

A block-tridiagonal solver with two-level parallelization for finite element-spectral codes

Jungpyo Lee*, John C. Wright

Plasma Science and Fusion Center, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA

ARTICLE INFO

Article history:

Received 25 September 2013

Received in revised form

29 April 2014

Accepted 9 June 2014

Available online 17 June 2014

Keywords:

Two-level parallel computing

Block tridiagonal solver

Partitioned Thomas method

Cyclic odd-even reduction

Plasma waves

ABSTRACT

Two-level parallelization is introduced to solve a massive block-tridiagonal matrix system. One-level is used for distributing blocks whose size is as large as the number of block rows due to the spectral basis, and the other level is used for parallelizing in the block row dimension. The purpose of the added parallelization dimension is to retard the saturation of the scaling due to communication overhead and inefficiencies in the single-level parallelization only distributing blocks. As a technique for parallelizing the tridiagonal matrix, the combined method of “Partitioned Thomas method” and “Cyclic Odd-Even Reduction” is implemented in an MPI-Fortran90 based finite element-spectral code (TORIC) that calculates the propagation of electromagnetic waves in a tokamak. The two-level parallel solver using thousands of processors shows more than 5 times improved computation speed with the optimized processor grid compared to the single-level parallel solver under the same conditions. Three-dimensional RF field reconstructions in a tokamak are shown as examples of the physics simulations that have been enabled by this algorithmic advance.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

The numerical solution of partial differential equations in two dimensions often produces a block-tridiagonal matrix system. This tridiagonal structure generally appears by the discretization along one coordinate in which only adjacent mesh points or elements are coupled. If the coupling along the second dimension is in a local basis, the resulting blocks can be sparse or banded. When a global basis is used (e.g. Fourier spectral basis), the size of the blocks may be comparable to the number of block rows and each block can be massive. For larger problems, in-core memory may not be sufficient to hold even a few blocks, and so the blocks must be distributed across several cores. Thus, the parallelization of the solver for the “massive block”-tridiagonal system is required for not only the scaling of the computation speed but also the distribution of the memory.

The electromagnetic wave code (TORIC) [1] is an example of a system producing a “massive block”-tridiagonal matrix that solves Maxwell’s equations and calculates the wave propagation and damping [2–4] which are important in magnetic confinement

fusion research. We have shown in previous work [5] that significantly improved performance is achieved in solving this block-tridiagonal system by introducing parallelism along the block rows in addition to within the dense blocks. In this article, we explain in detail this two-level parallelization of the solver using a three-dimensional (3-D) processor configuration and how operation counts and parallel saturation effects in different parts of the algorithm explain the efficient scaling of the computation speed that has been observed.

The TORIC code is written in MPI-Fortran90, and uses the spectral collocation and finite element methods (FEM) to solve Maxwell’s equations in Galerkin’s weak variational form,

$$\int dV \vec{F}^* \cdot \left\{ -\frac{c^2}{\omega^2} \nabla \times \nabla \times \vec{E} + \vec{E} + \frac{4\pi i}{\omega} (\vec{J}_P + \vec{J}_A) \right\} = 0, \quad (1)$$

where c is the speed of light in a vacuum, \vec{E} is the electric field, \vec{J}_P is the plasma current response to the electric field, \vec{J}_A is the applied antenna current, ω is the prescribed frequency of the applied antenna current, and dV is the differential volume. Here, \vec{F} is a vector function in the space spanned by the basis of \vec{E} , satisfying the same boundary condition as \vec{E} . Eq. (1) is solved for three components of the electric field vector (radial, poloidal and toroidal direction) and their radial derivatives. The radial dimension is discretized by finite elements expressed in the cubic Hermite polynomial

* Corresponding author. Tel.: +1 6179998033.

E-mail addresses: jungpyo@cims.nyu.edu, jungpyo@psfc.mit.edu (J. Lee), jwright@psfc.mit.edu (J.C. Wright).

basis,¹ and the poloidal and toroidal dimensions use a Fourier spectral basis [1,5], as shown in

$$\vec{E}(r, \phi, \theta) = \sum_{n=-l_n}^{l_n} \sum_{m=-l_m}^{l_m} \vec{E}^{n,m}(r) e^{i(n\phi+m\theta)}, \quad (2)$$

where r is the radial coordinate, ϕ and θ are the toroidal and the poloidal angle, n and m are the toroidal and poloidal spectral modes, and l_n and l_m are the maximum toroidal and poloidal mode numbers considered, respectively.

While the toroidal spectral modes are decoupled due to the toroidal axisymmetry of a tokamak, the poloidal spectral modes are coupled by the dependence of the static magnetic field on the poloidal angle. For a fixed toroidal spectral mode, the wave equation in Eq. (1) reduces to a two-dimensional (2-D) problem (radial and poloidal). The constitutive relation between the plasma current and the electric field for each poloidal mode (m) at a radial position (r) is given by

$$\vec{J}_p^{n,m}(r) = -\frac{i\omega}{4\pi} \bar{\bar{\chi}} \cdot \vec{E}^{n,m}(r), \quad (3)$$

where $\bar{\bar{\chi}}(\omega, n, m)$ is the susceptibility tensor that is anisotropic in the directions parallel and perpendicular to the static magnetic field (see [1,2] for the derivations of $\bar{\bar{\chi}}$). Here, the electric field, $\vec{E}^{n,m}(r)$, is radially discretized by n_1 finite elements (i.e. $r = r_i$ where the index of the elements are $i = 1, \dots, n_1$).

Using Eqs. (2)–(3) and the given boundary condition for \vec{J}_A , Eq. (1) results in a master matrix that has a radial dimension of n_1 block rows, and each row has three blocks, \underline{L}_i , \underline{D}_i and \underline{R}_i due to the adjacent radial mesh interactions that are introduced through the cubic Hermite polynomial basis set of the FEM representation. The size of each block is $n_2 \times n_2$, and contains the poloidal Fourier spectral information. The dimension n_2 is equal to six times the poloidal spectral mode number (i.e. $n_2 = 6(2l_m + 1)$), where the factor of six is from the three components of the electric field and their derivatives. The discrete block-tridiagonal system form of Eq. (1) consists of the matrix equation,

$$\underline{L}_i \cdot \vec{x}_{i-1} + \underline{D}_i \cdot \vec{x}_i + \underline{R}_i \cdot \vec{x}_{i+1} = \vec{y}_i \quad \text{for } i = 1, \dots, n_1, \quad (4)$$

where each \vec{x}_i and \vec{y}_i is a complex vector, every element in \underline{L}_1 and \underline{R}_{n_1} is zero, and \vec{y}_1 is determined by boundary conditions. The total master matrix size is $(n_1 n_2) \times (n_1 n_2)$, with typical values of n_1 and n_2 for a small size problem being about 200 and 1000 respectively (see Fig. 1).

Many methods have been investigated to parallelize the solution of this block-tridiagonal system by considering it as either just a “tridiagonal” system [6–9] or a “block” matrix system [10]. The “tridiagonal” system is parallelized in the block row dimension, and the “block” matrix system is parallelized by distributing the blocks. In other words, in some cases the parallel algorithm for a block tridiagonal system is adopted in which the rows of the master matrix are divided among a one-dimensional (1-D) processor grid and they are calculated with a “cyclic reduction method” [6] or “partitioned Thomas method” [7,8] as in the simple tridiagonal problem. The “cyclic reduction method” of solution uses the simultaneous odd row elimination with logarithmic recursive steps, $O(\log(n_1))$, “partitioned Thomas method” requires $O(n_1/p_1)$ elimination steps in divided groups (see Table 1). The “block” matrix system approach keeps the serial routine of the Thomas algorithm

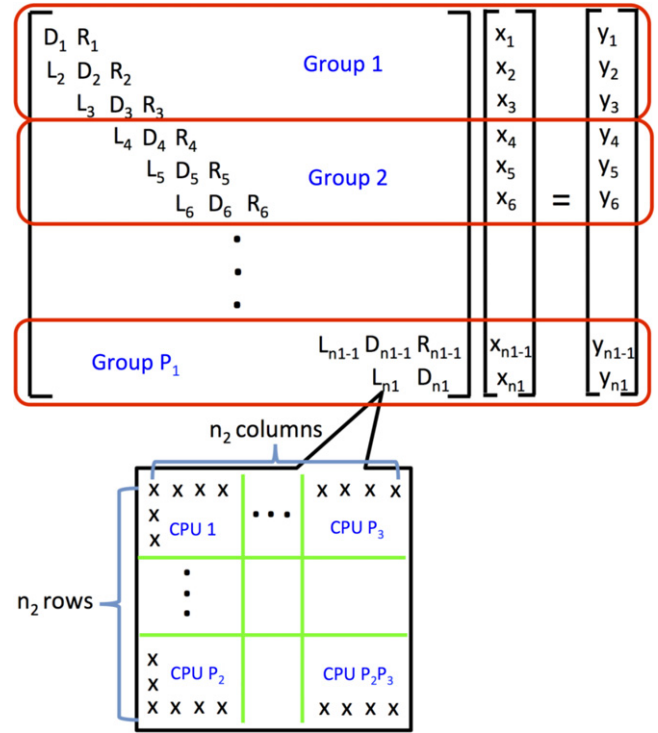


Fig. 1. A schematic of the two-level parallelization using 3-D processor configuration for a block-tridiagonal system. The size of each block, L, D and R is $n_2 \times n_2$ and there are n_1 rows of the blocks. The rows are divided into P_1 groups, and the elements of each block are assigned to $P_2 P_3$ processors. Thus, every element has a 3-dimensional index corresponding to the assigned processor among the total number of processors, $P_{tot} = P_1 P_2 P_3$.

