NASA/CR-1998-208736
ICASE Report No. 98-48

Parallelization of the Pipelined Thomas Algorithm

A. Povitsky
ICASE, Hampton, Virginia

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA

Operated by Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-97046

November 1998

PARALLELIZATION OF THE PIPELINED THOMAS ALGORITHM

A. POVITSKY *

Abstract. In this study the following questions are addressed. Is it possible to improve the parallelization efficiency of the Thomas algorithm? How should the Thomas algorithm be formulated in order to get solved lines that are used as data for other computational tasks while processors are idle?

To answer these questions, two-step pipelined algorithms (PAs) are introduced formally. It is shown that the idle processor time is invariant with respect to the order of backward and forward steps in PAs starting from one outermost processor. The advantage of PAs starting from two outermost processors is small. Versions of the pipelined Thomas algorithms considered here fall into the category of PAs.

These results show that the parallelization efficiency of the Thomas algorithm cannot be improved directly. However, the processor idle time can be used if some data has been computed by the time processors become idle. To achieve this goal the Immediate Backward pipelined Thomas Algorithm (IB-PTA) is developed in this article. The backward step is computed immediately after the forward step has been completed for the first portion of lines. This enables the completion of the Thomas algorithm for some of these lines before processors become idle. An algorithm for generating a static processor schedule recursively is developed. This schedule is used to switch between forward and backward computations and to control communications between processors. The advantage of the IB-PTA over the basic PTA is the presence of solved lines, which are available for other computations, by the time processors become idle.

Key words. Thomas algorithm, band matrix, pipelined algorithms, parallel computations, implicit numerical methods, MIMD computer

Subject classification. Computer Science

1. Introduction. Implicit methods are widely used in computational mechanics. Usually, the systems of linearized and discretized equations obtained are of a bandwidth type.

A very efficient direct solver, known as the Thomas algorithm, is used for solution of these systems of equations in computational mechanics [1]. The Thomas algorithm represents a version of the Gauss Elimination method for band matrix systems. For multi-dimensional cases, the Alternative Direction Implicit (ADI) methods are based on solution of linear systems with band matrices corresponding to finite-difference discretization schemes in each direction.

The usual way of parallelizing numerical schemes for MIMD computers is to partition the computational field into multiple subdomains, then to allocate each subdomain to a different processor. This is done so as to minimize communication, delay and processor idle time.

Parallelization of implicit solvers that use the Thomas algorithm for the solution of large banded linear systems of equations is hindered by global spatial data dependencies. Parallel versions of the Thomas algorithm are of the pipelined type. A pipeline in a parallel program involves each processor performing the same set of operations on a successive (continuous) stream of data. Pipelines occur

*ICASE, NASA Langley Research Center, Hampton, VA 23681-2199, e-mail: aeralpo@icase.edu. This research was supported by the National Aeronautic and Space Administration under NASA Contract No. NAS1-97046 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-2199.

due to the recurrence of data within a loop [2]. The main disadvantage is that during the pipelined process some processors will be idle when we switch from the forward to the backward steps of the Thomas algorithm.

In order to avoid pipelining, some parallel Thomas algorithms include the reduction of a banded system of equations on P "slave" processors, the solution of the reduced system of size $O(P)$ on the "master" processor, the broadcast of this solution to the "slaves", and simultaneous computation of the final solution on P processors. This algorithm includes global communications of "slave-master" type and additional computations on each "slave" processor. F. Gustavson and A. Gupta [3] recently have proposed such an improved parallel algorithm for tri-diagonal systems. This algorithm has a redundancy of two: it requires twice as many computational operations per grid point as the serial Thomas algorithm.

Implementation of internal boundary conditions eliminates far-field data dependencies, allowing band matrix systems to be solved independently on each processor. However, modification of either the finite-difference approximation or the implicitness of the scheme due to interface boundary conditions can cause accuracy, stability and convergence properties to deteriorate relative to the original serial method [4].

Naik et al. [5] reduced the parallelization penalty of the solution of tri-diagonal systems by sending necessary information to neighboring processors for groups of rendered lines at forward and backward steps of the Thomas algorithm. The authors of [5] derived the optimal number of lines to be solved per message as a function of computation time per grid point and communication time. This pipelined Thomas algorithm (PTA) does not need global communication and performs the same computations as the serial Thomas algorithm; however, at two points in this algorithm each processor has to wait for all processors ahead.

There is no available systematic study about various formulations of the pipelined Thomas algorithms, therefore we begin with theoretical proofs about processor idle time for these algorithms, using the unified approach of two-step pipelined algorithms.

In this paper a new pipelined Thomas algorithm is developed. This algorithm is designed for parallel multi-domain solution of band matrix linear systems. We call it the Immediate Backward pipelined Thomas Algorithm (IB-PTA): the backward step is performed line-by-line immediately after the forward step has been completed for these lines. This algorithm provides exactly the same solution as the serial Thomas algorithm.

The advantage of the IB-PTA over the basic PTA is that some lines have been completed by the backward step of the Thomas algorithm before the processors are idle. In non-linear and multi-dimensional problems the IB-PTA may be used for other computational tasks at the time when processors are idle. These tasks include computation of the Jacobians of the original non-linear systems, computation of right hand sides of ADI equations and computation by the Thomas algorithm in the next spatial direction. Use of the IB-PTA should lead to a considerable reduction of idle processor time.

Practical use of the proposed IB-PTA is based on a static processor schedule, which has been created before computations by the Thomas algorithm are executed. The rest of the article describes the recursive algorithm for the assignment of this schedule.

Implementation of the IB-PTA within a 3-D PDE solver is the scope of another study [6].

The article contains four sections. In section 2, various formulations of the Thomas pipelined algorithms are discussed. In section 3, a formal definition and study of two-step pipelined algorithms

(PA) is presented. In section 4, the procedure to generate the computational and communicational schedule of processors is presented.

2. The pipelined Thomas algorithms. The theory of pipelined Thomas algorithms developed in this paper can be applied to various types of matrices, including block bandwidth and n-width band matrices. In numerical analysis we usually have to solve the system of $N \times L$ equations which is split to L banded systems of N equations. Each banded system corresponds to one line of numerical grid, and commonly $N \neq L$. The set of L systems where each system is a scalar tridiagonal matrix $N \times N$ is taken here as an example:

$$a_{k,l}x_{k-1,l} + b_{k,l}x_{k,l} + c_{k,l}x_{k+1,l} = f_{k,l}, \quad k = 1, ..., N, \quad l = 1, ..., L, \tag{1}$$

where $a_{k,l}$ are the coefficients, $x_{k,l}$ are the unknown variables, and L is the number of lines.
The version of the Thomas algorithm for the scalar tridiagonal case is as follows. The forward (the first) step of the serial Thomas algorithm is

$$d_{1,l} = b_{1,l}, \quad d_{k,l} = b_{k,l} - a_{k,l}\frac{c_{k-1,l}}{d_{k-1,l}}, \quad k = 2, ..., N,$$

$$g_{1,l} = \frac{f_{1,l}}{d_{1,l}}, \quad g_{k,l} = \frac{-a_{k,l}g_{k-1,l} + f_{k,l}}{d_{k,l}}, \quad k = 2, ..., N. \tag{2}$$

The backward (the second) step is

$$x_{N,l} = g_{N,l}, \quad x_{k,l} = g_{k,l} - x_{k+1,l}\frac{c_{k,l}}{d_{k,l}}, \quad k = N-1, ..., 1. \tag{3}$$

For pipelined Thomas parallel algorithms considered in this article, the coefficients of Eq. (1) are mapped onto P processors so that the subset
$\{a_{k,l},\ b_{k,l},\ c_{k,l}, |\ \ k = N(p-1)/P + 1, ..., Np/P, \ \ l = 1, ..., L\}$ belongs to the pth processor.

