer among subdomains in the pre-conditioner, which is the must-have for typical large-scale structural problems, its design and implementation strategy is also very important, especially in a highly parallel environment with over thousands of computational nodes.

There are roughly three types of DDM-based scheme. One is so called Schwartz type. The next one is primary, Schur complement or iterative substructuring type. The third one is dual, Lagrange multiplier type (FETI, as an example). The first one is based on overlapping domain decomposition, while others are based on non-overlapping one. Here in this paper, we restrict our focus on the second, iterative substructuring type scheme. Because there are some commonalities among these three types of DDM-based scheme in the implementation stage, however, we may refer techniques found in other two types of DDM scheme as needed.

2.1 Subdomain local solver

For every DDM iteration, the subdomain local FE solver is invoked at least once, and typically a few times. The majority of computational time, for example, more than 95 %, is consumed by the local solver. Therefore, the design and the implementation of the local solver are critical for the total performance of the DDM code. Assuming a structural analysis, the flow of DDM algorithm without any pre-conditioner is shown in Fig. 1.

Fig.1: Flow of DDM algorithm.



In a typical DDM formulation, subdomain local Schur complement (LSC) matrix, say, $\mathbf{S_i}$, and a given vector $\boldsymbol{v}$ are multiplied. However, in actual design and implementation of the DDM code, the LSC matrix, S_i itself is never formed or evaluated explicitly. Instead, matrix-vector product $\mathbf{S_i}\,\boldsymbol{v}$ is evaluated through a finite element analysis of this subdomain with Dirichlet boundary conditions.

Typical design and implementation of this subdomain local FE analysis are based on a direct solver. First, for each subdomain, the part of the stiffness matrix related to the internal degrees of freedom (DOF) is factorized once. Then, for each DDM iteration, the forward and back-substitution is performed on the factorized matrix. As the direct solver, a band or a skyline solver can be used with appropriate subdomain-level mesh renumbering. If the size of subdomain is relatively large, a sparse direct solver may also be used, and it is faster and consumes less memory than other solvers. Through our experiences, however, conversely if the subdomain size is small, the efficiency gain is small or even negative because of set-up cost and overhead of the sparse direct solver.

An iterative solver can also be used in the subdomain local FE analysis. If it is applied to the essential pure-DDM iteration stage, the iterative solver is required to produce a very accurate solution until machine precision, for example, 15 digits in case of double precision number. In case of the pre-condition stage, inexact solver can also be used. In this stage, the subdomain local solver serves as the smoother of a multi-grid solver. For example, Neumann-Neumann pre-conditioner can be substituted to a simple diagonal scaling. In case of FETI family, Dirichlet pre-conditioner is often simplified to

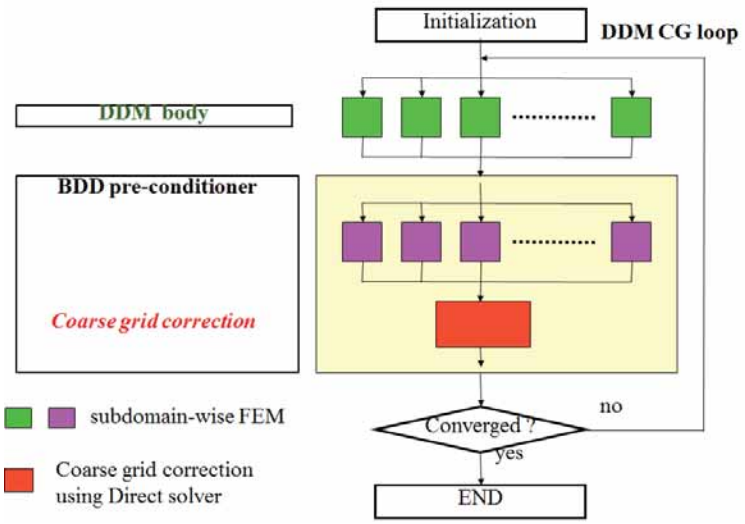
a lumped pre-conditioner.

2.2 Global or coarse grid solver

A global solver is required to reduce the number of DDM iterations. If the number of subdomains is large, which is very typical in a large-scale analysis, it is desirable to equip some kinds of global solver. In case of structural problems, it is well known that the convergence ratio of such problems is usually very bad. Furthermore, in a practical application found in civil, plant engineering, aerospace and vehicle industries, the geometry model is highly complicated, and relatively thin or long geometrical features are found everywhere in the model. A pre-conditioner which contains global data transfer among subdomains are essential for such cases.

We have been using the Balancing Domain Decomposition (BDD) pre-conditioners [4]. This pre-conditioner employs a coarse grid correction stage. The flow of DDM algorithm with the BDD pre-conditioner is shown in Fig. 2.

Fig.2: Flow of BDD algorithm.



The size of the coarse problem is almost proportional to the number of subdomains. In case of structural problems, each subdomain in the coarse space has 6 degrees of freedom, which is related to the rigid body motion of the structural problems. The size

of the coarse problem could easily reach the order of a million degrees of freedom, if a top-level supercomputer with thousands of computational nodes is utilized. However, because this global solver for the coarse grid is, by nature, global, it can easily become the primary bottleneck in such a highly parallel environment.

A coarse problem can be solved either a direct or an iterative solver. In case of structural problems, however, because the coarse matrix is highly ill-conditioned, it is typically handled by the direct solver. In this case, once the matrix is factorized at the beginning of the DDM solver, only forward and back-substitution is required for each DDM iteration.

3 Design and Implementation of DDM Code : Our Approach

Through decades of our experience of porting our code, ADVENTURE, and using onto various types of HPC and supercomputing environments, we have been trying and testing effective design and implementation strategies of DDM codes. Here in this section, some of these techniques especially applicable to the modern HPC environment are described in detail. They are categorized into two parts: subdomain local FE solver and global solver for coarse grid. Performance benchmark on multi-core workstation, PC clusters and supercomputers including RIKEN K Computer are demonstrated.

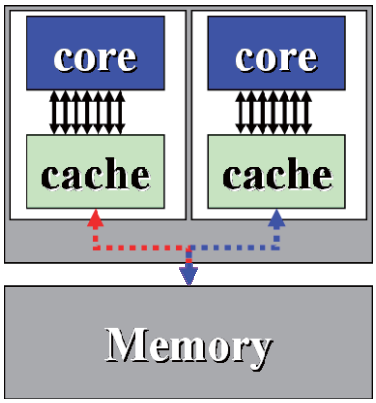
3.1 Modern HPC Environment

Recent supercomputers have tens of thousands of processor cores. Each computational node of such a distributed memory parallel computer typically contains one or more multi-core / many core scalar CPU. Then, these nodes are connected each other by a high speed interconnect, and they form a distributed memory parallel computer. For example, RIKEN K Computer, which has 80 thousand computational nodes and almost half-million cores, is shown in Fig. 3.

Fig.3: Image of RIKEN K Computer (from AICS home page).

As shown in Fig. 4, each scalar CPU has multiple cores. Then, each core contains associated blocks of cache memory and multiple floating point units. Furthermore, each floating point (FP) unit handles multiple data items, for example, 2 or 4 double-precision numbers, packed into a wider register. These FP instructions are called SIMD, single instruction multiple data. Intel and the compatible CPU employ SSE and AVX instruction set. Other CPU venders also have their own. In case of Fujituu SPARC64 series including K Computer and FX10, with HPC-ACE instruction set, two double precision data can be handled simultaneously. For example, Fujitsuu SPARC64 VIII fx in RIKEN K Computer runs in 2.0 GHz. Each CPU has 8 cores, and each core has 2 sets of add and multiply FP SIMD units. For each clock cycle, it can execute 2 x 2 x 2 x 2 = 8 FP operations. In total, This CPU can handle 64 FP operations with 2.0 GHz, thus 128 G FLOPS in total.

Fig.4: Multi-core scalar CPU architecture.



Thanks to these powerful HPC environments, a large scale finite element

analysis with hundreds of millions of degrees of freedom (DOF) can be performed quickly and easily, if it is just a linear static analysis. However, in case of a dynamic and non-linear elasto-plastic analysis, used typically at the design and maintenance phase in nuclear power plant, aerospace and architectural industries, a lot of time and increment steps are required. Such a huge scale FE analysis job could take hours or even days of computational time to run on a top-level supercomputer. Therefore, further speed-up and performance optimization of the parallel finite element codes are desired.

3.2 Design and Implementation of Subdomain Local Solver

The most critical performance bottleneck in a DDM solver is the subdomain local FE solver. Typically, a DDM code utilizes a direct solver in its subdomain-level FEM calculation. With a subdomain, after the factorization of its stiffness matrix has been performed, only forward and backward substitution suffices at each DDM step. In this approach, the subdomain stiffness matrix is factorized at the first iteration of DDM, and stored on memory as a band, skyline or other sparse storage form. For each DDM iteration step, the factorized matrix is read from memory into processor cache and it is used for the solution step of the linear system. This approach can be called DS (Direct solver-based matrix Storage). However, on a cache-based scalar CPU, especially in case of multi-core / many core, such a memory access intensive implementation may not scale well.

Here, we propose three new matrix-storage free approaches, DSF (Direct solver-based matrix Storage-Free), ISF (Iterative solver-based matrix Storage-Free) and IS (Iterative solver-based matrix Storage). They perform almost all the calculation on the cache of a CPU. Then, DSF and ISF store no data other than mesh on memory. Therefore, they require less memory, and are more scalable on multi-core architecture.

In addition, we also propose another DS type design and implementation, called local Schur complement approach (DS-LSC). While it is categorized in DS type and data are stored on memory, this approach explicitly evaluate and stores a local Schur complement matrix, $\mathbf{S_i}$, for each subdomain. It is efficient than the traditional DS based on repeated forward and back-substitution of the factorized matrix, if the subdomain size is reasonable large. And it can enjoy high utilization ratio of CPU floating point resources when the local Schur complement matrix is evaluated on each subdomain.

3.2.1 Direct Storage-Free (DSF) Approach

Direct solver-based matrix Storage-Free (DSF) approach is a kind of matrix storage-free implementation, which requires memory only for the mesh and analysis result data. While DS can utilize the coefficient matrix data already factorized and stored on memory, DSF requires a matrix factorization at each invocation of the subdomain local FE solver. Of course, the number of required FP operations is more than that of DS. However, the issue here is, is it actually slower than DS, and if it is slower, how much ? Because DSF is memory-saving implementation, much larger scale analysis can be performed within a limited memory space.

In our experience, a DSF code runs comparable against a DS code, if the subdomain size is small enough, for example, less than 500 DOF per subdomain. We have tested on a shared-memory workstation with 24 cores in total. DSF run faster than DS on this platform.

3.2.2 Iterative Storage (IS) and Strage-Free (ISF) Approaches

Both of Iterative solver-based matrix Storage (IS) and Iterative solver-based matrix Storage-Free (ISF) approaches utilize an iterative solver for the subdomain local FE analysis. The former stores the coefficient matrix and its pre-conditioner data if necessary on memory. The latter produces these data on the fly on each DDM iteration step. They can also be classified by the type of pre-conditioner. We have implemented diagonal scaling, SSOR, IC(0) and ICT pre-conditioners. Pre-conditioners whose data are costly to re-produce on the fly can be used only with IS type local solver.

The simplest IS or ISF type local solvers are ISF-diag and IS-diag, which use diagonal scaling only. ISF type implementation, which forms on the fly the coefficient matrix and the pre-conditioning data at the beginning of each invocation of the local solver, is available.

Another simple pre-conditioner is SSOR. In ISF-SSOR and IS-SSOR, a non-zero component storage CG solver with SSOR pre-conditioning and Eisenstat's trick [7] can be used. Because SSOR pre-conditioner utilizes coefficient matrix as a pre-conditioning data, it can also be implemented as an ISF type local solver.