[11], and, for each block operation, uses a parallelized matrix computation algorithm such as ScaLAPACK [12] in a two-dimensional (2-D) processor grid.

These two parallelization methods have different scaling and saturation characteristics at large number of processors. Combining the methods for two-level parallelization in a three-dimensional (3-D) processor grid overcomes their saturation limitations and achieves better scaling. This dual scalability by two-level parallelization has also been used in a solver [13] that employs a cyclic reduction method for parallelizing block rows and multithreaded routines (OpenMP, GotoBLAS) for manipulating blocks. This BCYCLIC solver [13] and our solver have similar features of algorithmic advantage for efficient dual scaling as will be shown in Fig. 2, while they are suitable for application to different sizes of block-tridiagonal systems since they have different memory management. In particular, the BCYCLIC solver is efficient for systems where several block rows can be stored in a single node for multithreaded block operations, while our solver does not have a limitation on the use of out-of-core memory or multiple nodes for splitting large block sizes that cannot be stored in the core memory of a single node, since it uses ScaLAPACK instead of LAPACK/BLACS with threads used in [13]. The block size of TORIC is typically too large to store several block rows in a node because of the global spectral basis.

A single-level parallel solver in TORIC was implemented with the serial Thomas algorithm along the block rows and 2-D parallel operations for the blocks [14]. ScaLAPACK (using routines: PZGEMM, PZGEADD, PZGETRS, PZGETRF) [12] was used for all matrix operations including the generalized matrix algebra of the Thomas algorithm. All elements in each block are distributed in the uniform 2-D processor grid whose dimension sizes are the most similar integers to each other. For efficient communication, the logical block size used by ScaLAPACK is set by $b \times b$ and so the

¹ The cubic Hermite polynomial is smooth in C^1 space, which is required to solve Eq. (1) using integration by parts.

Table 1
Comparison of the parallel algorithms for block-tridiagonal matrix with n_1 block rows where each block is size $n_2 \times n_2$. The total number of processors $P_{\text{tot}} = P_1 P_2 P_3$, n_1 is parallelized in P_1 groups and each block in a group is parallelized on the $P_2 P_3$ processor. Here, M , A , and D are the computation time for block multiplication, addition, and division, respectively, which can be modeled in Eqs. (5)–(7).

	Computation time per processor	Maximum memory in a processor (unit of complex data format)
Cyclic odd–even reduction algorithm [6,13]	$\left(\log_2 \frac{P_1}{2} + \left(\frac{n_1}{P_1} - 2\right)\right) \times (6M + 2A + D)$	$\frac{n_1}{P_1} \frac{(2n_2^2 + 2n_2)}{P_2 P_3}$
Partitioned Thomas algorithm [7]	$\frac{n_1}{P_1} (4M + 2A + 2D) + P_1 (M + A + 2D)$	$\frac{n_1}{P_1} \frac{(2n_2^2 + 2n_2)}{P_2 P_3}$
Combined algorithm [9]	$\frac{n_1}{P_1} (4M + 2A + 2D) + \log_2 \frac{P_1}{2} \times (6M + 2A + D)$	$\frac{n_1}{P_1} \frac{(2n_2^2 + 2n_2)}{P_2 P_3}$
Serial Thomas algorithm ($P_1 = 1$) [20]	$n_1 (M + A + D)$	$n_1 (n_2^2 + 2n_2) / P_2 P_3$

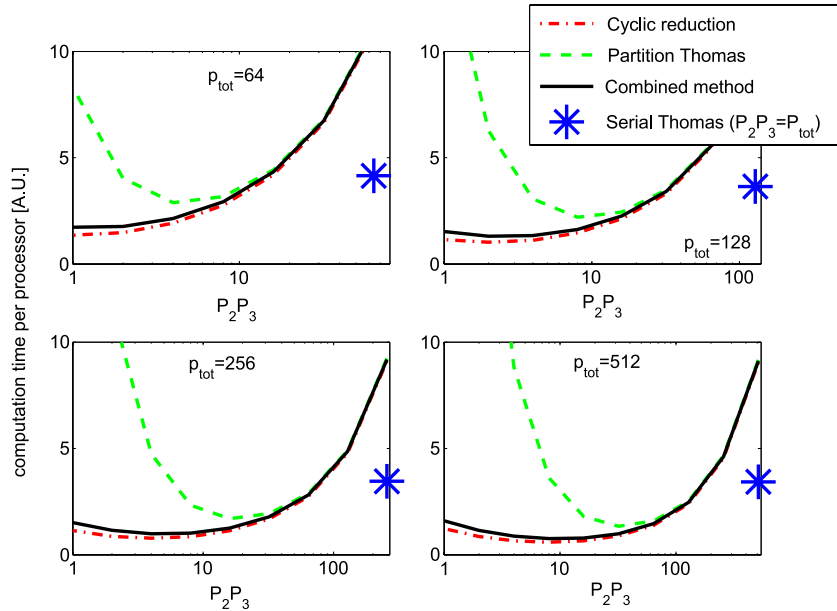


Fig. 2. Estimation of the computation time per processor based on Table 1 for $n_1 = 270$ and $n_2 = 1530$. The time for two-level parallel algorithms (cyclic odd–even reduction, partitioned Thomas method, and combined method) is estimated in terms of $P_2 P_3$ for various P_{tot} , and the time for the single-level parallel algorithm (serial Thomas) is marked as an asterisk at $P_2 P_3 = P_{\text{tot}}$. The saturation effects are included by the model in Eqs. (5)–(7) for M , A , and D .

number of processors (P_{tot}) is constrained to be less than $(n_2/b)^2$. A small logical block size may be chosen to increase the number of processors available for use at the cost of increased communication (e.g. $b \simeq 72$). We have found through experimentation that communication degrades performance even as this constraint is approached. Only when $(n_2/b)^2$ is much larger than P_{tot} , this single-level solver implementation is efficient.

This limitation on the number of processors that can be used efficiently with the single-level solver may present load balancing problems and result in many idle processors in a large, integrated multi-component simulation such as those carried out in the CSWIM Fusion Simulation Project [15] and the transport analysis code TRANSP [16]. For a relatively small problem, $n_1 = 270$ and $n_2 = 1530$ with 20 processors, the completion time to run TORIC is about one hour. Our purpose is to reduce this time to the order of minutes by the use of about 1000 processors through improved scaling. We will demonstrate that the two-level parallel solver using 3-D processor configuration within the dense blocks (2-D) and along the block rows (1-D) can achieve good scaling for small and large problems.

The plan of this paper is as follows: In Section 2, we compare the characteristics of several algorithms for parallelization along the block rows. According to our semi-empirical model, the combined algorithm of the partitioned Thomas method and cyclic reduction, which was developed in [9], is selected for our solver because of the efficient dual scalability and stability. In Section 3, the implementation of the combined algorithm as the two-level parallel solver is

explained, and in Section 4, we present test results of the computation speed for the two-level parallel solver and compare it with the single-level parallel solver. We also discuss accuracy, stability and memory management of the solver. Section 5 introduces two examples for a physics problem to which the two-level parallel solver can be applied: three-dimensional (in space) wave field simulations in the ion cyclotron range of frequencies (ICRF) and the lower hybrid range of frequencies (LHRF). The two-level parallel solver makes these computationally intense problems feasible to solve. Conclusions of this paper are given in Section 6.

2. Selection of the parallel algorithm

To select a parallel algorithm along the block rows adequate for the two-level parallel solver, we compared the matrix operation count and the required memory for the algorithms typically used for block-tridiagonal matrix solvers in Table 1. The common parallel block-tridiagonal solvers use a single-level parallelization using 1-D processor grid (i.e. $P_{\text{tot}} = P_1$), because the algorithm, whether it is the Partitioned Thomas method [7] or odd–even cyclic reduction [6], is easily applicable to a 1-D processor grid, and the usual block size n_2 is much smaller than the number of blocks n_1 . However, for the massive block system such as produced with the coupled FEM-Spectral method, n_2 is as big as n_1 . Storing several blocks each of size $n_2 \times n_2$ using in-core memory would be impossible for such a massive system. Thus we parallelize each block as well,

giving the two-level parallelization for the required memory and desired calculation speed.

If the block operation time is ideally reduced by the number of processors used in the operation, it is always most efficient in terms of number of floating point operations and memory to use the Thomas algorithm with a serial calculation in rows and parallelized block operations on the 2-D processor grid (i.e. $P_{\text{tot}} = P_2 P_3$). However, the additional operation for the parallelization and the increased communication between the processors deteriorates the improvement in speed from parallelization as the number of processors increases and becomes comparable to the square root of the size of a block divided by the logical block size (e.g. $P_{\text{tot}} \sim (n_2/72)^2$). Also, beyond this limit, additional processors have no work to do and remain idle. A good way to avoid both memory and speed problems and retain full utilization of processors is to add another dimension for the parallelization, so the total processor grid configuration becomes three dimensional (i.e. $P_{\text{tot}} = P_1 P_2 P_3$) in the two-level parallelization.