We now briefly describe the pipelined Thomas Algorithm (PTA) [5]. Rendering the lth line, the pth processor receives coefficients $d_{N(p-1)/P,l},\ g_{N(p-1)/P,l}$ from the $(p-1)^{th}$ processor; computes the forward step coefficients $d_{k,l}$ and $g_{k,l}$, where $k = N(p-1)/P + 1, ..., Np/P$, sends coefficients $d_{Np/P,l},\ g_{Np/P,l}$ to the $(p+1)^{th}$ processor and repeats computations (2) for the next lines until all the forward step computations are completed. The order of computation of lines by the forward and backward steps of the PTA is shown in Fig. 1a. The three upper lines in Fig. 1a are computed by the forward step computations. After completion of all forward step computations specific to a single processor, the pth processor (except last) has to wait for the completion of the forward step computations by all processors ahead of it. The last outermost (P^{th}) processor starts the backward step computations (3) first. Other processors proceed with the backward step computations in the similar manner as the forward step computations. The three lower lines in Fig. 1a represent the backward step computations. The coefficients d and g of the forward step at interface boundaries are required for beginning of computations in the $(p+1)^{th}$ processor; similarly the computed x values of the backward step are required for the beginning of computations in the $(p-1)^{th}$ processor.

Processors are idle between the forward and backward steps (see Fig 3a), and there is no data available for other computational tasks by this time.

The algorithm denoted as the Immediate Backward PTA (IB-PTA) is shown in Fig. 1b. It is assumed that the computational time per part of a line, corresponding to the steps performed by a single processor, is equal to unity for both forward and backward steps. Actually these times are

different for forward and backward steps of Thomas algorithm. This issue will be discussed in the next sections. First, lines are computed by the forward step until the first line is completed on the last processor (the two upper lines in Fig. 1b). Then the backward step computations for each line start immediately after the completion of the forward step on the last processor (the next four lines in Fig. 1b). Each processor switches between the forward and backward steps of the Thomas algorithm as shown in Fig. 2a at two sequential time units. Each processor communicates with its neighbors to get necessary data for beginning of either the forward or backward computations for next line. Finally, remaining lines are computed by the backward step computations and there are no available lines for the forward step computations (the two lower lines in Fig. 1b).

Two other versions of the pipelined Thomas algorithms are shown in Fig. 1c-d. These formulations of the PTA, denoted as the first-last pipelined Thomas algorithms (FL-PTAs), start the forward step computations from the two outermost processors (indeed, each line starts from one side). Each processor switches between pipelined computations in increasing (from the first to the Pth processor) and decreasing directions. Four upper lines in Fig. 1c are computed by forward step computations. Outermost processors start their forward step computations simultaneously, and middle processors are idle waiting for available data. Then outermost processors start the backward computations simultaneously and perform it in the same way as the forward step computations (the four lower lines in Fig. 1c). Combination of the FL-PTA and the IB-PTA leads to the First-Last Immediate Backward Pipelined Thomas algorithm (FL-IB-PTA), where four fluxes of data are treated simultaneously (Fig. 1d). Outermost processors start the forward step computations simultaneously as in the case of the FL-PTA. The backward step computations for each line start immediately after completion of this line by the forward step computations.

Now we can calculate the processor idle time for the basic PTA and the IB-PTA. The idle time for the pth processor is time between the start of the forward step computations and completion of the backward step computations on the pth processor, when this processor is idle. The maximum processor idle time for the system of P processors is equal to the maximum of the idle times of individual processors.

In the basic PTA, the pth processor waits for the completion of the forward step computations for the last line by $P-p$ processors ahead of it, and for the completion of the backward step computations for the first line by the same $P-p$ processors. Thus, the processor idle time is equal to $2(P-p)$ for the pth processor (see Fig 3a).

Performing the IB-PTA, the pth processor has computed $2(P-p)+1$ lines by the forward step before it starts the backward step computations for the first line. At that point, each processor spends one half of its time computing new lines by the forward step and the other half computing the previous lines by the backward step. Once all lines are completed by the forward step, processors are busy only one half of the time executing the backward step computations (see Fig. 3b). At this time there are $2(P-p)+1$ lines uncompleted by the backward step. Thus, the idle time of the pth processor $2(P-p)$ is the same as for the PTA.

Obviously, there are a lot of possible versions of PTAs. In order to predict the processor idle time for any PTA, a unified approach is developed in the following section.

3. Theoretical estimation of processor idle time for the PTA.

3.1. Two-step pipelined algorithms. Suppose there are L lines of data. Each line is split equally between P processors and is treated as follows: The first (forward) step computations start from the first outermost processor and run in a pipelined way to the last outermost processor. The

second (backward) step for any line is performed from the Pth to the first processor after the first step has been completed for this line.

This algorithm is fully characterized by the $I_1(p,l)$ and $I_2(p,l)$ functions that are the times of the beginning of the first and second step computations, respectively, for the lth line on the pth processor. The computational time per fraction of a line assigned to a single processor is equal to unity for both (forward and backward) steps on a single processor.

To make the algorithm valid, the following relations must be satisfied for the I_1 and I_2 functions:

$$I_r(p,i) \neq I_s(p,j), \forall r,s \in \{1,2\}, \quad i \neq j; \tag{4}$$

$$I_1(p,i) \geq I_1(p-1,i)+1, \quad p=2,...,P; \tag{5}$$

$$I_2(P,i) \geq I_1(P,i)+1; \tag{6}$$

$$I_2(p,i) \geq I_2(p+1,i)+1, \quad p=P-1,...,1, \tag{7}$$

where $i,j=1,...,L$ are line numbers, $p=1,...,P$ are processor numbers, and $r \in \{1,2\}$ and $s \in \{1,2\}$ are indices specifying either the I_1 or I_2 function.

Condition (4) states that each processor computes one step (forward or backward) for a single line per time unit. The pipelined nature of the algorithms considered here leads to inequalities (5, 7) for the first and second steps, respectively. Inequality (6) corresponds to the condition that the second step starts after completion of the first step for each line.

Additionally, if there are some available lines that have been completed by a previous processor or that are ready to start from the outermost processor, the line with minimum sequential number is computed first. Thus, the outermost processors execute the forward and backward steps for lines in increasing order. This requirement, denoted as A-requirement, is made for convenience in the following discussion.

The maximum elapsed time is defined for a system of processors as the time interval between the beginning of computation of the first line and the completion of computation of the last line. For the PA, the elapsed time is equal to

$$T_e = I_2(1,L) - I_1(1,1) + 1. \tag{8}$$

The useful time of each processor is equal to $2L$, as computational time is equal to unity per fraction of line assigned to a single processor. The maximum processor idle time for the system of processors is defined as the maximum idle time over all processors performing the PA, and is equal to $T_e - 2L$.

The following theorem is valid for the PA.

THEOREM 3.1. *For the PA, the maximum elapsed time is greater than or equal to* $2L+2(P-1)$. *There exists functions* I_1 *and* I_2 *satisfying conditions (4-7) such that* $T_e = 2L+2(P-1)$ *for these* I_1 *and* I_2.

Proof. The proof of the first part of the theorem is done by the method of mathematical induction on the number of processors P.

For $P=1$ (a single processor is involved) the elapsed time is equal to $2L$ for any number of lines L.

Suppose for $p \leq P$ the theorem is satisfied. Let the $(P+1)^{th}$ processor render the same L lines. Assume, to show a contradiction, that for $P+1$ processors the elapsed time is

$$T_e(P+1) < 2L + 2((P+1)-1) \tag{9}$$

for some particular I_1 and I_2 satisfying (4-7).

Now switch off the first processor and consider the system of P processors $2, ..., P+1$ with I_1^{new} and I_2^{new} as follows:

$$I_r^{new}(p,i) = I_r(p+1,i), \quad p = 1, ..., P, \quad r \in \{1,2\} \tag{10}$$

Note that the functions $I_1(p,l)$ and $I_2(p,l)$, where $p = 2, .., P+1$, remain the same as for the assumed system of $P+1$ processors. This set $(I_1^{new},\ I_2^{new})$ satisfies (4-7). The elapsed time for this system of P processors is equal to

$$T_e(P) = I_2^{new}(1,L) - I_1^{new}(1,1) + 1 = I_2(2,L) - I_1(2,1) + 1. \tag{11}$$

According to conditions (5,7)

$$I_2(2,L) \leq I_2(1,L) - 1, \quad I_1(2,1) \geq I_1(1,1) + 1. \tag{12}$$

Therefore,

$$T_e(P) \leq T_e(P+1) - 2 < 2L + 2(P-1). \tag{13}$$

This contradiction of the induction hypothesis ($T_e(P) \geq 2L + 2(P-1)$) proves the first statement of the theorem.

One particular distribution of I_1 and I_2

$$I_1(p,l) = p + l - 1, \quad I_2(p,l) = P - p + l + (P + L - 1) = 2P + L + l - p - 1 \tag{14}$$

meets conditions (4-7) and

$$T_e = I_2(1,L) - I_1(1,1) + 1 = 2L + 2(P-1). \tag{15}$$

This proves the second statement of Theorem 3.1. □

The distribution (14) of I_1 and I_2 corresponds to the basic PTA (see the previous section).

