The Numerical Computation of Time-Dependent, Incompressible Fluid Flow

The Numerical Computation of Time-Dependent, Incompressible Fluid Flow

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam, op gezag van de
Rector Magnificus dr. S.K. Thoden van Velzen,
hoogleraar in de Faculteit der Tandheelkunde,
in het openbaar te verdedigen in de Aula der Universiteit
(Oude Lutherse Kerk, ingang Singel 441, hoek Spui)
op woensdag 7 september 1988 te klokke 13.30 uur

door

Johannes Hendrikus Maria ten Thije Boonkkamp

geboren te Neede in 1959

1988
Centrum voor Wiskunde en Informatica

De numerieke berekening van tijdsafhankelijke, onsamendrukbare vloeistofstromingen

Promotor : prof.dr. P.J. van der Houwen
Co-promotor : dr. J.G. Verwer
Faculteit : Wiskunde en Informatica

PREFACE

The main subject of this thesis is the numerical computation of time-dependent incompressible fluid flow. The thesis consists of two parts. Part I is an introductory part, and gives a short account of the mathematical formulation of incompressible fluid flow. Some important solution techniques are discussed.

Part II is the main part of the thesis, and consists of five papers on the numerical computation of time-dependent, incompressible fluid flow and related topics. Two of these papers have been published in, respectively, Appl. Num. Math. and SIAM J. Sci. Stat. Comput., and a third one has been accepted for publication in ZAMM. The other two papers have been submitted for publication. The papers are:

1. *On the odd-even hopscotch scheme for the numerical integration of time-dependent partial differential equations*, Appl. Num. Math. 3 (1987), 183-193. (together with J.G. Verwer).
2. *The odd-even hopscotch pressure correction scheme for the incompressible Navier-Stokes equations*, SIAM J. Sci. Stat. Comput. 9 (1988), 252-270.
3. *The odd-even hopscotch pressure correction scheme for the computation of free convection in a square cavity*, (submitted for publication).
4. *Vectorization of the odd-even hopscotch scheme and the alternating direction implicit scheme for the two-dimensional Burgers' equations*, (submitted for publication). (together with E.D. de Goede).
5. *Residual smoothing for accelerating the ADI iteration method for elliptic difference equations*, ZAMM, (to appear).

I would like to express my gratitude to all of those who contributed in some way or another to the realization of this thesis. I want to mention some of them explicitly. In particular, I want to express my gratitude to dr. J.G. Verwer for his stimulating supervision and for his many contributions to this thesis. I am also grateful to prof. dr. P.J. van der Houwen for his guidance as a promotor and for his many valuable suggestions during the preparation of the last paper. I wish to thank drs. E.D. de Goede for his co-operation in writing the fourth paper and drs. J.G. Blom for her assistance in using the computer system. Finally, I would like to thank the Centre for Mathematics and Computer Science for giving me the opportunity to carry out the investigations which led to this thesis.

Amsterdam, May 1988
J.H.M. ten Thije Boonkkamp

Contents

Part I. Time-dependent incompressible fluid flow: a brief account

1. Introduction

Part I of this thesis gives a short mathematical formulation of incompressible fluid flow, and serves as a background to the five papers presented in Part II. These papers are concerned with numerical methods for time-dependent, incompressible fluid flow and related topics.

Part I contains the following. In Section 2, three possible ways to formulate incompressible fluid flow are presented, viz. the primitive variable formulation, the vorticity-streamfunction formulation and the streamfunction-biharmonic formulation. The difference between these formulations originates from the choice of the variables. In the primitive variable formulation, the variables are the velocity and the pressure of the flow field. This formulation is, especially for three-dimensional computations, most frequently used. In this thesis, we restrict ourselves to the primitive variable formulation.

The most important solution techniques for the equations in primitive variable formulation are discussed in Section 3. We can roughly distinguish two types of methods, viz. implicit methods and pressure- correction methods. In implicit methods, the velocity and the pressure are computed simultaneously by iteration. In this thesis, the term pressure-correction method is used to denote methods for which the computation of the velocity and the pressure is decoupled. In these methods, the pressure is directly computed from a Poisson equation. In the literature, the term fractional-step methods is also used for these methods. However, we use the term pressure-correction methods, since the term fractional step methods also denotes another class of integration-techniques for time-dependent partial differential equations. In this thesis we concentrate on the pressure- correction methods.

In the pressure-correction methods, we can globally distinguish three

computational stages:
— space discretization
— time-integration
— solution of a Poisson equation for the pressure.

For space discretization there are basically three possibilities, viz. the finite difference method, the finite element method and the spectral method. Each of these methods has its own advantages and drawbacks. In this thesis, we will only consider the finite difference method.

There are many time-integration techniques, which one can use in the pressure-correction methods. We can distinguish explicit methods, implicit methods and splitting methods. In this thesis, mainly splitting methods are considered. Examples of splitting methods are the odd-even hopscotch (OEH) scheme and the alternating direction implicit (ADI) scheme.