The partitioned Thomas method uses a “divide and conquer” technique. The system (master matrix) is partitioned into P_1 subsystem, which proceeds with eliminations by the Thomas method simultaneously and results in “fill-in” blocks. The “fill-in” blocks are managed to find a solution through the communications between the subsystems [7]. Conversely, the cyclic odd–even reduction algorithm [6] has no matrix fill-in step that requires significant additional time in the partitioned Thomas method. The logarithmic cyclic reductions of the algorithm are the most efficient when both P_1 and n_1 are approximately a power of 2. When either P_1 or n_1 is not a power of 2, the cyclic algorithm is still available by modifying the distribution of processors and communicating each other as shown in [13]. For “massive block”-tridiagonal system, P_1 is typically assigned to be less than $n_1/2$ to treat the massive blocks using some processors of $P_2 P_3$. When P_1 is assigned to be less than $n_1/2$, it induces $n_1/2P_1 + n_1/4P_1 + \dots + 2 = n_1/P_1 - 2$ additional series of operation to the logarithmic reduction process in the cyclic reduction algorithm (see the first row of Table 1).

The combined algorithm of the partitioned Thomas method and the cyclic reduction was introduced in [9]. This combined algorithm was used in this case for the analysis of a solar tachocline problem to enhance both the speed and the stability of the calculation. It can alleviate the local pivoting instability problem of the cyclic reduction method because it is based on the partitioned Thomas method except that it uses cyclic reduction for dealing with the matrix fill-in and for communication between the P_1 groups. The computation time for the fill-in reduction process of the partitioned Thomas method [7], $P_1(M + A + 2D)$, is replaced by the term from the cyclic reduction algorithm [6] in the combined algorithm, $(\log_2 P_1 - 1) \times (6M + 2A + D)$ (see Table 1). Here M , A , and D are the computation time for block multiplication, addition, and division respectively. The contribution of this paper is a generalization of the work in [9] to include parallelization of the block operations and to characterize the performance properties of the resulting two-level parallelization.

We have developed an execution time model for the block operations, M , A , and D including the saturation effect in Eqs. (5)–(7), in order to compare the realistic speed of the algorithms. From the observation that the deterioration of scaling by parallelization in $P_2 P_3$ becomes severe as the number of processors approaches the saturation point, we set the exponential model in Eq. (8) as:

$$M = M_0 \frac{n_2^2}{(P_2 P_3)_{\text{eff}}} \quad (5)$$

$$A = A_0 \frac{n_2^2}{(P_2 P_3)_{\text{eff}}} \quad (6)$$

$$D = D_0 \frac{n_2^2}{(P_2 P_3)_{\text{eff}}} \quad (7)$$

$$(P_2 P_3)_{\text{eff}} = \left\{ (P_2 P_3)_{\text{sat}} * \left(1 - \exp \left(- \frac{(P_2 P_3)}{(P_2 P_3)_{\text{sat}}} \right) \right) \right\}^{\alpha_{(P_2 P_3)}} \quad (8)$$

The exponent parameter, $\alpha_{(P_2 P_3)}$, represents the non-ideal scaling because $(P_2 P_3)_{\text{eff}}$ becomes about $(P_2 P_3)^{\alpha_{(P_2 P_3)}}$ when $P_2 P_3$ is much smaller than $(P_2 P_3)_{\text{sat}}$. Ideally, $\alpha_{(P_2 P_3)}$ should be 1. However, from actual tests of the run time in Section 4, we can specify the parameters, $\alpha_{(P_2 P_3)} = 0.41$ and $(P_2 P_3)_{\text{sat}} = (n_2/191)^2$ from Fig. 4(a). These constants may not be generally true for all range of processors and for all architectures, but it can explain the results well in our test shown in Fig. 4. Also, we set the parameters, $M_0 = 0.5D_0 = n_2 A_0$, because the general speed of matrix multiplication in a well optimized computation code is about two times faster than that of matrix division based on experience when the matrix size is about 1000×1000 .

No communication saturation model is used for the P_1 decomposition in this comparison. For cyclic reduction steps in the first and third rows of Table 1, they have a natural algorithmic saturation from the $\log_2 P_1$ term that dominates over any communication saturation. Unlike the distribution of a matrix block among $P_2 P_3$ processors, increase of P_1 does not fragment matrix block among more processors but has the weaker effect of increasing communication boundaries between block row groups. Additionally, for the massive block system, we typically use $P_1 \ll n_1$ in which there is little saturation in the parallelization due to the communication.

The computation time per processor is estimated for the algorithms in Fig. 2 for a relatively small size system problem, $n_1 = 270$ and $n_2 = 1530$. Both the combined algorithm and the cyclic reduction algorithm show the similar performance of the two-level scaling and they have typically smaller computation time than the partitioned Thomas algorithm. This model is demonstrated by the real computation results in Fig. 6 that agree well with the models in Fig. 2 (compare the blue asterisk and the black curve in Fig. 2 with Fig. 6). Among the algorithms, we selected the combined algorithm because it is known to be more stable than the original cyclic reduction algorithm [9]. Some stability issues of the original cyclic reduction were also fixed in the Buneman version of cyclic reduction with a reordering of operations [17–19].

3. Code implementations

One approach for implementing the 3-D grid for the two-level solver is to use a context array in BLACS [21], in which each context uses the 2-D processor grid as does the single-level solver [12]. In BLACS, a context indicates a group within a boundary of an MPI communicator. Under the default context having the total number of processors, it is possible to assign multiple sub-contexts (groups) by mapping each processor. Also, we can communicate across sub-contexts when needed in a tridiagonal algorithm.

The combined algorithms of the partitioned Thomas method and cyclic odd–even reduction [9] can be summarized as three forward reduction steps and two back substitution steps. By partitioning the whole system into P_1 groups and applying the Thomas algorithm in each group, we can achieve a subsystem containing P_1 rows to be communicated across the groups using the cyclic reduction. Once we achieve the solution for a row after the cyclic reduction, the solution is substituted to find another row solution within the subsystem, and then finally substituted backward as in the Thomas algorithm for each group.

3.1. Divided forward elimination and odd–even cyclic reductions

The three forward steps are described in Fig. 3. The first step is for serial elimination of the L_i block by the previous row

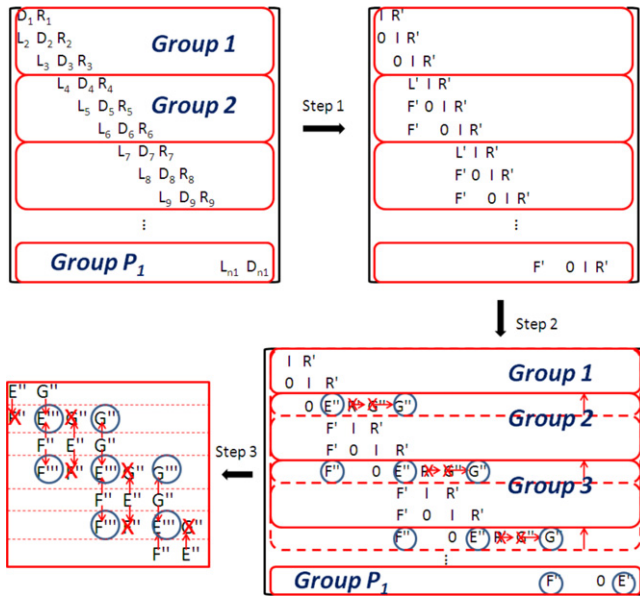


Fig. 3. Description of the combined algorithm for the forward elimination (Step 1), the block redistribution (Step 2), and the cyclic odd-even reduction (Step 3).

($i-1$ th row) as in the Thomas algorithm, but this process is executed simultaneously in every group like a typical partition method. During the first step, the elimination processes create the fill-in blocks “F” at the last non-zero column of the previous group except for the first group (see Fig. 3step 1).

Step 2 is the preliminary process for step 3, the cyclic reduction step which requires tridiagonal form. To construct the upper level tridiagonal form composed of the circled blocks in the first row of each group (see Fig. 3step 2), we need to move the blocks G'' in the first row to the column where the block E'' of the next group is located. Before carrying out the redistribution, the matrices in the last row in each group must be transmitted to the next group. Then, the received right block R' is eliminated by the appropriate linear operation with the following row, and the elimination by the next row is repeated until the block G'' moves into the desired position.

The block tridiagonal system formed by the E'' , F'' and G'' blocks from each processor group can be reduced by a typical odd-even cyclic reduction in step 3 as shown in Fig. 3. This step is for the communication of information between groups, so the portion of the total run time for this step is increased as P_1 is increased. This reduction is carried out in $\log_2((P_1 + 1)/2)$ steps because P_1 should be $2^n - 1$ instead of 2^n where n is an integer, and it requires the total number of processors to be several times ($2^n - 1$) (i.e. $P_{tot} = P_2 P_3 (2^n - 1)$). This characteristic could be a weak point of this algorithm in a practical computation environment, because a node in a cluster consists of 2^n processors typically. However, the problem can be solved by assigning ($2^n - 1$) nodes that have m processors in a node, which results in no free processor on certain nodes and has minimal communication across the nodes.

3.2. Cyclic substitutions and divided backward substitutions

In the end of the cyclic reduction in step 3 only one block E'' remains, so we can obtain a part of the full solution by taking $x_i = E''^{-1} y_i$. This part of the solution is substituted to find a full solution vector, \tilde{x} , in step 4 and step 5. In step 4, the cyclic back substitution is executed in $\log_2((P_1 + 1)/2)$ steps, the same as in the cyclic reduction step. Then, in each group, the serial back substitution continues simultaneously in step 5. In this step, each group P_i except the first one should have the information of the solution in the previous group P_{i-1} to evaluate the terms contributed by the fill-in blocks F'' in the solution.