There is the following corollary from Theorem 3.1.

COROLLARY 3.2. *If the maximum idle time for P processors performing PA is equal to $2(P-1)$ then the idle time of the pth processor is equal to $2(P-p)$.*

Proof. The pth processor starts to render its fraction of the first line $p-1$ time units later than the first processor, and the pth processor starts to render its fraction of the last line $p-1$ time units earlier than the first processor. Thus, the following relations hold for the I_1 and I_2 functions corresponding to the first and the last lines:

$$I_1(p,1) = I_1(1,1) + p - 1, \quad I_2(1,L) = I_2(p,L) + p - 1. \tag{16}$$

The maximum idle time for the system of P processors is calculated as follows:

$$T_e(P) = I_2(1,L) - I_1(1,1) = I_2(p,L) + (p-1) - (I_1(p,L) - (p-1)) = T_e(p) + 2(p-1). \tag{17}$$

Therefore,

$$T_e(p) = T_e(P) - 2(p-1). \tag{18}$$

According to Theorem 3.1 $T_e(P) = 2L + 2(P-1)$, the elapsed time of the pth processor is given by

$$T_e(p) = 2L + 2(P-1) - 2(p-1) = 2L + 2(P-p). \tag{19}$$

The useful time of each processor is equal to $2L$; therefore, the idle time is equal to $2(P-p)$. □

3.2. Two-step and first-last pipelined algorithms. We define the two-step and first-last pipelined algorithms (FL-PA) starting from two outermost processors. Subsets Λ_1 and Λ_2 include lines starting from the first and the Pth processors, respectively. There are two more fluxes of data than for the one-way PAs defined in the previous subsection. The functions I_3 and I_4 are starting times of the forward and backward computations, respectively, for lines belonging to Λ_2. These functions have to satisfy the following relations similar to (5-7):

$$I_3(p,i) \geq I_3(p+1,i) + 1, \quad p = P-1, ..., 1; \tag{20}$$

$$I_4(1,i) \geq I_3(1,i) + 1; \tag{21}$$

$$I_4(p,i) \geq I_4(p-1,i) + 1, \quad p = 2, ..., P; \tag{22}$$

where $i \in \Lambda_2$.

Conditions (5-7) must be satisfied for $i \in \Lambda_1$. The condition (4) with $r, s \in \{1,2,3,4\}$ and A-requirement must be satisfied for all functions I_r, where $r \in \{1,2,3,4\}$. According to the definition of the maximum elapsed time for P processors, this time for FL-PAs is given by

$$T_e = max(I_2(1,L_1), I_4(P,L_2)) - min(I_1(1,F_1), I_3(P,F_2)) + 1 \tag{23}$$

where $F_1,\ F_2,\ L_1,\ L_2$ are the first and last lines in subsets Λ_1 and Λ_2, respectively.

We define a passage as a set of starting times of the pipelined forward or backward computations on P processors for the ith line:

$$\{I_r(p,i) | p = 1, ..., P\}, \quad \{I_s(p,i) | p = P, ..., 1\}, \tag{24}$$

where $r \in \{1,3\},\ s \in \{2,4\}$.

The FL-PAs drive the $|\Lambda_1| + |\Lambda_2| = L$ passages from the first to the last processor governed by I_1 and I_4 functions and the L passages from the last to the first processor governed by I_2 and I_3 functions.

Now we will define the following reformulation of the FL-PA. This reformulated algorithm uses the same passages as the original FL-PA, i.e.,

$$\forall i, r_1 \quad \exists j, r_2 : \quad I^{new}_{r_1}(p,i) = I_{r_2}(p,j); \tag{25}$$

$$\forall i, s_1 \quad \exists j, s_2 : \quad I^{new}_{s_1}(p,i) = I_{s_2}(p,j); \tag{26}$$

where $p = 1, ..., P,\ r_1, r_2 \in \{1,3\},\ s_1, s_2 \in \{2,4\}$.

The lines start from the first processor in the following order: $F_1, ..., L_1$ (the forward step) then $F_2, ..., L_2$ (the backward step), and from the last processor, $F_2, ..., L_2$ (the forward step), and then $F_1, ..., L_1$ (the backward step). Here indices 1 and 2 correspond to subsets Λ_1 and Λ_2, respectively. Now all backward computations starting from an outermost processor are performed later than the forward computations starting from the same outermost processor:

$$\forall i_1 \in \Lambda_1 \;\; \forall i_2 \in \Lambda_2 : \;\; I_1^{new}(p, i_1) < I_4^{new}(p, i_2), \;\; I_3^{new}(p, i_2) < I_2^{new}(p, i_1). \tag{27}$$

Obviously, conditions (5,7,20,22) are satisfied for each passage of the FL-PA. The reformulated algorithm meets these conditions as it uses the same passages as the original one. Conditions (6,21) are satisfied because each processor completes forward step computations for each line no later than and begins backward step computations for each line no earlier than in the original algorithm.

The forward step computations for the first lines (F_1 and F_2) for the reformulated and original FL-PA start at the same time. The backward step computations for the last lines for the reformulated and original FL-PA (L_1 and L_2) are completed at the same times. Thus, the maximum elapsed time for this algorithm is equal to that for the original algorithm for the same number of processors.

LEMMA 3.3. *The reformulation of the FL-PA described above does not change the maximum elapsed time.*

We define the symmetric FL-PA (SFL-PA) as a particular case of the FL-PA so that

$$I_1(p, i_1) = I_3(P - p + 1, i_2), \;\; I_2(p, i_1) = I_4(P - p + 1, i_2), \tag{28}$$

where $|\Lambda_1| = |\Lambda_2| = L/2, i_1 \in \Lambda_1, \;\; i_2 \in \Lambda_2, \;\; p = 1, ..., P$. This definition is valid for an even number of processors P and an even total number of lines L.

All first-last pipelined Thomas algorithms described in the previous section fall into the class of SFL-PA (see Fig. 1c,d).

In a more common sense, one may reformulate any FL-PA in such a way that the lines belonging to Λ_1 start from the last outermost processor and the lines belonging to Λ_2 start from the first processor. Obviously, this reformulation does not change the maximum elapsed time. Therefore, T_e is a symmetric function of its arguments ($T_e(I_1, I_2, I_3, I_4) = T_e(I_3, I_4, I_1, I_2)$), and the case of symmetric FL-PA $I_1 = I_3, \;\; I_2 = I_4$ corresponds to the local minimum of elapsed time of FL-PAs.

THEOREM 3.4. *For the SFL-PA, the maximum elapsed time is greater than or equal to* $2L+2(P-2)$. *There exist functions* $I_1, \; I_2, \; I_3$ *and* I_4 *meeting conditions (4-7, 20-22) such that* $T_e = 2L+2(P-2)$ *for these I functions.*

Proof. The method of mathematical induction on the number of processors P with the induction step equal to two is used to prove the first part of the theorem.

For $P = 2$ and $L = 2$ the processors can perform computations with zero idle time. The first and the second processor start the forward step computations simultaneously for the first and the second line. At the end of the first time unit, they exchange the interface data and complete the forward step (the first processor renders the second line and vice versa). Processors compute the backward step for two lines the same way. The functions $I_r, \; r = 1, 2, 3, 4$ are given by:

$$\begin{aligned} &I_1(1,1) = 1, \;\; I_1(2,1) = 2, \;\; I_3(2,2) = 1, \;\; I_3(2,1) = 2, \\ &I_2(1,1) = 4, \;\; I_2(2,1) = 3, \;\; I_3(1,2) = 3, \;\; I_3(2,2) = 4. \end{aligned} \tag{29}$$

For $P = 2$ and $L = 2m, \; m > 1$, the lines are coupled and treated as described above couple by couple. The elapsed time is $T_e = 2L = 2L + 2(2 - 2)$.

Suppose that the statement of induction is valid for $p \leq P$. Assume that for $p = P+2$ the elapsed time is $T_e(P+2) < 2L + 2((P+2)-2)$ for some set of I satisfying conditions (4-7, 20-22).

Let's reformulate this algorithm according to Lemma 3.3. This procedure does not change the elapsed time. Obviously, the reformulated algorithm is also symmetrical.