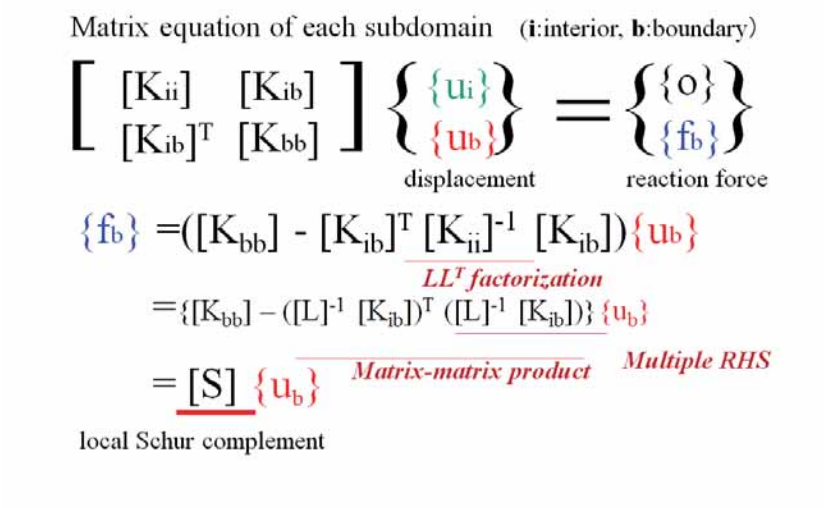
IS-ICT employs ICT (incomplete Cholesky factorization with threshold) pre-conditioner to accelerate CG iterations. An ICT pre-conditioning matrix is evaluated at the beginning of the DDM solver. IS-ICT solver runs comparably well against a traditional DS implementation on most of the modern multi-core scalar CPU.

3.2.3 Local Schur Complement (DS-LSC) Approach

In case of DS approach, a new type of local solver implementation, DS-LSC, is developed. In this approach, a local Schur complement (LSC) matrix for each subdomain is utilized. To do so, they have to be explicitly evaluated and stored on memory.

First, with each subdomain, its local Schur complement (LSC) matrix is explicitly evaluated through relatively heavy matrix operations, as shown in Fig. 5. Then, instead of solving subdomain local FE problem, the product between LSC matrix and vector is evaluated at each DDM iteration step. Also, at the evaluation phase of LSC matrix, which is mainly composed of matrix-matrix product operations, highly efficient utilization of CPU resources through on-cache floating point (FP) operations using level 3 BLAS routines can be expected. This LSC approach is faster and it consumes less memory than the conventional subdomain local solver approach, with forward and back-substitution on skyline storage, if the subdomain size becomes larger.

Fig.5: Local Schur complement approach.



3.3 Design and Implementation of Global Solver for Coarse Grid

To reduce the number of DDM iterations, a pre-conditioner for DDM is effective. The BDD (Balancing Domain Decomposition) pre-conditioner [3] is one of the effective pre-conditioners for DDM. This pre-conditioner mainly consists of two components, Neumann-Neumann pre-conditioner, which requires invocation of another subdomain local FE solver, and coarse grid correction, which is a global operation.

The latter stage, the coarse grid correction, is very important to suppress the increase of DDM iterations caused by the growth of the number of subdomains. However, this stage requires solving at each DDM step another medium scale linear equation about the coarse problem, whose matrix size is proportional to the number of subdomains. In case of a large scale analysis with many subdomains, this stage can easily become a performance bottleneck, especially on a parallel computer environment.

For example, in case of RIKEN K Computer, this computer has 80 thousands of computational nodes. A full-node job requires at least 80 thousands subdomains. To consider load balancing on a practical problem with complicated geometry, probably 320 thousands subdomains, which is 4 times more, will be desirable. Because this is a structural analysis, each subdomain corresponds to 6 degrees of freedom in a coarse space, 3 translation and 3 rotation components. Thus, in total, about 2 million DOF has to be solved in this coarse problem.

3.3.1 Sparse Direct Solver Approach

A sparse direct solver can be utilized to handle the large scale coarse problem. For example, using the sparse direct solver, PARDISO, in Intel Math Kernel Library, a linear system with hundreds of thousands of DOF, which is produced from a coarse problem with tens of thousands of sub-domains, can be handled easily on an ordinary PC or a workstation with a few GB of memory. However, Intel MKL can be used only on x86 platform. It cannot be used on RIKEN K Computer, whose CPU is made by Fujitsuu and SPARC-compatible.

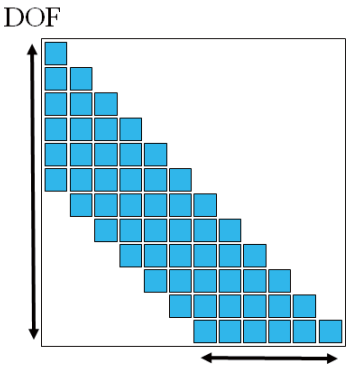
Some free software packages of sparse direct solvers, such as MUMPS, Paradiso (free version) and SuperLU, are available. Some of them run on distributed memory environment. We have tested MUMPS on K Computer. It runs reasonably well, although some performance problems occur in forward and back-substitution stages.

3.3.2 Parallel Skyline Approach

Although a sparse direct solver is desirable ideally, in reality, it could have portability and performance problems. Most of these freely distributed packages run reasonable well on the x86-compatible platform and the performance is highly optimized on the platform. However, they may usually have portability problems on relatively minor or new hardware platforms, including RIKEN K Computer, IBM BlueGene and emerging GPU-based supercomputers.

To work around these issues, we also employ a self-made parallel skyline solver, specially tuned for solving a repeated RHS problem on distributed memory parallel computers. As shown in Fig. 6, the parallelization is performed in block-wise direction in each row of blocks.

Fig.6: Parallel block skyline / band solver.



3.4 Performance Benchmark on Various HPC Environments

First, the proposed subdomain local FE solvers, DSF (Direct solver-based matrix Storage-Free) and ISF (Iterative solver-based matrix Storage-Free) are benchmarked. Compared with a traditional memory access intensive approach, called DS (Direct solver-based matrix Storage), these new matrix storage-free approaches can scale well

on multi-core / many core CPU architectures. Some performance benchmark results using these design approaches on a multi-core workstation and a PC cluster are tested. These new matrix storage-free approaches can scale well on multi-core / many core CPU architectures of a 24 core workstation. A 500 million DOF problem was successfully solved on a PC cluster. Also, local solver benchmark measured on K Computer is shown in Fig. 7.

Fig.7: Benchmark of local solver on K Computer.

Unit: m. sec.

Local solver type	Peak ratio	subdomain DOF 6500	10000	15000
DS-Sky-NZ	1%	*6.74*	*14.1*	*27.5*
DS-LSC	9%	0.74	0.94	1.31
(prepare LSC)	39%	1630	2880	6250
IS-diag	6%	22.7	40.2	74.8
ISF-diag-EBE	18%	33.0	60.4	101
IS-ICT	6%	14.0	22.0	36.7
IS-ICT-EBE	15%	18.9	31.2	47.2

Then, performance benchmark is carried out on various types of massively parallel supercomputers, such as T2K (Hitachi HA8000), Oakleaf FX (Fujitsuu PRIMEHPC FX10) in The University of Tokyo, and RIKEN K Computer. Large scale structural problems with near 1 billion degrees of freedom are successfully solved on T2K and RIKEN K Computer, using 4096 nodes within a few minutes. As a preliminary benchmark result using a beta version of the C compiler on K Computer, more than 30 % of peak FP performance is obtained on K Computer using 4096 nodes, 32768 cores. Performance benchmark of large scale structural problems on K Computer is shown in Fig. 8.

Fig.8: Benchmark of large scale plate-bending problems on K Computer.

total DOF	Subdomain DOF	Num. Subdomains	Num. nodes	Num. cores	time (unit : sec)
0.2B	15000	17000	4096	32768	36
0.3B	6500	66000	4096	32768	60
0.5B	10000	66000	4096	32768	83
0.8B	15000	66000	4096	32768	151

4 Conclusions

The Domain Decomposition Method (DDM) is one of the effective parallel finite element schemes. We have been developing an FE-based parallel structural analysis code, ADVENTURE_Solid, based on DDM, with Balancing Domain Decomposition (BDD) pre-conditioners.

Performance benchmark is carried out on various types of massively parallel supercomputers, such as Hitachi HA8000 (T2K), Fujitsuu PRIMEHPC FX10 and RIKEN K Computer. Large scale structural problems with 1 billion degrees of freedom are successfully solved on HA8000-based T2K supercomputer in Univ. of Tokyo, which has quad-core AMD Opteron processors. Each linear step can be performed in a few minutes with 4096 cores. Also, more than 30 % of peak FP performance is obtained on K Computer using 4096 nodes, 32768 cores. The implementation will be introduced to the future version of open-source CAE system, ADVENTURE.

Acknowledgment

This work is financially supported by the JST CREST Project on Simulation for Quake

Proof Capability of Nuclear Power Plants, Japan. Part of the results is obtained by early access to the K Computer at the RIKEN Advanced Institute for Computational Science (AICS).

References

[1] ADVEVTURE project home page, http://adventure.sys.t.u-tokyo.ac.jp

[2] G. Yagawa and R. Shioya: Computing Systems in Engineering, Vol. 4 (1993), p. 495.

[3] S. Yoshimura, R. Shioya, H. Noguchi and T. Miyamura, Advanced general purpose computational mechanics system for large-scale analysis and design, Journal of Computational and Applied Mathematics, 49 (2002) 279-296.

[4] J. Mandel: Commun. Numer. Meth. Eng., Vol. 9 (1993), p. 233

[5] M. Ogino, R. Shioya, H. Kawai and S. Yoshimura: Seismic Response Analysis of Nuclear Pressure Vessel Model with ADVENTURE System on the Earth Simulator, *Journal of the Earth Simulator*, Vol. 2, pp. 41–54, 2005.

[6] H. Kawai, M. Ogino, R. Shioya and S. Yoshimura: Performance Tuning of Hierarchical Domain-decomposition Method on Multi-core Architecture, USNCCM10.

[7] Y. Saad, Iterative Methods for Sparse Linear Systems, (SIAM 2003).

A Hybrid CPU-GPU Implementation of Finite Element Method based on the Domain Decomposition Method

Masao Ogino[1], Ryohei Fujise[2], and Hiroshi Kanayama[3]

1 Information Technology Center, Nagoya University, JAPAN
2 Graduate School of Engineering, Kyushu University, JAPAN
3 Faculty of Engineering, Kyushu University, JAPAN
masao.ogino@cc.nagoya-u.ac.jp, fujise@cm.mech.kyushu-u.ac.jp,
kanayama@mech.kyushu-u.ac.jp

1 Introduction

The Domain Decomposition Method (DDM) is well known as an effective parallel finite element method. In particular, the Hierarchical Domain Decomposition Method (HDDM) by Yagawa and Shioya [1] is one of the most effective methods for large-scale problems due to its excellent parallel performance and suitability for various kinds of parallel computers. The HDDM was successfully applied to problems with 100 million Degrees Of Freedom (DOFs) by Shioya and Yagawa [2]. To overcome an ill-conditioned problem, the Balancing Domain Decomposition (BDD) method was proposed by Mandel [3] in which a coarse grid correction is added to the Neumann-Neumann (N-N) preconditioner by DeRoeck and Letallec [4]. The method was further extended to problems with jumps in coefficients by Mandel and Brezina [5], to plate problems by LeTallec et al. [6], to Stokes problems by Goldfeld [7], to Maxwell's problems by Toselli [8], and to structural problems with linear multipoint constraints by Miyamura [9]. The method was also successfully applied to large-scale problems by Shioya et al. [10] and Yamada et al. [11], to a vector supercomputer by Ogino et al. [12], and an inexact balancing method was proposed to solve the issue of the coarse grid correction by Ogino et al. [13]. To guarantee that the coarse matrix of the coarse problem is positive definite and also improve the sparsity pattern of the coarse matrix, the Balancing Domain Decomposition based on Constraints (BDDC) method was proposed by LeTallec et al. [14], and Dohrmann [15].

