**Centrum voor Wiskunde en Informatica**
Centre for Mathematics and Computer Science

E.D. de Goede, J.H.M. ten Thije Boonkkamp

# Vectorization of the odd-even hopscotch scheme and the alternating direction implicit scheme for the two-dimensional Burgers' equations

Department of Numerical Mathematics Report NM-R8720 November

BIBLIOTHEEK
Centrum voor Wiskunde en Informatica
Amsterdam

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Copyright © Stichting Mathematisch Centrum, Amsterdam

# Vectorization of the Odd-Even Hopscotch Scheme and the Alternating Direction Implicit Scheme for the Two-Dimensional Burgers' Equations

E.D. de Goede & J.H.M. ten Thije Boonkkamp
*Centre for Mathematics and Computer Science*
*P.O. Box 4079, 1009 AB Amsterdam, The Netherlands*

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.

*1980 Mathematics subject classification* : Primary 65V05, Secondary 65M05, 76DXX
*Key words and Phrases* : vector computers, Burgers' equations, odd-even hopscotch scheme, alternating direction implicit scheme, vectorization.
*Note* : This report will be submitted for publication elsewhere.

## 1. Introduction

This report is written as a contribution to a project to develop 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 FORTRAN 77 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 time step. 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 time step 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 [15], which is described in [3,9], and a method developed by Wubs and De Goede [3]. By the approach of Wubs and De Goede, the tridiagonal systems are solved by a combination of explicit and implicit calculations, thus resulting in an alternating direction explicit-implicit (ADEI) scheme. It appears that the variant of the partition method and the method developed by Wubs and De Goede are suitable methods for use on vector computers.

The purpose of this paper is to report our experience in vectorizing both schemes for the two-

Report NM-R8720
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

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 (1-pipe) CDC Cyber 205 and a Cray X-MP/24. We used one (portable) code on both machines.

Section 2 contains a brief summary of the conceptual features of vector computers, which are relevant to the present application. In Section 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 Section 5, we compare the accuracy and performance of both schemes. Finally, Section 6 contains some concluding remarks.

## 2. Vector processing

Many scientific and technical problems use a large number of data, and can only be solved by using today's supercomputers. In general, identical calculations have to be performed on these data. This has led to the development of vector computers. Vector computers belong to the class of Single Instruction stream - Multiple Data stream (SIMD) machines [6]. A vector computer can perform the same operation on a set of data (i.e. a vector) by means of one vector instruction. On a conventional (scalar) computer each operation on each pair of operands requires one instruction. These computers belong to the class of Single Instruction stream - Single Data stream (SISD) machines.

The vector instructions fall into two main categories : those that perform floating-point arithmetic, and those that may be called data-motion instructions (for example, instructions to compress or expand an array using an index-list). If data cannot be structured into vectors, vector computers do not out-perform fast conventional computers. The need for vector data-motion instructions also becomes apparent when one considers 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 without any intermediate registers. A Cray-computer allows vectors whereby the number of memory locations between consecutive elements (the so-called stride) is constant.

The time needed for completing a vector instruction contains two components. The first is the start-up time, that is the time elapsed before the first result is computed. Due to the start-up time it is beneficial to work with longer vectors [6]. The vector processor is better utilized if a single operation is performed on a longer vector, instead of using several operations to compute the same number of results. The second component is proportional to the vector length and is called the streamtime. Both for a CDC Cyber 205 and a Cray-computer, the result rate for an add or multiply instruction is 1 result per cycle (clock period).

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 vectorization. The nature of vector operations is such that only DO-loops might be translated into vector instructions. Specific characteristics of a given DO-loop determine its vectorizability [1]. It is not always possible to vectorize a code, like in the following example :

```
      DO 10  I = 1,N
             A(I + 1) = A(I) + S                                        (2.1)
   10 CONTINUE
```

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 code. We have used such instructions e.g. in the following situation : When a recursion does not refer to previously computed values within the same DO-loop, then the compiler can be instructed to ignore the seemingly dependency, and to vectorize the loop, 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 .

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 sixteen 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. For example, if the stride is an 8-fold, then there is a substantial delay in the vector operations. So, in order to obtain a good performance on vector computers it is important to consider the data structure very carefully (see Section 4).