Now switch off the two outermost processors. Define the new set $I_r^{new}, r \in \{1,2,3,4\}$ on P processors

$$\begin{aligned} I_1^{new}(p,i_1) &= I_1(p+1,i_1), \\ I_3^{new}(p,i_2) &= I_3(p+1,i_2), \\ I_2^{new}(p,i_1) &= I_2(p+1,i_1) - 2, \\ I_4^{new}(p,i_2) &= I_4(p+1,i_2) - 2, \end{aligned} \tag{30}$$

where $p = 1, ..., P, \quad i_{1,2} = 1, ..., L/2$.

The SFL-PA characterized by the set I^{new} starts the backward computations two time units earlier than the assumed SFL-PA on $P+2$ processors. There is no forward or backward step computations on switched-off processors; therefore, the backward step computations on the second and on the $(P+1)^{th}$ processors can start immediately after completion the forward computations for the last lines ($L_1 \in \Lambda_1, L_2 \in \Lambda_2$).

According to Lemma 3.3, forward computations for $i_1 \in \Lambda_1$ are executed earlier than backward computations for $i_2 \in \Lambda_2$. According to the definition of symmetrical FL-PA, all forward computations are performed earlier than all backward computations on all processors. Therefore, condition (4) is satisfied for I corresponding to different, forward and backward, pipelined computations. The set I^{new} meets conditions (4-7, 20-22) as $I_{1,3}^{new}$ are equal to $I_{1,3}$ and $I_{2,4}^{new}$ are equal to $I_{2,4} - 2$.

The elapsed time for this SFL-PA on P processors is derived in the same manner as in the proof of Theorem 3.1 (see (11)):

$$\begin{aligned} T_e(P) = I_2^{new}(1,L_1) - I_1^{new}(1,F_1) + 1 = \\ I_2(2,L_1) - I_1(2,F_1) + 1 = (I_2(1,L_1) - 3) - (I_1(1,F_1) + 1) = \\ T_e(P+2) - 4 < 2L + 2(P-2). \end{aligned} \tag{31}$$

This contradiction of the induction hypothesis proves the first statement of the theorem.

One example with the minimum elapsed time is presented here for $P = 2q$ and $L = 2m$:

$$I_1 = \begin{cases} p + l - 1 & \text{if } p \leq q \\ m + q - 1 + (p-q) + l - 1 & \text{if } p > q; \end{cases} \qquad 1 \leq l \leq m \tag{32}$$

$$I_3 = \begin{cases} P - p + l - m & \text{if } p > q \\ m + q - 1 + q - p + l - m & \text{if } p \leq q. \end{cases} \qquad m+1 \leq l \leq L \tag{33}$$

The functions I_2 and I_4 are analogous to I_3 and I_1, respectively. One half of the lines is rendered by the forward step beginning from the first processor and the other half is computed starting from the last processor. After completion of the first half of lines by the processors $1, ..., q$ and simultaneous completion of the second half by the processors $P, P-1, ..., q+1$, processors q and $q+1$ exchange the interface data and then continue to render lines by the forward step. The first q processors now render the second half of lines and vice versa. After a simultaneous completion of the forward step,

processors compute the backward step the same way. For both groups of processors, it takes $m+q-1$ units of time to render m lines either by backward or forward step.

These subsets of q processors are referred to as single super-processors, and subsets Λ_1 and Λ_2 of m lines are referred to as single super-lines. Thus, there are two super-processors and two super-lines, as in the basis of induction, which has been considered in the proof of this Theorem. Thus, the elapsed time is equal to $4(m+q-1) = 2L + 2(P-2)$. □

Therefore, SFL-PAs have only a small advantage in terms of the maximum idle processor time over the basic PTA.

3.3. Applications to the pipelined Thomas algorithms. All pipelined Thomas algorithms described in the second section belong to PAs. Explicit derivation of the I functions can be a cumbersome task for some versions of PTA. It is difficult to do this for the First-Last Immediate Backward pipelined Thomas Algorithm (Fig. 1d) since it treats four fluxes of data simultaneously. Instead, one can use Theorems 3.1 and 3.4 to estimate the elapsed time and the processor idle time.

The first two algorithms, the basic PTA (Fig. 1a) and the Immediate Backward pipelined Thomas Algorithm (IB-PTA) (Fig. 1b) have equal minimum elapsed times (Theorem 3.1). The first-last algorithms (Fig. 1c,d) have a small advantage of two time units idle time for an even number of lines (Theorem 3.4). Thus, these pipelined Thomas algorithms do not reduce the processor idle time considerably. However, for the IB-PTA and for the FL-IB-PTA there are completed lines by the time processors become idle. These processors can be used for other computational tasks while they are idle from the Thomas algorithm.

For real MIMD computers, there is a latency time of communications between processors. Therefore, it is advantageous to solve a number of lines per single message, i.e., to transfer coefficients (for the forward step) or the solution (for the backward step) for a number of lines.

The computational times per node may not be equal for forward and backward steps. For example, there are three multiplication operations per grid point for the forward step and two multiplication operations per grid point for the backward step of the tridiagonal Thomas algorithm. The number of portions of lines may be different for forward and backward steps of the pipelined Thomas algorithm. Those lines that have been completed by the forward step are gathered in groups of lines on the last processor (see the next section for details). Sets Λ_f and Λ_b include portions of lines (not single lines) for the forward and backward computations, respectively. The same number of lines is treated by forward and backward steps:

$$|\Lambda_f| K_1 = |\Lambda_b| K_2, \tag{34}$$

where K_1 and K_2 are numbers of lines solved per message for forward and backward steps, respectively.

To avoid idle time, the amount of computational work should be equal for the forward and backward steps of the Immediate Backward PTAs. Thus, numbers of lines solved per one message for forward and backward steps must satisfy the following equation:

$$N K_1 g_1 = N K_2 g_2, \tag{35}$$

where $g_1,\ g_2$ are computational times per grid node for forward and backward steps. Therefore, the computational time per fraction of a portion of lines assigned to a single processor is taken to be equal to unity for forward and backward step computations. As in the previous cases of the line-by-line treatment, processors switch between forward and backward steps of the IB-PTA. However,

processors may treat several portions of lines by the forward step computations prior to switching to the backward step computations (compare Fig. 2a and b).

To analyze these versions of the pipelined Thomas algorithms which render lines portion-by-portion, we define the extended pipelined algorithms (EPA) that are analogous to the PAs defined in subsection 3.1. The I functions must satisfy relations analogous to (4-7), where i, j are the numbers of portions of lines. The condition (6) is given by

$$I_1(P, i) < I_2(P, j), \tag{36}$$

where $iK_1 \leq jK_2$. This condition states that all lines belonging to the ith portion has been completed by the forward step computations prior to the start of the backward step computations for any of these lines.

The first-last portion-by-portion algorithms are analogous to FL-PTAs (subsection 3.2). The I functions should satisfy conditions analogous to (4-7) and (20-22). We define a symmetric FL-EPA, denoted as SFL-EPA, as follows:

$$I_1(p, i_{1f}) = I_3(P - p + 1, i_{2f}), \quad I_2(p, i_{1b}) = I_4(P - p + 1, i_{2b}), \tag{37}$$

where $|\Lambda_{1f}| = |\Lambda_{2f}| = L_f/2, |\Lambda_{1b}| = |\Lambda_{2b}| = L_b/2, \quad i_{1f} \in \Lambda_{1f}, \quad i_{2f} \in \Lambda_{2f}, \quad i_{1b} \in \Lambda_{1b}, \quad i_{2b} \in \Lambda_{2b}, \quad p = 1, ..., P$.

The proofs of Theorems 3.1 and 3.4 can be used straightforwardly to prove the following Theorem.

THEOREM 3.5. *For the EPA, the maximum elapsed time of the system of processors is greater than or equal to*

$$T_e = L_f + L_b + 2(P - 1).$$

For the SFL-EPA, the maximum elapsed time is greater than or equal to

$$T_e = L_f + L_b + 2(P - 2).$$

The various formulations of the PTA considered in the previous section can be formulated in this case where portions of lines are treated per single interface message. If the same number of lines, say, K_1 for the forward step and K_2 for the backward step, are solved per message for the basic EPA and the IB-EPA, the minimum of processor idle times is equal for these algorithms.

