SIAM J. SCI. STAT. COMPUT.
Vol. 11, No. 2, pp. 354-367, March 1990

009

VECTORIZATION OF THE ODD-EVEN HOPSCOTCH SCHEME AND THE ALTERNATING DIRECTION IMPLICIT SCHEME FOR THE TWO-DIMENSIONAL BURGERS EQUATIONS*

E. D. DE GOEDE† AND J. H. M. TEN THIJE BOONKKAMP‡

Abstract. A vectorized version of the odd-even hopscotch (OEH) scheme and the alternation direction implicit (ADI) scheme have been implemented on vector computers for solving the two-dimensional Burgers equations on a rectangular domain. This paper examines the efficiency of both schemes on vector computers. Data structures and techniques employed in vectorizing both schemes are discussed, accompanied by performance details.

Key words. vector computers, Burgers equations, odd-even hopscotch scheme, alternating direction implicit scheme, vectorization

AMS(MOS) subject classifications. primary 65V05; secondary 65M05, 76DXX

1. Introduction. This report is written as a contribution to a project for developing numerical software for vector- and parallel computers. Vectorized versions of the odd-even hopscotch (OEH) scheme and the alternating direction implicit (ADI) scheme are developed in FORTRAN77 for the two-dimensional Burgers equations. In the near future, the vectorized codes will be combined with a pressure correction technique [8], [13] in order to solve the time-dependent, incompressible, Navier-Stokes equations.

The OEH scheme and the ADI scheme are integration schemes for time-dependent partial differential equations (PDEs) and are applicable to wide classes of problems. The OEH scheme has shown to be an efficient scheme on serial (scalar) computers, in the sense that it is fast per timestep. Moreover, the scheme is relatively easy to implement. Due to its near-explicitness the OEH scheme is also very suitable for use on vector computers. A detailed discussion of the OEH scheme is given in [4]. The ADI scheme we consider in this report is the Peaceman-Rachford scheme [11]. The ADI scheme is more expensive per timestep than the OEH scheme since it requires the solution of tridiagonal systems of equations. However, the ADI scheme is more robust than the OEH scheme.

For the solution of the tridiagonal systems we use the Gaussian elimination method, a variant of the partition method of Wang [17], which is described in [3], [9], and a method developed by De Goede and Wubs [3]. By the approach of De Goede and Wubs, the tridiagonal systems are solved by a combination of explicit and implicit calculations, thus resulting in an alternating direction explicit-implicit (ADEI) scheme. Since the Gaussian elimination method is a sequential method, this method seems to be unsuitable for use on vector computers. However, for the two-dimensional ADI scheme a number of (independent) tridiagonal systems must be solved. Therefore, this method allows vectorization across the systems. Moreover, this method does not increase the operation count, unlike the above-mentioned partition methods. Furthermore, it turns out that also the partition method and the explicit-implicit method are efficient on vector computers.

* Received by the editors November 18, 1987; accepted for publication (in revised form) November 30, 1988.

† Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, the Netherlands.
‡ Nederlandse Philips Bedryven B.V., P.O. Box 80000, 5600 JA Eindhoven, the Netherlands.

The purpose of this paper is to report our experience in vectorizing both schemes for the two-dimensional Burgers equations. Much effort has been spent in optimizing the FORTRAN code for vector computers, avoiding the explicit use of assembler code. The experiments have been carried out on a (2-pipe) CDC Cyber 205 and a Cray X-MP/24. We used one (portable) code on both machines. Since the code contains many long vector operations, it is our opinion that on other vector machines (such as the NEC SX-2, Fujitsu VP200, Alliant FX/8, etc.) we will also obtain good performances.

Section 2 contains a brief summary of the conceptual features of vector computers, which are relevant to the present application. In § 3 a description of the OEH scheme and the ADI scheme is given. Section 4 is devoted to the description of the techniques used for vectorizing both the OEH scheme and the ADI scheme. In § 5, we compare the accuracy and performance of both schemes. Finally, § 6 contains some concluding remarks.

2. Vector processing. Vector operations fall into two main categories: those that perform floating-point arithmetic, and those that may be called data-motion operations (for example, operations to compress or expand an array using an index-list). The need for vector data-motion operations also becomes apparent when we consider the definition of a vector on a CDC Cyber 205: a vector is a set of similar elements occupying consecutive memory locations. The reason for this vector definition is that when performing vector operations on a CDC Cyber 205 the input elements stream directly from the memory to the vector pipes (arithmetic units) and the output elements stream directly back into the memory. A Cray-computer accepts vectors for which the number of memory locations between consecutive elements (the so-called stride) is constant.

To enhance an effective data flow rate in order to match the computation rate of vector computers, the memory is divided into memory banks that may operate concurrently. For example, the memory of the CDC Cyber 205 is divided into 16 memory stacks, each of which is divided into eight independent banks. When one memory stack is busy with a memory request, further references to the same stack cannot be made. If a vector operation calls for an operand whose elements are located w words apart in the memory (i.e., stride w), then the data flow rate might be reduced due to the memory conflicts and thus result in a longer vector operation time. So, in order to obtain a good performance on vector computers it is important to consider the data structure very carefully (see § 4) [6].

For an efficient use of vector computers, the compiler plays an important role. The compiler translates FORTRAN DO-loops into vector machine instructions, if possible. This process is called auto-vectorization. The nature of vector operations is such that only DO-loops are candidates for vector operations. Specific characteristics of a given DO-loop determine its vectorizability [1]. It is not always possible to vectorize a code, as in the following example:

$$\begin{aligned} &\text{DO 10 } I = 1, N \\ &\qquad A(I+1) = A(I) + S \qquad (2.1) \\ &\text{10 CONTINUE} \end{aligned}$$

Because in vector processing the arguments must be determinable before the operation starts, this loop cannot be vectorized. This restriction is known as recursion; it conflicts with the nature of vector processing.