In recent years, many countries prepare the petascale computing systems. Especially, many supercomputers consisting of multicore processors and Graphical Processing Unit (GPU) are developed. To improve the computational performance on such new computer architecture, linear solvers of subdomain problems of the BDD method on the multicore processors was evaluated by Ogino et al. [16] and Kawai et al. [17], and Papadrakakis et al. [18] presented a hybrid CPU-GPU implementation.

In this paper, to develop a reusable and maintainable parallel finite element analysis system for a GPU-equipped computer system, a hybrid CPU-GPU implementation of the BDD method is presented. In Section 2, DDM algorithm is described, while in Section 3, a CPU-GPU hybrid implementation is presented. In Section 4, some numerical examples are demonstrated. Finally, concluding remarks are presented in Section 5.

2 Domain Decomposition Method

A Schur complement equation

The DDM is based on a non-overlapping domain decomposition and the Schur complement equation. Consider a system of linear algebraic equations arising from a finite element discretization of a linear elliptic and self-adjoint boundary value problem in the domain Ω.

$$\mathbf{A}\,\mathbf{x} = \mathbf{b} \qquad (1)$$

where **A** is the global stiffness matrix, **u** an unknown vector, and **f** a given vector. In elastostatic problems, the matrix **A** is symmetric positive-definite. The domain Ω is decomposed into non-overlapping N subdomains $\Omega^{(1)}$,..., $\Omega^{(N)}$, for which the union of all subdomains boundaries is $\Gamma = \bigcup_{i=1}^{N} \partial\Omega^{(i)} \setminus \partial\Omega_D$. Here, let $\partial\Omega_D$ be the Dirichlet boundary. Next, let $\mathbf{R}^{(i)}$ denote the 0-1 matrix that restricts the global to local DOFs mapping for $\Omega^{(i)}$. Let $\mathbf{x}^{(i)}$ be the unknown vector in $\Omega^{(i)}$, and then $\mathbf{x}^{(i)} = \mathbf{R}^{(i)}\mathbf{x}$. The stiffness matrix and the ginve vector are then described by assembling local them; $\mathbf{A} = \sum_{i=1}^{N} \mathbf{R}^{(i)T}\mathbf{A}^{(i)}\mathbf{R}^{(i)}$, $= \sum_{i=1}^{N} \mathbf{R}^{(i)T}\mathbf{b}^{(i)}$.

Each $\mathbf{x}^{(i)}$ in $\Omega^{(i)}$ is classified as either (B) interface DOFs, if corresponding to the interface of $\Omega^{(i)}$ with other subdomains, or (I) subdomain-interior DOFs, if having only interior DOFs in $\Omega^{(i)}$. With this classification, Eq. (1) is then represented as follows.

$$\begin{bmatrix} \mathbf{A}_{II} & \mathbf{A}_{IB} \\ \mathbf{A}_{IB}^{T} & \mathbf{A}_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{x}_I \\ \mathbf{x}_B \end{bmatrix} = \begin{bmatrix} \mathbf{b}_I \\ \mathbf{b}_B \end{bmatrix} \qquad (2)$$

where subscripts I and B correspond to the interior DOFs and the interface DOFs, respectively, and each block matrice and vectors are as follows.

$$\mathbf{A}_{II} = \begin{bmatrix} \mathbf{A}_{II}^{(1)} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{A}_{II}^{(N)} \end{bmatrix}, \mathbf{A}_{IB} = \begin{bmatrix} \mathbf{A}_{IB}^{(1)}\mathbf{R}_B^{(1)} \\ \vdots \\ \mathbf{A}_{IB}^{(N)}\mathbf{R}_B^{(N)} \end{bmatrix}, \mathbf{A}_{BB} = \sum_{i=1}^{N} \mathbf{R}_B^{(i)T}\mathbf{A}_{BB}^{(i)}\mathbf{R}_B^{(i)},$$
$$\mathbf{x}_I = \begin{bmatrix} \mathbf{x}_I^{(1)} \\ \vdots \\ \mathbf{x}_I^{(N)} \end{bmatrix}, \mathbf{b}_I = \begin{bmatrix} \mathbf{b}_I^{(1)} \\ \vdots \\ \mathbf{b}_I^{(N)} \end{bmatrix}, \ \mathbf{x}_B^{(i)} = \mathbf{R}_B^{(i)}\mathbf{x}_B, \mathbf{b}_B = \sum_{i=1}^{N} \mathbf{R}_B^{(i)T}\mathbf{b}_B^{(i)} \qquad (3)$$

Considering Eq. (2), the Schur complement matrix **S** of $\mathbf{A}_{II}$ of **A** is;

$$\mathbf{S} = \mathbf{A}_{BB} - \mathbf{A}_{IB}^{T}\mathbf{A}_{II}^{-1}\mathbf{A}_{IB} \qquad (4)$$

Here, Eq. (4) is also represented by sum of local Schur complement matrices;

$$\mathbf{S} = \sum_{i=1}^{N} \mathbf{R}_B^{(i)T}\mathbf{S}^{(i)}\mathbf{R}_B^{(i)}, \ \mathbf{S}^{(i)} = \mathbf{A}_{BB}^{(i)} - \mathbf{A}_{IB}^{(i)T}\mathbf{A}_{II}^{(i)-1}\mathbf{A}_{IB}^{(i)} \qquad (5)$$

Using Eq. (4), Eq. (1) can be solved by three steps:

DDM Step 1: Reduction

$$\mathbf{A}_{II}\,\mathbf{x}_I = \mathbf{b}_I \qquad (6)$$

$$\mathbf{g} = \mathbf{b}_B - \mathbf{A}_{IB}^{T}\mathbf{x}_I \qquad (7)$$

DDM Step 2: Solution

$$\mathbf{S}\,\mathbf{x}_{\mathrm{B}} = \mathbf{g} \qquad (8)$$

DDM Step 3: Expansion

$$\widetilde{\mathbf{b}_{\mathrm{I}}} = \mathbf{b}_{\mathrm{I}} - \mathbf{A}_{\mathrm{IB}}\mathbf{x}_{\mathrm{B}} \qquad (9)$$

$$\mathbf{A}_{\mathrm{II}}\,\mathbf{x}_{\mathrm{I}} = \widetilde{\mathbf{b}_{\mathrm{I}}} \qquad (10)$$

In DDM three steps, most of computation time is spent in solving Eq. (8). Equation (8) is called the Schur complement equation, and then can be solved by the iterative methods, such as the Conjugate Gradient (CG) method. Besides, the product of the Schur complement matrix and a vector is needed. The matrix-vector multiplication can be performed without the construction of $\mathbf{S}$ explicitly.

Matrix-Vector Multiplication $\mathbf{q}^{\mathrm{n}} = \mathbf{S}\mathbf{p}^{\mathrm{n}}$

$$\widetilde{\mathbf{b}_{\mathrm{I}}}^{\mathrm{n}} = -\mathbf{A}_{\mathrm{IB}}\mathbf{p}^{\mathrm{n}} \qquad (11)$$

$$\mathbf{A}_{\mathrm{II}}\,\mathbf{x}_{\mathrm{I}}^{\mathrm{n}} = \widetilde{\mathbf{b}_{\mathrm{I}}}^{\mathrm{n}} \qquad (12)$$

$$\mathbf{q}^{\mathrm{n}} = \mathbf{A}_{\mathrm{IB}}^{\mathrm{T}}\mathbf{x}_{\mathrm{I}}^{\mathrm{n}} + \mathbf{A}_{\mathrm{BB}}\mathbf{p}^{\mathrm{n}} \qquad (13)$$

A BDD Preconditioning

When solving Eq. (8), the preconditioned CG method requires the solution of following auxiliary problem in each iteration.

$$\mathbf{M}\,\mathbf{z}^{\mathrm{n}} = \mathbf{r}^{\mathrm{n}} \qquad (14)$$

where $\mathbf{M}$ is a preconditioning matrix, assumed to be a symmetric positive-definite matrix. $\mathbf{z}^{\mathrm{n}}$ is a preconditioned vector, and $\mathbf{r}^{\mathrm{n}}$ is a residual vector in n-th CG iteration. In this paper, the BDD-DIAG preconditioner by Ogino et al. [13] is employed. The BDD-DIAG has a BDD-like algorithm, and its preconditioner is described as follows.

$$\mathbf{M}^{-1} = \left(\mathbf{I} - \mathbf{R}_0^{\mathrm{T}}\mathbf{S}_0^{-1}\mathbf{R}_0\mathbf{S}\right)\mathbf{T}_{\mathrm{DIAG}} \qquad (15)$$

where $\mathbf{R}_0$ is a weighted restriction operator from the interface DOFs to coarse DOFs, $\mathbf{S}_0$ a coarse matrix, $\mathbf{T}_{\mathrm{DIAG}}$ a diagonal-scaling operator.

The preconditioner also requires the product of the Schur complement matrix and a vector. Obviously, Eqs. (11)-(13) can be used.

3 Implementation of DDM on CPU-GPU

The DDM algorithm consists of three type operations of a subdomain-level, an interface-level, and a coarse-level. The subdomain-level operation includes subddomain-independent matrix/vector calculation, and data communication for summation. Next, the interface-level operation includes vector calculation, and data communication for summation. On the other hand, a coarse-level operation requires one matrix calculation in all. In the following, we present how to implement subdomain-level operations on a CPU-GPU computer system.

A MPI-OpenMP Parallel Implementation of Matrix-Vector Multiplication

To perform Eqs. (11)-(13), all MPI processes calculate subdomain-independent matrix/vector operations for assigned subdomains. Subdomain-level operations are as follows:

Step 1: $\tilde{\mathbf{b}}_{\mathrm{I}}^{\mathrm{n(i)}} \Leftarrow \mathrm{SpMV}\left(\mathbf{A}_{\mathrm{IB}}^{(\mathrm{i})}, \mathbf{R}_{\mathrm{B}}^{(\mathrm{i})}\mathbf{p}^{\mathrm{n}}\right)\ , \mathrm{i} = 1, \ldots, \mathrm{N}^{p}$

Step 2: $\mathbf{x}_{\mathrm{I}}^{\mathrm{n(i)}} \Leftarrow \mathrm{Solve}\left(\mathbf{A}_{\mathrm{II}}^{(\mathrm{i})}, \tilde{\mathbf{b}}_{\mathrm{I}}^{\mathrm{n(i)}}\right)\ , \mathrm{i} = 1, \ldots, \mathrm{N}^{p}$

Step 3: $\mathbf{q}^{\mathrm{n(i)}} \Leftarrow \mathrm{SpMV}\left(\mathbf{A}_{\mathrm{IB}}^{(\mathrm{i})\mathrm{T}}, \mathbf{x}_{\mathrm{I}}^{\mathrm{n(i)}}\right) + \mathrm{SpMV}\left(\mathbf{A}_{\mathrm{BB}}^{(\mathrm{i})}, \mathbf{R}_{\mathrm{B}}^{(\mathrm{i})}\mathbf{p}^{\mathrm{n}}\right)\ , \mathrm{i} = 1, \ldots, \mathrm{N}^{p}$

Step 4: $\mathbf{q}^{\mathrm{n}} \Leftarrow \mathrm{AXPY}\left(1.0,\ \mathbf{R}_{\mathrm{B}}^{(\mathrm{i})\mathrm{T}}\mathbf{q}^{\mathrm{n(i)}},\ \mathbf{q}^{\mathrm{n}}\right)\ , \mathrm{i} = 1, \ldots, \mathrm{N}^{p}$