In the first paper, the OEH scheme is discussed for multi-dimensional convection-diffusion equations. In particular, the von Neumann stability for a class of linear convection-diffusion equations is examined. The scheme is tested by applying it to the Burgers' equations, which is a simplified model for the equations of incompressible fluid flow in primitive variable formulation.

The fourth paper is devoted to vectorization aspects of both the OEH scheme and the ADI scheme, for solving the Burgers' equations. Both schemes are implemented on the vector computers Cyber 205 and Cray X-MP/24. Data structures and techniques employed in vectorizing both schemes are discussed. An assessment of the performance of both schemes is given.

In the second paper, the OEH time-integration scheme is combined with an efficient Poisson-solver for the computation of incompressible fluid flow. The resulting scheme is called the odd-even hopscotch pressure-correction (OEH-PC) scheme. Accuracy and performance details of the scheme are examined for a simple test problem. A more practical test problem concerns the flow through a reservoir. The third paper is an extension of the second paper. In this paper, the OEH-PC scheme is used to compute free convection in a cavity, i.e. the flow in a cavity caused by a temperature gradient. In this case, the set of equations has to be extended with a convection-diffusion equation for the temperature.

Although there are nowadays many efficient Poisson-solvers available such as e.g. the Fast Fourier Transform method and many multigrid methods, the fifth paper still presents another approach to solving general elliptic equations (including the Poisson equation). In this paper, the classical ADI iteration method is combined with residual smoothing. Residual smoothing is a simple technique, which can be useful to accelerate the convergence of iterative methods for elliptic difference equations. When applied in the proper way, residual smoothing can considerably reduce the number of iterations and the computing time of the ADI iteration method.

2. General description of incompressible fluid flow

The flow of fluids and gases plays an important role in many fields of science and engineering. With the aid of modern computers, it is nowadays possible to simulate complicated flows. An important class of flows are the incompressible flows. A fluid flow is said to be incompressible when the density ρ of a fluid particle is constant along its trajectory in the flow. In most cases one usually assumes that ρ is constant in the whole flow field, so that this condition is satisfied. In general, the flow of a fluid and the flow of a gas at low speed (low compared with the speed of sound) can be considered incompressible. A few examples are the flow in a channel [3,15,24], the flow in a cavity [8,9,10,11,15,23,25] and the flow past a blunt body [7,13]. This section gives a short mathematical description of incompressible fluid flow, which is the basis of many computer simulations. For a more comprehensive discussion of the matter the reader is referred to e.g. [1,21].

The flow of an incompressible fluid is governed by the conservation of mass and momentum. The conservation of mass is described by the continuity equation

$$\nabla \cdot \mathbf{u} = 0, \tag{2.1}$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x},t)$ is the velocity field of the flow. The Navier-Stokes equations (or momentum equations) describe the conservation of momentum of the fluid flow. These equations read

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u} + \mathbf{f}. \tag{2.2}$$

In (2.2) $p = p(\mathbf{x},t)$ is the (scaled) pressure of the flow and $\mathbf{f} = \mathbf{f}(\mathbf{x},t)$ represents an external force acting on the fluid (e.g. the gravitational force). The parameter Re is the so-called Reynolds-number. The system (2.1)-(2.2) is defined on an connected space domain $\Omega \subset R^d$ ($d=2$ or $d=3$) and has to be completed with the following initial- and boundary conditions:
initial conditions:

$$\mathbf{u} = \mathbf{u}^\circ \text{ for } t = 0 \tag{2.3}$$

boundary conditions:

$$\mathbf{u} = \mathbf{u}_\Gamma \text{ on } \Gamma = \partial\Omega, \tag{2.4}$$

or more general

$$B\mathbf{u} = 0 \text{ on } \Gamma = \partial\Omega, \tag{2.4'}$$

where B is a general boundary operator. In (2.3), the initial field $\mathbf{u}^\circ$ has to satisfy the constraint $\nabla \cdot \mathbf{u}^\circ = 0$. If (2.4) applies, then $\mathbf{u}_\Gamma$ must satisfy the condition

$$\oint_\Gamma \mathbf{u}_\Gamma \cdot \mathbf{n} ds = 0, \tag{2.5}$$

where $\mathbf{n}$ is the normal unit vector to Γ. There are in general no pressure

boundary conditions available. The flow of an incompressible fluid is fully described by the equations (2.1)-(2.4). These equations are often referred to as the primitive variable formulation of the flow.

For numerical computation, the main difficulties associated with the primitive variable formulation are

(i) Equation (2.1) does not contain a time-derivative, therefore the application of a time-integration technique to (2.1)-(2.2) is not straightforward.
(ii) The lack of boundary conditions for the pressure.
(iii) The momentum equations (2.2) are nonlinear. Especially for large Re, when the convective term $(\mathbf{u}\cdot\nabla)\mathbf{u}$ dominates, these equations are therefore difficult to solve.