In many situations the compiler can be instructed to generate more efficient codes. We have used such instructions, e.g., in the following situation. The compiler can be instructed to vectorize DO-loops, ignoring possible vector dependencies, by inserting a so-called comment-directive:

for the CFT77 compiler (Cray X-MP/24): CDIR$ IVDEP
and for the VAST compiler (Cyber 205): CVD$ NODEPCHK

3. The OEH scheme and the ADI scheme for the two-dimensional Burgers equations. Consider the two-dimensional Burgers equations:

$$\begin{aligned} u_t &= f_1(u, v) \quad \text{with } f_1(u, v) = -uu_x - vu_y + (u_{xx} + u_{yy})/\mathrm{Re}, \\ v_t &= f_2(u, v) \quad \text{with } f_2(u, v) = -uv_x - vv_y + (v_{xx} + v_{yy})/\mathrm{Re}, \end{aligned} \tag{3.1}$$

with Re denoting the Reynolds number and u and v the velocity components in x- and y-direction, respectively. On the boundary Γ of the connected space domain Ω, we prescribe the Dirichlet conditions

$$u = u_\Gamma, \qquad v = v_\Gamma.$$

The Burgers equations have the same convective and viscous terms as the incompressible Navier-Stokes equations, although the pressure gradient terms are not retained. Also a solution to the Burgers equations would not, in general, satisfy the continuity equation. These equations possess the desirable property that exact solutions can be constructed by means of the Cole-Hopf transformation [2]. This enables us to compare the numerical solution of the Burgers equations with the exact solution.

In this section we give a description of the OEH scheme and the ADI scheme for the Burgers equations. The space discretization is discussed in § 3.1 and the time integration in § 3.2.

3.1. Space discretization. For the space discretization the computational domain is covered by a $N \times M$ rectangular staggered grid, with h and k being the grid sizes in x- and y-direction, respectively (see Fig. 1). In a staggered grid different variables are defined at different grid points. The reason for this choice is that in continuation to this report we want to apply the OEH scheme and the ADI scheme to the incompressible Navier-Stokes equations, for which a staggered grid is most suitable [14].

In what follows, U is a grid function approximating the velocity u (likewise for V, F_1, and F_2) with components U_{ij}. The components U_{ij} are numbered lexicographically. The application of standard second-order central differences converts (3.1) into

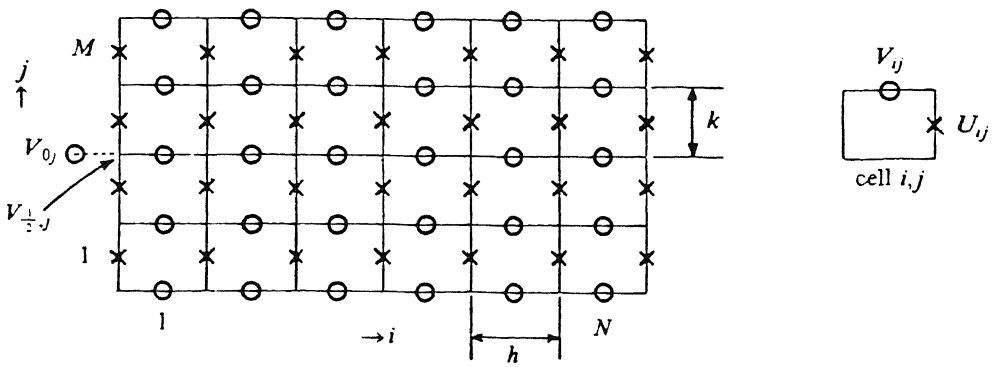


FIG. 1. *The staggered grid.*

the system of ordinary differential equations (ODEs):

$$\text{(3.2a)}\quad \frac{d}{dt}U_{ij} = F_{1,ij}(U, V),\ i=1,\cdots,N-1,\ j=1,\cdots,M \qquad \text{(interior}\times\text{-points)},$$

$$\text{(3.2b)}\quad \frac{d}{dt}V_{ij} = F_{2,ij}(U, V),\ i=1,\cdots,N,\quad j=1,\cdots,M-1 \quad \text{(interior O-points)},$$

where

$$\text{(3.3a)}\quad F_{1,ij} = -U_{ij}\frac{(U_{i+1,j}-U_{i-1,j})}{2h} - \bar{V}_{ij}\frac{(U_{i,j+1}-U_{i,j-1})}{2k} + \frac{1}{\mathrm{Re}}\frac{(U_{i+1,j}-2U_{ij}+U_{i-1,j})}{h^2} + \frac{1}{\mathrm{Re}}\frac{(U_{i,j+1}-2U_{ij}+U_{i,j-1})}{k^2},$$

$$\text{(3.3b)}\quad F_{2,ij} = -\bar{U}_{ij}\frac{(V_{i+1,j}-V_{i-1,j})}{2h} - V_{ij}\frac{(V_{i,j+1}-V_{i,j-1})}{2k} + \frac{1}{\mathrm{Re}}\frac{(V_{i+1,j}-2V_{ij}+V_{i-1,j})}{h^2} + \frac{1}{\mathrm{Re}}\frac{(V_{i,j+1}-2V_{ij}+V_{i,j-1})}{k^2}.$$

In (3.3a) $\bar{V}_{ij}$ represents an approximation to V at the $\times$-points; likewise in (3.3b) $\bar{U}_{ij}$ represents an approximation to U at the O-points. The values of $\bar{U}_{ij}$ and $\bar{V}_{ij}$ are determined by averaging over neighbouring values. For the ADI scheme $\bar{U}_{ij}$ and $\bar{V}_{ij}$ are trivially defined by

$$\text{(3.4a)}\quad \bar{U}_{ij} = \tfrac{1}{4}(U_{ij}+U_{i,j+1}+U_{i-1,j}+U_{i-1,j+1}), \qquad \bar{V}_{ij} = \tfrac{1}{4}(V_{ij}+V_{i,j-1}+V_{i+1,j}+V_{i+1,j-1}).$$

However, for the OEH scheme we choose

$$\text{(3.4b)}\quad \bar{U}_{ij} = \tfrac{1}{2}(U_{i-1,j}+U_{i,j+1}), \qquad \bar{V}_{ij} = \tfrac{1}{2}(V_{i,j-1}+V_{i+1,j}).$$

The reason for this choice will become apparent in § 3.2.1.

