Tatra Mt. Math. Publ. **30** (2005), 93–100

TATRA MOUNTAINS
Mathematical Publications

ASYMPTOTICAL BEHAVIOUR OF THE SPEED-UP OF ONE PARALLEL ALGORITHM

Pavol Purcz

ABSTRACT. An earlier suggested parallel "ring" algorithm for solving the spatially one-dimensional initial-boundary-value problem (IBVP) for a parabolic equation using an explicit difference method is shortly described. Asymptotical behaviour of the speed-up function of this parallel algorithm is studied. The speed-up function is determined as the ratio between necessary times for realization of the algorithm in sequentional and parallel cases. Theoretical estimates of the speed-up function show the significant speed-up of the parallel algorithm in comparison with the serial one for large values of the parameter q, where q is the maximum of values computed by one processor during one time level. It is shown that the coefficient of the speed-up tends to number of using processors, if the parameter q tends to infinity.

1. Introduction

Numerical applications tend to be dominant in parallel scientific computing. The importance of boundary-value problems (BVP) for engineering applications motivates continuous development of fast numerical algorithms and their solution as well as the effective usage of parallel computational systems. A number of various approaches have been suggested, and some of them can be found in [1], [2], [3], [7], [9] and in other papers.

Some theoretical aspects of using an explicit difference method in the case of spatially one-dimensional initial-boundary-value problem (IBVP) were suggested by Tyrtyshnikov [11] as a possible approach to the development of a parallel algorithm. Some properties of recurrence relations like those of the explicit difference method were studied by Kogge [6]. In this paper, such an algorithm is shortly presented also in the case of spatially one-dimensional IBVP. This parallel algorithm in one-, two- and three-dimensional case is described in detail

2000 Mathematics Subject Classification: 35K15, 35K20, 35-04.
Keywords: parallel algorithm, parabolic equation, explicit difference method, speed-up.

Supported by grant VEGA 1/1006/04 9150.

in [8, 10]. Some theoretical estimates of the speed-up function of the algorithm in case of spatially one- and m-dimensional IBVP are given.

2. One-dimensional IBVP problem

Let us consider a spatially one-dimensional IBVP for a parabolic equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad t > 0, \quad 0 < x < l_x, \tag{1}$$

$$u(0, x) = \varphi(x), \quad 0 < x < l_x, \tag{2}$$

$$u(t, 0) = \alpha(t), \quad t > 0, \tag{3}$$

$$u(t, l_x) = \beta(t), \quad t > 0, \tag{4}$$

where φ, α, β are sufficiently smooth given functions such that

$$\varphi(0) = \alpha(0),$$
$$\varphi(l_x) = \beta(0).$$

We look for a sufficiently smooth function $u(t, x)$, which satisfies the equation (1) with conditions (2)–(4). For the numerical solution of the problem (1)–(4) let us consider the following explicit difference method of the form

$$\frac{u_i^k - u_i^{k-1}}{\tau} = \frac{u_{i-1}^{k-1} - 2u_i^{k-1} + u_{i+1}^{k-1}}{h_x^2}, \tag{5}$$

where $\tau > 0$ ($h_x > 0$) is time (spatial) step of discretization, u_i^k are approximate values of the solution at the points (k, i), ($i = 1, 2, \ldots, n_x - 1$, $k = 1, 2, \ldots, n_x \times h_x = l_x$). Moreover, $u_0^k = \alpha(t_k)$, $u_{n_x}^k = \beta(t_k)$, and $u_i^0 = \varphi(x_i)$, where $x_i = ih_x$, $t_k = k\tau$.

Let us consider a natural number $n_x = q \cdot n$, where $q \geq 1$, $n \geq 2$. Then the explicit difference scheme (5) can be realized on a parallel computer with n processors $P_1, P_2, \ldots, P_n$.