The flow of an incompressible fluid can also be formulated in terms of the vorticity ω and the stream function ψ. The vorticity ω is defined by

$$\omega = \nabla\times\mathbf{u}. \tag{2.6}$$

An equation for ω can be derived by applying the curl-operator to equation (2.2); the pressure term is then eliminated. This gives

$$\omega_t + (\mathbf{u}\cdot\nabla)\omega - (\omega\cdot\nabla)\mathbf{u} = \frac{1}{\mathrm{Re}}\nabla^2\omega + \nabla\times\mathbf{f}. \tag{2.7}$$

Since the velocity field $\mathbf{u}$ is solenoidal, i.e. $\mathbf{u}$ satisfies (2.1), $\mathbf{u}$ can be written as

$$\mathbf{u} = \nabla\times\psi \text{ with } \nabla\cdot\psi = 0, \tag{2.8}$$

where ψ is the solenoidal streamfuntion. The condition $\nabla\cdot\psi = 0$ is usually imposed on the streamfunction, but is not strictly necessary in order to obtain a solenoidal velocity field. From (2.6) and (2.8) one can easily derive the following equation for the streamfunction

$$\nabla^2\psi + \omega = 0. \tag{2.9}$$

The equations (2.7) and (2.9), together with appropriate initial- and boundary-conditions, is called the vorticity-streamfunction formulation of the flow.

The vorticity-streamfunction formulation becomes especially attractive for two-dimensional flow problems. Let in this case the velocity field be given by $\mathbf{u}(\mathbf{x},t) = u(x,y,t)\mathbf{i} + v(x,y,t)\mathbf{j}$. Then one can easily see that both ω and ψ have only a component in z-direction, i.e. $\omega = \omega(x,y,t)\mathbf{k}$ and $\psi = \psi(x,y,t)\mathbf{k}$. The equations for ω and ψ reduce to

$$\omega_t + (\mathbf{u}\cdot\nabla)\omega = \frac{1}{\mathrm{Re}}\nabla^2\omega \tag{2.7'}$$

$$\nabla^2\psi + \omega = 0. \tag{2.9'}$$

Note that in (2.7') we have assumed that the external force $\mathbf{f}$ satisfies $\nabla\times\mathbf{f} = 0$. The velocity components u and v can in this case be computed from the streamfunction ψ via the rule

$$u = \psi_y,\ v = -\psi_x. \tag{2.8'}$$

If the velocity $\mathbf{u}$ is known on $\Gamma = \partial\Omega$, then both Dirichlet- and Neumann-

boundary conditions for ψ can be derived (Cf (2.8′)). Boundary conditions for ω are generally not available. Techniques to derive artificial boundary conditions for ω are extensively described in [20].

In conclusion, we can say that the vorticity-streamfunction formulation is a feasible method for two-dimensional flow problems, since the number of variables is only two. For three-dimensional problems, however, the number of variables is six, and therefore the method is difficult to apply in this case. A major difficulty of the method, for both two- and three-dimensional problems, is the lack of boundary conditions for the vorticity.

A third way to describe incompressible fluid flow is the so-called streamfuction-biharmonic formulation. Since this formulation is (practically) never used in three-dimensional computations, we restrict ourselves to the two-dimensional case. Elimination of the vorticity ω in the vorticity-streamfunction formulation gives the following equation for ψ (Cf. (2.7′)-(2.9′))

$$(\nabla^2\psi)_t + \psi_y(\nabla^2\psi)_x - \psi_x(\nabla^2\psi)_y = \frac{1}{\mathrm{Re}}\nabla^4\psi. \tag{2.10}$$

Supplied with the proper Dirichlet- and Newmann- boundary conditions for ψ, equation (2.10) completely determines the flow.

The primitive variable formulation is often used, both for two-dimensional and three-dimensional flow problems, see e.g. [6,8,12,13,14,15]. The vorticity-streamfunction formulation is also often used, but almost always for two-dimsional flows, see e.g. [3,9,22]. An exception are [17,18], where this method is described for three-dimensional flow problems. The streamfunction- biharmonic formulation is seldom used, see e.g. [2]. An evaluation of all three methods for steady plane flow is given in [4].

In what follows, we will restrict ourselves to the primitive variable formulation (2.1)-(2.4). Section 3 gives a short description of some important solution methods for these equations. This section serves as a background to the five papers in Part II, and is of a rather technical nature.