Step 5: $\mathbf{q}^{\mathrm{n}} \Leftarrow \mathrm{Allreduce}(\mathbf{q}^{\mathrm{n}})$

Besides, subdomain steps 1-4 can be done using not only a step-by-step approach but also subdomain-by-subdomain one. From a viewpoint of an effective cache memory use, subdomain-by-subdomain approach is much better. Figure 1 shows a conceptual diagram of a MPI-OpenMP hybrid parallel implementation of the matrix-vector multiplication. From the figure, a dynamic load balancing by OpenMP function is employed. Since the task inside a loop have relatively large granularity, the chunk size is set one. In here, subdomain step 4 may cause a memory access contention of $\mathbf{q}^{\mathrm{n}}$, and critical directive is used. After all assigned subdomain calculation is computed, results are exchanged by MPI function.

```
#pragma omp parallel for schedule(dynamic,1)
for (i = 0; i < N^p; i++) {
   f_I^(i) ⇐ SpMV(A_IB^(i), R_B^(i) p^n)
   u_I^(i) ⇐ Solve(A_II^(i), f_I^(i))
   q^(i) ⇐ SpMV(A_IB^(i)T, u_I^(i)) + SpMV(A_BB^(i), R_B^(i) p^n)
#pragma omp critical
   q^n ⇐ AXPY(1.0, R_B^(i)T q^(i), q^n)
}
q^n ⇐ Allreduce(q^n)
```

Figure 1 A MPI-OpenMP hybrid implementation of DDM

A CPU-GPU Hybrid Implementation of Sudomain-level Operations

From Figure 1, a subdomain is dynamic-assignment to a OpenMP thread. If a computer system has one GPU, one thread performs a controller for the GPU. Figure 2 shows a conceptual diagram of a GPU implementation of one subdomain calculation. From the figure, a data transfer from the host memory to the device memory is needed before *SpMV* and *Solve* functions, and a result should be gotten from the device memory after that. In here, matrices are kept on the device memory during the analysis if there is

enough memory.

In this implementation, the GPU requires two-type calculations of a sparse matrix and vector multiplication (*SpMV*), and to solve a linear system (*Solve*). From viewpoints of reusable and maintainable implementation, the CUBLAS [19] and CUSPARSE [20] libraries are used. Firstly, *SpMV* can be performed by CUSPARSE, e.g. *cusparseDcsrmv* function. On the other hand, a direct method or an iterative method approach can be considered to solve a linear system. To examine these approaches, numerical examples are performed in Section 4.

copy $\mathbf{A}_{\mathrm{IB}}^{(\mathrm{i})}, \mathbf{R}_{\mathrm{B}}^{(\mathrm{i})}\mathbf{p}^{\mathrm{n}}$ *from CPU to GPU*
$\mathbf{f}_{\mathrm{I}}^{(\mathrm{i})} \Leftarrow \mathrm{SpMV}\left(\mathbf{A}_{\mathrm{IB}}^{(\mathrm{i})}, \mathbf{R}_{\mathrm{B}}^{(\mathrm{i})}\mathbf{p}^{\mathrm{n}}\right)$ *on GPU*
copy $\mathbf{f}_{\mathrm{I}}^{(\mathrm{i})}$ *from GPU to CPU*
copy $\mathbf{A}_{\mathrm{II}}^{(\mathrm{i})}, \mathbf{f}_{\mathrm{I}}^{(\mathrm{i})}$ *from CPU to GPU*
$\mathbf{u}_{\mathrm{I}}^{(\mathrm{i})} \Leftarrow \mathrm{Solve}\left(\mathbf{A}_{\mathrm{II}}^{(\mathrm{i})}, \mathbf{f}_{\mathrm{I}}^{(\mathrm{i})}\right)$ *on GPU*
copy $\mathbf{u}_{\mathrm{I}}^{(\mathrm{i})}$ *from CPU to GPU*
copy $\mathbf{A}_{\mathrm{IB}}^{(\mathrm{i})\mathrm{T}}, \mathbf{u}_{\mathrm{I}}^{(\mathrm{i})}, \mathbf{A}_{\mathrm{BB}}^{(\mathrm{i})}, \mathbf{R}_{\mathrm{B}}^{(\mathrm{i})}\mathbf{p}^{\mathrm{n}}$ *from CPU to GPU*
$\mathbf{q}^{(\mathrm{i})} \Leftarrow \mathrm{SpMV}\left(\mathbf{A}_{\mathrm{IB}}^{(\mathrm{i})\mathrm{T}}, \mathbf{u}_{\mathrm{I}}^{(\mathrm{i})}\right) + \mathrm{SpMV}\left(\mathbf{A}_{\mathrm{BB}}^{(\mathrm{i})}, \mathbf{R}_{\mathrm{B}}^{(\mathrm{i})}\mathbf{p}^{\mathrm{n}}\right)$ *on GPU*
copy $\mathbf{q}^{(\mathrm{i})}$ *from CPU to GPU*
#pragma omp critical
$\mathbf{q}^{\mathrm{n}} \Leftarrow \mathrm{AXPY}\left(1.0, \mathbf{R}_{\mathrm{B}}^{(\mathrm{i})\mathrm{T}}\mathbf{q}^{(\mathrm{i})}, \mathbf{q}^{\mathrm{n}}\right)$ *on CPU*

Figure 2 A GPU implementation of one subdomain calculation

4 Numerical Examples

As a test, an elastostatic stress analysis of a quadratic tetrahedral mesh is considered. All computation in this paper is performed on a GPU computer system consisting of Intel Xeon W3670 and NVIDIA Tesla C2075. GCC 4.1.2 and CUDA Toolkit 4.2 are used.

A Direct Method Approach

To solve Eq. (12) on the GPU, a direct method is considered as follows;

Step I: $\mathbf{A}_{\mathrm{II}}^{(\mathrm{i})} = \mathbf{L}_{\mathrm{II}}^{(\mathrm{i})}\mathbf{L}_{\mathrm{II}}^{(\mathrm{i})\mathrm{T}}$ Cholesky factorization on CPU by an in-house code

Step II: $\mathbf{L}_{\mathrm{II}}^{(\mathrm{i})}\mathbf{y}_{\mathrm{I}}^{(\mathrm{i})\mathrm{n}} = \widetilde{\mathbf{b}}_{\mathrm{I}}^{(\mathrm{i})\mathrm{n}}$ Forward substitution on GPU by *cusparseDcsrsv_solve*

Step III: $\mathbf{L}_{\mathrm{II}}^{(\mathrm{i})\mathrm{T}}\mathbf{x}_{\mathrm{I}}^{(\mathrm{i})\mathrm{n}} = \mathbf{y}_{\mathrm{I}}^{(\mathrm{i})\mathrm{n}}$ Backward substitution on GPU by *cusparseDcsrsv_solve*

Table 1 shows a computational time of *Step II* and *III*. From the table, these calculations of GPU is rather slow compared with CPU.

Table 1 Computational time [sec] of *cusparseDcsrsv_solve*

DOFs	1 K	6 K	10 K
CPU	2.2e-4	8.8e-3	1.8e-2
GPU	1.3e-2	2.3e-1	4.6e-1

An Iterative Method Approach

To solve Eq. (12) on the GPU, an iterative method is considered. In this model, since a coefficient matrix of Eq. (12) is a symmetric positive definite, the Conjugate Gradient (CG) method is used. A matrix-vector multiplication and vector operations are implemented by CUSPARSE and CUBLAS, respectively. A sample code of the CG method is shown in Figure 3.

```
int advlas_solve_by_cg_cuda(...) {
...
  cudaStat = cudaMalloc((void**)&devValA, nnz*sizeof(devValA[0]));
  cudaStat = cudaMalloc((void**)&r, n*sizeof(r[0]));
  cudaStat = cudaMalloc((void**)&p, n*sizeof(p[0]));
...
  cudaStat = cudaMemcpy(devValA, csrValA, (size_t)(nnz*sizeof(devValA[0])),
                                    cudaMemcpyHostToDevice);
...
 while (1) {
    cusparseDcsrmv(sHandle, CUSPARSE_OPERATION_NON_TRANSPOSE, n, n, nnz, &c_one,
              descrA, devValA, devRowPtrA, devColIndA, p, &c_zero, q);
    cublasDdot(bHandle, n, p, 1, q, 1, &alpha);
    alpha = rr / alpha;
    cublasDaxpy(bHandle, n, &alpha, p, 1, devU, 1);
    alpha *= -1.0;
    cublasDaxpy(bHandle, n, &alpha, q, 1, r, 1);
    cublasDdot(bHandle, n, r, 1, r, 1, &rr_new);
...
  }
  cudaStat = cudaMemcpy(u, devU, (size_t)(n*sizeof(u[0])),
                                    cudaMemcpyDeviceToHost);
...
}
```

Figure 3 A sample code of the CG method using CUDA libraries

Table 2 shows results of computation time using a symmetric or general matrix type. Problem size is 13K or 129K, and the Jacobi preconditioner is used. As previously noted, the coefficient matrix of this model is symmetric, however, a general type which stores all nonzero entries was faster than a symmetric type on the GPU. It is reason that symmetric type requires calculating the transpose of a matrix, and the CPU can utilize a cache memory for it, however, CUSPARSE on the GPU is rather slow. Next, Table 3 shows results of the Jacobi preconditioner or the Gauss-Seidel (GS) preconditioner. In fact, the GS preconditioner reduced the iteration counts compared with the Jacobi preconditioner, but it takes much more time on the GPU. It is caused by the same reason of the direct method approach. Consequently, a combination of the general matrix type and the Jacobi preconditioner is the fastest implementation in the CG method approach.

Table 2 Computational time using different matrix type

	Matrix type	13 K DOFs [sec]	129 K DOFs [sec]
CPU	symmetric	4.36	53.35
CPU	general	5.68	72.34
GPU	symmetric	17.27	83.76
GPU	general	1.28	9.71

Table 3 Computational time using different preconditioner in solving 13K DOFs

	Matrix type	Preconditioner	Iteration counts	Time [sec]
CPU	symmetric	Jacobi	4,039	4.36
CPU	symmetric	GS	1,469	3.74
GPU	general	Jacobi	4,039	1.28
GPU	general	GS	1,469	21.01

Computational performances of the BDD-DIAG method

Finally, a CPU-GPU hybrid computation was performed in a large-scale analysis. Based on the above result, the CG method with the general matrix type and the Jacobi preconditioner is employed for the GPU in solving Eq. (12). On the other hand, a combination of the symmetric type and the GS preconditioner is used for the CPU. Figure 4 plots results of number of subdomains vs. computational time. The red line means results of a CPU-GPU hybrid computation, and the blue line is CPU only. From the figure, CPU-GPU hybrid computation was 1.5 times faster than CPU only in case of 10K or more DOFs per subdomain.

Here, the green line in Figure 4 means results of the direct method approach on the CPU. A CPU-optimized code shows the best performance of computational time.

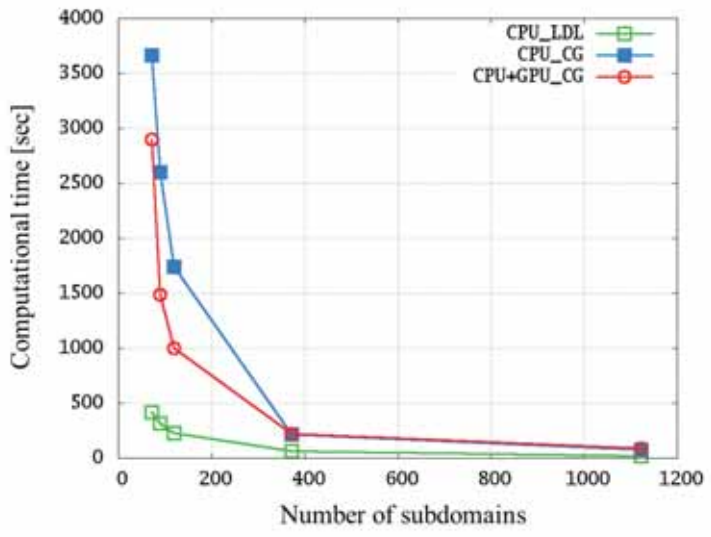


Figure 4 Number of subdomains vs. computational time

5 Conclusion

We have proposed a CPU-GPU hybrid implementation of the DDM using CUBLAS and CUSPARSE libraries. A subdomain loop is parallelized by OpenMP, and one thread performs a controller of the GPU. On the GPU, a sparse matrix and vector multiplication is performed by CUSPARSE, and a linear system is solved by an iterative method using CUBLAS and CUSPARSE.