4. Result and discussions

4.1. Computation speed of the solver

The computation speed of the solver using the combined algorithm is evaluated with various 3-D processor grids, $[P_1, P_2, P_3]$, for three different size problems as shown in Fig. 4. Each graph shows the result for different decompositions of the processor grid. The red line indicates the single-level solver scaling which saturates due to large communication time well before the number of processors is comparable to $(n_2/72)^2$, which is the ratio of block size to the logical block size of ScaLAPACK. However, the graphs for the two-level solver show fairly good and stable scaling. For the large number of processors in Fig. 4(c), the two-level solver with an optimized configuration is about 10 times faster than the single-level solver. This test for wall-clock time was conducted on the massively parallel Franklin computing system at the National Energy Research Scientific Computing Center (NERSC). Each of Franklin's compute nodes consists of a 2.3 GHz quad-core AMD Opteron processor (Budapest) with a theoretical peak performance of 9.2 GFlop/sec per core. Each core has 2 GB of memory.

In the log-log graph of run time as a function of the number of processors, an ideal scaling from parallelization has a slope of -1 . Although the ideal scaling is hard to achieve because of increasing communication, the two-level solver shows a much steeper slope than the single-level solver. We can consider the slope as the average efficiency of the scaling. For the small size problem in Fig. 4(a), we evaluate the speed-up² ($=T(P_{ref}) \times P_{ref}/T(P_{tot})$) by selecting the reference point as the first point ($P_{ref} = 8$) of the single-level solver. For example, the two-level solver with $P_2 P_3 = 16$ (magenta line with plus symbols, slope = -0.55) shows that the speed-up by the different total number of processors ($P_{tot} = 16, 48, 112, 240, 496, 1008, 2032$) are respectively 10.4, 12.1, 24.6, 43.7, 64.3, 84.6, and 120.1. Their corresponding efficiencies ($=T(P_{ref}) \times P_{ref}/(T(P_{tot}) \times P_{tot})$) are 65, 25, 22, 18, 12, 8.4, and 5.9%. For the medium size problem in Fig. 4(b), using the first point ($P_{ref} = 64$) of the single-level solver (red line) for the reference point, the speed-up of the two-level solver with $P_2 P_3 = 16$ (light blue line with diamond symbols, slope = -0.70) by the different total number of processors ($P_{tot} = 112, 240, 496, 1008, 2032$) are 69.0, 138.2, 250.2, 381.6, and 522.0. Their corresponding efficiencies are 61, 57, 50, 37, and 25%. We note for the cases in Fig. 4(a) and (b) the processor number and speed-up are comparable to those in [13] (see Fig. 4 in [13]). For the large size problem in Fig. 4(c), we may use the first point ($P_{ref} = 1920$) of the two-level solver with $P_2 P_3 = 128$ for the reference point, because the single-level solver (red line) shows saturation of the scaling already at this point. Then, the speedup of the two-level solver with $P_2 P_3 = 128$ (green line with cross symbols, slope = -0.79) by the different total number of processors ($P_{tot} = 3968, 8064, 16256$) are 3498, 6508, and 10160. Their corresponding efficiencies of the scaling are 88%, 80%, and 60%, respectively. As the results are represented as lines on a log-log graph, the efficiencies based on the linear speed-up decrease with increasing number of processors.

The non-ideal scaling parameters for $P_2 P_3$ used in the model of Section 2 can be inferred from the red graph in Fig. 4(a) in which all processors are used for the parallelization in blocks ($P_{tot} = P_2 P_3$). The exponent parameter, $\alpha_{(P_2 P_3)} = 0.41$, is obtained by the average slope in the log-log graph before the saturation point. The saturation point for $n_2 = 1530$ is around $P_2 P_3 = 64$ where the graph begins to be flat, giving the parameter $(P_2 P_3)_{sat} = (n_2/191)^2$.

² A serial solver using a processor ($P_{ref} = 1$) is an ideal reference point for the scaling by the parallelization. However, in this test, the reference point $P_{ref} > 1$ is used due to the constraint of the required memory for the massive system.

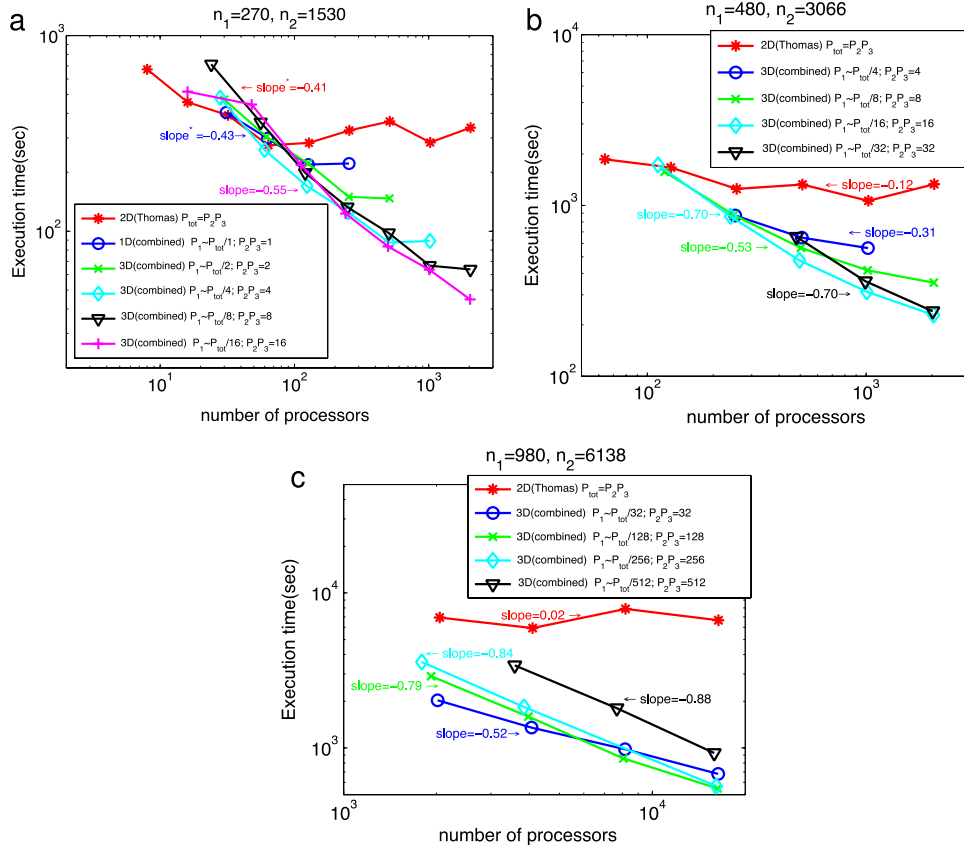


Fig. 4. Comparison of the solver run time in terms of various 3-D processor grid configurations $[P_1, P_2, P_3]$ with different size problem (a) $[n_1, n_2] = (270, 1530)$, (b) $[n_1, n_2] = (480, 3066)$, and (c) $[n_1, n_2] = (980, 6138)$. The red graph corresponds to the single-level parallel solver that uses of the serial Thomas algorithm along the block rows and 2-D parallel operations for blocks. The other graphs (blue, green, cyan, and black) correspond to the two-level solver that uses 1-D parallelization by the combined algorithm in Section 2 along block rows and uses 2-D parallel operations for blocks. The slope values next to the lines indicate the average of the slopes of the graphs, and the “slope*” indicates the average slope before the saturation point. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The red graphs in Fig. 4(b) and (c) showing the full saturations beyond $P_2P_3 = 256$ and beyond $P_2P_3 = 1024$, respectively, are also consistent with the saturation points in Fig. 4(a). For the two-level parallel solver using 3-D processor grid shows retardation of the saturation because P_1 can be used to make P_2P_3 less than $(P_2P_3)_{sat}$. Also, as shown in Fig. 4(a)–(c), the slopes become generally steeper for the larger size problem or for smaller processor number, as they are far from the saturation point. These facts validate the exponential form of the model used in Section 2.

Fig. 5 shows the allocated computation time for the steps of the combined algorithm within the run time of the two-level solver. As the number of groups P_1 is increased at a fixed number of block rows n_1 , the dominant time usage is shifted from step 1 and 2 to step 3 in Fig. 3 due to fewer partitioned blocks per group and more cyclic reductions. The graphs in Fig. 5 are in accordance with the expected theoretical operation of the combined algorithm in Section 2. Since the operation count of the partitioned Thomas part in the step 1 and step 2 is proportional to n_1/P_1 , the slope is about -1 . But the cyclic reduction part in step 3 results in a logarithmic increase of the graph because the operation count is proportional to $\log_2 P_1$, as indicated in Table 1. For large P_1 with a fixed P_2P_3 , the run time of step 3 is dominant component of the total run time, which implies the algorithmic saturation of the parallelization in P_1 . This algorithmic saturation is shown in the reduced slope of the black line in Fig. 4(a) when $\log_2 P_1 \geq n_1/P_1$. Note that the flat slope of all graphs except the red graph in Fig. 4(a) when $P_1 > n_1/2$ is not due to the algorithmic saturation but due to the constraint of the parallelization by $P_1 \leq n_1/2$.