For the treatment of the boundary conditions, we apply a simple reflection technique [13]. Consider, e.g., (3.2b) and (3.3b) at the O-points $(1, j)$, which involves the outside value $V_{0,j}$. The reflection technique consists of writing the boundary value $V_{1/2,j}$ as a mean value of its neighbouring values $V_{0,j}$ and $V_{1,j}$, so that $V_{0,j} = 2V_{1/2,j} - V_{1,j}$ (see Fig. 1).

3.2. Time integration. Let $\mathbf{U} = (U, V)^T$ and $\mathbf{F}(\mathbf{U}) = (F_1(U, V), F_2(U, V))^T$; then (3.2) can be written in the vector form

$$\text{(3.5)}\quad \frac{d}{dt}\mathbf{U} = \mathbf{F}(\mathbf{U}).$$

For reasons of computational feasibility, we apply a two-term splitting formula for the numerical integration of (3.5). Let

$$\mathbf{F}(\mathbf{U}) = \mathbf{F}_1(\mathbf{U}) + \mathbf{F}_2(\mathbf{U}),$$

and consider the two-stage formula

$$\text{(3.6)}\quad \begin{aligned} \mathbf{U}^{n+1/2} &= \mathbf{U}^n + \tfrac{1}{2}\tau[\mathbf{F}_1(\mathbf{U}^{n+1/2}) + \mathbf{F}_2(\mathbf{U}^n)], \\ \mathbf{U}^{n+1} &= \mathbf{U}^{n+1/2} + \tfrac{1}{2}\tau[\mathbf{F}_1(\mathbf{U}^{n+1/2}) + \mathbf{F}_2(\mathbf{U}^{n+1})], \end{aligned}$$

with τ denoting the timestep. It can be easily verified that this integration formula is second-order consistent for any ODE system (3.5) [7]. Both the OEH scheme and the ADI scheme are special cases of (3.6).

3.2.1. The OEH scheme. In what follows, U_{ij}^n denotes the discrete approximation to u at the grid point (ih, jk) at time level $t_n = n\tau$ (likewise for V_{ij}^n). The OEH scheme for (3.5) is given by the numerical integration formula [4]

$$U_{ij}^{n+1} - \tau\theta_{ij}^{n+1} F_{ij}(\mathbf{U}^{n+1}) = U_{ij}^n + \tau\theta_{ij}^n F_{ij}(\mathbf{U}^n), \tag{3.7}$$

where $U_{ij}^n = (U_{ij}^n, V_{ij}^n)^T$ (likewise for $F_{ij}(\mathbf{U}^n)$). The function θ_{ij}^n is defined by

$$\theta_{ij}^n = \begin{cases} 1 & \text{if } n+i+j \text{ is odd} \quad \text{(odd points)}, \\ 0 & \text{if } n+i+j \text{ is even} \quad \text{(even points)}. \end{cases} \tag{3.8}$$

Writing down two successive steps of (3.7) yields

$$U_{ij}^{n+1} = U_{ij}^n + \tau\theta_{ij}^n F_{ij}(\mathbf{U}^n) + \tau\theta_{ij}^{n+1} F_{ij}(\mathbf{U}^{n+1}), \tag{3.9a}$$

$$U_{ij}^{n+2} = U_{ij}^{n+1} + \tau\theta_{ij}^{n+1} F_{ij}(\mathbf{U}^{n+1}) + \tau\theta_{ij}^{n+2} F_{ij}(\mathbf{U}^{n+2}). \tag{3.9b}$$

Let $\mathbf{F}_O(\mathbf{U})$ and $\mathbf{F}_E(\mathbf{U})$ denote the restriction of $\mathbf{F}(\mathbf{U})$ to the odd and even points respectively, then replacing τ by $\tau/2$, (3.9) can be written in the form

$$\mathbf{U}^{n+1/2} = \mathbf{U}^n + \tfrac{1}{2}\tau[\mathbf{F}_E(\mathbf{U}^{n+1/2}) + \mathbf{F}_O(\mathbf{U}^n)], \tag{3.10a}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^{n+1/2} + \tfrac{1}{2}\tau[\mathbf{F}_E(\mathbf{U}^{n+1/2}) + \mathbf{F}_O(\mathbf{U}^{n+1})]. \tag{3.10b}$$

The order of computation for the OEH scheme is

$$\mathbf{U}_O^{n+1/2} = \mathbf{U}_O^n + \tfrac{1}{2}\tau\mathbf{F}_O(\mathbf{U}^n) \qquad (= 2\mathbf{U}_O^n - \mathbf{U}_O^{n-1/2} \text{ if } n \geqq 1), \tag{3.11a}$$

$$\mathbf{U}_E^{n+1/2} = \mathbf{U}_E^n + \tfrac{1}{2}\tau\mathbf{F}_E(\mathbf{U}^{n+1/2}), \tag{3.11b}$$

$$\mathbf{U}_E^{n+1} = \mathbf{U}_E^{n+1/2} + \tfrac{1}{2}\tau\mathbf{F}_E(\mathbf{U}^{n+1/2}) \qquad (= 2\mathbf{U}_E^{n+1/2} - \mathbf{U}_E^n), \tag{3.11c}$$

$$\mathbf{U}_O^{n+1} = \mathbf{U}_O^{n+1/2} + \tfrac{1}{2}\tau\mathbf{F}_O(\mathbf{U}^{n+1}). \tag{3.11d}$$

Note that (3.11a) is just the forward Euler rule at the odd points, whereas (3.11b) is the backward Euler rule at the even points. For (3.11c) and (3.11d) it is just vice versa. Substituting (3.4b) into (3.3), it can be easily verified that in (3.11) there exists an odd-even coupling between the variables, i.e., a variable at an odd point is only coupled to variables at even points and vice versa. Because of this odd-even coupling and the alternating use of the forward- and backward Euler rule, scheme (3.11) is only diagonally implicit. Note that the computation of the forward Euler rule in (3.11a) and (3.11c) can be economized by using a simple interpolation formula. The scheme thus obtained is called the fast form of the OEH scheme [4].