4. Generation of processor schedule. At each time unit each processor either performs forward or backward step computations or is idle. For the basic PTA the processor scheduling is simple: each processor executes the forward step, becomes idle and then executes the backward step. Each processor starts computations immediately after the corresponding data are available from its neighbor. In this case, communications govern computational tasks and there is no necessity of the processor scheduling before computations.

For other considered versions of the Thomas pipelined algorithms, there are more simultaneous fluxes of data in a system of processors and the order of processor tasks is more complicated. For example, completed forward step coefficients from the previous processor are not transfered to the current processor immediately after they have been computed, as this processor may execute the backward step computations at the next time unit. Our earlier attempts to govern processors by communications have led to cumbersome code patterns. The static processor schedule should be

computed prior to running the PTA. This schedule governs the consequence of computations and communications on each processor.

To set up this schedule, let us define the variable $J(p,i)$ that governs the sequence of the forward and backward computations and the idle state:

$$J(p,i) = \begin{cases} +1 & \text{forward step computations} \\ 0 & \text{processor is idle} \\ -1 & \text{backward step computations,} \end{cases} \tag{38}$$

where p is the number of processors and i is the serial number of the time unit. The first time unit on each processor corresponds to computation of the first portion of lines by the forward step on this processor. Therefore, the $J(p,i)$ and $J(p+1,i)$ correspond to two following time units. Use of this variable is more convenient for computer programming of the pipelined Thomas algorithms rather than the explicit use of the $I(p,l)$ functions.

We confine ourselves to one-way PTAs in this section.

THEOREM 4.1. *For any given pipelined Thomas algorithm on P processors with the minimum elapsed time there exists the PTA on $P+1$ processors with the same processor schedule for all processors except the first (outermost) processor such that the elapsed time of this PTA is minimum.*

Proof. Consider the pipelined Thomas algorithm with minimum elapsed time $(2L+2(P-1))$ on P processors. We construct here a schedule for the additional processor meeting conditions of this theorem.

Renumber processors $1,...,P$ to $2,...,P+1$. First, we formulate the schedule for the first (new) processor as follows:

$$\begin{aligned} J(1,i) &= 1 \quad \text{if} \quad J(2,i)=1 \\ J(1,i+2) &= -1 \quad \text{if} \quad J(2,i)=-1 \\ J(1,i) &= 0 \quad \text{otherwise,} \end{aligned} \tag{39}$$

where index i runs from 1 to $2L+2(P-1)$ on the second processor.

There are two more elapsed time units on the first processor than on the second one, therefore, this algorithm has the minimum elapsed time, $2L+2(P-1)+2$, on $(P+1)$ processors.

Now we have to check that the first processor computes no more than one portion of lines at each time unit (condition (4) in terms of I functions). This means that the assignment (39) does not assign 1 and -1 to the same i^{th} time unit on the first processor. This situation may occur only as a result of a switch from the backward step to the forward step on the second processor such that $J(2,i_c-2)=-1$ and $J(2,i_c)=1$. Note that the switch from the backward step to the forward step, such that $J(2,i_c-2)=-1, J(2,i_c-1)=1$ and $J(2,i_c)=-1$, does not lead to this collision.

Consider the first switch leading to the collision. The assignment (39) maps one-to-one the set $\{1 \le i \le i_c-1 | J(2,i) \ne 0\}$ into the set $\{1 \le i \le i_c+1 | J(1,i) \ne 0\}$. There exists at least two $J(1,i), i=1,...,i_c+1$ that are equal to zero. As $J(1,i_c)=1$ there exists at least one $1 \le i \le i_c-1$ such that $J(1,i)=0$. Consequently, there is an idle time unit on the first processor which may be used for the forward step computations preceding those at the i_c^{th} time unit on the second processor.

After every switch from the backward to the forward step, there is a switch from the forward to the backward step, as the backward step computations always perform last. Consider a switch from the forward to the backward step, i.e., $J(2,i_e)=1$ and $J(2,i_e+1)=-1$. The value $J(1,i_e+1)$ is

equal to zero unless $J(2, i_e - 1) = -1$ (see (39)). In this case there is no collision due to switching from the backward to the forward step (see above). After each collision, an additional vacant time unit appears due to the forward-to-backward switch, and it might be used to resolve the next collision. Therefore, we adopt the following single-valued assignment:

$$\begin{aligned} J(1, i_{min}) &= 1 \quad \text{if} \quad J(2, i) = 1 \\ J(1, i+2) &= -1 \quad \text{if} \quad J(2, i) = -1 \\ J(1, i) &= 0 \quad \text{otherwise,} \end{aligned} \tag{40}$$

where $i_{min} = \min(1 \leq j \leq i | J(1, j) = 0)$.

The obtained schedule meets the pipelined nature of PTAs (5,7) as for any portion of lines the forward computations are performed earlier on the first processor than on the second one, and the backward computations are performed earlier on the second processor than in the first one. Obviously, the inequality (6) is satisfied, as there are no changes of the schedule of the last outermost processor. This proves the theorem. □

Let us define a variable that governs communications between processors:

$$Com(p, i) = \begin{cases} 0 & \text{processors } p \text{ and } p+1 \text{ do not communicate} \\ 1 & \text{send forward-step coefficients from } p \text{ to } p+1 \\ 2 & \text{receive solution from } p+1 \text{ to } p \\ 3 & \text{send forward-step coefficients and receive solution,} \end{cases} \tag{41}$$

where p refers to communications between the pth and the $(p+1)^{th}$ computers, i is the end of the ith time unit (or beginning of the $(i+1)^{th}$ time unit) on the pth processor.

If $J(p, i)$ is given, the variable $Com(p, i)$ is computed by

$$Com(p, i) = \begin{cases} 1 & \text{if } J(p+1, i) = 1 \\ 2 & \text{if } J(p+1, i-1) = -1 \text{ and } J(p+1, i) \neq 1 \\ 3 & \text{if } J(p+1, i-1) = -1 \text{ and } J(p+1, i) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{42}$$

The end of the ith time unit on the pth processor corresponds to the beginning of the ith time unit on the $(p+1)^{th}$ processor. If $J(p+1, i) = 1$, the $(p+1)^{th}$ processor receives the forward step coefficients. The first line in (42) defines the transfer of the forward step coefficients from the pth to the $(p+1)^{th}$ processor.

The $(p+1)^{th}$ processor sends the backward step solution to the pth processor immediately after the completion of their computations. The second line of (42) defines the solution transfer for backward pipelined computations.

The third line of (42) corresponds to the case of simultaneous transfer of the forward step coefficients from the pth to the $(p+1)^{th}$ processor and the backward step solution from the $(p+1)^{th}$ to the pth processor. If the send and receive operations are performed simultaneously, parallelism of communications appears as an additional advantage of the IB-PTA.

One recursive algorithm that assigns the schedule of computations to the pth processor and the schedule of communications between the pth and the $(p+1)^{th}$ processors for a given schedule of the $(p+1)^{th}$ processor, is shown in Fig. 4. First, this algorithm assigns zero values to J and Com variables corresponding to the pth processor. For negative $J(p+1, i)$ the values $J(p, i+2) = -1$ and $Com(p, i+1) = 2$ are assigned (see (40) and (42)).

For positive $J(p+1,i)$, the value of $Com(p,I)$ is equal either to 1 or to 3. The latter case is realized if $J(p+1,i-1)=-1$ (see (42)), and the previous $Com(p,i)=2$ is reassigned. Then the search for a vacant time unit i_{min} (see (40)) is realized and the assignment $J(p,i_{min})=1$ is completed. The entire loop is repeated $2(P-p)+L_f+L_b$ times according to Theorem 3.5.

Theorem 4.1 states that the algorithm is correct. The only reason that the algorithm is unable to find a vacant time unit for the forward step computations (error warning) is an erroneous schedule ($J(P,i)$) on the last outermost processor (see below).

Thus, a valid schedule should be assigned to the Pth outermost processor before recursive computations of the time schedule for other processors are executed. This schedule must satisfy condition (4). To keep the minimum idle time for the system of P processors, the Pth processor should execute forward and backward step computations in a contiguous way. If the outermost processor is idle some time, we cannot obtain a schedule corresponding to the minimum elapsed time on other processors.

The assignment of $J(P,i)$ for the basic PTA is simple:

$$J(P,i)=1 \quad \text{if} \quad i=1,...,L_f \tag{43}$$
$$J(P,i)=-1 \quad \text{if} \quad i=L_f+1,...,L_f+L_b.$$

The algorithm for assignment of $J(P,i)$ for the IB-PTA is shown in Fig. 5. First, the forward step computations are performed for the first portion of lines. The variable KS is equal to the number of available lines which have been completed by the forward step and not yet rendered by the backward step.