3. Some solution methods for the primitive variable formulation

In this section we consider some important solution methods for the equations of time-dependent, incompressible fluid flow in primitive variable formulation. This section does not contain a comprehensive survey; it merely serves to demonstrate the essential features of some important solution methods. Consider to this purpose the equations (in primitive variable formulation) for incompressible fluid flow

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}) - \nabla p \text{ with } f(\mathbf{u}) = -(\mathbf{u}\cdot\nabla)\mathbf{u} + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{u},\ \mathbf{x}\in\Omega,\ t>0 \tag{3.1}$$

$$\nabla\cdot\mathbf{u} = 0,\ \mathbf{x}\in\Omega,\ t>0. \tag{3.2}$$

We assume that there are no external forces acting on the fluid. Appropriate initial- and boundary- conditions for the velocity field $\mathbf{u}$ are assumed, but play no essential role in what follows.

For introducing the various solution techniques, we follow the method of

lines approach. Thus, suppose that by a suitable space discretization technique (e.g. the finite difference technique) the set of partial differential equations (3.1)-(3.2) is converted into the following set of ordinary differential equations coupled with a set of algebraic equations

$$\dot{\mathbf{U}} = \mathbf{F}(\mathbf{U}) - GP \tag{3.3}$$

$$D\mathbf{U} = B. \tag{3.4}$$

In (3.3) and (3.4), $\mathbf{U}$ and P are gridfunctions defined on a space grid covering Ω and $\mathbf{F}(\mathbf{U})$ is the discrete approximation of $\mathbf{f}(\mathbf{u})$. G and D are the discrete gradient- and divergence-operator, respectively, and B is a term containing boundary values for the velocity $\mathbf{u}$.

We basically distinguish two classes of methods, viz. the pressure-correction methods and the implicit methods. In the pressure-correction methods, the computation of $\mathbf{U}$ and P is decoupled and the discrete pressure P is computed from a Poisson equation replacing the continuity equation. In the implicit methods, however, $\mathbf{U}$ and P are computed simultaneously from the momentum equations and the continuity equation, by an iterative procedure.

3.1. Pressure-correction methods

Many pressure-correction methods are inspired by the projection method of CHORIN [6]. Therefore, this method is briefly described first. Then an outline of some pressure-correction methods is given.

The principle of the projection method of Chorin can be explained as follows. Equation (3.1) can be written in the form

$$\mathbf{f}(\mathbf{u}) = \mathbf{u}_t + \nabla p. \tag{3.1'}$$

The vector $\mathbf{f}(\mathbf{u})$ is thus decomposed into the sum of a vector with zero divergence ($\mathbf{u}_t$) and a vector with zero curl (∇p). This decomposition exists and is uniquely determined.

Let $\mathbf{U}^n$ denote a fully discrete approximation of $\mathbf{u}(\mathbf{x},t)$ at time level $t_n = n\tau$, where τ is the time step (similar for P^n). At the first stage of the projection method, an auxiliary field $\mathbf{U}^*$ is computed from the rule

$$\alpha_0 \mathbf{U}^* + \sum_{j=1}^{r} \alpha_j \mathbf{U}^{n+1-j} = \mathfrak{F}(\mathbf{U}) := \beta_0 \tau \mathbf{F}(\mathbf{U}^*) + \sum_{j=1}^{r} \beta_j \tau \mathbf{F}(\mathbf{U}^{n+1-j}). \tag{3.5}$$

Equation (3.5) can be interpreted as a linear multistep scheme [16] for (3.3), from which the pressure terms GP^{n+1-j} are omitted. Other time-integration techniques, which do not fit into this formulation (such as e.g. ADI) are also possible.

In general, the auxiliary field $\mathbf{U}^*$ is not (discretely) divergence free. Therefore, $\mathbf{U}^*$ is projected onto the subspace of (discretely) divergence free vectors in the following way. Analogous to (3.1'), we perform the decomposition

$$\mathfrak{F}(\mathbf{U}) = \alpha_0 \mathbf{U}^* + \sum_{j=1}^{r} \alpha_j \mathbf{U}^{n+1-j} = (\alpha_0 \mathbf{U}^{n+1} + \sum_{j=1}^{r} \alpha_j \mathbf{U}^{n+1-j}) + GP^{n+1}. \tag{3.6}$$

In (3.6) the term between brackets is an approximation of $\dot{\mathbf{U}}^n$ and P^{n+1} is an approximation of P at time-level $t_{n+1} = (n+1)\tau$. Since the vectors $\mathbf{U}^{n+1-j}$ $(j = 1(1)r)$ are (discretely) divergence free, it is sufficient to perform the decomposition

$$\alpha_0 \mathbf{U}^* = \alpha_0 \mathbf{U}^{n+1} + GP^{n+1}, \tag{3.7a}$$

where $\mathbf{U}^{n+1}$ is (discretely) divergence free, i.e. (Cf. (3.4))

$$D\mathbf{U}^{n+1} = B^{n+1}. \tag{3.7b}$$

The variables $\mathbf{U}^{n+1}$ and P^{n+1} have to be solved (simultaneously) from (3.7a) and (3.7b). The solution method proposed by Chorin is therefore an iterative method.