3.2.2. The ADI scheme. For the ADI scheme we use the splitting formula

$$\mathbf{F}(\mathbf{U}) = \mathbf{F}_x(\mathbf{U}) + \mathbf{F}_y(\mathbf{U}),$$

where $\mathbf{F}_x$ and $\mathbf{F}_y$ represent the space discretization terms containing the x- and y-derivatives, respectively. For the Burgers equations such a splitting is possible, because there are no mixed derivatives. So, the ADI scheme for (3.5) is given by [11]

$$\mathbf{U}^{n+1/2} = \mathbf{U}^n + \tfrac{1}{2}\tau[\mathbf{F}_x(\mathbf{U}^{n+1/2}) + \mathbf{F}_y(\mathbf{U}^n)], \tag{3.12a}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^{n+1/2} + \tfrac{1}{2}\tau[\mathbf{F}_x(\mathbf{U}^{n+1/2}) + \mathbf{F}_y(\mathbf{U}^{n+1})]. \tag{3.12b}$$

Note that (3.12a) is explicit in the y-direction and implicit in the x-direction, and vice versa in (3.12b). Since there is a 3-point coupling in each direction, the ADI scheme

can be implemented such that only nonlinear tridiagonal systems must be solved at each step.

In order to obtain linear systems, the terms $\mathbf{F}_x(\mathbf{U}^{n+1/2})$ in (3.12a) and $\mathbf{F}_y(\mathbf{U}^{n+1})$ in (3.12b), which can be written in the form (cf. (3.3))

$$\text{(3.13a)} \qquad \mathbf{F}_x(\mathbf{U}^{n+1/2}) = A(U^{n+1/2})\mathbf{U}^{n+1/2} \quad \text{and} \quad \mathbf{F}_y(\mathbf{U}^{n+1}) = B(V^{n+1})\mathbf{U}^{n+1},$$

are linearized as follows:

$$\text{(3.13b)} \qquad \mathbf{F}'_x(\mathbf{U}^{n+1/2}) = A(U^*)\mathbf{U}^{n+1/2} \quad \text{and} \quad \mathbf{F}'_y(\mathbf{U}^{n+1}) = B(V^*)\mathbf{U}^{n+1},$$

with A and B tridiagonal matrices and U^* and V^* approximations to $U^{n+1/2}$ and V^{n+1}, respectively. To maintain second-order accuracy, the approximations U^* and V^* are given by (see [12])

$$U^* = \tfrac{3}{2}U^n - \tfrac{1}{2}U^{n-1}, \qquad V^* = 2V^{n+1/2} - V^n.$$

Now, the ADI scheme only requires the solution of linear tridiagonal systems. In § 4.2 we will discuss some algorithms for the solution of these systems. Due to the linearization process, it is not possible to formulate a fast form for the ADI scheme, as for the OEH scheme (cf. (3.11a) and (3.11c)).

3.3. Stability. Finally, we make some remarks about the stability of both the OEH scheme and the ADI scheme. Consider to this purpose the linear convection–diffusion equation

$$\text{(3.14)} \qquad u_t = -q_1 u_x - q_2 u_y + (u_{xx} + u_{yy})/\mathrm{Re}.$$

Here, the vector $(q_1, q_2)^T$ represents a constant velocity. Now suppose that for the space discretization we use standard central differences, with constant grid sizes h and k in x- and y-direction, respectively. Then von Neumann stability analysis applied to the OEH scheme (3.10) yields the following necessary and sufficient timestep restriction [14], [15] for (3.14):

$$\tau^2\left(\frac{1}{h^2}+\frac{1}{k^2}\right)(q_1^2+q_2^2) \leqq 4.$$

This inequality shows that the OEH scheme is conditionally stable ($\tau = O(h)$), independent of Re. The ADI scheme for the linear equation (3.14) is unconditionally stable in the sense of von Neumann stability [10].

Remark. A drawback of the OEH scheme is the so-called Du Fort–Frankel (DFF) deficiency [14], [15]. By this we mean that for τ, h, $k \to 0$, the solution of the OEH scheme converges to the solution of the problem

$$\text{(3.15)} \qquad \begin{aligned} u_t &= -uu_x - vu_y + (u_{xx} + u_{yy})/\mathrm{Re} - \frac{1}{\mathrm{Re}}\tau^2\left(\frac{1}{h^2}+\frac{1}{k^2}\right)u_{tt}, \\ v_t &= -uv_x - vv_y + (v_{xx} + v_{yy})/\mathrm{Re} - \frac{1}{\mathrm{Re}}\tau^2\left(\frac{1}{h^2}+\frac{1}{k^2}\right)v_{tt}. \end{aligned}$$

In general, for convergence it is thus necessary that $\tau = o(\max(h, k))$.

4. Implementations. In this section we describe implementation techniques for vectorizing both the OEH scheme and the ADI scheme for use on vector computers. It is our goal to implement the schemes in such a way that they perform efficiently on

vector computers. We utilize the vector processing concepts discussed in § 2. The programs have been written in the ANSI FORTRAN77. Thus, the resulting software is portable.

4.1. The OEH scheme. The OEH scheme is based upon the alternating use of the forward and backward Euler rule. Because of the 5-point coupling that exists between the variables, the OEH scheme is diagonally implicit (see § 3.2.1). Specifically, the scheme only requires scalar divisions and no nonlinear equations must be solved.

The obvious choice for the ordering of the grid points is the red-black or chess-board ordering, where all the four neighbours of each point belong to another colour. The grid points may be subdivided accordingly into two vectors that contain the red and black points, respectively. The grid points are numbered along horizontal grid lines. The OEH scheme is performed in four stages (see (3.11a)-(3.11d)). For example, in the first stage the values in the red points are updated using the value in the red point itself and old values in neighbouring black points (i.e., the forward Euler rule), then in the second stage the values in the black points are updated using the old value in the black point and new values in red points (i.e., the backward Euler rule). Throughout the code the elements of the two vectors are stored in consecutive memory elements (i.e., stride 1), which is, in general, an advantage on vector computers. Moreover, no data reorderings must be performed.