So far as a CPU-optimized code and all processor core of the CPU are used, CPU only is faster than CPU-GPU hybrid. However, in solving huge-scale problems, the DOFs per one subdomain is expected to become even larger. In that case, GPU would be an efficient solution.

Acknowledgements

This research was supported by CREST, JST. One of the authors (M. Ogino) was also supported by KAKENHI 24760062.

References

[1] Yagawa, G., Shioya, R., Parallel finite elements on a massively parallel computer with domain decomposition, *Comput. Syst. Eng.*, 4 (1994), pp.495-503.
[2] Shioya, R., Yagawa, G., Parallel finite element analysis of 100 million dofs based on the hierarchical domain decomposition method (in Japanese), *Trans. JSCES*, 3 (2001), pp.201-206.
[3] Mandel, J., Balancing domain decomposition, *Commun. Numer. Methods Eng.*, 9 (1993), pp.223-241.
[4] DeRoeck, Y., LeTallec, P., Analysis and test of a local domain decomposition preconditioner, *DDM4*, (1991).
[5] Mandel, J., Brezina, M., Balancing domain decomposition for problems with large jumps in coefficients, *Math. Comput.*, 65 (1996), pp.1387-1401.
[6] Mandel, J., Brezina, M., Balancing domain decomposition: Theory and performance in two and three dimensions, *UCD/CCM Report*, 2 (1993).
[7] Goldfeld, P., Balancing Neumann-Neumann for (in)compressible linear elasticity and (generalized) stokes -parallel implementation, *DDM14*, (2002), pp.209-216.
[8] Toselli, A., Neumann-Neumann methods for vector field problems, *Electron. Trans. Numer. Anal.*, 11 (2000), pp.1-24.
[9] Miyamura, T., Incorporation of multipoint constraints into the balancing domain decomposition method and its parallel implementation, *Int. J. Numer. Methods Eng.*, 69(2) (2006), pp.326-346.
[10] Shioya, R., Kanayama, H., Tagami, D., Ogino, M., 3D large scale structural analysis using a balancing domain decomposition method (in Japanese), *Trans. JSCES*, 2000 (2000), p. 20000017.
[11] Yamada, T., Ogino, M., Yoshimura, S., Prediction and numerical validation of optimal number of subdomains on balancing domain decomposition method (in Japanese), *Trans. JSCES*, 2009 (2009), p.20090014.
[12] Ogino, M., Shioya, R., Kawai, H., Yoshimura, S., Seismic response analysis of nuclear pressure vessel model with adventure system on the Earth Simulator, *J. Earth*

Simulator, 2 (2005), pp. 41-54.
[13] Ogino, M., Shioya, R., Kanayama, H., An inexact balancing preconditioner for large-scale structural analysis, *J. Comput. Sci. Technol.*, 2-1 (2008), pp.150-161.
[14] LeTallec, P., Mandel, J., Vidrascu, M., A Neumann-Neumann domain decomposition algorithm for solving plate and shell problems, *SIAM J. Numer. Anal.*, 35 (1998), pp.836-867.
[15] Dohrmann, C., A preconditioner for substructuring based on constrained energy minimization, *SIAM J. Sci. Comput.*, 25 (2003), pp.246-258.
[16] Ogino, M., Kawai, H., Shioya, S., Yoshimura, S., Parallel implementation of a balancing domain decomposition method for multi-core processors, *BWCCA2010*, (2010), pp.56-61
[17] Kawai, H., Ogino,M., Shioya, R., Yoshimura, S., , Large scale elasto-plastic analysis using domain decomposition method optimized for multi-core cpu architecture, *Key Eng. Mater.*, 462-463 (2011), pp.605-610.
[18] Papadrakakis, M., Stavroulakis, G., Karatarakis, A., A new era in scientific computing: Domain decomposition methods in hybrid CPU/GPU architectures, *Comput. Methods Appl. Mech. Eng.*, 200 (2011), pp.1490-1508.
[19] CUBLAS, http://developer.nvidia.com/cuda/cublas
[20] CUSPARSE, http://developer.nvidia.com/cuda/cusparse

ADVENTURE_Thermal-An Open Source Module for Large Scale Heat Conduction Problems

A.M.M.Mukaddes[1], Masao Ogino[2], and Ryuji Shioya[1]

1 Centre for Computational Mechanics Research (CCMR), Toyo University, JAPAN
2 Information Technology Centre, Nagoya University, JAPAN
mukaddes1975@gmail.com, masao.ogino@cc.nagoya-u.ac.jp, shioya@toyo.jp

1 Introduction

An open source module has been developed to solve large scale heat conduction problems in the parallel computer. Both steady and unsteady heat conduction problems with all types of boundary conditions can be solved using this module. The basic algorithm is the Hierarchical Domain Decomposition Method (HDDM) [1]. In this method, the whole domain is divided into parts and then each part is divided into subdomains. Finite element analysis is performed on each subdomain. The Conjugate Gradient (CG) method is used to solve the interface problem that is constituted by share nodes of subdomains. The performance of the domain decomposition method depends on the number of CG iterations and the computation time per iteration.

A suitable preconditioner may decrease the number of CG iterations. A simplified diagonal scaling is used in the CG method [2]. Balancing Domain Decomposition (BDD) [3, 4] is also implemented as another efficient option. This method is shown very efficient for large scale scalar problems [4]. BDD also shows good performance in the structural analysis [5] and flow problems [6]. Though BDD reduces the number of CG iterations and computation time, it suffers from the problem of high storage. A Diagonal based Inexact Balancing Domain Decomposition (IBDD-DIAG) [7] is implemented, which requires less storage than the BDD method.

CG method that is employed to solve the interface problem uses the Sparse Matrix Vector multiplication (SpMxV) as its basic operation. Sparse matrices, by definition are populated by many zeroes and thus special storage schemes are used to enable efficient storage and computational operations. These representations usually store the non-zero values of the matrix with additional indexing information about the position of these values [8]. The non-zero sparse matrix storage formats used in this module are Compressed Sparse Row (CSR) [8], Coordinate storage (COO), Compressed Sparse Column (CSC), Incremental Compressed Sparse Row (ICSR) [9] , Block Compressed Sparse Row (BCSR)[10], Modified Compressed Row (MSR) [11,12] and Skyline Storage (SKY). The COO format suffers the problem of using diagonal elements for the special purpose. A diagonal based coordinate storage format is proposed here to overcome this issue. All CSR type storage formats outperform the skyline method in terms of storage and computation time. Again in the construction part of the BDD type preconditioners, matrix vector multiplication is necessary thus sparse matrix storage schemes are used there too.

For the time integration of unsteady heat conduction problems, backward Euler method

and Crank-Nicolson method are used. These are not completely stable method but the stability depends on the time difference. The temperature distribution from this module can be used to measure the thermal stress using another ADVENTURE module, called ADVENTURE_Solid [13].

2 Heat Conduction Problems

A heat conduction problem is considered in a domain Ω. Defining $\bar{f}$ as the internal heat generation, $\bar{u}$ temperature applied on the boundary Γ_u, Q the heat flux applied on the boundary Γ_Q and $h(u-u_c)$ the convection heat transfer on the boundary Γ_c, the fundamental equations of this heat conduction problem are given by:

$$\begin{aligned} & q = -k\ grad\ u && in\ \Omega \\ & \rho C\dot{u} = -div\ q + \bar{f} && in\ \Omega \\ & q \cdot n = \bar{Q} && in\ \Gamma_Q \\ & q \cdot n = h(u-u_c) && in\ \Gamma_c \\ & u = \bar{u} && in\ \Gamma_u \end{aligned} \tag{1}$$

where u is the temperature, q the heat flux, k the thermal conductivity, h the convective heat transfer coefficient, u_c the external temperature and n an outer normal unit vector, ρ the density, C the specific heat respectively. The finite element (quadratic tetrahedral) discretization of (1) yields a linear system of the form

$$Ku = f \tag{2}$$

where

K is the global stiffness matrix,

u is an unknown vector of temperature,

and f is a known vector.

3 Domain Decomposition Method

The domain decomposition method decomposes the domain Ω into N non-overlapping subdomains, $\{\Omega_i\}_{i=1,\ldots,N}$. Thus the stiffness matrix K of equation (2) could be generated by subassembly:

$$K = \sum_{i=1}^{N} R^{(i)} K^{(i)} R^{(i)T} \tag{3}$$

where $R^{(i)T}$ is the 0-1 matrix which translates the global indices of the nodes into local (subdomain) numbering. Denoting $u^{(i)}$ as the vector corresponding to the nodes in $\Omega^{(i)}$, it can be expressed as $u^{(i)} = R^{(i)T}u$. Each $u^{(i)}$ is split into degrees of freedom $u_B^{(i)}$, which correspond to $\partial\Omega^{(i)}$, called interface degrees of freedom, $u_I^{(i)}$ for interior degrees of freedom and $u_T^{(i)}$ for essential boundary conditions (temperature for heat conduction problem). The subdomain matrix $K^{(i)}$, vector $u^{(i)}$ are then split accordingly:

$$K^{(i)} = \begin{pmatrix} K_{II}^{(i)} & K_{IB}^{(i)} & K_{IT}^{(i)} \\ K_{BI}^{(i)} & K_{BB}^{(i)} & K_{BT}^{(i)} \\ K_{TI}^{(i)} & K_{TB}^{(i)} & K_{TT}^{(i)} \end{pmatrix}, \qquad \begin{Bmatrix} u_I^{(i)} \\ u_B^{(i)} \\ u_T^{(i)} \end{Bmatrix}. \tag{4}$$

Similarly equation (2) can be written as

$$\begin{bmatrix} K_{II} & K_{IB} & K_{IT} \\ K_{BI} & K_{BB} & K_{BT} \\ K_{TI} & K_{TB} & K_{TT} \end{bmatrix} \begin{Bmatrix} u_I \\ u_B \\ u_T \end{Bmatrix} = \begin{Bmatrix} f_I \\ f_B \\ f_T \end{Bmatrix}. \tag{5}$$

After eliminating the interior degrees of freedom, problem (5) is reduced to a problem on interface

$$Su_B = g \tag{6}$$

where the Schur complement $S = \sum_{i=1}^{N} R_B^{(i)} S^{(i)} R_B^{(i)T}$ is assumed to be positive definite, u_B is the vector of the unknown variables on the interface, g is a known vector and $S^{(i)}$ are the local Schur complements of subdomain $i = 1,\ldots,N$, assumed to be positive semi-definite. The problem (6) is solved by the CG method. The flow of algorithm of the basic domain decomposition method is shown below.