An alternative approach, which we adopt in the second an third paper of Part II, is to multiply (3.7a) by D, which gives (using (3.7b))

$$LP^{n+1} = \alpha_0 (D\mathbf{U}^* - B^{n+1}), \tag{3.8}$$

where $L := DG$ is the discrete Laplace operator. The approximate pressure P^{n+1} is computed from the 'Poisson' equation (3.8) and $\mathbf{U}^{n+1}$ is then computed from (3.7a). To summarize, we get the following scheme

$$\alpha_0 \mathbf{U}^* + \sum_{j=1}^{r} \alpha_j \mathbf{U}^{n+1-j} = \beta_0 \tau \mathbf{F}(\mathbf{U}^*) + \sum_{j=1}^{r} \beta_j \tau \mathbf{F}(\mathbf{U}^{n+1-j}) \tag{3.9a}$$

$$LP^{n+1} = \alpha_0 (D\mathbf{U}^* - B^{n+1}) \tag{3.9b}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^* - \alpha_0^{-1} GP^{n+1}. \tag{3.9c}$$

This scheme is the general form of many pressure-correction schemes.

A very simple explicit version of this scheme is the following one proposed by FORTIN et al [8]

$$\mathbf{U}^* = \mathbf{U}^n + \tau \mathbf{F}(\mathbf{U}^n) \tag{3.10a}$$

$$LP^{n+1} = \frac{1}{\tau} (D\mathbf{U}^* - B^{n+1}) \tag{3.10b}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^* - \tau GP^{n+1}. \tag{3.10c}$$

At the first stage, the auxiliary field $\mathbf{U}^*$ is computed by the explicit Euler rule. This scheme is very similar to the well-known MAC-method of HARLOW and WELCH [12]. An advantage of this scheme is its simplicity, however, a drawback is the severe time step restriction for stability.

An implicit pressure-correction scheme, based on the Crank-Nicolson scheme, is the following

$$\mathbf{U}^* = \mathbf{U}^n + \tfrac{1}{2}\tau (\mathbf{F}(\mathbf{U}^*) + \mathbf{F}(\mathbf{U}^n)) \tag{3.11a}$$

$$LP^{n+1} = \frac{1}{\tau} (D\mathbf{U}^* - B^{n+1}) \tag{3.11b}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^* - \tau GP^{n+1}. \tag{3.11c}$$

This scheme is , in a certain sense, unconditionally stable, however it requires the solution of a nonlinear set of equations each time step. To overcome this problem, MOIN and KIM proposed the following pressure-correction scheme [15]. Let $\mathbf{F}(\mathbf{U}) = -\mathbf{N}(\mathbf{U}) + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{u}$, where $\mathbf{N}(\mathbf{u})$ is the discretization of the (nonlinear) convective term in the Navier-Stokes equations (Cf. (3.1)). Their pressure-correction scheme then reads

$$\mathbf{U}^* = \mathbf{U}^n - \tfrac{1}{2}\tau(3\mathbf{N}(\mathbf{U}^n) - \mathbf{N}(\mathbf{U}^{n-1})) + \frac{\tau}{2Re}(\nabla^2\mathbf{U}^* + \nabla^2\mathbf{U}^n) \quad (3.12a)$$

$$LP^{n+1} = \frac{1}{\tau}(D\mathbf{U}^* - B^{n+1}) \quad (3.12b)$$

$$\mathbf{U}^{n+1} = \mathbf{U}^* - \tau GP^{n+1}. \quad (3.12c)$$

At the first stage (3.12a) the auxiliary field $\mathbf{U}^*$ is computed using the second order Adams-Bashforth scheme for the convective term and the Crank-Nicolson scheme for the viscous term. Note that this scheme is only conditionally stable with respect to the convective term, but has no viscous stability restriction.

Another pressure-correction method, which is slightly different from (3.9a)-(3.9c) is due to VAN KAN [14]. To introduce this method, consider the Crank-Nicolson scheme for (3.3)

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \tfrac{1}{2}\tau(\mathbf{F}(\mathbf{U}^{n+1}) + \mathbf{F}(\mathbf{U}^n)) - \tfrac{1}{2}\tau(GP^{n+1} + GP^n), \quad (3.13)$$

coupled with the set of algebraic equations (3.7b). Note that the pressure P has to appear implicitly in (3.13), in order to satisfy (3.7b). The computation of $\mathbf{U}^{n+1}$ and P^{n+1} from (3.13) and (3.7b) is decoupled in a predictor-corrector fashion. First, substitute P^n for P^{n+1} in (3.13). This defines a predictor $\mathbf{U}^*$ as follows

$$\mathbf{U}^* = \mathbf{U}^n + \tfrac{1}{2}\tau(\mathbf{F}(\mathbf{U}^*) + \mathbf{F}(\mathbf{U}^n)) - \tau GP^n. \quad (3.14)$$