Note that the two vectors are not confined to one horizontal grid line, but they extend over the whole grid. This was done in order to achieve improved performance through utilization of longer vectors. As a penalty for using those longer vectors, the values in the boundary points are overwritten, thus destroying the correct boundary values. To restore the correct boundary values, these values are stored separately. Moreover, the first and the last grid points of each horizontal line must be of the same colour to maintain the red-black ordering. Thus, the number of grid points in horizontal direction ($= N$) has to be odd.

The OEH scheme requires minimal storage. In our implementation we used only one extra array of length $NM/2$, which is one fourth of the total number of unknowns. Hence, the total storage amounts approximately to $2.5NM$ memory locations.

4.2. The ADI scheme. The ADI scheme for two-dimensional problems requires the solution of tridiagonal systems along horizontal and vertical grid lines, respectively. Tridiagonal systems form an important class of linear algebraic equations. Consequently, efficient algorithms have been developed for the solution of such systems. The tridiagonal systems can be viewed in various ways. For example, for (3.12a) we have M tridiagonal (linear) systems of order N. The first method we use to solve this system is the Gaussian elimination method. Since the M systems are uncoupled, we can vectorize across the systems, thus resulting into vector operations of length M. On the other hand, the M individual systems can be combined in a single tridiagonal system of order NM to obtain longer vectors. As a consequence, extra memory is needed. Due to the large memory capacities of today's vector computers, it is possible to execute programs with large memory requirements. For example, on the Cyber 205 and the Cray X-MP/24 the maximal memory size is about four million 64-bit words.

Several methods have been proposed to achieve efficient methods for such large systems on vector computers. In this report we use a variant of the partition method of Wang [3], [9], which will be discussed briefly now. First, the tridiagonal matrix is partitioned into a $l \times l$ block tridiagonal matrix with each block a $m \times m$ matrix. The method starts by reducing the tridiagonal system to a tridiagonal system of order l using vector operations. Then the reduced system is solved by Gaussian elimination.

Finally, the other unknowns are solved by back substitution using again vector operations. Although the variant of the partition method has a higher operation count than the Gaussian elimination method, this is an efficient method on vector computers since the vector length of the operations is much higher.

For this variant of the partition method, it is plausible that the off-diagonal elements of the reduced system are very small relative to the main diagonal elements [3], [5], which is confirmed by numerical experiments. Following Van der Vorst [16] and De Goede and Wubs [3], the solution of the reduced tridiagonal system is approximated by a truncated Neumann series. The resulting explicit-implicit method is advantageous for use on vector computers. The price to be paid for the approximation of the reduced system is a possible drop in accuracy. However, due to the relatively small off-diagonal elements, this approach hardly affects the accuracy.

As said in § 2, for the performance on vector computers the data structure is very important. For the ADI scheme, tridiagonal systems must be solved along horizontal grid lines and vertical grid lines, respectively. If the arrays are ordered horizontally, then the x-differences can be calculated efficiently. Likewise, if the arrays are ordered vertically, then the y-differences can be calculated efficiently. These two orderings imply that during the performance of the ADI scheme, reorderings must be performed to change from horizontal to vertical lines and vice versa. The reordering operations have been implemented as efficiently as possible.

Moreover, during the solution of the tridiagonal systems, the variant of the partition method requires vector operations with stride m. The Cray-computer is hardly hampered by a stride unequal to one. However, the CDC Cyber 205 requires contiguous vectors (i.e., stride 1). Therefore, compress/expand instructions are necessary to restructure the vectors. The alternative is to reorder in advance the data structure to obtain contiguous vectors. On the CDC Cyber 205 this alternative may be useful. Both versions have been implemented.

For each of the implementations, the storage requirements are significantly larger than for the OEH scheme, viz. about $9NM$ memory locations.

Summarizing, for the Peaceman-Rachford ADI scheme the following implementations for the solution of the tridiagonal systems have been used:

ADIGE (Gaussian elimination),
ADIW (a variant of the partition method of Wang (stride m)),
ADEI (the explicit-implicit scheme (stride m)),
ADIW1 (ADIW with an extra reordering of the data structure (stride 1)),
ADEI1 (ADEI with an extra reordering of the data structure (stride 1)).

5. Performance. In this section we report on the accuracy and performance of the OEH scheme and the ADI scheme on vector computers. For this purpose, we have applied the schemes to a moving wave front problem. In general, moving wave front problems are difficult to compute since the solution contains sharp gradients, both in space and time. This necessitates the use of small timesteps and, when employing a uniform grid, a small grid size. Therefore, such problems are time and memory consuming and the application of powerful computers (such as, e.g., vector computers) is obvious.

In our experiments the following vector computers and FORTRAN compilers have been used:

(i) (2-pipe) CDC Cyber 205 (SARA, Amsterdam, the Netherlands), max. 200 MFLOP/s, FORTRAN 200 compiler, (the VAST (version 1.22W) precompiler of Pacific Sierra Research Corporation is used).

(ii) Cray X-MP/24 (Cray Research, Bracknell, UK), max. 235 MFLOP/s. FORTRAN CFT77 (version 1.3) compiler.

An exact solution of the Burgers equations can be generated by using the Cole-Hopf transformation [2],

$$u = \frac{2}{\mathrm{Re}} \frac{\phi_x}{\phi} \quad \text{and} \quad v = -\frac{2}{\mathrm{Re}} \frac{\phi_y}{\phi}, \tag{5.1a}$$

where ϕ is the solution of

$$\phi_t = \frac{1}{\mathrm{Re}} (\phi_{xx} + \phi_{yy}). \tag{5.1b}$$

In our test problem we choose $\phi = f_1 + f_2$ [18], with

$$\begin{aligned} f_1(x, y, t) &= \exp((-12(x+y)+9t) * \mathrm{Re}/32), \\ f_2(x, y, t) &= \exp((-4(x+2y)+5t) * \mathrm{Re}/16), \end{aligned} \tag{5.2}$$

which yields the exact solution

$$u = \frac{1}{4} * \frac{3f_1 + 2f_2}{f_1 + f_2} = \frac{3}{4} - \frac{1}{4} * \frac{1}{1+\exp((-4x+4y-t) * \mathrm{Re}/32)}, \tag{5.3a}$$

$$v = \frac{1}{4} * \frac{3f_1 + 4f_2}{f_1 + f_2} = \frac{3}{4} + \frac{1}{4} * \frac{1}{1+\exp((-4x+4y-t) * \mathrm{Re}/32)}. \tag{5.3b}$$

The solution represents a wave front at $y = x + 0.25t$. The speed of propagation is $0.125\sqrt{2}$ and is perpendicular to the wave front. For increasing values of Re, the wave front becomes sharper. In Fig. 2 the exact solution for u is shown at $t = 2.5$ for Re = 100; 1,000; 10,000.