At the beginning each processor computes values u_i^0 at $q+1$ points lying on the axis O_x using the initial condition (2). Then each processor starts computation of values in the first time level according to the difference scheme (5). In order to compute each value u_i^1, $i = 1, \ldots, n_x - 1$, we need three values u_{i-1}^0, u_i^0, u_{i+1}^0 from the previous time level. By the same procedure it is possible to determine next values u_i^k in the next time level. The values computed by one processor create points lying on tops, edges or inside a triangle. The set of values in all mentioned points is said to be a triangular block of data. If some processor P_i has not enough data for computing values u_i^k, the next phase of the computational process—a data transfer between processors must be determined. By analyzing missing data for the next computation of the processor P_i it is not difficult to find out that those are values at the points lying in the two next marginal levels on the one side of the triangle created by neighbouring processor. After finishing the data transfer a new stage of computation using already transferred values follows. In this phase each processor P_i may compute next values u_i^k using the difference scheme (5) and the set of data in its memory. All values obtained at this computation stage by one processor lie inside of the considered area. We can visualize them as points lying inside or on the borders of a square. The set of values at these points is said to be a square block of data. So the whole process of computations and data transfers is regularly repeated. Fig. 1 schematically describes the "ring" implementation of whole process in the case of 5 processors and the width of q points for one processor, where q is the maximum of values which are computed by one processor during one time level. Generally, for n processors the whole data exchange process can be written in the form:

$$P_{(i \bmod n)+1} \to P_i \to P_{(n+i-2 \bmod n)+1}\,, \qquad i = 1, 2, \ldots, n\,. \tag{6}$$

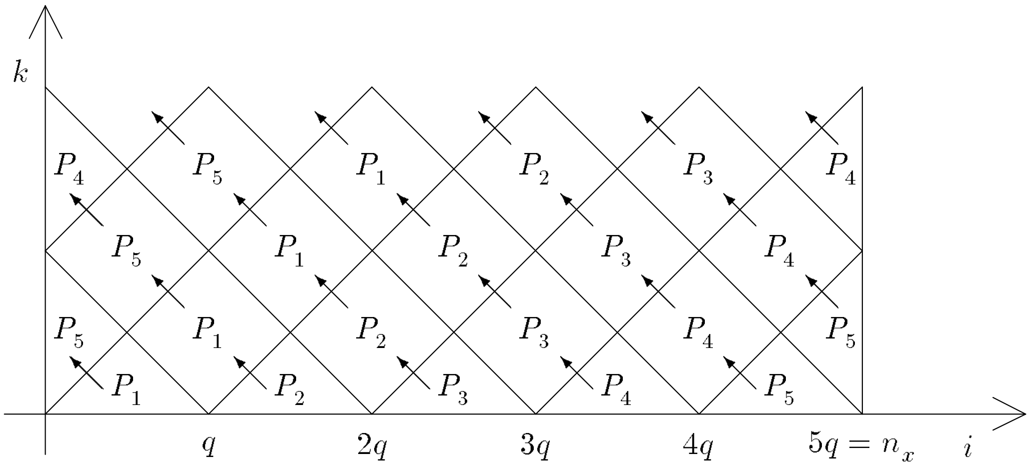

FIGURE 1. The parallel algorithm with five processors ($n = 5$). The arrows show the direction of the data transfer.

3. Speed-up function of the algorithm

3.1 Studied problem

Let us consider a multi-processor computational system which performs two kinds of actions:

a) numerical operations (addition, multiplication, …),

b) information interchanges.

For many computational systems the mean time t_o of the action of the type (a) is often much less than the time t_c of the action of the type (b), i.e., $t_o \ll t_c$. Let us suppose that $t_c \approx w t_o$; $w \gg 1$. The term information interchanges, or shortly interchanges represents transfer of data between processors. When the performance of an algorithm is necessary for NO numerical operations and NC interchanges, then the time t of its realization under the above conditions will have the form:

$$t = \mathrm{NO} \cdot t_o + \mathrm{NC} \cdot t_c . \tag{7}$$

A detailed discussion of the speed-up and communication complexity can be found in [4], [5] and [12].

Other splitting strategies of the computational effort between processors can be found in [13], [14], [15] and elsewhere.