Scheduling the IB-PTA for the Pth processor, the backward step computations begin for the next portion of K_2 lines once these lines have been completed by the forward step ($KS \geq K_2$). Otherwise, one more portion of lines must be completed by the forward step computations.

For example, consider the tri-diagonal Thomas algorithm with $K_2/K_1 = g_1/g_2 = 1.5$. [1] Once two first portions of K_1 lines have been completed by the forward step, $1.5K_1$ lines from these two portions form the first portion for the backward step computations. The remaining $0.5K_1$ lines and the third portion of lines form the second portion of K_2 lines for the backward step computations. This loop repeats until the completion of lines.

The processor schedules for the IB-PTA and the basic PTA are shown in Fig. 6a,b respectively. The pth column of values of J corresponds to the pth processor (from 1 to P). Columns are shifted so as each horizontal line corresponds to a single time moment in terms of wall clock. Arrows $--->$, $<---$ and $<-->$ denote the send, receive and send-receive communications that correspond to the values of the variable Com (see (42)). One can find that the maximum elapsed time (length of the first column) is the same for the PTA and the IB-PTA. Of course, the idle time (the number of zeros) is also the same. The advantage of the IB-PTA is that the processors become idle after completion of some lines, therefore, the processors may be used for other computational tasks using these data.

5. Conclusions. The ways of efficient parallelization of two-step (forward-backward) pipelined algorithms (PAs) originated from the solution of banded matrix linear systems on parallel computers are discussed. The recursive algorithm for the assignment of the processor computation and communication schedule for PAs is derived. This scheduling algorithm can be used to execute multiblock computations where different lines come to a processor from different neighbors, to reduce idle time

[1] the exact ratio is compiler- and computer-dependent

by scheduling other computational tasks while processors are idle from PA computations and to implement those versions of PAs which can improve the parallelization efficiency.

It is shown theoretically that the processor idle time of the pipelined Thomas algorithms cannot be improved directly.

To remedy the problem of the processor idle time, the new parallel version of the Thomas algorithm (IB-PTA) is proposed and denoted as the Immediate Backward Pipelined Thomas Algorithm (IB-PTA). The backward step is processed immediately after the forward step has been completed for the first portion of lines. Thus, part of the lines have been computed by the Thomas algorithm before processors become idle, and the idle processors can perform other computational data-dependent tasks. The obtained processor schedule for the proposed IB-PTA and basic PTA are presented. These schedules confirm our theoretical results about PAs.

6. Acknowledgment. The author wishes to thank Prof. Alex Pothen (ODU and ICASE) and Dr. Stephen Guattery (ICASE) for careful reading of the manuscript and useful suggestions.

REFERENCES

[1] Hirsch, Ch. *Numerical computation of internal and external flows, Vol. 1: Fundamentals of numerical discretization,* John Wiley and Sons, Chichester, 1994.

[2] Johnson, S.P, Leggett, P.F., Ierotheou, C.S. et al., *Computer Aided Parallelization Tools (CAPTools). User Manual,* University of Greenwich, UK, 1996, http://www.gre.ac.uk/~captool/.

[3] Gustafson, F.G. and Gupta, A., *A New Parallel Algorithm for Tridiagonal Symmetric Positive Definite Systems of Equations,* Proceedings of the Third International Workshop of Applied Parallel Computing, PARA'96, pp. 341-349.

[4] Povitsky, A. and Wolfshtein, M., *Multi-domain Implicit Numerical Scheme,* International Journal for Numerical Methods in Fluids, 25 (1997), pp. 547-566.

[5] Naik, N.H., Naik, V.K. and Nicoules, M., *Parallelization of a Class of Implicit Finite Difference Schemes in Computational Fluid Dynamics*, International Journal of High Speed Computing, 5, 1993, pp. 1-50.

[6] Povitsky, A., *Parallel Directionally Split Solver Based on Reformulation of Pipelined Thomas Algorithm,* ICASE Report No. 98-45.

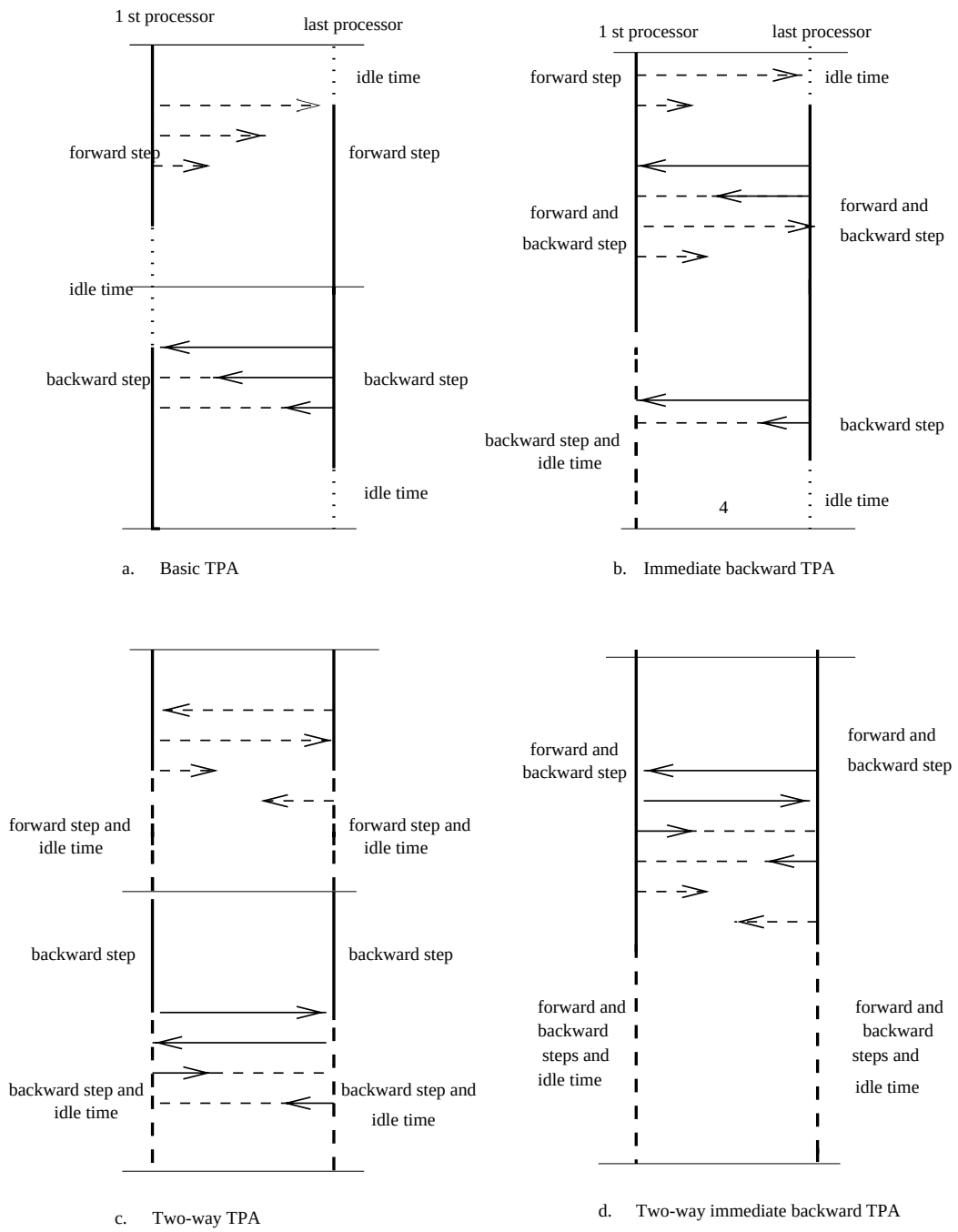

FIG. 1. *Examples of pipelined Thomas algorithms (a) basic Pipelined Thomas Algorithm (PTA); (b) Immediate Backward Pipelined Thomas Algorithm (IB-PTA); (c) First-Last Pipelined Thomas Algorithm (FL-PTA); (d) First-Last Immediate Backward Pipelined Thomas Algorithm (FL-IB-PTA), where - - - > denotes the forward step, ⟶ denotes the backward step*