## 3. The OEH scheme and the ADI scheme for the two-dimensional Burgers' equations

Consider the two-dimensional Burgers' equations

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

with $u$ and $v$ the velocities in $x$- and $y$-directions and Re denoting the Reynolds number. On the boundary $\Gamma$ of the connected space domain $\Omega$, we prescribe the Dirichlet conditions

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

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 Section 3.1 and the time integration in Section 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 $y$-directions 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 [13].

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 the system of ordinary differential equations (ODEs)

$$\begin{aligned} \frac{d}{dt} U_{ij} &= F_{1,ij}(\mathrm{U},\mathrm{V})\,, \qquad i=1,\ldots,N-1 \ , \ j=1,\ldots,M \qquad \text{(interior } \times \text{ -points)} \\ \frac{d}{dt} V_{ij} &= F_{2,ij}(\mathrm{U},\mathrm{V})\,, \qquad i=1,\ldots,N \ , \ j=1,\ldots,M-1 \qquad \text{(interior } \bigcirc \text{ -points)}\,, \end{aligned} \tag{3.2}$$

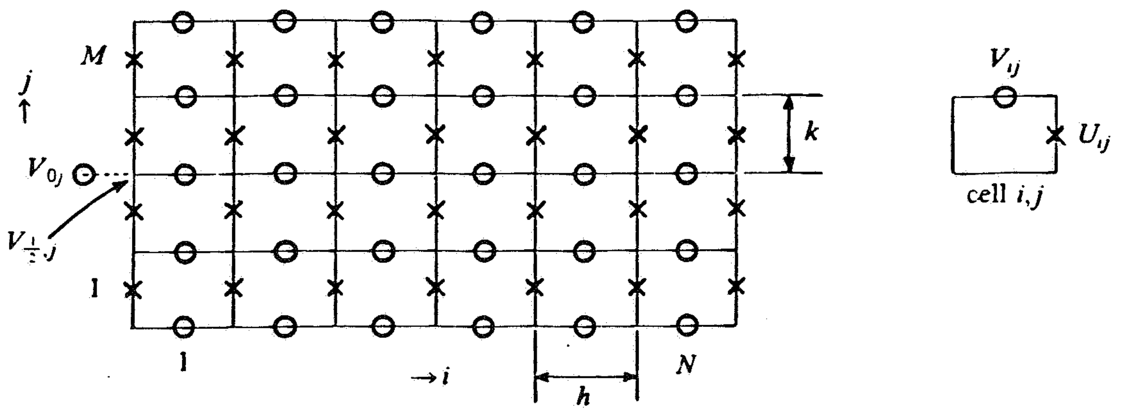

Fig 1. The staggered grid

where

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

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

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

$$\overline{U}_{ij}=\frac{1}{4}(U_{ij}+U_{i,j+1}+U_{i-1,j}+U_{i-1,j+1}),\quad \overline{V}_{ij}=\frac{1}{4}(V_{ij}+V_{i,j-1}+V_{i+1,j}+V_{i+1,j-1}). \tag{3.4a}$$

However, for the OEH scheme we choose

$$\overline{U}_{ij} = \frac{1}{2}(U_{i-1,j} + U_{i,j+1}),\quad \overline{V}_{ij} = \frac{1}{2}(V_{i,j-1} + V_{i+1,j}). \tag{3.4b}$$

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

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

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

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 second-stage formula

$$\begin{aligned} \mathbf{U}^{n+½} &= \mathbf{U}^n + \frac{1}{2}\tau[\mathbf{F}_1(\mathbf{U}^{n+½}) + \mathbf{F}_2(\mathbf{U}^n)] \\ \mathbf{U}^{n+1} &= \mathbf{U}^{n+½} + \frac{1}{2}\tau[\mathbf{F}_1(\mathbf{U}^{n+½}) + \mathbf{F}_2(\mathbf{U}^{n+1})]. \end{aligned} \tag{3.6}$$

with $\tau$ denoting the time step. 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 $\theta_{ij}^n$ is defined by