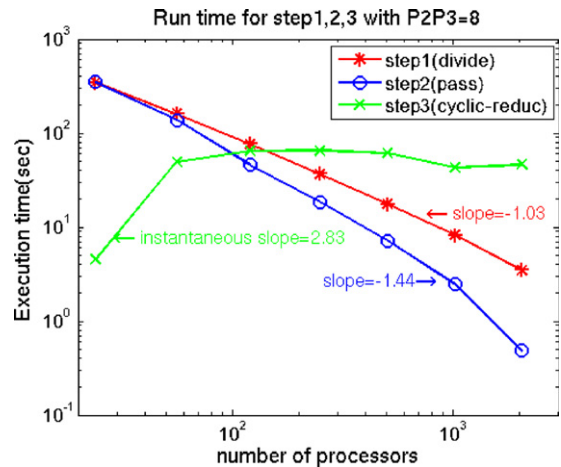


Fig. 5. Run time of the forward reduction steps in the two-level solver with a small size problem $[n_1, n_2] = (270, 1530)$. Step 1 is a divided forward elimination process. Step 2 is a preliminary process for making the tridiagonal form needed for Step 3, which is a typical cyclic odd–even reduction process. Step 3 shows the logarithmic increase as indicated in Table 1. The summation of the three run times of steps 1, 2, and 3 corresponds to the yellow graph with triangle symbols in Fig. 4(a).

An enhanced improvement in speed-up relative to ideal scaling (i.e. the steeper slope than -1) is seen in Fig. 5 for step 2 for large P_1 . The reason for this could be specific to the algorithm, i.e. the matrix operations of the first row in each group are much smaller than for the rest of the rows in the group, and some of the remaining rows

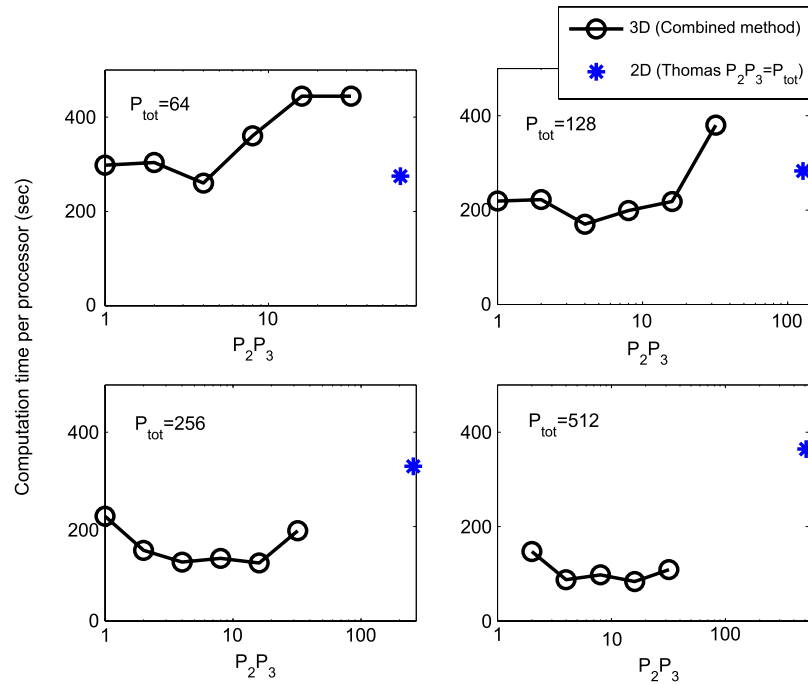


Fig. 6. Comparison of the solver run time in terms of P_2P_3 with a small size problem $[n_1, n_2] = (270, 1530)$. Compare this result with the estimation for the combined algorithm (black line) and Thomas algorithm (blue asterisk) in Fig. 2.

become the first rows in a group as we divide into more groups with increased P_1 .

This saturation of the single-level parallelization either in P_1 or in P_2P_3 implies the existence of an optimal 3-D processor configuration for the two-level parallelization. We found that the optimal 3-D grid exists at the minimum total run time when $P_2P_3 \approx 16$ for $n_2 = 1530$ and $n_2 = 3066$, and $P_2P_3 \approx 128$ for $n_2 = 6138$, provided the number of processors is big enough. Fig. 6 shows the run time comparison in terms of P_2P_3 for $n_2 = 1530$. As mentioned in Section 2, the results of Fig. 6 are reasonably consistent with the non-ideal scaling model we developed for the algorithm comparison. Thus, the optimal grid for the minimal computation time can be estimated from the modeling in Section 2 without the full test such as shown in Fig. 6.

It is important to point out that the two-level parallel solver is not always faster than the single-level parallel solver because the serial Thomas algorithm has fewer matrix operations and thus works well with a small number of processors far before the saturation point. For example, the single-level solver shows the faster computation speed than any 3-D configuration of two-level solver below $P_{\text{tot}} = 32$ in Fig. 4(a).

From Table 2 obtained by the IPM monitoring tool [22] at NERSC, we can compare the saturation effect from MPI communication for the two solvers. Although this tool does not measure the communication time in a specific subroutine, it does monitor the total run time including pre-processing and post-processing, and we can see a remarkable difference between the single-level parallel solver and the two-level parallel solver. As the total number of processors (P_{tot}) is increased, MPI communication time increases at a much faster rate for the single-level solver than it does for the two-level solver. When P_{tot} changes from 32 to 2048, the ratio of the communication time to the total run time increases about three times for single-level solver, while the ratio for the two-level solver increases only about two times. Also, the drop of the average floating point operation speed (Gflop/s) in terms of the total core number for the single-level solver is more severe than the speed drop-off for the two-level solver. Both of these observations demonstrate the retarded saturation that can be credited for the reduced communication of the two-level solver.

In the second column of Table 2, even before the saturation point (e.g. $P_{\text{tot}} = 32$), we can see about 3 times higher average execution speed for the two-level solver than for the single-level solver (compare Table 2(a) with (d)). This is also a significant benefit of the use of the two-level solver. The difference of the execution speed may depend on the efficiency of the calculation by ScaLAPACK in a given block size with a different number of processors (note that the single-level solver uses all cores whereas the two-level solver uses the fraction P_2P_3/P_{tot} of cores for the ScaLAPACK operations). Hence, the two-level solver algorithm has more efficient data processing (e.g. fewer cache misses and calculation in a larger loop) as well as less communication overhead. Furthermore, from Table 2(e) and (f), we can see that the scaling by ScaLAPACK is less efficient than that by the combined algorithm along the block rows.

4.2. Other properties of the solver

The required in-core memory for the two-level solver is about two times that for the single-level solver because of the block fill-ins (see the second column of Table 1 and the third column of Table 2). When the matrix size is $n_1 = 270$ and $n_2 = 1530$, the allocated memory per core is about 2 GB using 16 processors with the two-level solver, so it prevents us from using the two-level solver with processors less than 16 and the single-level solver with less than 8 processors (see Fig. 4(a)). An out-of-core method would enable the two-level solver to work with a small number of processors, and indeed we have observed no significant degradation in computation speed in our testing of the out-of-core algorithm.

The memory management of this solver has a different character than the multithreaded solver discussed in [13]. The optimization of the memory management for fast computation depends on architecture. Although threading is known to be faster than MPI communication, it relies on uniform memory access on a node. For some architecture, the effective memory access and number of threads are limited to the memory and number of cores on only one of the dies. For example, the Hopper machine at NERSC has a NUMA

Table 2

Measurement of the average MPI communication time percentage of the total run time (the first column), the average floating point operation speed per core (the second column), and the average memory usage per core (the third column) by IPM which is the NERSC developed performance monitoring tool for MPI programs [22]. This result is for a small size problem $[n_1, n_2] = (270, 1530)$ in terms of various processor grid configurations and solver types.

	% comm	gflop/s	gbyte
(a) Single-level solver ($P_{\text{tot}} = P_2 P_3 = 32$)	26.2	0.719	0.522
(b) Single-level solver ($P_{\text{tot}} = P_2 P_3 = 128$)	38.4	0.438	0.292
(c) Single-level solver ($P_{\text{tot}} = P_2 P_3 = 2048$)	78.1	0.110	0.188
(d) Two-level solver ($P_{\text{tot}} = 32, P_2 P_3 = 1$)	34.6	2.596	1.486
(e) Two-level solver ($P_{\text{tot}} = 128, P_2 P_3 = 1$)	53.5	1.714	1.051
(f) Two-level solver ($P_{\text{tot}} = 128, P_2 P_3 = 16$)	48.7	1.158	0.392
(g) Two-level solver ($P_{\text{tot}} = 2048, P_2 P_3 = 16$)	64.2	0.568	0.262

architecture and so only $1/4$ or 6 threads could be used on each node out of 24 cores available and so only $1/4$ of the memory could be used as well. This places a constraint on the number of threads and amount of memory that can be efficiently used for block decomposition and limits the algorithm to SMP machines for solving large blocks. In [13], this limitation is acknowledged and they indicate plans to extend BCYCLIC to hybrid MPI to use more memory and multiple nodes in block decomposition. However, threaded applications typically have a smaller memory footprint per process due to sharing of parts of the executable and common data structures.