With the purpose of testing the (order of) accuracy of the schemes, we first compare the exact solution of the Burgers equations with the numerical solution obtained for grid sizes $h = k = 1/17, 1/33, 1/65, 1/129$ and for timesteps $\tau = 1/10, 1/20, 1/40, 1/80, 1/160, 1/320$ (provided that the time integration is stable). The computational domain is $\Omega = [0, 1] \times [0, 1]$ and the time integration interval is $[0, 2.5]$. We prescribe time-dependent Dirichlet boundary conditions that are taken from the exact solution and we choose Re = 100. For the time integration we use the OEH scheme, the ADIW scheme, and the ADEI scheme (see § 4).

To measure the accuracy of the numerical solution we define

$$\mathrm{cd}_\infty = -{}^{10}\log(\|\text{global error at } t = 2.5\|_\infty), \tag{5.4}$$

denoting the number of correct digits in the numerical approximation at the endpoint $t = 2.5$.

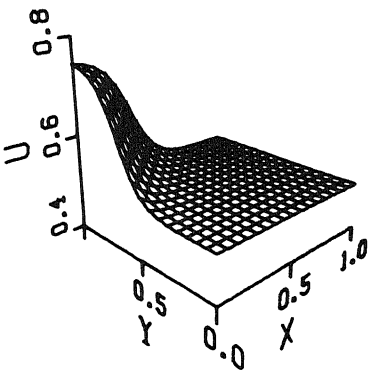
Since $\max|u(x, y, t)| = 0.75$ and $\max|v(x, y, t)| = 1.0$, von Neumann stability analysis applied to the OEH scheme suggests the timestep restriction

$$\tau \leqq \tfrac{4}{5}\sqrt{2}h. \tag{5.5}$$

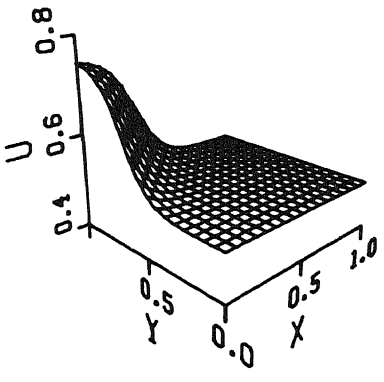
In Table 5.1 we list the cd_∞-values for all three schemes. We only list the cd_∞-values for the u-field; for the v-field we obtain nearly the same results.

First consider the OEH scheme. From Table 5.1 we can conclude the following:

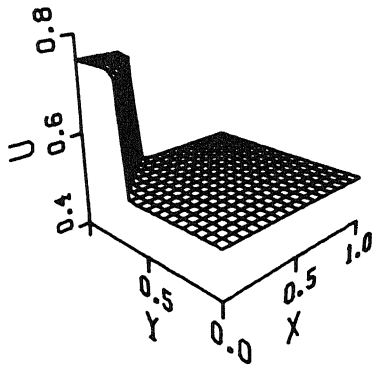
(i) For small timesteps (e.g. $\tau = 1/320$) the time integration error is neglectable, and we can observe the second-order behaviour in space (${}^{10}\log(4) \approx 0.6$). On a fine grid (e.g., $h = 1/129$) we can observe the second-order behaviour in time since the space discretization error is neglectable.



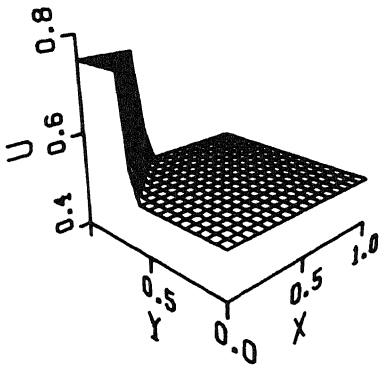
Re = 100



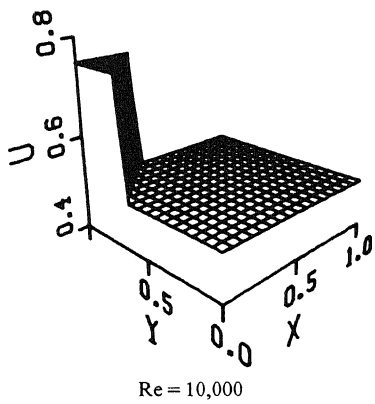
Re = 100



Re = 1,000

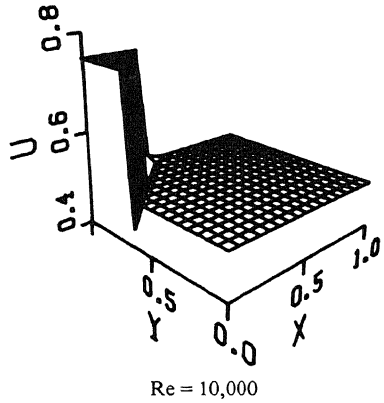


Re = 1,000



Re = 10,000

FIG. 2. *Exact solutions* (5.3a) *for* Re = 100; 1,000; 10,000.



Re = 10,000

FIG. 3. *Corresponding numerical solutions.*

(ii) For τ fixed and $h \to 0$ the accuracy decreases if τ/h is sufficiently large. This is caused by the DFF deficiency (cf. (3.15)).