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

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 (3.9) can be rewritten in the form (3.6), replacing $\tau$ by $\tau/2$,

$$\mathbf{U}^{n+½} = \mathbf{U}^n + \frac{1}{2}\tau[\mathbf{F}_E(\mathbf{U}^{n+½}) + \mathbf{F}_O(\mathbf{U}^n)] \tag{3.10a}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^{n+½} + \frac{1}{2}\tau[\mathbf{F}_E(\mathbf{U}^{n+½}) + \mathbf{F}_O(\mathbf{U}^{n+1})]. \tag{3.10b}$$

The order of computation for the OEH scheme is

$$\mathbf{U}_O^{n+½} = \mathbf{U}_O^n + \frac{1}{2}\tau \mathbf{F}_O(\mathbf{U}^n) \qquad ( = 2\mathbf{U}_E^n - \mathbf{U}_E^{n-½} \text{ if } n \geqslant 1 ) \tag{3.11a}$$

$$\mathbf{U}_E^{n+½} = \mathbf{U}_E^n + \frac{1}{2}\tau \mathbf{F}_E(\mathbf{U}^{n+½}) \tag{3.11b}$$

$$\mathbf{U}_E^{n+1} = \mathbf{U}_E^{n+½} + \frac{1}{2}\tau \mathbf{F}_E(\mathbf{U}^{n+½}) \qquad ( = 2\mathbf{U}_E^{n+½} - \mathbf{U}_E^n ) \tag{3.11c}$$

$$\mathbf{U}_O^{n+1} = \mathbf{U}_O^{n+½} + \frac{1}{2}\tau \mathbf{F}_O(\mathbf{U}^{n+1}) \tag{3.11d}$$

Notice 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 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. Notice 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 discretizations of the 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 [11]

$$\mathbf{U}^{n+½} = \mathbf{U}^n + \frac{1}{2}\tau[\mathbf{F}_x(\mathbf{U}^{n+½}) + \mathbf{F}_y(\mathbf{U}^n)] \tag{3.12a}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^{n+½} + \frac{1}{2}\tau[\mathbf{F}_x(\mathbf{U}^{n+½}) + \mathbf{F}_y(\mathbf{U}^{n+1})]. \tag{3.12b}$$

Notice that (3.12a) is explicit in the $y$-direction and implicit in the $x$-direction, and vice versa in (3.12b). Since there exists a 3-point coupling in each direction, the ADI scheme can be implemented such that only nonlinear tridiagonal systems have to be solved.

In order to obtain linear systems, we linearize the terms $\mathbf{F}_x(\mathbf{U}^{n+½})$ in (3.12a) and $\mathbf{F}_y(\mathbf{U}^{n+1})$ in (3.12b), which can be written in the form (cf. (3.3))

$$\mathbf{F}_x(\mathbf{U}^{n+½}) = A(U^{n+½})\mathbf{U}^{n+½} \quad \text{and} \quad \mathbf{F}_y(\mathbf{U}^{n+1}) = B(V^{n+1})\mathbf{U}^{n+1} \tag{3.13a}$$

by

$$\mathbf{F}'_x(\mathbf{U}^{n+½}) = A(U^*)\mathbf{U}^{n+½} \quad \text{and} \quad \mathbf{F}'_y(\mathbf{U}^{n+1}) = B(V^*)\mathbf{U}^{n+1}, \tag{3.13b}$$

with $A$ and $B$ tridiagonal matrices and $U^*$ and $V^*$ approximations to $U^{n+½}$ and $V^{n+1}$ respectively. To maintain second-order accuracy, the approximations $U^*$ and $V^*$ are calculated by (see [12])

$$U^* = \frac{3}{2}U^n - \frac{1}{2}U^{n-1}$$

$$V^* = 2V^{n+½} - V^n.$$

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

## *3.3. Stability*

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

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

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 $y$-directions respectively. Then von Neumann stability analysis applied to the OEH scheme (3.10) yields the following necessary and sufficient time step restriction [13,14] for (3.14)

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

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 [13,14]. By this we mean that for $\tau,h,k \to 0$, the solution of the OEH scheme converges to the solution of the problem

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

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

## 4. IMPLEMENTATION

In this section we describe implementation techniques for vectorizing both the OEH scheme and the ADI scheme 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 Section 2. The implementations have been written in the ANSI FORTRAN language known as FORTRAN 77. Thus, the resulting software is portable and auto-vectorizable.