Substitution of $\mathbf{U}^*$ in the right and side of (3.13) (instead of $\mathbf{U}^{n+1}$) defines the corrector, also denoted by $\mathbf{U}^{n+1}$:

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \tfrac{1}{2}\tau(\mathbf{F}(\mathbf{U}^*) + \mathbf{F}(\mathbf{U}^n)) - \tfrac{1}{2}\tau(GP^{n+1} + GP^n). \quad (3.15)$$

From (3.14) and (3.15) one can easily see that

$$\mathbf{U}^{n+1} - \mathbf{U}^* = -\tfrac{1}{2}\tau GQ^n, \; Q^n := P^{n+1} - P^n. \quad (3.16)$$

Applying the operator D to (3.16) and using (3.7b), we obtain the following 'Poisson' equation for the pressure-increment Q^n:

$$LQ^n = \frac{2}{\tau}(D\mathbf{U}^* - B^{n+1}). \quad (3.17)$$

After the computation of Q^n, the new velocity field $\mathbf{U}^{n+1}$ and the new pressure P^{n+1} can be directly computed from (3.16). To summarize, we get the

following scheme

$$\mathbf{U}^* = \mathbf{U}^n + \tfrac{1}{2}\tau(\mathbf{F}(\mathbf{U}^*) + \mathbf{F}(\mathbf{U}^n)) - \tau GP^n \tag{3.18a}$$

$$LQ^n = \frac{2}{\tau}(D\mathbf{U}^* - B^{n+1}),\ P^{n+1} = P^n + Q^n \tag{3.18b}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^* - \tfrac{1}{2}\tau GQ^n . \tag{3.18c}$$

Note that the schemes (3.11) and (3.18) are very similar. They only differ in the treatment of the pressure-term. Finally, it should be noted that van Kan [14] uses the ADI scheme for the time integration, rather than the Crank-Nicolson scheme.

In the second paper of Part II, a pressure-correction scheme similar to (3.18) is used for the computation of incompressible fluid flow. The time-integration scheme used in this paper is the OEH scheme instead of the Crank-Nicolson scheme. An extension of this scheme, for the computation of free convection, is discussed in the third paper of Part II.

3.2. Implicit methods

Another class of methods are the implicit methods. These methods are based on an implicit discretization of the momentum- and continuity - equation. The resulting (nonlinear) algebraic system is then solved by iteration. We stress, that for these methods the equation of continuity is not replaced by a Poisson equation for the pressure. We demonstrate this technique by two examples.

The first example is due to PEYRET [19]. Consider again the Crank-Nicolson scheme (3.13) coupled with the set of equations (3.7b). Thus, for the sake of completeness, consider the following system

$$\mathbf{U}^{n+1} - \tfrac{1}{2}\tau\mathbf{F}(\mathbf{U}^{n+1}) + \tfrac{1}{2}\tau GP^{n+1} = \mathbf{U}^n + \tfrac{1}{2}\tau\mathbf{F}(\mathbf{U}^n) - \tfrac{1}{2}\tau GP^n \tag{3.19a}$$

$$D\mathbf{U}^{n+1} = B^{n+1} . \tag{3.19b}$$

Due to its implicitness, the scheme (3.19) has favourable stability properties. Let in the two-dimensional case $\mathbf{U} = (U,V)^T$, then the system (3.19a)-(3.19c) can be written in the symbolic form

$$\mathfrak{L}_u(U^{n+1}, V^{n+1}, P^{n+1}) = 0 \tag{3.20a}$$

$$\mathfrak{L}_v(U^{n+1}, V^{n+1}, P^{n+1}) = 0 \tag{3.20b}$$

$$\mathfrak{D}(U^{n+1}, V^{n+1}) = 0. \tag{3.20c}$$

The first two equations represent the momentum equations and equation (3.20c) represents the continuity equation. The iterative procedure for (3.20) proposed by Peyret is the following

$$U^{n+1,k+1} - U^{n+1,k} + \alpha\mathfrak{L}_u(U^{n+1,k}, V^{n+1,k}, P^{n+1,k}) = 0, U^{n+1,0} = U^n . \tag{3.21a}$$

$$V^{n+1,k+1} - V^{n+1,k} + \alpha\mathfrak{L}_v(U^{n+1,k+1}, V^{n+1,k}, P^{n+1,k}) = 0, V^{n+1,0} = V^n . \tag{3.21b}$$

$$P^{n+1,k+1} - P^{n+1,k} + \beta\mathfrak{D}(U^{n+1,k+1}, V^{n+1,k+1}) = 0, P^{n+1,0} = P^n . \tag{3.21c}$$

The superscripts k and n denote, respectively, the index of iteration and the time-level. The iteration parameters α and β have to be chosen properly, in order to ensure convergence. In general, the performance of the iterative method is very sensitive to the choice of the parameter values α and β, and it is very difficult to determine the optimum values of these parameters.