3.2 Speed-up function in one-dimensional case

We suppose that points u_i^0, $i = 0, 1, \dots, n_x$, given by initial condition $\varphi(x)$, are computed in advance and stored in memory. We consider a layer which is $nq + 1$ points long and q points wide. This layer regularly repeats in all considered parallel process. Each processor needs (after expressing an element u_i^k) five arithmetical operations for computation of the values u_i^k, $i = 1, \dots, n_x$, $k > 0$, according to the difference scheme (5). In general, the number of arithmetical operations for the computation of one point u_i^k inside the considered layer and on its border is not the same. It depends on the computational complexity of the points specified by boundary conditions. We denote the number of arithmetical operations needed for computation of the values u_0^k and $u_{n_x}^k$ by $r(\alpha)$ and $r(\beta)$, respectively. Then the whole number of arithmetical operations needed for computation of the values in the considered area in the sequentional case is

$$\mathrm{NO}_{\mathrm{seq}} = 5(n_1 q - 1)q + \big[r(\alpha) + r(\beta)\big]q = \big[5n_1 q + P\mathrm{seq}_0(q)\big]q$$

and in the parallel case it is

$$\mathrm{NO}_{\mathrm{par}} = \max\Big\{5q^2;\ 5\frac{q^2}{2} + 5\Big(\frac{q^2}{2} - q\Big) + \big[r(\alpha) + r(\beta)\big]q\Big\} = \big[5q + P\mathrm{par}_0(q)\big]q .$$

On a parallel computer with n processors ordered in geometrical visualization as a ring, we can suppose, according to (6), the corresponding connections for data transfer between each two neighbouring processors (i.e., processor P_1 is connected with processor P_2, processor P_2 is connected with processor P_3, etc. and processor P_n is connected with processor P_1). In the parallel case each processor needs according to the description of the algorithm q interchanges in the data exchange process in the considered area. Then

$$\mathrm{NC}_{\mathrm{seq}} = 0 ,$$
$$\mathrm{NC}_{\mathrm{par}} = q = qPC\mathrm{par}_0(q) ,$$

where $P\text{seq}_0(q)$, $P\text{par}_0(q)$ and $PC\text{par}_0(q)$ are 0-degree polynomials of the variable q. (In this subsection, in case $m = 1$ these polynomials are constants.) Summarizing,

$$t_{\text{seq}} = \text{NO}_{\text{seq}}\, t_o + \text{NC}_{\text{seq}}\, t_c = \left[5n_1 q + P\text{seq}_0(q)\right] q t_o$$
$$t_{\text{par}} = \text{NO}_{\text{par}}\, t_o + \text{NC}_{\text{par}}\, t_c = \left[5q + P\text{par}_0(q)\right] q t_o + q \cdot PC\text{par}_0(q) t_c\,.$$

Assuming $t_c \approx w t_o$; $w \gg 1$, we can write the last equation in the form:

$$t_{\text{par}} = \left[5q + P\text{par}_0(q)\right] q t_o + q PC\text{par}_0(q) w t_o\,.$$

Now we are able to define the speed-up function $f(n,q)$ as the ratio of time needed for the realization of the algorithm in sequential case and the time in parallel case. This gives:

$$f(n,q) = \frac{t_{\text{seq}}}{t_{\text{par}}} = \frac{5nq + P\text{seq}_0(q)}{5q + P\text{par}_0(q) + wPC\text{par}_0(q)}\,. \tag{8}$$

3.3 Speed-up function in m-dimensional case

Let us consider m-dimensional IBVP and a natural number expressed in the form $\prod_{j=1}^{m} n_j$. Then explicit difference scheme

$$\frac{u^{k}_{i_1 i_2 \ldots i_m} - u^{k-1}_{i_1 i_2 \ldots i_m}}{\tau} = \sum_{j=1}^{m} \frac{u^{k-1}_{i_1 i_2 \ldots i_{j-1} \ldots i_m} - 2u^{k-1}_{i_1 i_2 \ldots i_m} + u^{k-1}_{i_1 i_2 \ldots i_{j+1} \ldots i_m}}{h^2_{x_j}} \tag{9}$$

can be realized on a parallel computer with $\prod_{j=1}^{m} n_j$ processors ordered in geometrical visualization in a spatially m-dimensional cube of size $n_1 \times n_2 \times \cdots \times n_m$. Similarly as in one-dimensional case, we can suppose the corresponding connections for data transfer between each neighbouring processors. A term neighbouring processors means the processor standing (in ordering of the spatially m-dimensional cube) in the direction of each axis in this m-dimensional space. Moreover, both the first and the last processor ordered in one line (or in the direction in one axis), are also connected and create the ring-ordering connection in this one direction. The number of arithmetical operations needed for the computation of the values according to the difference scheme (9) can be determined via mathematical induction.