### *4.1. The OEH scheme*

The OEH scheme consists of 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 Section 3.2.1). Specifically, the scheme only requires scalar divisions and no nonlinear equations have to 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 accordingly be divided into two vectors, which 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-d)). 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 a pleasant feature for the performance on vector computers. Moreover, no data reorderings have to be performed.

The reader may notice that the two vectors are not confined to one horizontal grid line, but 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 copied in advance into another vector. Moreover, the first and the last grid points of each horizontal line have to 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 $2.5NM$ memory locations.

### *4.2. The ADI scheme*

The ADI scheme for two-dimensional problems involves the solution of tridiagonal sets of equations along horizontal and vertical grid lines respectively. These sets of equations can be viewed in various ways. For example, for (3.12a) we have $M-1$ tridiagonal (linear) systems of order $N$. However, the $M-1$ individual systems can be combined to a single tridiagonal system of order $N(M-1)$. We prefer the latter choice in order 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 the maximal size of the main memory is about 1 million 64-bit words, whereas the maximal memory size on the Cray X-MP/24 is about 4 million 64-bit words.

Tridiagonal systems form an important class of linear algebraic equations. Consequently, efficient algorithms have been developed for the solution of such systems. The Gaussian elimination method has proven to be an efficient method on scalar computers. Unfortunately, due to the recursive nature of this method, the operations have to be evaluated one at a time. Therefore, the (sequential) Gaussian elimination method is unsuitable for use on vector- and parallel computers. Several methods have been proposed to achieve efficient methods 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, the method is more efficient on a vector computer because of its vector operations.

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. It was the approach of Wubs and De Goede, to approximate the solution of the reduced tridiagonal system by a truncated Neumann series [3]. The resulting explicit-implicit method is advantageous for use on vector computers. The prize 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 Section 2, for the performance on vector computers the data structure is very important. For the ADI scheme tridiagonal systems have to 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 have to be performed to change from horizontal to vertical lines and vice versa. The reordering operations have been implemented as efficient 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, the following implementations for the ADI-type schemes have been used :

| | |
|---|---|
| ADIW | the Peaceman-Rachford scheme where a variant of the partition method of Wang is used for the solution of the tridiagonal systems (stride $m$), |
| ADIW1 | ADIW with an extra reordering of the data structure (stride 1), |
| ADIGE | the Peaceman-Rachford scheme where Gaussian elimination is used for the solution of the tridiagonal systems, |
| ADEI | the Peaceman-Rachford scheme where the method developed by Wubs and De Goede is used for the solution of the tridiagonal systems (stride $m$). |
| 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 time steps and, when employing a uniform grid, a small grid size. Therefore, such problems are time- and memory consuming and the application of vector computers is obvious.

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

(i) CDC Cyber 205 ( SARA, Amsterdam, The Netherlands), max. 100 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, U.K.), 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 $\phi$ is the solution of

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

In our test problem we choose $\phi = f_1 + f_2$ [16], with

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

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 $= 100,1000,10000$.

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 time steps $\tau = 1/10, 1/20, 1/40, 1/80, 1/160, 1/320$ (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 which 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 Section 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 time step restriction

$$\tau \leqslant \frac{4}{5}\sqrt{2}\,h\,. \tag{5.5}$$

In Table 5.1 we list the $\mathrm{cd}_\infty$-values for all three schemes. We only list the $\mathrm{cd}_\infty$-values for the $u$-field; for the $v$-field we obtain nearly the same results.

| scheme | $h^{-1}$ | correct digits for $u$-field ($\infty$ - 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 |

Table 5.1. $\mathrm{cd}_\infty$-values for the OEH, ADIW and ADEI scheme.

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

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

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

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

Now, consider both ADI-type schemes. In the same way as for the OEH scheme, one 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 $\times$ 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 Section 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 time steps without paying attention to the accuracy or stability.

| scheme | Execution times (in seconds) | | |
|---|---|---|---|
| | Cyber 205 (vectorized code) | Cray X-MP/24 | Cyber 205 (scalar code) |
| OEH | 3.0 | 1.0 | 15.4 (5.1) |
| ADIW | 35.0 | 8.7 | 181.6 (5.2) |
| ADIW1 | 26.6 | 8.9 | 187.1 (7.0) |
| ADIGE | 57.6 | 16.6 | 121.4 (2.1) |
| ADEI | 30.6 | 6.1 | 176.6 (5.8) |
| ADEI1 | 21.2 | 6.8 | 182.2 (8.6) |

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

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 have to be solved and no data reorderings have to be performed. For the scalar code, it is fair to say that the ratio of the execution time for the ADI-type schemes 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. However, for the Gaussian elimination method the speed-up factor is only two. Due to its recursive nature the Gaussian elimination method is unsuitable for use on vector computers.