Step 0: Initialize: $u_B^{(0)}$

(1a) Find: $u_I^{(0)}$, $\begin{bmatrix} K_{II} & K_{IB} & K_{IT} \end{bmatrix} \begin{Bmatrix} u_I^{(0)} \\ u_B^{(0)} \\ u_T^{(0)} \end{Bmatrix} = f_I$

(1b) Calculate: $g^{(0)} = w^{(0)} = \begin{bmatrix} K_{BI} & K_{BB} & K_{BT} \end{bmatrix} \begin{Bmatrix} u_I^{(0)} \\ u_B^{(0)} \\ u_T \end{Bmatrix} - f_B$

Step 2: (iteration) (2a) Find $v^{(k)}$: $\begin{bmatrix} K_{II} & K_{IB} & K_{IT} \end{bmatrix} \begin{Bmatrix} v^{(k)} \\ w^{(k)} \\ 0 \end{Bmatrix} = 0$

(2b) Find: $Sw^{(k)} = \begin{bmatrix} K_{BI} & K_{BB} & K_{BT} \end{bmatrix} \begin{Bmatrix} v^{(k)} \\ w^{(k)} \\ 0 \end{Bmatrix}$

2(c) Update for each k iteration: $u_B^{(k)}, g^{(k)} and\ w^{(k)}$

$$\rho^{(k)} = \frac{\left(g^{(k)} \cdot g^{(k)}\right)}{\left(w^{(k)} \cdot Sw^{(k)}\right)}$$

$$u_b^{(k+1)} = u_b^{(k)} - \rho^{(k)} w^{(k)}$$

$$g^{(k+1)} = g^{(k)} - \rho^{(k)} Sw^{(k)}$$

$$If \left(\frac{\left(g^{(k+1)} \cdot g^{(k+1)}\right)}{\left(g^{(0)} \cdot g^{(0)}\right)} \right) \prec \varepsilon \ /* convergence */ /$$

$$\gamma^{(k)} = \frac{\left(g^{(k+1)} \cdot g^{(k+1)}\right)}{\left(g^{(k)} \cdot g^{(k)}\right)}$$

$$w^{(k+1)} = \gamma^{(k)} w^{(k)} + g^{(k+1)}$$

We remark that the Schur complement matrix is not constructed explicitly, though the multiplication of a vector with the Schur complement is required in the CG method. The action of the matrix S on a typical vector x can be implemented by using the block elements of the matrix $K^{(i)}$. Again the

$$\begin{aligned} y &= S \cdot x \\ &= \left\{ \sum_{i=1}^{N} R_B^{(i)} S^{(i)} R_B^{(i)T} \right\} \cdot x \\ &= \sum_{i=1}^{N} R_B^{(i)} S^{(i)} x^{(i)} \\ &= \sum_{i=1}^{N} R_B^{(i)} y^{(i)} \end{aligned} \tag{7}$$

where, $$y^{(i)} = S^{(i)} x^{(i)} = \left(K_{BB}^{(i)} - K_{IB}^{(i)T} \left(K_{II}^{(i)} \right)^{-1} K_{IB}^{(i)} \right) x^{(i)}. \tag{8}$$

action of $S^{(i)}$ on a vector can be implemented by solving a *Dirichlet* problem on $\Omega^{(i)}$ and it needs subdomain wise matrix-vector multiplication twice in each iteration as shown in equation (8).

A Balancing Domain Decomposition (BDD) method is implemented in the CG method to accelerate the CG iteration.

3.1 Hierarchical Domain Decomposition Method

For constructing the DDM algorithms for parallel computers, a good principle is to divide the original domain into parts, which are further decomposed into smaller subdomains as shown in figure-1. This is called Hierarchical Domain Decomposition Method (HDDM) which is first introduced in [1].

This hierarchically structured DDM classifies processors into 3 groups, 'Grand Parent', 'Parent' and 'Child'. One of the processors is assigned as Grand Parent, a few as Parent, and others as Child. The number of processors assigned as Parent is the same as that of parts. The number of Child processors can be varied; and it affects the parallel performance.

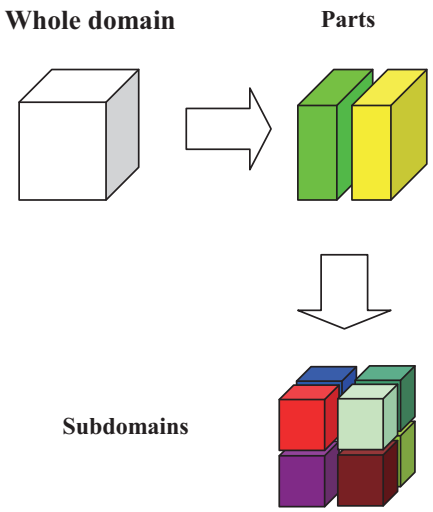


Figure-1 Hierarchical domain decomposition method

4 Balancing Domain Decomposition

The BDD preconditioner proposed by Jan Mandel is constructed by solutions of the local Neumann-Neumann problems on subdomains coupled with a coarse problem in a coarse space. The BDD preconditioner is of the form:

$$M_{BDD}^{-1} = Q_c + (I - Q_c S) Q_l (I - S Q_c) \qquad (9)$$

where Q_l is the local level part [3] and Q_c is the coarse level part of the preconditioner.

4.1 Coarse Level

The application of the coarse term $Q_c = R_0 (R_0^T S R_0)^{-1} R_0^T$ amounts to the solution of a coarse problem whose coefficient matrix is $S_0 = R_0^T S R_0$. The operator R_0 translates the coarse degrees of freedom to the corresponding global degrees of freedom and is defined by

$$R_0 = \left[R_B^{(1)} D^{(1)} Z^{(1)}, ..., R_B^{(N)} D^{(N)} Z^{(N)} \right] . \qquad (10)$$

For the scalar heat conductive problem, $Z^{(i)}$ is a column constant vector and can be defined by

$$Z^{(i)} = (1...1)^T \qquad (11)$$

where the number of element "1" is for each interface point in subdomain i. The operator R_0 is a n by N matrix, where n is the dimension of S.

The implementation of the BDD preconditioner (9) goes as follows :

Step 1: Balance the original residual by solving the coarse problem for an unknown vector $\lambda \in \Re^N$ *:*

$$S_0 \lambda = R_0^T r . \qquad (12)$$

Step 2: Set

$$s = r - S R_0 \lambda . \qquad (13)$$

Step 3: Solve Neumann-Neumann problems and average these results

$$\bar{u} = \sum_{i=1}^{N} R_B^{(i)} D^{(i)} S^{(i)+} D^{(i)T} R_B^{(i)T} s . \qquad (14)$$

Step 4: Compute

$$\bar{s} = r - S\bar{u} . \qquad (15)$$

Step 5: Solve the coarse problem again for an unknown vector $\mu \in \Re^N$

$$S_0 \mu = R_0^T \bar{s} . \qquad (16)$$

Step 6: Find the preconditioned vector

$$z = \bar{u} + R_0 \mu . \qquad (17)$$

The equation (9) can also be expressed as:

$$M_{BDD}^{-1} = \left(P + (I - P) Q_l S (I - P) \right) S^{-1} \qquad (18)$$

where $P = Q_c S$ is the $S-$orthogonal projection onto the coarse space.

The construction of the BDD preconditioner uses the matrix vector multiplication in

equation (13) and (15) where sparse matrix storages schemes are used.

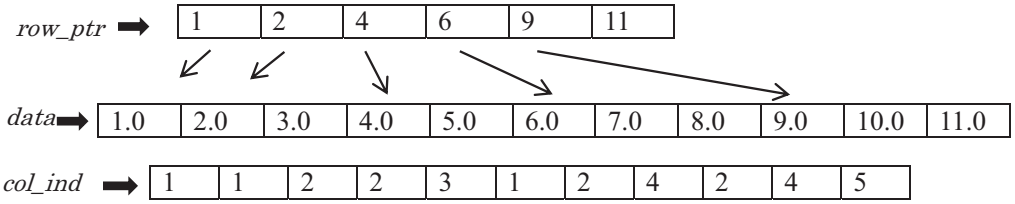
5. Sparse Matrix Storage Schemes

In order to take the advantage of the large number of zero elements, special formats are used to store sparse subdomain matrices. The main goal is to represent only the non-zero elements (*nnz*) considering the memory requirements and the computation time. Several sparse matrix storage formats implemented in the ADVNTURE_Thermal module are discussed below.

Compressed Sparse Row (CSR)

Compressed sparse row format is popular and the most general purpose storage format for the sparse matrix. The elements are stored using three arrays: *data*, *row_ptr* and *col_ind*. The array *data* of length *nnz* contains the non-zero elements of *A* row by row, *col_ind* of length *nnz* contains the column indices which correspond to the non-zero elements in the vector *data*. The integer vector *row_ptr* of length *nrows* contains the pointers to the beginning of each row in the array *data* and *col_ind*. With the *row_ptr* array we can easily compute the number of non-zero elements in the i^{th} row as *row_ptr*[i+1] - *row_ptr*[i]. The CSR representation of an example symmetric matrix *A*:

$$A = \begin{bmatrix} 1.0 & 2.0 & 0.0 & 6.0 & 0.0 \\ 2.0 & 3.0 & 4.0 & 7.0 & 9.0 \\ 0.0 & 4.0 & 5.0 & 0.0 & 0.0 \\ 6.0 & 7.0 & 0.0 & 8.0 & 10.0 \\ 0.0 & 9.0 & 0.0 & 10.0 & 11.0 \end{bmatrix}$$



The working set of the CSR format:

$$ws = (nnz \times (idx\_b + val\_b) + (nrows + 1) \times idx\_b)$$

where *nnz* is number of non-zeros, *idx_b* and *val_b* are bytes required for integer value and floats and *nrows* is the number of rows.

Coordinate Storage (COO)

The simplest sparse matrix storage structure is COO. It uses three arrays of length *nnz* to store the sparse matrix: *data*, *row_ind* and *col_ind.* The array *data* stores the non-zero

elements of the sparse matrix. The *row_ind* and *col_ind* store the row indices and column indices of the corresponding element. The COO storage format of the example matrix A is shown below.

data	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0
row_ind	1	2	2	3	3	4	4	4	5	5	5
col_ind	1	1	2	2	3	1	2	4	2	4	5

The working set of the COO format:

$$ws = (nnz \times (2 \times idx\_b + val\_b))$$

Diagonal Coordinate Storage (DCOO)

It uses one array of length *nnz* to store the sparse matrix: *data*, and two arrays of length *nnz-1* to store the row and column indices: *row_ind* and *col_ind.* The array *data* stores the diagonal elements first and then stores off-diagonal elements. The *row_ind* and *col_ind* store the row indices and column indices of the off-diagonal elements only. The DCOO storage format of the example matrix *A* is shown below.

data	1.0	3.0	5.0	8.0	11.0	2.0	4.0	6.0	7.0	9.0	10.0
row_ind						2	3	4	4	5	5
col_ind						1	2	1	2	2	4

The working set of the DCOO format:

$$ws = (nnz \times valu\_b + 2 \times (nnz - nrows) \times idx\_b)$$

Compressed Sparse Column (CSC)

In this sparse matrix storing format the elements are stored using three arrays: *data*, *col_ptr* and *row_ind*. The array *data* of length *nnz* contains the non-zero elements of *A* column by column, *row_ind* of length *nnz* contains the row indices which correspond to the non-zero elements in the array *data*. The integer array *col_ptr* of length *nrows* contains the pointers to the beginning of each column in the array *data* and *row_ind*. With the *col_ptr* array we can easily compute the number of non-zero elements in the i^{th} column as *col_ptr*[i+1] - *col_ptr*[i].

col_ptr	1	4	8	9	11	11					
data	1.0	2.0	6.0	3.0	4.0	7.0	9.0	5.0	8.0	10.0	11.0
row_ind	1	2	4	2	3	4	5	3	4	5	5

The working set of the CSC format:

$$ws = (nnz \times (idx\_b + val\_b) + (ncols + 1) \times idx\_b)$$

Modified Sparse Row (MSR)

The modified sparse row format has only two parallel arrays of equal length (*nnz +1*): A real array *data* and an integer array *col_ind.* The first *n* position in data contains the diagonal elements of the matrix in order. The position *n+1* of the array data is not used, or may sometimes be used to carry other information concerning the matrix. Starting at position *n+1*, the non-zero elements of data excluding the diagonal elements, are stored by row. The *n+1* first position of *col_ind* contains the pointer to the beginning of each row in data. The rest position of *col_ind* represents its column indices which corresponds to the non-zero elements (off-diagonal) in the array data. Thus for matrix *A*, the two arrays are shown in below.

data	1.0	3.0	5.0	8.0	11.0		2.0	4.0	6.0	7.0	9.0	10.0

col_ind	1	2	4	6	9	11	1	2	1	2	2	4

The restriction of MSR method is that principal diagonal element of the coefficient matrix must be non-zero.