Another very interesting example is due to SOH and GOODRICH [23]. Their method is also based on the system (3.19). If we write $\mathbf{U}^{n+1} = \mathbf{U}^n + \delta\mathbf{U}^n$, $P^{n+1} = P^n + \delta P^n$ and $B^{n+1} = B^n + \delta B^n$, then the equations (3.19a)-(3.19b) can be rewritten as

$$\delta\mathbf{U}^n - \alpha(\mathbf{F}(\mathbf{U}^n + \delta\mathbf{U}^n) - G(P^n + \delta P^n)) = \alpha(\mathbf{F}(\mathbf{U}^n) - GP^n) \tag{3.22a}$$

$$D(\delta\mathbf{U}^n) = \delta B^n, \tag{3.22b}$$

where $\alpha = \tau/2$. In the spirit of the method of false transient [17], we can associate with (3.22a)-(3.22b) the auxiliary system

$$\frac{\partial}{\partial t^*}(\delta\tilde{\mathbf{U}}) + \delta\tilde{\mathbf{U}} - \alpha(\mathbf{F}(\mathbf{U}^n + \delta\tilde{\mathbf{U}}) - G(P^n + \delta\tilde{P})) = \alpha(\mathbf{F}(\mathbf{U}^n) - GP^n) \tag{3.23a}$$

$$\beta\frac{\partial}{\partial t^*}(\delta\tilde{P}) + (D(\delta\tilde{\mathbf{U}}) - \delta B^n) = 0, \tag{3.23b}$$

where t^* is a (non-physical) pseudo-time, β is the so-called artificial compressibility coefficient and $\delta\tilde{\mathbf{U}} = \mathbf{U}^* - \mathbf{U}^n$, $\delta\tilde{P} = P^* - P^n$; here the asterisk denotes a transient value in pseudo-time. From (3.22) and (3.23), we can see that the solution of the (nonlinear) system (3.22) is equal to the steady solution of the 'transient' system (3.23), assuming the latter exists. Thus , a time-integration technique for (3.23) can be interpreted as an iteration method for the solution of (3.22). Various explicit or implicit schemes can be used for the time-integration of (3.23). Soh and Goodrich use a factored ADI scheme [23]. Since no accuracy in the pseudo-time t^* is required, the corresponding time step τ^* can be chosen such that the convergence to a steady state is optimal. In this case, the parameter value β should be chosen carefully, for optimal performance. Notice that this method is very similar to the artificial compressibility method, which is a time-marching technique for the computation of the solution of the steady Navier-Stokes equations [5], applied at each time step.

References

[1] G.K. Batchelor, *An introduction to fluid dynamics,* Cambridge University Press, Cambridge, 1983.

[2] M. Bourcier and C. François, *Intégration numérique des équations de Navier-Stokes dans un domaine carré,* Rech. Aérosp. 131 (1969), 23-33.

[3] J.S. Bramley and S.C.R. Dennis, *A numerical treatment of two- dimensional flow in a branching channel,* Lecture Notes in Physics, No. 170, E. Krause, New York, 1982, 155-160.

[4] T. Cebeci et. al., *Studies of numerical methods for the plane Navier-Stokes equations,* Comput. Meth. Appl. Mech. Eng., 27 (1981), 13-44.

[5] A.J. Chorin, *A numerical method for solving incompressible viscous flow problems,* J. Comput. Phys., 2 (1967), 12-26.

[6] A.J. Chorin, *Numerical solution of the Navier-Stokes equations,* Math. Comp., 22 (1968), 745-762.

[7] S.C.R. Dennis and A.N. Staniforth, *A numerical method for calculating the initial flow past a cylinder in a viscous fluid,* Lecture Notes in Physics, No. 8, M. Holt, New York, 1971, 343-349.

[8] M. Fortin, R. Peyret and R. Temam, *Résolution numérique des équations de Navier-Stokes pour un fluide incompressible,* J. Méc., 10 (1971), 357-390.

[9] U. Ghia, K.N. Ghia and C.T. Shin, *High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method,* J. Comput. Phys., 48 (1982), 387-411.

[10] K. Gustafson and K. Halasi, *Vortex dynamics of cavity flows,* J. Comput. Phys., 64 (1986), 279-319.

[11] K. Gustafson and K. Halasi, *Cavity flow dynamics at higher Reynolds number and higher aspect ratio,* J. Comput. Phys., 70 (1987), 271-283.

[12] F.H. Harlow and J.E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluids with free surface,* Phys. Fluids, 8 (1965), 2182-2189.

[13] C.W. Hirt and J.L. Cook, *Calculating three-dimensional flows around structures and over rough terrain,* J. Comput. Phys., 10 (1972), 324-340.