(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 about three times faster than the CDC Cyber 205. This is due to a smaller clock cycle (8.5 ns vs. 20 ns) and to a better compiler.

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 one may expect oscillations in the solution. Therefore, the $cd_\infty$-value, as defined in (5.4), is a too strict measure for the accuracy. Instead, we define

$$cd_1 = -{}^{10}\log(\,\|\text{ global error at } t = 2.5\,\|_1\,)\,.$$

We start our computations for $\text{Re} = 100$ on a $33 \times 33$ grid. On each grid and for each Re-number we choose the time step as large as possible such that $cd_1 \geqslant 3$. As soon as $cd_1 < 3$ for each time step we switch to the next finer grid and choose an appropriate time step. 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 time step 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.

| | 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) | | |
| 1000 | 2.51 | 3.42 | | | | 3.19 (1/80) | | |
| 1500 | | 3.06 | | | | 2.91(1/160) | 3.36 (1/80) | |
| 2000 | | 2.86 | 3.74 | | | | 3.15 (1/80) | |
| 3000 | | | 3.39 | | | | 3.09(1/160) | |
| 4000 | | | 3.18 | | | | 2.81(1/160) | 3.01 (1/80) |
| 5000 | | | 3.03 | | | | | 3.23(1/160) |
| 6000 | | | 2.88 | 3.70 | | | | 3.32(1/320) |
| 7000 | | | | 3.59 | | | | |
| 10000 | | | | 3.35 | | | | |

Table 5.3. $cd_1$-values for the OEH and ADIW scheme for increasing values of Re with $t = 2.5$.

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 time step. 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 Re = 100,1000,10000 computed with the OEH scheme.

## 6. Concluding remarks

In this paper we compared 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 we used the following three methods for the solution of the tridiagonal systems: the Gaussian elimination method (ADIGE), a variant of the partition method of Wang (ADIW) and the method developed by Wubs and De Goede (ADEI).

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 Section 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 have to be linearized in some way (cf (3.13)). Moreover, the OEH scheme can be extended to multi-dimensional problems in a straightforward manner, contrary to the ADI-type schemes.

The OEH scheme has the following disadvantages over the ADI-type schemes :

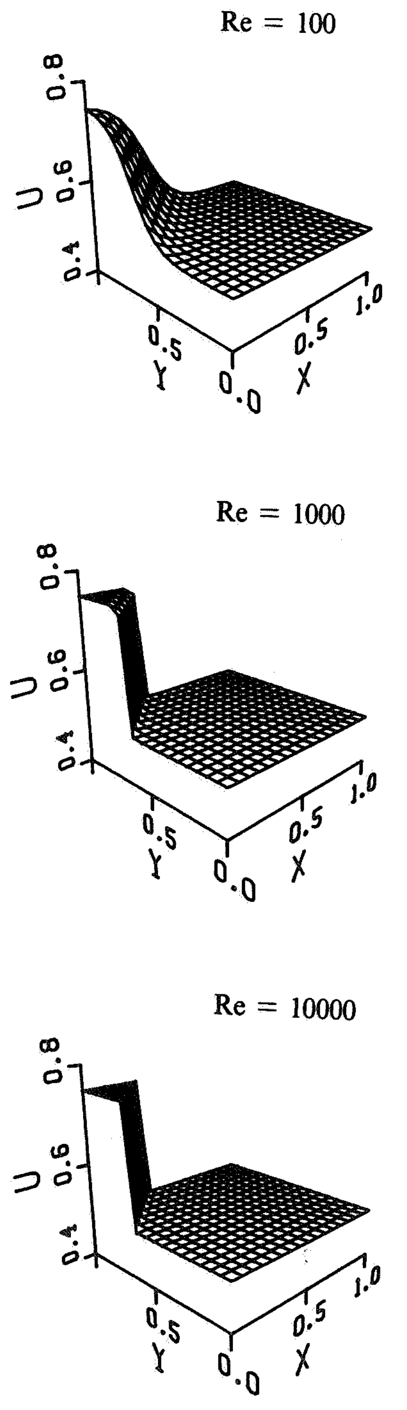

Fig. 2. Exact solutions (5.3a) for Re = 100, 1000, 10000.