The working set of the MSR format:

$$ws = (nnz + 1) \times (idx\_b + val\_b)$$

Variable Block Sparse Row (VBR)

The idea of this format is to exploit the non-zeros in contiguous locations by packing them. Unlike fixed-size blocking, the blocks will have variable lengths. As in fixed size blocking, if we know the column index of the first non-zero in a block, then we will also know the column indices of all its other non-zeros.This storage format requires an array *nz_ptr* (of length the number of blocks) in addition to the other three arrays used in CSR: an array *data* stores the non-zero elements of *A* row by row, an array *col_ind* (of length the number of blocks) stores the column number of the first non-zero for each block, and an array *row_ptr* (of length the number of rows) to point to the position where block of each row starts. The last element of the *row_ptr* is the number of blocks. The array *nz_ptr* stores the location of the first non-zero of each block in the array *data.* An example of VBR is

nz_ptr	1	2	4	6	8	9	10

data	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0

col_ind	1	1	2	1	4	2	4

row_ptr	1	2	3	4	6	7

The working set of the VBR format:

$$ws = (nnz \times val_b + 2 \times nblocks \times idx_b) + nrows \times idx\_b)$$

Incremental Compressed Sparse Row (ICSR)

The incremental compressed sparse row is a variant of MSR. Instead of 1D index itself, the difference with the 1D index of the previous nonzero is stored, as an increment. This storage format has two arrays with the following function: *data* a real array of length *nnz +1,* stores non-zero elements of matrix *A* and *incr* an integer array of length *nnz + 1* stores the increment with the previous non-zero. ICSR representation of matrix *A* is shown as:

data	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0
incr	0	1	1	2	1	1	1	2	2	2	1

The working set of the ICSR format:

$$ws = (nnz \times (idx\_b + val\_b)$$

Skyline or Variable Band (SKY)

The Skyline representation becomes popular for direct solvers especially when pivoting is not necessary. For symmetric matrices, this representation only stores the lower triangular matrix and hence requires half storage space. The matrix elements are stored using three arrays: *data, row_ptrn, col_ind.* The array *data* stores the elements of *A* row by row, *col_ind* contain column number of first element of each row and *row_ptr* array points to the start of every row.

row_ptr	1	2	4	6	9

data	1.0	2.0	3.0	4.0	5.0	6.0	7.0	0.0	8.0	9.0	0.0	10.0	11.0

col_ind	1	1	2	1	2

This storage format stores some zero elements while other methods explained above do not.

5. Performance Evaluation

5.1 Model description

The model used to investigate the performance of ADVENTURE_Thermal in different aspects is High Temperature Test Reactor (HTTR) shown in Figure-2. The model parameters are given in Table-1.

Figure-2 High Temperature Test Reactor (HTTR)

Table-1 Parameters of HTTR

Model	meshing points	elements	subdomains	Parts	processors
Model 1	1,893,340	1,167,268	4800	6	6
Model 2	13,853,784	9,338,144	76800	96	96

5.2 Performance of Balancing Domain Decomposition

Balancing Domain Decomposition method has been implemented in the ADVENTURE_Thermal module. Implementation of the method makes the module faster. The CG method converges rapidly with the BDD method as shown in figure-3. The results depicted in the figure-4 shows that BDD method is independent of number of subdomains. The method is compared with simplified diagonal scaling and CG method without a preconditioner. Variations of the BDD method are also added in the module as well as investigated. BDD is found the efficient solver for the large scale scalar problems. BDD-DIAG and IBDD-DIAG are also independent of number of subdomains as shown in figure-4.

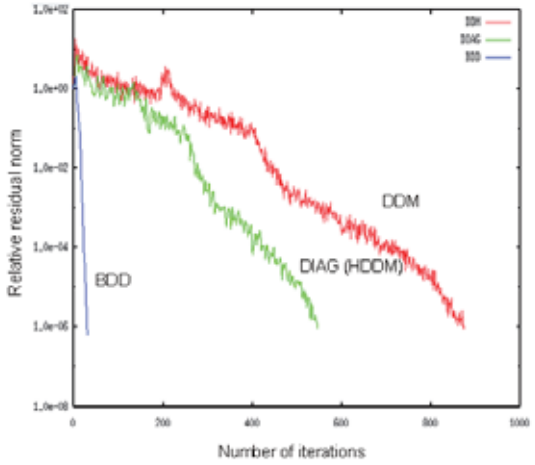


Figure-3 Convergence behavior of CG method

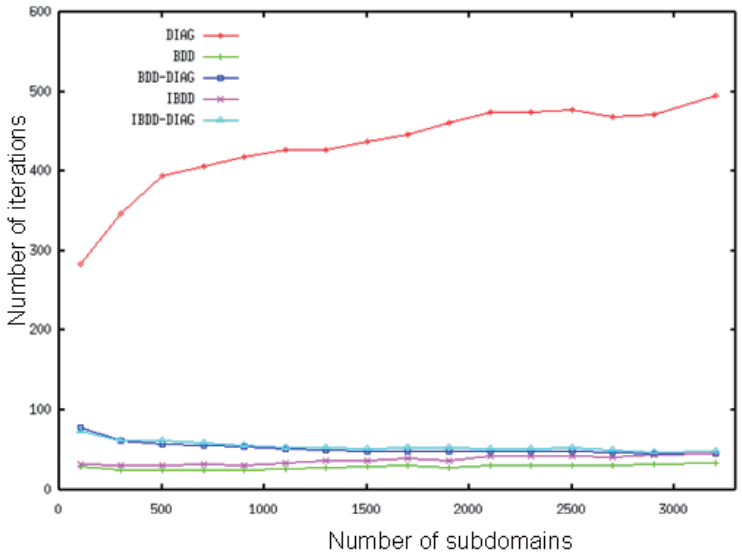


Figure-4 Independency of BDD type methods on number of subdomains

5.3 Performance of Sparse Matrix Storage Schemes

The performance of the ADVENTURE_Thermal with different sparse matrix storage schemes is shown in figure-5, figure-6 and figure-7. CSR type storage schemes reduce the computation time and memory compared to the skyline storage as shown in figure-5 and table-2. The model is analyzed in the FX10 machine of University of Tokyo. A large scale problem (model-2) is analyzed using the 96 processors of FX10 and results are shown in figure-6 and table-2. Though skyline method shows good performance in terms of floating point operations per second it takes more time compared to the CSR type storage schemes. In FX10 machine, each node has 16 processors. 6 processors of six nodes and 6 processors of one node are used to analyze the model-1. Results are shown in figure-5 and figure-6. DCOO shows better performance compared to COO but still it is slower than CSR. In COO storage schemes diagonal elements should be stored in a separate array that is concern of the memory.

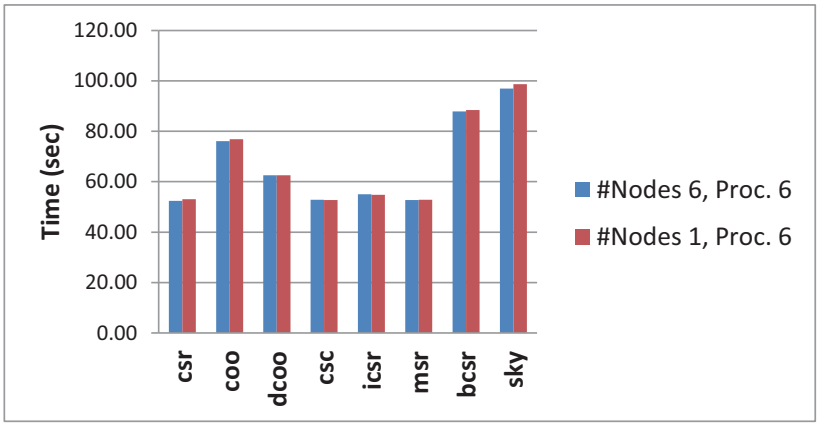


Figure-6 Computation time of different sparse storage schemes (model-1, SpMxV)

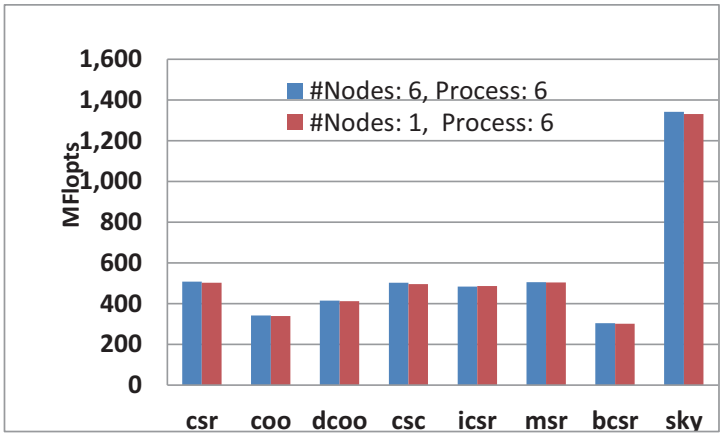


Figure-7 Floating point operations for difference storage schemes (model-1, SpM x V)

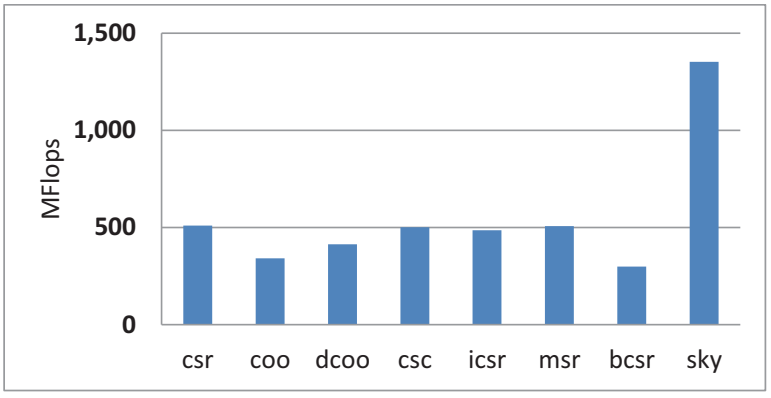


Figure-8 Flops of SpMxV for different storage schemes (model-2, SpMxV)

Table-2 Total Computation time, memory and Mflops-peak ratio (model-2)

	SpMxV Time (sec)	Total Time (sec)	Memory (MB)/Proc.	SpM x V Mflops/peak
CSR	35	172	100	3.44
COO	51	188	109	2.31
DCOO	42	183	105	2.79
CSC	37	175	100	3.39
ICSR	37	174	990	3.28
MSR	35	173	996	3.43
BCSR	60	199	955	2.02
SKY	68	213	166	9.14

6 Conclusions

A module to solve large scale heat conduction problems has been developed using the domain decomposition method. An efficient preconditioner has been implemented which accelerates the iterative domain decomposition method. Several sparse matrix storage schemes are used in the matrix-vector multiplication operation of the conjugate gradient method. CSR type sparse matrix storage schemes are found as efficient schemes. User can solve large scale steady and unsteady heat conduction problems in different parallel computing environments using the developed module.

Acknowledgement

This research is financially supported by one of JST, CREST projects named “Development of Numerical Library Based on Hierarchical Domain Decomposition for Post Petascale Simulation”.