The combined algorithm of the two-level solver can handle non-powers of two for the number of block rows and the number of processors. For the combined algorithm using the original cyclic reduction, n_1 can be arbitrary number times power of two. Also, the total number of processors for two-level solver is constrained to be $P_{\text{tot}} = P_2 P_3 (2^n - 1)$, which is more flexible than the single-level solver for 1-D parallelization by the original cyclic reduction having $P_{\text{tot}} = (2^n - 1)$. The modifications³ in the original cyclic reduction algorithm in [13] to remove the constraint on the number of processors are useful and could be applied to this solver in the cyclic reduction step as well to permit completely arbitrary processor counts.

To demonstrate the accuracy of the solvers, we compare the solutions, \vec{x}_i in Eq. (4), as well as a representative value of the solution (e.g. a wave power calculation using the electric field solution in TORIC). The values obtained by the two-level solver agree well with the result of the single-level solver (to within 0.01%). Also, the two-level solver shows excellent stability of the result in terms of the varying processor number (to within 0.01%). This precision may be a characteristic of the new algorithm. Because the sequential eliminations in step 1 are executed in divided groups, the accumulated error can be smaller than that of the single-level solver which does the sequential elimination for all range of radial components by the Thomas algorithm. However, from another viewpoint, the local pivoting in the divided groups of the two-level solver instead of the global pivoting in the serial Thomas algorithm may induce instability of the solution. Many people have investigated the relevant stability of the tridiagonal system with the partitioned Thomas algorithm [23] and cyclic reduction algorithm [17–19] and have developed techniques to ensure numerical stability regarding the use of the pivoting [24,25].

Our solver uses the algorithm shown in Fig. 3, where no block operations in the schematic depend on the right hand side. So we can use this method for multiple right hand sides, if needed. However, in the current solver, multiple right hand sides should

be given altogether before the block operations since the algorithm normalizes the diagonal block as identity and eliminates the lower side band block during the solution. Therefore, the stored blocks in the memory after the forward process cannot be used in the system with a new right hand side. If we repeat all the block operations from the beginning for a new right hand side, total computation time would depend linearly on the rank of columns of right hand side. In the case that we need to solve this system many times with a new right hand side (e.g. iteration process), it is more efficient to store all needed blocks for later use with the new right hand side, instead of repeating all block operations. Then, in the combined method, the required memory for this purpose, $O((4n_1 + p_1)n_2^2)$, becomes more than two times the original minimal memory storage, $O(2n_1 n_2^2)$ (see Table 1). This is because the solver must save more n_1 diagonal blocks before normalization during step 1 in Fig. 3, more n_1 blocks corresponding to temporary moving blocks G'' during step 2, and more p_1 blocks during cyclic elimination step 3. Moreover, if sufficient memory storage is allowed, then calculating an inversion of the master matrix with an identity right hand side matrix could be another efficient solution method for a problem with a significant number of right hand sides.

Because the speed of the solver is determined by the processor taking the most time, a well distributed load over all processors is important. When the solver is integrated with pre-processing and post-processing in an independent computation code, the versatile parallelization may help the load balance. In Fig. 7, the most unbalanced of the work load in TORIC occurred during pre-processing because the blocks would be trivial such as an identity or zero matrix for the last several processors. All processors are blocked by an MPI barrier until they reach the backward substitution step, so the last several processors usually are free at the end of the runtime for step 3. We may use this expected unbalance by assigning more work during the solver time to the free processors. Then, the imbalance would be dissolved after step 1 and makes the two-level solver faster.

5. Applications

The 3-D block-tridiagonal parallel solver can be applied to many physics problems that are computationally intensive. One example is the study of plasma heating and current drive by mode-converted waves in the ion cyclotron range of frequencies (ICRF) in a tokamak. The energy or momentum of the wave is transferred to the plasma by several resonant mechanisms [26]. The code (TORIC) utilizing the two-level solver is used to solve Maxwell's equations given in Eq. (1), with the plasma response impressed by an ICRF wave injected by a 4-strap antenna in the Alcator C-Mod tokamak [27–29]. By the linear summation of 2-D electric field solutions from each toroidal mode (see Fig. 8) weighted by the antenna toroidal spectrum profile, we can obtain the wave electric field in 3-D geometry as shown in Fig. 9. Although the 3-D reconstruction

³ The cyclic reduction with arbitrary number of rows and processors may result in a non-uniform work load over processors and a few percent ($O(1/\log_2 P_2)$) increase of the computation time than the perfect recursion with powers of two rows and processors

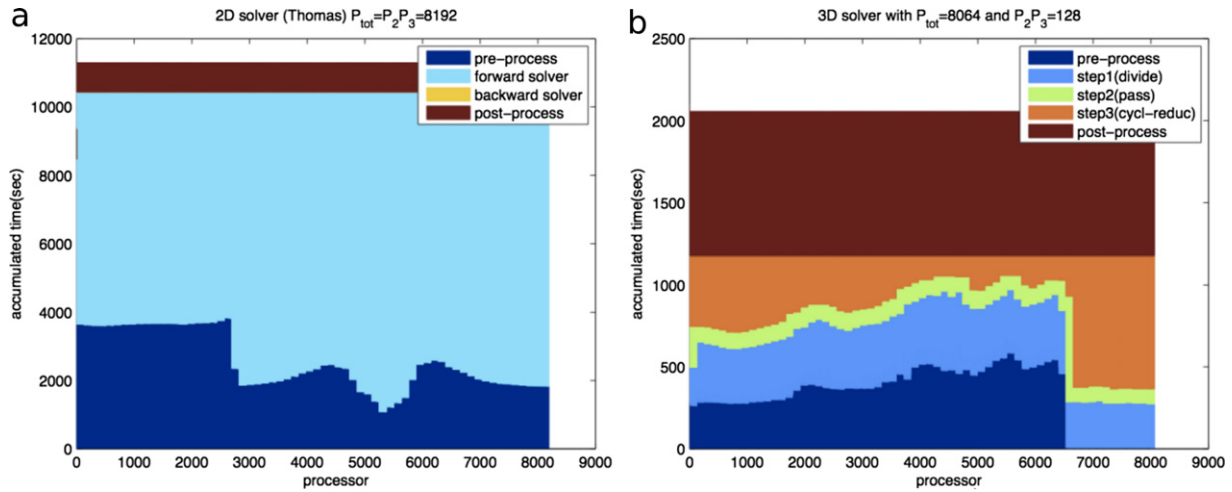


Fig. 7. Work load distribution to each processor for a large problem $[n_1, n_2] = (980, 6138)$. X-axis indicates a processor index, and y-axis is the accumulated run time of (a) the single-level solver using 2-D grid and (b) two-level solver using 3-D grid.

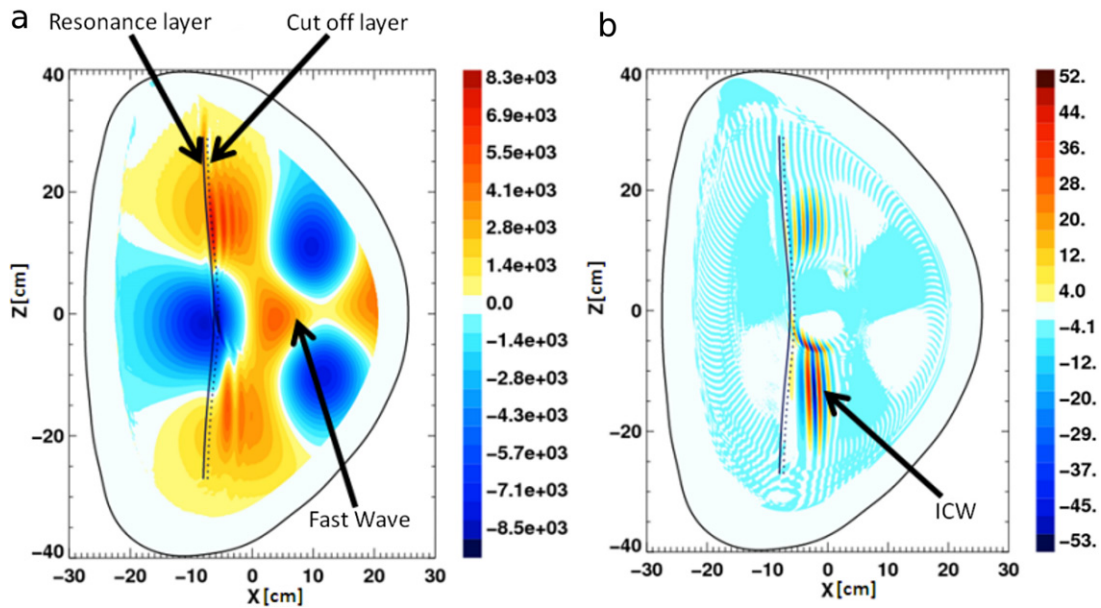


Fig. 8. 2-D contour plot of the electric field of an ICRF wave in Alcator C-Mod from a TORIC wave simulation with one positive toroidal mode ($n_\phi = 9$). (a) The real part of the right hand polarized electric field in the plane perpendicular to the static magnetic field, and (b) the real part of the parallel electric field. The unit of the electric field is [V/m], but is normalized to the square root of power absorption, $\sqrt{P_{abs}[MW]}$. In the poloidal cross section, the outer line corresponds to the vessel of the tokamak, the solid vertical line is the ion–ion hybrid resonance layer where mode conversion can occur, and the dashed vertical line is the cut-off layer of the fast wave [26]. The center of the antenna is located at about $(X, Z) = (22 \text{ cm}, 0 \text{ cm})$. Plasma parameters for this discharge are static magnetic field ($B_T = 8.0T$), plasma current ($I_p = 1.2 \text{ MA}$), central electron temperature ($T_{e0} = 4 \text{ keV}$), central electron density ($n_{e0} = 1.4e^{20} \text{ m}^{-3}$), and ion concentration ($D, {}^3\text{He}, \text{H}) = (0.44, 0.22, 0.08) \times n_e$.

with many toroidal modes is useful to distinguish the different wave modes supported by the plasma wave dispersion relation, it is computationally prohibitive using the single-level parallelization in many computer facilities. Using 511 poloidal modes to resolve a millimeter short wavelength mode converted wave, the total computation time for 40 toroidal modes using the single-level parallelization is about $3000 \text{ s} \times 40 \text{ modes} \times 256 \text{ cores} \cong 8500 \text{ CPU-Hours}$. However, using the two-level parallel solver, the CPU computation time is more than threefold decreased, as shown in Fig. 4(b), thus making the 3-D reconstruction with many toroidal modes feasible.