Lemma. *The number of arithmetical operations needed for the computation of the values according to the difference scheme* (9) *in case of spatially* m*-dimensional* IBVP *is equal to* $\text{NOseq} = 4m + 1$.

P r o o f . The first step was determined in the previous subsection. This means that the statement of our lemma holds for $m = 1$. Let us consider spatially l-dimensional IBVP ($l \geq 1$) along with corresponding explicit difference scheme in the form:

$$\frac{u^k_{i_1 i_2 \ldots i_l} - u^{k-1}_{i_1 i_2 \ldots i_l}}{\tau} = \sum_{j=1}^{l} \frac{u^{k-1}_{i_1 i_2 \ldots i_{j-1} \ldots i_l} - 2u^{k-1}_{i_1 i_2 \ldots i_l} + u^{k-1}_{i_1 i_2 \ldots i_{j+1} \ldots i_l}}{h^2_{x_j}} . \tag{10}$$

Let us consider that $\mathrm{NOseq} = 4l + 1$.

Now, let us consider spatially $l + 1$-dimensional IBVP. Then to right side of the explicit difference scheme (10) we must add a term

$$\frac{u^{k-1}_{i_1 i_2 \ldots i_l} - 2u^{k-1}_{i_1 i_2 \ldots i_{l+1}} + u^{k-1}_{i_1 i_2 \ldots i_{l+2}}}{h^2_{x_{l+1}}}$$

which means that addition of each next one dimension results in performing extra 4 numerical operations. Then, $\mathrm{NOseq} = 4l + 1 + 4 = 4(l + 1) + 1$ and the proof is finished. □

Thus

$$\mathrm{NOseq} = \left[(4m + 1)\, q^m \prod_{j=1}^{m} n_j + P\mathrm{seq}_{m-1}(q)\right] q\,, \tag{11}$$

$$\mathrm{NOpar} = \left[(4m + 1)\, q^m + P\mathrm{par}_{m-1}(q)\right] q\,, \tag{12}$$

$$\mathrm{NCseq} = 0\,, \tag{13}$$

$$\mathrm{NCpar} = qPC\mathrm{par}_{m-1}(q)\,, \tag{14}$$

where $P\mathrm{seq}_{m-1}(q)$, $P\mathrm{par}_{m-1}(q)$ and $PC\mathrm{par}_{m-1}(q)$ are $(m-1)$-degree polynomials of the variable q. Relations (11)–(12) follow directly from the extension of one-dimensional IBVP to m-dimensional one. Relation (14) follows from the facts that the transferred data lie on the border of spatially $(m+1)$-dimensional objects, created in geometrical visualization by each processor in a parallel process. These objects lie in a similar layer, which is described in the previous subsection. The considered layer is $\prod_{j=1}^{m} (n_j q + 1)$ long (in the spatial axises) and q point wide (in the time axis).

To sum up,

$$t_{\mathrm{seq}} = \mathrm{NO}_{\mathrm{seq}}\, t_o + \mathrm{NC}_{\mathrm{seq}}\, t_c = \left[(4m + 1)\, q^m \prod_{j=1}^{m} n_j + P\mathrm{seq}_{m-1}(q)\right] q t_o\,,$$

$$t_{\mathrm{par}} = \mathrm{NO}_{\mathrm{par}}\, t_o + \mathrm{NC}_{\mathrm{par}}\, t_c = \left[(4m + 1)\, q^m + P\mathrm{par}_{m-1}(q)\right] q t_o + qPC\mathrm{par}_{m-1}(q) t_c\,.$$

Assuming $t_c \approx wt_o$; $w \gg 1$, we can write the last equation in the following form:

$$t_{\text{par}} = \left[(4m+1)q^m + P\text{par}_{m-1}(q)\right]qt_o + qPC\text{par}_{m-1}(q)wt_o\,.$$

Hence, in general, for spatially m-dimensional IBVP we have the following theorem.