(iii) When looking along diagonals (τ/h constant) we observe a second-order behaviour if τ/h is small enough. For larger values of τ/h the scheme fails to converge due to the DFF deficiency.

TABLE 5.1
cd_x-values for the OEH, ADIW, *and* ADEI *schemes.*

Scheme	h^{-1}	Correct digits for u-field (∞-norm)					
		$\tau=1/10$	$\tau=1/20$	$\tau=1/40$	$\tau=1/80$	$\tau=1/160$	$\tau=1/320$
OEH	17		2.54	2.55	2.55	2.55	2.55
	33			3.05	3.21	3.20	3.20
	65			2.82	3.37	3.73	3.85
	129				2.84	3.44	3.98
ADIW	17	1.92	2.29	2.48	2.51	2.52	2.52
	33	2.10	2.56	2.89	3.07	3.15	3.18
	65	2.24	2.75	3.19	3.51	3.68	3.77
	129	2.25	2.77	3.26	3.67	3.99	4.19
ADEI	17	1.92	2.29	2.48	2.51	2.52	2.52
	33	2.12	2.57	2.89	3.07	3.15	3.18
	65	2.24	2.75	3.19	3.51	3.68	3.77
	129	2.25	2.78	3.26	3.67	3.98	4.17

Now, consider both ADI-type schemes. In the same way as for the OEH scheme, we can observe second-order behaviour in space and time. In general, the accuracy of the OEH scheme is comparable with that of the ADI-type schemes. However, especially on the finest grid the ADI-type schemes are more accurate than the OEH scheme, because the latter suffers from the DFF deficiency. Note that the accuracy results for the ADIW scheme and the ADEI scheme are comparable. So, the accuracy is hardly reduced if the tridiagonal systems are solved by the approximating method.

Table 5.2 presents the execution times obtained for a single example, namely, for a 129×129 grid with $t=2.5$, $\tau=1/80$, and $\mathrm{Re}=100$. We compare the OEH scheme with the five implementations of ADI-type schemes (see § 4). As an illustration, the implementations have also been performed without vectorization on the CDC Cyber 205 (scalar code). In parentheses we list the ratio in performance of the vectorized code to the scalar code. We emphasize that Table 5.2 contains the execution times for the computation of 200 timesteps without paying attention to the accuracy or stability.

From this experiment we can draw the following conclusions:

(i) On both vector computers the OEH scheme is considerably faster than the implementations of the ADI-type schemes. This is due to the fact that no systems of equations must be solved and no data reorderings must be performed. For the scalar code, it is fair to say that the ratio of the execution time for the ADI-type schemes

TABLE 5.2
Execution times in seconds for a 129×129 *grid with* $t=2.5$, $\tau=1/80$ *and* $\mathrm{Re}=100$.

Scheme	Execution times (in seconds)		
	Cyber 205 (vectorized code)	Cray X-MP/24	Cyber 205 (scalar code)
OEH	1.8	1.0	15.4 (8.6)
ADIGE	15.0	6.0	118.2 (7.9)
ADIW	27.3	8.7	181.6 (6.6)
ADEI	22.4	6.1	176.6 (7.8)
ADIW1	18.6	8.9	187.1 (10.0)
ADEI1	12.6	6.7	182.2 (14.4)

to the OEH scheme is misleading. For example, the data reorderings (from x-ordering to y-ordering and vice versa) are uneconomical for use on scalar computers. So, the scalar code for the ADI-type schemes is far from optimal.

(ii) On the CDC Cyber 205 the vectorized code is much faster than the scalar code. For all schemes a considerable speed-up factor is obtained.

(iii) On the CDC Cyber 205 it is beneficial to reorder the data structure to obtain contiguous vectors (compare ADIW with ADIW1 and ADEI with ADEI1). The speed-up in performance justifies the overhead due to the data reordering. On the Cray X-MP/24 this does not hold since the Cray is hardly hampered by a stride unequal to one.

(iv) In general, the Cray X-MP/24 is considerably faster than the (2-pipe) CDC Cyber 205. This is due to a smaller clock cycle and a better compiler.

(v) On the CDC Cyber 205 the fastest method is ADEI1 (i.e., the explicit-implicit method with an extra reordering of the data structure). On the CRAY X-MP/24 however, the Gaussian elimination method and the explicit-implicit method ADEI are the fastest methods.

Finally, we examine the accuracy behaviour of the OEH scheme and the ADIW scheme for increasing values of Re. In this experiment we compute the numerical solution at $T=2.5$ and use the grid size values $h=k=1/33$, $1/65$, $1/129$, $1/257$. Especially for large values of Re we may expect oscillations in the solution. Therefore, the cd_∞-value, as defined in (5.4), is a too strict measure for the accuracy. Instead, we define

$$\mathrm{cd}_1 = -^{10}\log(\|\text{global error at } t=2.5\|_1).$$

We start our computations for $\mathrm{Re}=100$ on a 33×33 grid. On each grid and for each Re-number we choose the timestep as large as possible such that $\mathrm{cd}_1 \geqq 3$. As soon as $\mathrm{cd}_1 < 3$ for each timestep we switch to the next finer grid and choose an appropriate timestep. In Table 5.3 we list the cd_1-values for the u-field for increasing values of Re; for the v-field we find nearly the same results. For the ADIW scheme the timestep is listed in parentheses. In this experiment we used the ADIW scheme; however, nearly the same results would have been obtained for the ADEI scheme.

TABLE 5.3
cd_1-values for the OEH *and* ADIW *scheme for increasing values of* Re.