In the tokamak geometry (toroidally symmetric), Maxwell's equations together with a constitutive relation for the plasma current determine the dispersion relation in terms of the perpendicular wave vector (k_\perp) and parallel wave vector (k_\parallel) relative to the static magnetic field direction, the wave frequency (ω), and the

density and temperature of ions and electrons [2,26]. For the range of ion cyclotron frequency wave, the plasma dispersion equation has three different roots for k_\perp^2 that correspond to the fast magnetosonic wave (FW) mode, the ion cyclotron wave (ICW) mode and the ion Bernstein wave (IBW) mode. The fast wave mode has the long wavelength (several centimeter) that is determined by the smallest magnitude root of k_\perp^2 , and this mode is dominant in the low temperature region in front of ICRF antenna. The fast wave mode is used for accessing the core plasma from the antenna, but it is evanescent beyond a cut-off layer before approaching an ion–ion hybrid resonance layer where mode conversion occurs in the core. If the distance between the cut-off layer and the resonance layer is small enough for tunneling of the wave as shown in Fig. 8, a large portion of the fast wave is not reflected but converted to either the ICW mode [30] or the IBW mode [31]. The wavelength of both converted modes is small, on the order of the ion gyroradius (ICW: a

few millimeters, IBW: sub-millimeter). While the electrostatic IBW is supported by kinetic effects in the high electron temperature of the core plasma, the electromagnetic ICW requires a large k_{\parallel} whose parallel phase velocity is smaller than the electron thermal velocity. Thus, ICW can exist not only in the core but also off-axis where the poloidal magnetic field results in an upshift of the wave k_{\parallel} [32]. Also, the ICW is susceptible to collisionless electron Landau damping (ELD) due to its large k_{\parallel} and its polarization results in a relatively large parallel electric field compared to the fast wave and the IBW. As shown in many experiments, increasing the concentration of He^3 in D and H plasmas moves the mode conversion layer farther from the core, resulting in mode conversion preferentially to ICW rather than mode conversion to IBW [33].

In Fig. 9, a TORIC simulation performed with the two-level parallel solver shows the fast wave (yellow and green) and the mode converted ICW (red and blue). The FW has a dominant right handed polarized electric field component in the perpendicular plane and appears as the centimeter scale (big) contours in front of the antenna in Fig. 8(a). For the component of the electric field parallel to the static magnetic field which is important for electron Landau damping, ICW is dominant and it appears as the long strips with the oscillating amplitude in a few millimeters as shown in Fig. 8(b) and Fig. 9. Two groups of strips are located above and below the mid-plane, but they are propagating in different toroidal directions as shown in Fig. 9, in expectation with the backward propagation property of the ICW (from negative to positive in the x -direction of Fig. 8). Also, an asymmetric toroidal mode profile results in a difference in the intensity and wavelength of the ICW propagating above and below the mid-plane. This is because both the toroidal and poloidal mode numbers contribute to the large k_{\parallel} of the ICW, and the positive poloidal mode effectively increases k_{\parallel} for the ICW below the mid-plane while the negative poloidal mode number decreases k_{\parallel} for the ICW above the mid-plane due to the backward propagating nature of the wave [34]. We also notice that the ICW is damped strongly in the parallel direction by electron Landau damping, which could be useful for localized current drive in the tokamak [27–30].

The 3-D reconstructed picture as shown in Fig. 9 is useful for understanding the toroidal dependency of many experimental diagnostics that must be taken into account when validating the numerical code against experiment. In fact the two-level parallel solver has made it feasible to carry out 3-D reconstructions of mode converted ICRF waves that were needed to simulate the signal of a Phase Contrast Imaging (PCI) diagnostic in the Alcator C-Mod tokamak that was used to detect the mode converted waves [33]. The diagnostic was displaced toroidally from the ICRF antennas necessitating 3-D field reconstructions in order to know precisely the wave field at the diagnostic location.

The two-level solver is also useful to analyze lower hybrid (LH) frequency range waves using TORLH [35], which is modified from TORIC to focus on the fast and slow wave mode by neglecting other thermal wave modes. To resolve the short wavelength of the slow mode (≤ 1 mm), TORLH typically requires higher spatial resolutions (i.e. $n_1 \geq 1000$ and $n_2 \geq 3000$) than TORIC. The slow wave is electrostatic, and it is damped by electron Landau damping that causes velocity space diffusion of the non-thermal fast electrons. The non-Maxwellian electron distribution function evolves consistently with a balance between the energy transfer from the wave and the electron collisions, which requires iteration between the wave solver (TORLH) and the Fokker–Planck equation solver (CQL3D [36]). The iterations make the analysis computationally more intensive. Fig. 10 shows 3-D contours of the parallel electric field of lower hybrid waves reconstructed by the results of the iteration between TORLH and CQL3D using many toroidal modes as shown in Fig. 9 for ICRF waves. Using the two-level solver, the

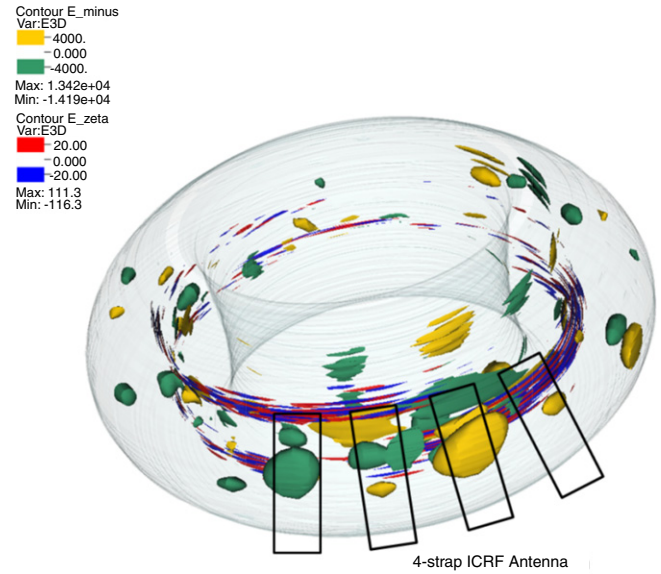


Fig. 9. 3-D contour plot of the electric field of an ICRF wave in Alcator C-Mod that was obtained by superposing the 2-D TORIC wave simulations of 41 toroidal modes (from $n = -20$ to $n = 20$). The real part of the parallel electric field (red and blue, [V/m]) represents an ion cyclotron wave mode (a short radial wavelength), and the right hand polarized electric field in the perpendicular plane to the static magnetic field (yellow and green, [V/m]) indicates the fast magnetosonic wave mode (a long radial wavelength). The peak toroidal mode of the antenna profile is $n = 7$, the total ICRF power absorption is about 0.85 MW, and the other plasma parameters are same as indicated in Fig. 8. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

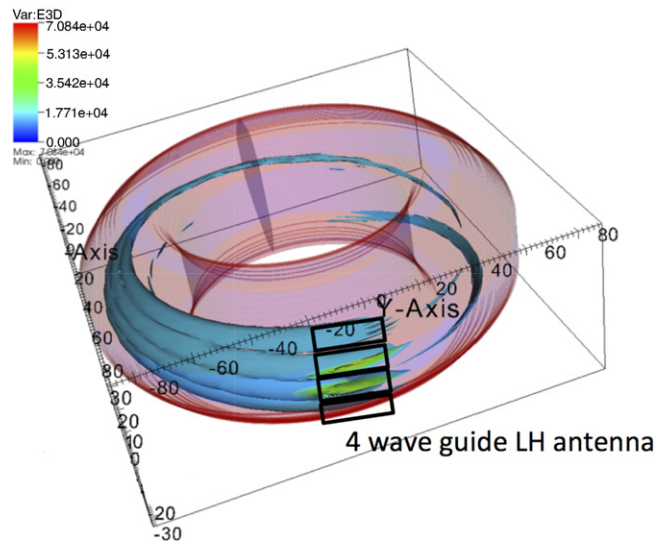


Fig. 10. 3-D contour plot of the electric field of an LH wave in Alcator C-Mod that was obtained by superposing the 2-D TORLH wave simulations of many toroidal modes (from $n = -157$ to $n = -179$, which corresponds to $n_{\parallel} = -1.81$ to $n_{\parallel} = -2.06$).

required computation time for the reconstruction is reduced significantly (more than four times) than when using the single-level solver.