Theorem.

$$f(n_1, n_2, \ldots, n_m, q) = \frac{(4m+1)q^m \prod\limits_{j=1}^{m} n_j + P\text{seq}_{m-1}(q)}{(4m+1)q^m + P\text{par}_{m-1}(q) + wPC\text{par}_{m-1}(q)}\,. \tag{15}$$

Corollary.

$$\lim_{q\to\infty} f(n_1, n_2, \ldots, n_m, q) = \prod_{j=1}^{m} n_j\,. \tag{16}$$

4. Conclusion

Now, we can analyze the speed-up function of the considered parallel algorithm in both cases, spatially one- or m-dimensional IBVP. As it can be seen, by using a given multi-processor computational system, both the number of processors n or $\prod\limits_{j=1}^{m} n_j$ and the coefficient w become constants. Solving a concrete IBVP on the given computational system, a number of arithmetical operations needed for computation of the values using boundary conditions becomes also constant. Then the speed-up function depends only on parameter the q—maximum of values which are computed by one processor during one time level; this is related to the selected density of data by spatial discretization of the considered area. If the parameter q is small, in some cases it could happen that the sequentional algorithm will be more effective than the parallel one. On the other side, the theoretical estimates of the speed-up function show the significant speed-up for large values of the parameter q in comparison with the serial implementation of the difference method. Moreover, the asymptotical behaviour shows that, provided the parameter q tends to infinity, then the speed-up tends to an ideal value—number of processors.

REFERENCES

[1] BURRAGE, K.: *Parallel methods for initial value problems*, Appl. Numer. Math. **11** (1993), 5–25.

[2] CRANK, J.—NICOLSON, P.: *A practical method for numerical evaluation of solutions of PDEs of the heat-conduction type*, Proc. Camb. Philos. Soc. **43** (1947), 60–67.

[3] FREEMAN, T. L.—PHILLIPS, C.: *Parallel Numerical Algorithms*, Prentice Hall, New York, 1992.

[4] GALLIVAN, K.—PLEMMONS, R. J.—SAMEH, A. H.: *Algorithms for dense linear algebra computations*, SIAM Rev. **32** (1990), 54–135.

[5] GOLUB, G. H.—VAN LOAN, C. F.: *Matrix Computations*, The Johns Hopkins University Press, Baltimore and London, 1989.

[6] KOGGE, P. M.: *Parallel solution of recurrence problems*, IBM J. Res. Dev. **18** (1974), 138–148.

[7] ORTEGA, J. M.—VOIGT, R. G.: *Solution of PDE on Vector and Parallel Computers*, SIAM, Philadelphia, 1985.

[8] PURCZ, P.: *Generalization of the parallel algorithm for spatially two—and three—dimensional Dirichlet problem for a parabolic equation*, J. Electrical Engineering **50** (1999), 36–39.

[9] PEACEMAN, D. W—RACHFORD, H. H.: *The numerical solution of parabolic and elliptic differential equations*, J. Soc. Indust. Appl. Math. **3** (1955), 28–41.

[10] PURCZ, P.: *Parallel algorithm for spatially one- and two-dimensional initial-boundary--value problem for a parabolic equation*, Kybernetika **37** (2001), 171–181.

[11] TYRTYSHNIKOV, E. E.: *Parallelization of some numerical methods*, in: Numerical Solution of Partial Differential Equation, Lecture at the Slovak Mathematical Society Seminar, Košice, Slovakia, 1992.

[12] TYRTYSHNIKOV, E. E.: *Blocks algorithms of linear algebra*, in: Computational Processes and Systems, N9, Nauka, Moscow, 1993, pp. 3–34.

[13] ZOETEWEIJ, P.: *A coordination-based framework for distributed constraint solving*, in: Recent Advances in Constraint (B. O'Sullivan, ed.), Lecture Notes Comput. Sci., Vol. 2627, Springer-Verlag, Berlin, 2003, pp. 171–184.

[14] ADVE, V. S.—MELLOR-CRUMMEY, J.—ANDERSON, M.—WANG, J. CH.—REED, D. A.—KENNEDY, K.: *An integrated compilation and performance analysis environment for data parallel programs*, in: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM), San Diego, California, USA, 1995, pp. 50.

[15] PINI, G., PUTTI, M.: *Parallel finite element Laplace transform method for the non-equilibrium Groundwater transport equation*, Internat. J. Numer. Meth. Eng. **40** (1997), 2653–2664.

Received December 2, 2003

Department of Mathematics
Faculty of Civil Engineering
Technical University of Košice
Vysokoškolská 4
SK–042 00 Košice
SLOVAKIA
E-mail: Pavol.Purcz@tuke.sk