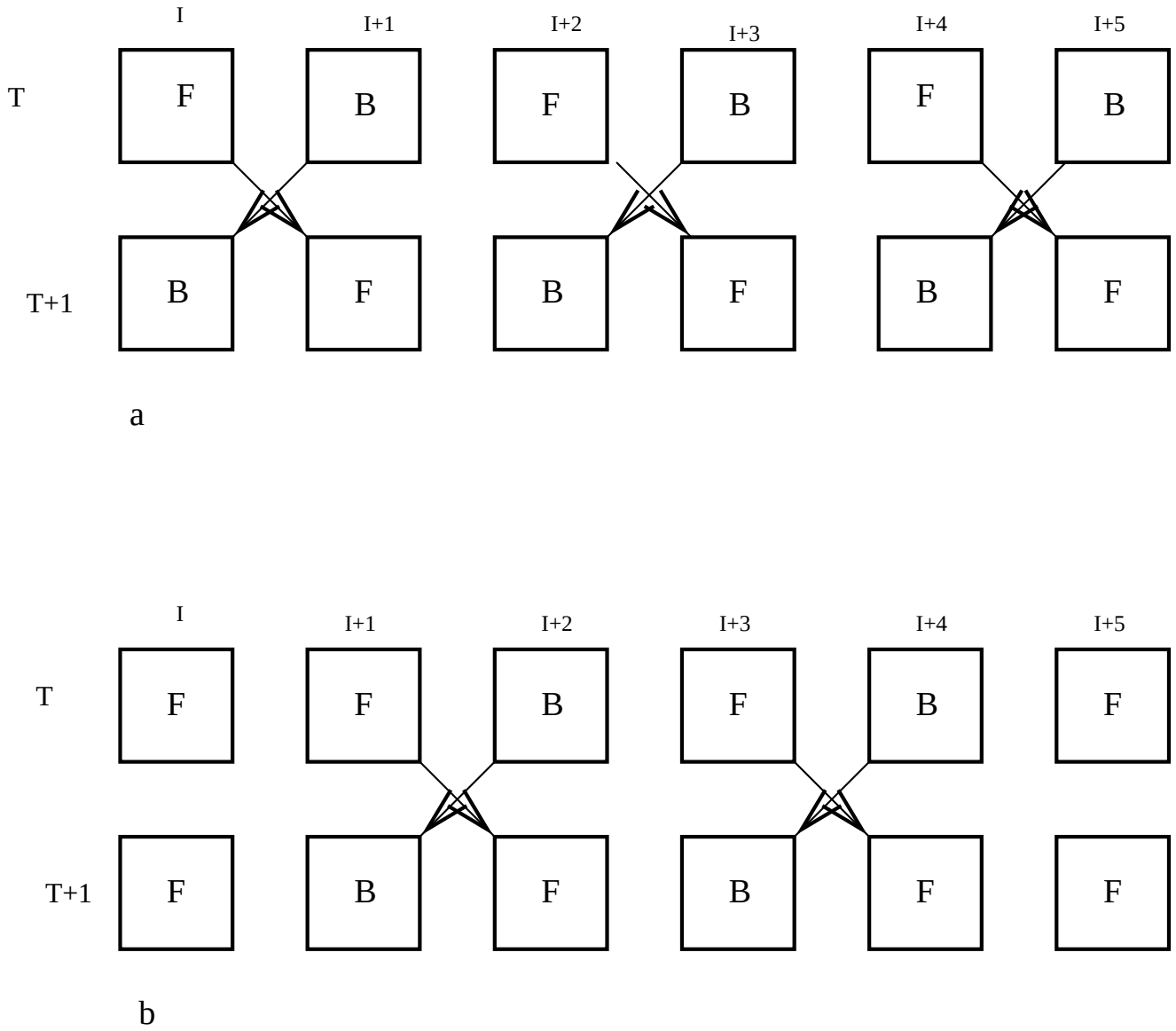

FIG. 2. *Sequence of forward and backward steps for the IB-PTA at two sequential time units, where I,...,I+5 are the numbers of processors, F is the forward step, B is the backward step, arrows denote data transfer between processors; (a) equal computational times for the F and B steps* ($g_1 = g_2$)*; (b) non-equal computational times for the F and B steps, corresponding to the tri-diagonal Thomas algorithm* ($g_1 = 1.5g_2$)

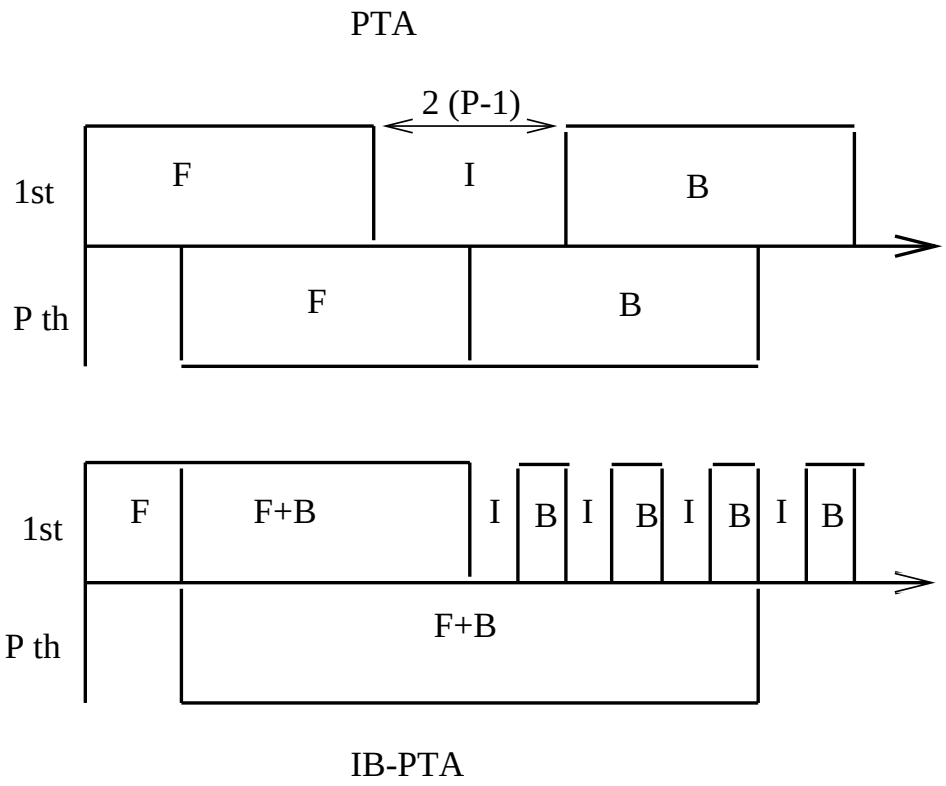

FIG. 3. *Pipelined Thomas algorithms: F-the forward step, B-the backward step, I-the idle state of processors, (a) basic pipelined Thomas Algorithm (PTA); (b) Immediate Backward Pipelined Thomas Algorithm (IB-PTA)*