The 3-D reconstruction of the LH resonance cone behavior shown in Fig. 10 is an important feature of the wave propagation as the resonance cone indicates the group velocity path of the electrostatic LH wave branch [5]. The toroidal dependence of the wave path could be very important when examining possible parasitic absorption mechanisms in the scrape off layer (SOL) that could be responsible for the density limit seen in LH current drive experiments such as collisional damping [37] or parametric decay

instability [38], as these mechanisms depend critically on the propagation path in the SOL.

6. Conclusion

The optimized distribution of total processors by two-level parallelization for a massive block-tridiagonal system is shown to be beneficial for faster computation by reducing the communication overhead when a large number of processors are used. The two-level parallel solver contains 1-D parallelization in block rows using the combined methods [9] of “Partition Thomas method” [7] and “Cyclic Odd–Even Reduction” [6], and 2-D parallelization for manipulating blocks themselves using ScaLAPACK [12]. A semi-empirical model to estimate the computation speed of several algorithms is established, and it is verified by test that an optimal point exists among various processor grid configurations. Using the two-level parallelization with the combined method, we can obtain system flexibility in terms of the number of block rows and processor configuration. Although the two-level solver requires about twice the memory of the single-level solver using a “Thomas algorithm”, it shows much higher floating point operation rate, with good accuracy and stability of the solution. As an application of the two-level solver, the intensive computations of mode-converted ICRF waves and lower hybrid waves in a tokamak are demonstrated, where the new solver makes expensive 3-D reconstructions of the wave fields computationally feasible, thus making it possible to confirm important physics of wave propagation and damping. We expect this technique will be useful for other applications that generate block-tridiagonal systems with large blocks.

Acknowledgments

We would like to thank Dr. P. Garaud for supplying the source code of his 1D block-tridiagonal solver using the combined method and Paul Bonoli for helpful guidance. We would also like to thank Dr. S.P. Hirshman for very useful discussions and insights. We also thank the reviewers for giving constructive comments. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the US Department of Energy under Contract No. DE-AC02-05CH11231. This work is also supported by USDoE awards DE-FC02-99ER54512 and the DOE Wave–Particle SciDAC Center (Scientific Discovery through Advanced Computing) Contract No. DE-FC02-01ER54648.

References

- [1] M. Brambilla, Numerical simulation of ion cyclotron waves in tokamak plasmas, *Plasma Phys. Control. Fusion* 41 (1999) 1–34.
- [2] T.H. Stix, *Waves in Plasmas*, American Institute of Physics, New York, 1992.
- [3] K. Appert, T. Hellsten, O. Sauter, et al., Computing of RF heating and current drive in tokamaks, *Comput. Phys. Commun.* 4 (1986) 73.
- [4] N.J. Fisch, Theory of current drives in plasmas, *Rev.Mod.Phys* 59 (1987) 175–234.
- [5] J.C. Wright, J.-P. Lee, E. Valeo, P. Bonoli, C.K. Phillips, E.F. Jaeger, R.W. Harvey, Challenges in self-consistent full-wave simulations of lower hybrid waves, *IEEE Trans. Plasma Sci.* 39 (2010) 2136–2143.
- [6] H.S. Stone, Parallel tridiagonal equation solvers, *ACM Trans. Math. Softw.* 1 (1975) 289–307.
- [7] H.H. Wang, A parallel method for tridiagonal equations, *ACM Trans. Math. Softw.* 7 (1981) 170–183.
- [8] V. Mehrmann, Divide and conquer methods for block tridiagonal systems, *Parallel Comput.* 19 (1993) 257–279.
- [9] P. Garaud, J.D. Garaud, Dynamics of the solar tachocline—II. The stratified case, *Mon. Not. R. Astron. Soc.* 391 (2008) 1239–1258.
- [10] J.C. Wright, P.T. Bonoli, E.D. Azevedo, M. Brambilla, Ultrahigh resolution simulations of mode converted ion cyclotron waves and lower hybrid waves, *Comput. Phys. Comm.* 164 (2004) 330–335.
- [11] L.H. Thomas, Elliptic problems in linear difference equations over a network, *Watson Sci. Comput. Lab. Rept.* (1949) Columbia University, New York.
- [12] SIAM, *ScaLAPACK Users' Guide*, 1997, ISBN 0-89871-397-8.
- [13] S.P. Hirshman, K.P. Perumalla, V.E. Lynch, R. Sanchez, *J. Comput. Phys.* 229 (2010) 6392.
- [14] J.C. Wright, P.T. Bonoli, et al., Full wave simulations of fast wave mode conversion and lower hybrid wave propagation in tokamaks, *Phys. Plasmas* 11 (2004) 2473.
- [15] D. Batchelor, et al. Advances in simulation of wave interactions with extended MHD phenomena, in: H. Simon (Ed.), *SciDAC 2009*, 14–18 June (2009), California, USA, vol. 180, *Journal of Physics: Conference Series*, Institute of Physics (2009) 012054.
- [16] R.I. Hawryluk, An Empirical Approach to Tokamak Transport, in: *Course on Physics of Plasma Close to Thermonuclear Conditions 1*, Coppi, B.(Ed.), CEC, Brussels, (1980), pp. 19–46.
- [17] O. Buneman, *A Compact Non-iterative Poisson Solver*, Report 294, Stanford Univ. Inst. for Plasma Research, Stanford, CA, 1969.
- [18] Buzbee, G.H. Golub, C.W. Nielson, On direct methods for solving Poisson's equations, *SIAM J. Numer. Anal.* 7 (1970) 627.
- [19] P. Yalamov, P. Pavlov, Stability of the block cyclic reduction, *Linear Algebra Appl.* 249 (1996) 341–358.
- [20] S.D. Conte, C. deBoor, *Elementary Numerical Analysis*, McGraw-Hill, New York, 1972.
- [21] J.J. Dongarra, R.C. Whaley, *BLACS Users' Guide*, 1997, <http://www.netlib.org/blacs/lawn94.ps>.
- [22] D. Skinner, W. Kramer, Understanding the Causes of Performance Variability in HPC Workloads, *IEEE International Symposium on Workload Characterization, IISWC05*, Oct (2005), (LBNL-61229).
- [23] V. Pavlov, D. Todorova, Stabilization and Experience with the Partitioning Method for Tridiagonal Systems, *Numerical Analysis and Its Applications, WNAA'96 Rouse*, Bulgaria (1996) 380–387.
- [24] N.J. Highham, Stability of block LDL^T factorization of a symmetric tridiagonal matrix, *Linear Algebra Appl.* 287 (1999) 181–189.
- [25] G. Alaghand, Parallel pivoting combined with parallel reduction and fill-in control, *Parallel Comput.* 11 (1989) 201–221.
- [26] M. Brambilla, *Kinetic Theory of Plasma Waves: Homogeneous Plasmas*, Clarendon, Oxford, 1998.
- [27] P.T. Bonoli, M. Brambilla, E. Nelson-Melby, et al., Mode conversion electron heating in Alcator C-mod: theory and experiment, *Phy. Plasma* 7 (2000) 1886.
- [28] Y. Lin, S. Wukitch, P. Bonoli, et al., Investigation of ion cyclotron range of frequencies mode conversion at the ion–ion hybrid layer in Alcator C-Mod, *Phy. Plasma* 11 (2004) 2466–2472.
- [29] Y. Lin, J.E. Rice, S.J. Wukitch, et al., Observation of ion cyclotron range of frequencies mode conversion plasma flow drive on Alcator C-Mod, *Phy. Plasma* 16 (2009) 056102.
- [30] F.W. Perkins, Heating Tokamaks via the ion–cyclotron and ion–ion hybrid resonance, *Nucl. Fusion* 17 (1977) 1197.
- [31] I.B. Bernstein, Waves in a plasma in a magnetic field, *Phys. Rev.* 109 (1958) 10–21.
- [32] A.K. Ram, A. Bers, Propagation and damping of mode converted ion Bernstein waves in toroidal plasmas, *Phys. Fluids B* 3 (1991) 1059–1069.
- [33] N. Tsujii, M. Porkolab, P.T. Bonoli, Y. Lin, J.C. Wright, S.J. Wukitch, E.F. Jaeger, D.L. Green, R.W. Harvey, Measurements of ion cyclotron range of frequencies mode converted wave intensity with phase contrast imaging in Alcator C-Mod and comparison with full-wave simulations, *Phys. Plasmas* 19 (2012) 082508.
- [34] E.F. Jaeger, L.A. Berry, J.R. Myra, et al., Sheared poloidal flow driven by mode conversion in Tokamak plasmas, *Phys. Rev. Lett.* 90 (2003) 195001.
- [35] J.C. Wright, P.T. Bonoli, A.E. Schmidt, et al., An assessment of full wave effects on the propagation and absorption of lower hybrid waves, *Phys. Plasma* 6 (2009) 072502.
- [36] R.W. Harvey, M.G. McCoy, *Proc. IAEA TCM on Advances in Sim. and Modeling of Thermonuclear Plasmas*, available through USDOC/NTIS No. DE3002962.
- [37] G.M. Wallace, R.R. Parker, P.T. Bonoli, et al., Absorption of lower hybrid waves in the scrape off layer of a diverted tokamak, *Phys. Plasmas* 17 (2010) 082508.
- [38] S.G. Baek, R.R. Parker, S. Shiraiwa, et al., Measurements of ion cyclotron parametric decay of lower hybrid waves at the high-field side of Alcator C-Mod, *Plasma Phys. Control. Fusion* 55 (2013) 052001.