	Correct digits for u-field (1-norm)							
	OEH scheme				ADIW scheme			
Re	$h=1/33$ $\tau=1/40$	1/65 1/80	1/129 1/160	1/257 1/320	$h=1/33$	$h=1/65$	$h=1/129$	$h=1/257$
100	4.05				3.30 (1/10)			
500	3.08				2.95 (1/80)	3.37 (1/40)		
1,000	2.51	3.42				3.19 (1/80)		
1,500		3.06				2.91 (1/160)	3.36 (1/80)	
2,000		2.86	3.74				3.15 (1/80)	
3,000			3.39				3.09 (1/160)	
4,000			3.18				2.81 (1/160)	3.01 (1/80)
5,000			3.03					3.23 (1/160)
6,000			2.88	3.70				3.32 (1/320)
7,000				3.59				
10,000				3.35				

From Table 5.3 we can conclude the following:

(i) In order to obtain the prescribed accuracy, the ADI scheme requires in general a finer grid than the OEH scheme. This is possibly due to the linearization process of the ADI scheme (see (3.13)). Both schemes require a comparable timestep. So, for large Re-numbers the OEH scheme seems to be more suitable than the ADI-type schemes for the numerical solution of the Burgers equations, at least for the present type of solution.

(ii) The DFF-deficiency of the OEH scheme is virtually absent for large Re-numbers since the terms u_{tt}/Re and v_{tt}/Re are very small, except in a small region near the wave front (see (3.15)).

In Fig. 3 we present the numerical solution for the u-field for $\mathrm{Re} = 100$; 1,000; 10,000 computed with the OEH scheme.

6. Concluding remarks. In this paper we examined the efficiency and performance of the odd-even hopscotch (OEH) scheme and the alternating direction implicit (ADI) scheme on vector computers, viz. the CDC Cyber 205 and the Cray X-MP/24. For the ADI scheme the following methods for the solution of the tridiagonal linear systems have been used: the Gaussian elimination method (ADIGE), a variant of the partition method of Wang (ADIW) and the explicit-implicit method (ADEI). The vectorized codes were considerably faster than the corresponding scalar codes.

First, let us consider the advantages of the OEH scheme over the ADI-type schemes:

(i) On both vector computers the OEH scheme is considerably faster than the ADI-type schemes, due to the near-explicitness of the OEH scheme.

(ii) The OEH scheme has minimal storage requirements. In our implementations we used about four times more memory space for the ADI-type schemes than for the OEH scheme. This is due to the way in which the tridiagonal systems are solved (see § 4).

(iii) It is very easy to implement the OEH scheme for both linear and nonlinear problems. For the ADI-type schemes the nonlinear tridiagonal systems of equations must be linearized in some way (cf. (3.13)). Moreover, the OEH scheme can be extended to multidimensional problems in a straightforward manner, contrary to the ADI-type schemes.

The ADI-type schemes have the following advantages over the OEH scheme:

(i) The ADI-type schemes have a better stability behaviour than the OEH scheme.

(ii) The OEH scheme suffers from the Du Fort-Frankel (DFF) deficiency that in general, has a negative influence on the accuracy.

Comparing the ADI-type schemes, the ADEI scheme and the ADIGE scheme have a comparable performance on vector computers. However, for test problems with an irregular domain, the ADEI scheme is to be preferred since in that case vectorization across the systems requires extra operations. In the near future, we will extend the codes for application to the incompressible Navier-Stokes equations.

Acknowledgments. We wish to express our gratitude to the ZWO Werkgroep Gebruik Supercomputers (WGS) for providing the necessary computer time on the CDC Cyber 205 and the Cray X-MP/24.

REFERENCES

[1] CDC Cyber 200 FORTRAN reference manual, version 1, publ. number 60480200H, Sunnyvale, CA.

[2] C. A. J. FLETCHER, *A comparison of finite element and finite difference solutions of the one- and two-dimensional Burgers equation*, J. Comput. Phys., 51 (1983), pp. 159-188.

[3] E. D. DE GOEDE AND F. W. WUBS, *Explicit-implicit methods for time-dependent partial differential equations*, Report NM-R8703, Centre for Mathematics and Computer Science, Amsterdam, the Netherlands, 1987.

[4] A. R. GOURLAY, *Hopscotch: A fast second-order partial differential solver*, J. Inst. Maths. Appl., 6 (1970), pp. 375-390.

[5] D. HELLER, *Some aspects of the cyclic reduction algorithm for block tridiagonal linear systems*, SIAM J. Numer. Anal., 13 (1976), pp. 484-496.

[6] R. W. HOCKNEY AND C. R. JESSHOPE, *Parallel Computers: Architecture, Programming and Algorithms*, Adam Hilger, Bristol, 1981.

[7] P. J. VAN DER HOUWEN AND J. G. VERWER, *One-step splitting methods for semi-discrete parabolic equations*, Computing, 22 (1979), pp. 291-309.

[8] J. VAN KAN, *A second-order pressure correction method for viscous incompressible flow*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 870-891.

[9] P. H. MICHIELSE AND H. A. VAN DER VORST, *Data transport in Wang's partition method*, Parallel Computing, 7 (1988), pp. 87-95.

[10] A. R. MITCHELL AND D. F. GRIFFITHS, *The Finite Difference Method in Partial Differential Equations*, John Wiley, Chichester, 1980.

[11] D. W. PEACEMAN AND H. H. RACHFORD, JR., *The numerical solution of parabolic and elliptic differential equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28-41.

[12] B. P. SOMMEIJER, *An* ALGOL 68 *implementation of two splitting methods for semi-discretized parabolic differential equations*, Report NM-NN 15/77, Centre for Mathematics and Computer Science, Amsterdam, the Netherlands, 1977.

[13] R. PEYRET AND T. D. TAYLOR, *Computational Methods for Fluid Flow*, Springer-Verlag, Berlin, New York, 1983.

[14] J. H. M. TEN THIJE BOONKKAMP, *The odd-even hopscotch pressure correction scheme for the incompressible Navier-Stokes equations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 252-270.

[15] J. H. M. TEN THIJE BOONKKAMP AND J. G. VERWER, *On the odd-even hopscotch scheme for the numerical solution of time-dependent partial differential equations*, Appl. Numer. Math., 3 (1987), pp. 183-193.

[16] H. A. VAN DER VORST, *A vectorizable variant of some* ICCG *methods*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 86-92.

[17] H. H. WANG, *A parallel method for tridiagonal system equations*, ACM Trans. Math. Software, (1981), pp. 170-183.

[18] G. B. WHITHAM, *Linear and Nonlinear Waves*, John Wiley, New York, 1974.