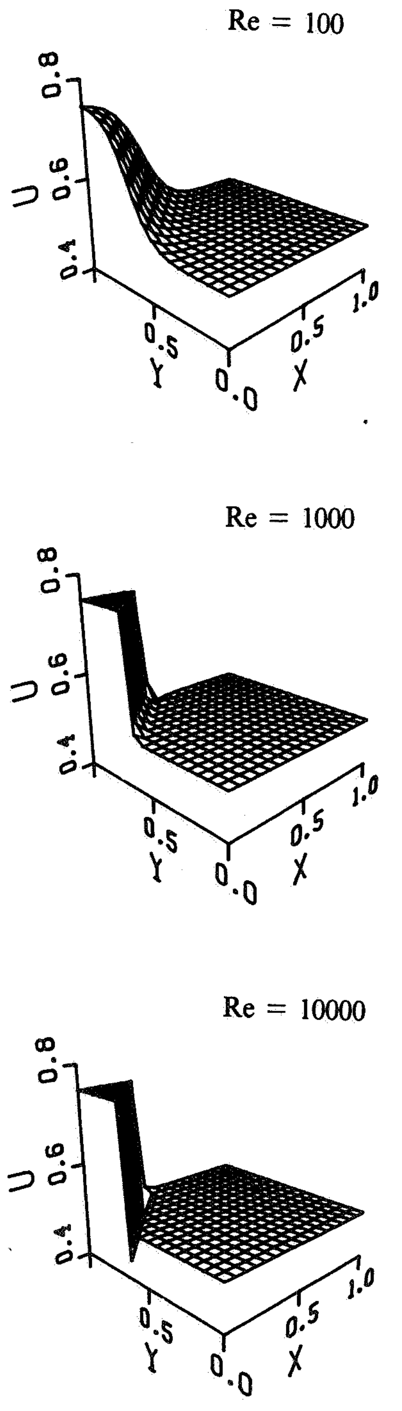

Fig. 3. Corresponding numerical solutions

(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, which in general has a negative influence on the accuracy.

Comparing the ADI-type schemes, it is evident that the ADEI scheme is in favour of the ADIW and ADIGE scheme, because it has a better performance on vector computers while the accuracy of both schemes is comparable.

In the near future, we will extend the codes for application to the incompressible Navier-Stokes equations.

ACKNOWLEDGEMENTS
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.

## 7. REFERENCES

[1] CDC Cyber 200 FORTRAN reference manual, version 1, publ. number 60480200H.
[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, 1987.
[4] A.R. GOURLAY, Hopscotch : a fast second-order partial differential solver, *J. Inst. Maths. Applics.*, 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, Ltd., 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. Stat. Comput.*, 7 (1986), pp. 870-891.
[9] P.H. MICHIELSE AND H.A. VAN DER VORST, *Data transport in Wang's partition method*, Report 86-32, Delft University of Technology, Delft, 1986.
[10] A.R. MITCHELL AND D.F. GRIFFITHS, *The finite difference method in partial differential equations*, Wiley, Chichester, 1980.
[11] D.W. PEACEMAN AND H.H. RACHFORD JR., The numerical solution of parabolic and elliptic differential equations, *J. Soc. Ind. 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, 1977.
[13] J.H.M. TEN THIJE BOONKKAMP, *The odd-even hopscotch pressure correction scheme for the incompressible Navier-Stokes equations*, to appear in SIAM J. Sci. Stat. Comput.
[14] 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.
[15] H.H. WANG, A parallel method for tridiagonal system equations, *ACM Trans. on Math. Softw.*, 7 (1981), pp. 170-183.
[16] G.B. WHITHAM, *Linear and nonlinear waves*, Wiley, New York, 1974.