References

1. Yagawa, G. and Shioya, R., Parallel Finite Elements on a Massively Parallel Computer with Domain Decomposition, Computing Systems in Engineering, 4:4-6 (1993), p.495-503.

2. Shioya, R., Kanayama, H., Mukaddes, A. M. M. and Ogino, M., Heat Conductive Analysis with Balancing Domain Decomposition, Theoretical and Applied Mechanics, Vol. 52(2003), p.43-53.

3. Mandel, J., Balancing Domain Decomposition, Comm. on Numerical Methods in Engineering, Vol. 9 (1993), p. 223-241.

4. Mukaddes, A.M.M., Ogino, M., Kanayama, H. and Shioya, R., A Scalable Balancing Domain Decomposition Based Preconditioner for Large Scale Heat Transfer Problems, submitted to JSME International Journal, Series B, Vol. 49, No. 2, pp. 533-540.

5. Shioya, R., Ogino, M., Kanayama, H. and Tagami, D., Large Scale Finite Element Analysis with a Balancing Domain Decomposition Method, Key Engineering Materials., 243-244 (2003), p.21-26.

6. Kanayama, H., Ogino, M., Takesue, N and A.M.M.Mukaddes, Finite Element Analysis for Statinary Incompressible Viscous Flows Using Balancing Domain Decomposition, Theoretical and Applied Mechanics, 54(2005), p.211-219.

7. Ogino, M., Shioya, R and Kanayama, H., An inexact balancing preconditioner for large-scale structural analysis, Journal of Computational Science and Technology, Vol. 2, No. 1 (2008), pp. 150-161.

8. Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J.M., Dongarra, J., Eijkhout, V., Pozo. R., Romine, C. and Dervorst, H.V., (1994), Templates for the solution of linear systems: building block for iterative methods. *SIAM*.

9. Bisseling, R. H., Parallel scientific computation, Oxford University Press (2004).

10. Pinar, A and Heath, T. M., Improving performance of sparse matrix-vector multiplication, Proceeding of ACM/IEEE Conference on Supercomputing, Article No. 30 (1999).

11. Saad, Y., SPARSEKIT: A basic tool kit for sparse matrix computations. Technical report, Computer Science Department, University of Minnesota, Minneapolis, MN 55455, Version 2 (1994).

12. Saad, Y., Krylov subspace methods on supercomputers, SIAM Journal of Scientific Static Computation. vol. 10, no. 6 (1989), pp. 1200-1232.

13. Adventure project, http://adventure.sys.t.u-tokyo.ac.jp/

MI レクチャーノートシリーズ刊行にあたり

本レクチャーノートシリーズは、文部科学省 21COE プログラム「機能数理学の構築と展開」(H.15-19 年度) において作成した COE Lecture Notes の続刊である。 今後、レクチャーノートは、文部科学省大学院教育改革支援プログラム「産業界が求める数学博士と新修士養成」(H19-21 年度) および、新しく採択された同グローバル COE プログラム「マス・フォア・インダストリ教育研究拠点」(H.20-24 年度) の推進において招聘する国内外の研究者による講義の講義録として出版するものである。

平成 20 年 7 月
グローバル COE プログラム
マス・フォア・インダストリ教育研究拠点
拠点リーダー　若山正人

Joint Research Workshop of Institute of Mathematics for Industry (IMI), Kyushu University

“Propagation of Ultra-large-scale Computation by the Domain-decomposition-method for Industrial Problems (PUCDIP 2012)”

発　行　2013年2月19日
編　集　金山　寛
発　行　九州大学マス・フォア・インダストリ研究所
九州大学大学院数理学研究院
グローバル COE プログラム「マス・フォア・インダストリ教育研究拠点」
〒819-0395 福岡市西区元岡744
九州大学伊都キャンパス数理学研究教育棟 GCOE 事務室
TEL 092-802-4404　FAX 092-802-4405
URL http://gcoe-mi.jp/

印　刷　城島印刷株式会社
〒810-0012 福岡市中央区白金 2 丁目 9 番 6 号
TEL 092-531-7102　FAX 092-524-4411

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note	Mitsuhiro T. NAKAO Kazuhiro YOKOYAMA	Computer Assisted Proofs - Numeric and Symbolic Approaches - 199pages	August 22, 2006
COE Lecture Note	M.J.Shai HARAN	Arithmetical Investigations - Representation theory, Orthogonal polynomials and Quantum interpolations- 174pages	August 22, 2006
COE Lecture Note Vol.3	Michal BENES Masato KIMURA Tatsuyuki NAKAKI	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2005 155pages	October 13, 2006
COE Lecture Note Vol.4	宮田　健治	辺要素有限要素法による磁界解析 - 機能数理学特別講義 21pages	May 15, 2007
COE Lecture Note Vol.5	Francois APERY	Univariate Elimination Subresultants - Bezout formula, Laurent series and vanishing conditions - 89pages	September 25, 2007
COE Lecture Note Vol.6	Michal BENES Masato KIMURA Tatsuyuki NAKAKI	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2006 209pages	October 12, 2007
COE Lecture Note Vol.7	若山　正人 中尾　充宏	九州大学産業技術数理研究センター キックオフミーティング 138pages	October 15, 2007
COE Lecture Note Vol.8	Alberto PARMEGGIANI	Introduction to the Spectral Theory of Non-Commutative Harmonic Oscillators 233pages	January 31, 2008
COE Lecture Note Vol.9	Michael I.TRIBELSKY	Introduction to Mathematical modeling 23pages	February 15, 2008
COE Lecture Note Vol.10	Jacques FARAUT	Infinite Dimensional Spherical Analysis 74pages	March 14, 2008
COE Lecture Note Vol.11	Gerrit van DIJK	Gelfand Pairs And Beyond 60pages	August 25, 2008
COE Lecture Note Vol.12	Faculty of Mathematics, Kyushu University	Consortium “MATH for INDUSTRY” First Forum 87pages	September 16, 2008
COE Lecture Note Vol.13	九州大学大学院 数理学研究院	プロシーディング「損保数理に現れる確率モデル」 ― 日新火災・九州大学 共同研究 2008 年 11 月 研究会 ― 82pages	February 6, 2009

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.14	Michal Beneš, Tohru Tsujikawa Shigetoshi Yazaki	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2008 77pages	February 12, 2009
COE Lecture Note Vol.15	Faculty of Mathematics, Kyushu University	International Workshop on Verified Computations and Related Topics 129pages	February 23, 2009
COE Lecture Note Vol.16	Alexander Samokhin	Volume Integral Equation Method in Problems of Mathematical Physics 50pages	February 24, 2009
COE Lecture Note Vol.17	矢嶋　　徹 及川　正行 梶原　健司 辻　　英一 福本　康秀	非線形波動の数理と物理 66pages	February 27, 2009
COE Lecture Note Vol.18	Tim Hoffmann	Discrete Differential Geometry of Curves and Surfaces 75pages	April 21, 2009
COE Lecture Note Vol.19	Ichiro Suzuki	The Pattern Formation Problem for Autonomous Mobile Robots —Special Lecture in Functional Mathematics— 23pages	April 30, 2009
COE Lecture Note Vol.20	Yasuhide Fukumoto Yasunori Maekawa	Math-for-Industry Tutorial: Spectral theories of non-Hermitian operators and their application 184pages	June 19, 2009
COE Lecture Note Vol.21	Faculty of Mathematics, Kyushu University	Forum “Math-for-Industry” Casimir Force, Casimir Operators and the Riemann Hypothesis 95pages	November 9, 2009
COE Lecture Note Vol.22	Masakazu Suzuki Hoon Hong Hirokazu Anai Chee Yap Yousuke Sato Hiroshi Yoshida	The Joint Conference of ASCM 2009 and MACIS 2009: Asian Symposium on Computer Mathematics Mathematical Aspects of Computer and Information Sciences 436pages	December 14, 2009
COE Lecture Note Vol.23	荒川　恒男 金子　昌信	多重ゼータ値入門 111pages	February 15, 2010
COE Lecture Note Vol.24	Fulton B.Gonzalez	Notes on Integral Geometry and Harmonic Analysis 125pages	March 12, 2010
COE Lecture Note Vol.25	Wayne Rossman	Discrete Constant Mean Curvature Surfaces via Conserved Quantities 130pages	May 31, 2010
COE Lecture Note Vol.26	Mihai Ciucu	Perfect Matchings and Applications 66pages	July 2, 2010

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.27	九州大学大学院 数理学研究院	Forum “Math-for-Industry” and Study Group Workshop Information security, visualization, and inverse problems, on the basis of optimization techniques 100pages	October 21, 2010
COE Lecture Note Vol.28	ANDREAS LANGER	MODULAR FORMS, ELLIPTIC AND MODULAR CURVES LECTURES AT KYUSHU UNIVERSITY 2010 62pages	November 26, 2010
COE Lecture Note Vol.29	木田　雅成 原田　昌晃 横山　俊一	Magma で広がる数学の世界 157pages	December 27, 2010
COE Lecture Note Vol.30	原　　隆 松井　　卓 廣島　文生	Mathematical Quantum Field Theory and Renormalization Theory 201pages	January 31, 2011
COE Lecture Note Vol.31	若山　正人 福本　康秀 高木　　剛 山本　昌宏	Study Group Workshop 2010 Lecture & Report 128pages	February 8, 2011
COE Lecture Note Vol.32	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2011 “TSUNAMI-Mathematical Modelling” Using Mathematics for Natural Disaster Prediction, Recovery and Provision for the Future 90pages	September 30, 2011
COE Lecture Note Vol.33	若山　正人 福本　康秀 高木　　剛 山本　昌宏	Study Group Workshop 2011 Lecture & Report 140pages	October 27, 2011
COE Lecture Note Vol.34	Adrian Muntean Vladimír Chalupecký	Homogenization Method and Multiscale Modeling 72pages	October 28, 2011
COE Lecture Note Vol.35	横山　俊一 夫　　紀恵 林　　卓也	計算機代数システムの進展 210pages	November 30, 2011
COE Lecture Note Vol.36	Michal Beneš Masato Kimura Shigetoshi Yazaki	Proceedings of Czech-Japanese Seminar in Applied Mathematics 2010 107pages	January 27, 2012
COE Lecture Note Vol.37	若山　正人 高木　　剛 Kirill Morozov 平岡　裕章 木村　正人 白井　朋之 西井　龍映 栄　伸一郎 穴井　宏和 福本　康秀	平成 23 年度 数学・数理科学と諸科学・産業との連携研究ワークショップ　拡がっていく数学　～期待される “見えない力” ～ 154pages	February 20, 2012

シリーズ既刊

Issue	Author／Editor	Title	Published
COE Lecture Note Vol.38	Fumio Hiroshima Itaru Sasaki Herbert Spohn Akito Suzuki	Enhanced Binding in Quantum Field Theory 204pages	March 12, 2012
COE Lecture Note Vol.39	Institute of Mathematics for Industry, Kyushu University	Multiscale Mathematics: Hierarchy of collective phenomena and interrelations between hierarchical structures 180pages	March 13, 2012
COE Lecture Note Vol.40	井ノ口順一 太田　泰広 筧　　三郎 梶原　健司 松浦　　望	離散可積分系・離散微分幾何チュートリアル 2012 152pages	March 15, 2012
COE Lecture Note Vol.41	Institute of Mathematics for Industry, Kyushu University	Forum “Math-for-Industry” 2012 “Information Recovery and Discovery” 91pages	October 22, 2012
COE Lecture Note Vol.42	佐伯　　修 若山　正人 山本　昌宏	Study Group Workshop 2012 Abstract, Lecture & Report 178pages	November 19, 2012
COE Lecture Note Vol.43	Institute of Mathematics for Industry, Kyushu University	Combinatorics and Numerical Analysis Joint Workshop 103pages	December 27, 2012
COE Lecture Note Vol.44	萩原　　学	モダン符号理論からポストモダン符号理論への展望 107pages	January 30, 2013