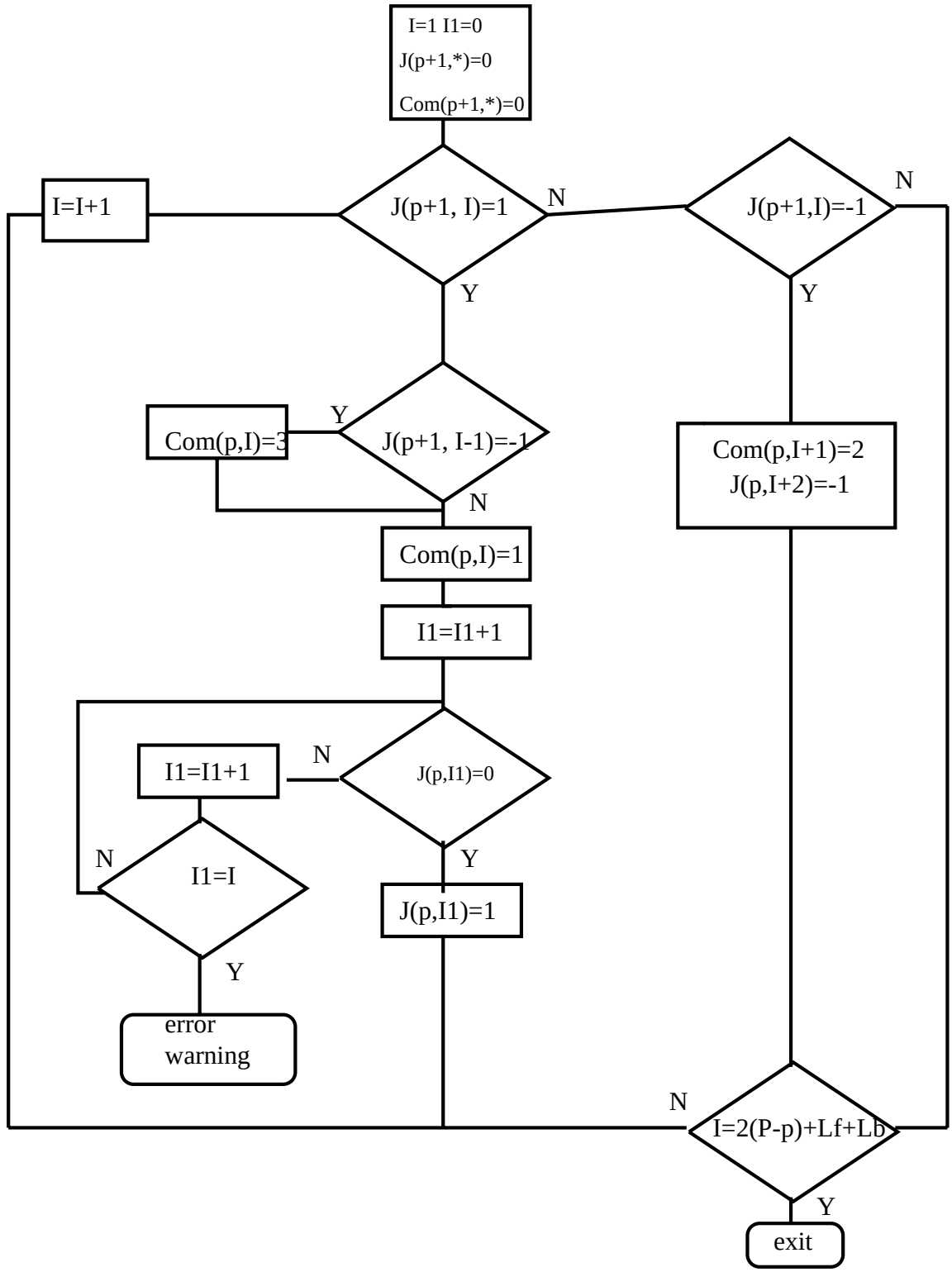

FIG. 4. *Assignment of processor schedule on the* $(p+1)^{th}$ *processor based on a given schedule on the* p^{th} *processor*

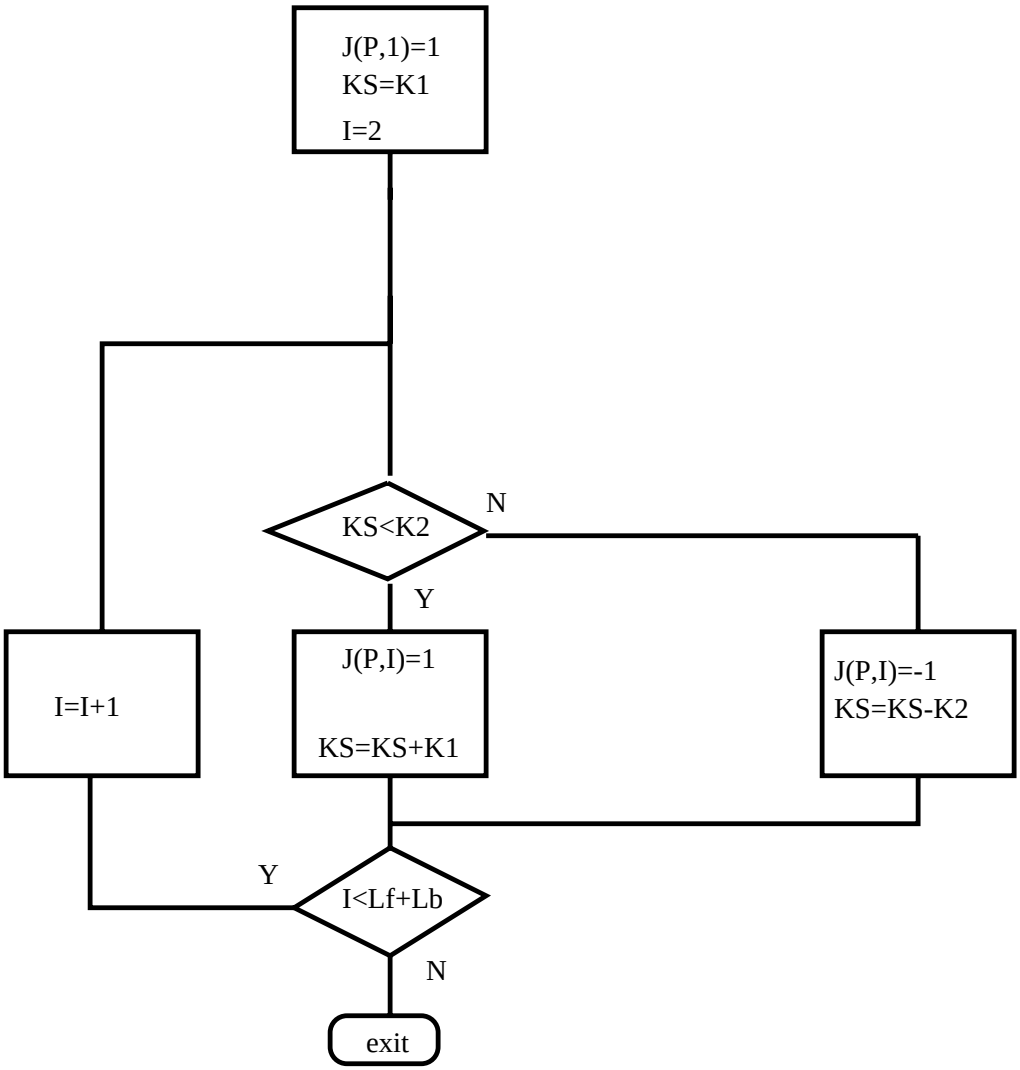

FIG. 5. *Assignment of the schedule on the last processor (P^{th}) for the IB-PTA*

a

```
 1
   --->
 1       1
   --->    --->
 1       1       1
   --->    --->    --->
 1       1       1       1
   --->    --->    --->    --->
 1       1       1       1       1
   --->    --->    --->    --->
 1       1       1       1       1
   --->    --->    --->
 1       1       1       1      -1
   --->    --->            <-->
 1       1       1      -1       1
   --->            <-->
 1       1      -1       1      -1
           <-->            <-->
 0      -1       1      -1       1
   <-->            <-->    --->
-1       1      -1       1       1
           <-->    --->
 0      -1       1       1      -1
   <---    --->            <-->
-1       0       1      -1       1
                   <-->
 0       0      -1       1      -1
           <---            <-->
 0      -1       0      -1       1
   <---                    <-->
-1       0       0      -1       1
                   <---
 0       0      -1       0      -1
           <---            <---
 0      -1       0      -1
   <---            <---
-1       0      -1
           <---
 0      -1
   <---
-1
```

b

```
 1
   --->
 1       1
   --->    --->
 1       1       1
   --->    --->    --->
 1       1       1       1
   --->    --->    --->    --->
 1       1       1       1       1
   --->    --->    --->    --->
 1       1       1       1       1
   --->    --->    --->    --->
 1       1       1       1       1
   --->    --->    --->    --->
 1       1       1       1       1
   --->    --->    --->    --->
 1       1       1       1       1
   --->    --->    --->    --->
 0       1       1       1       1
           --->    --->    --->
 0       0       1       1       1
                   --->    --->
 0       0       0       1       1
                           --->
 0       0       0       0       1
 0       0       0       0      -1
                           <---
 0       0       0      -1      -1
                   <---    <---
 0       0      -1      -1      -1
           <---    <---    <---
 0      -1      -1      -1      -1
   <---    <---    <---    <---
-1      -1      -1      -1      -1
   <---    <---    <---    <---
-1      -1      -1      -1      -1
   <---    <---    <---    <---
-1      -1      -1      -1
   <---    <---    <---
-1      -1      -1
   <---    <---
-1      -1
   <---
-1
```

FIG. 6. *Schedule of processors for the IB-PTA (a) and PTA (b),* $P = 5, L_f = 9, L_b = 6$