[14] J. van Kan, *A second-order accurate pressure-correction scheme for viscous incompressible flow,* SIAM J. Sci. Stat. Comput., 7 (1986), 870-891.

[15] J. Kim and P. Moin, *Application of a fractional-step method to incompressible Navier-Stokes equations,* J. Comput. Phys., 59 (1985), 308-323.

[16] J.D. Lambert, *Computational methods in ordinary differential equations,* John Wiley & Sons, London, 1973.

[17] G.D. Mallinson and G. de Vahl Davis, *The method of false transient for the solution of coupled elliptic equations,* J. Comput. Phys., 12 (1973), 435-461.

[18] G.D. Mallinson and G. de Vahl Davis, *Three-dimensional numerical convection in a box: a numerical study,* J. Fluid Mech., 83 (1973), 1-31.

[19] R. Peyret, *Unsteady evolution of a horizontal jet in a stratified fluid,* J. Fluid Mech., 78 (1976), 49-63.

[20] R. PEYRET and T.D. TAYLOR, *Computational methods for fluid flow,* Springer-Verlag, New York, 1983.
[21] H. SCHLICHTING, *Boundary-layer theory,* Mcgraw-Hill, New York, 1979.
[22] W.A. SHAY and D.H. SCHULTZ, *A second-order approximation to natural convection for large Rayleigh numbers and small Prandtl numbers,* Int. J. Num. Meth. Fluids, 5 (1985), 427-438.
[23] W.Y. SOH and J.W. GOODRICH, *Unsteady solution of incompressible Navier-Stokes equations,* accepted by J. Comput. Phys.
[24] T.D. TAYLOR and E. NDEFO, *Computation of viscous flow in a channel by the method of splitting,* Lecture Notes in Physics, No.8, M. Holt, New York, 1971, 356-364.
[25] S.Y. TUANN and M.D. OLSON, *Review of computing methods for recirculating flows,* J. Comput. Phys., 29 (1978), 1-19.

THE FIVE PAPERS

Applied Numerical Mathematics 3 (1987) 183–193
North-Holland

ON THE ODD-EVEN HOPSCOTCH SCHEME FOR THE NUMERICAL INTEGRATION OF TIME-DEPENDENT PARTIAL DIFFERENTIAL EQUATIONS

J.H.M. TEN THIJE BOONKKAMP and J.G. VERWER
Centre for Mathematics and Computer Science, 1098 SJ Amsterdam, The Netherlands

This paper is devoted to the odd-even hopscotch scheme for the numerical integration of time-dependent partial differential equations. Attention is focussed on two aspects. Firstly, via the equivalence to the combined leapfrog-Du Fort–Frankel method we derive the explicit expression of the critical time step for von Neumann stability for a class of multi-dimensional convection-diffusion equations. This expression can be derived directly by applying a useful stability theorem due to Hindmarsh, Gresho and Griffiths [9]. The interesting thing on the critical time step is that it is independent of the diffusion parameter and yet smaller than the critical time step for zero diffusion, but only in the multi-dimensional case. This curious phenomenon does not occur for the one-dimensional problem. Secondly, we consider the drawback of the Du Fort–Frankel accuracy deficiency of the hopscotch scheme. To overcome this deficiency we discuss global Richardson extrapolation in time. This simple device can always be used without reducing feasibility. Numerical examples are given to illustrate the outcome of the extrapolation.

1. Introduction

The subject of this paper is the odd-even hopscotch (OEH) method for the numerical integration in time of time-dependent partial differential equations (PDEs) (Gordon [2], Gourlay [3–5]). Attention is focussed on two aspects. First we consider the d-space dimensional convection-diffusion equation

$$u_t + (q \cdot \nabla) u = \epsilon \Delta u, \qquad t > 0, \quad x \in \mathbb{R}^d, \tag{1.1}$$

where $u(x, t) \in \mathbb{R}$ represents the convected and diffused variable, the vector $q = (q_1, \ldots, q_d)$ the (constant) velocity, and $\epsilon > 0$ a diffusion parameter. When combined with simple central differences the OEH method shows an equivalence to the combined leapfrog-Du Fort–Frankel method. Via this equivalence we derive the explicit expression of the critical time step for von Neumann stability for problem (1.1). This expression is easily found by applying a useful theorem due to Hindmarsh, Gresho and Griffiths [9]. This theorem plays an important role in their stability analysis of the forward Euler-central difference scheme.

The interesting thing on the critical time step is that it is independent of ϵ whereas it is smaller than the critical time step for zero diffusion, but only in the multi-dimensional case. We wish to remark that this pathological behaviour of the leapfrog-Du Fort–Frankel method has been observed earlier (see [13] and the references therein). However, to the best of our knowledge, the explicit expression of the critical time step is new.

An immediate consequence of this pathological behaviour is that adding artificial diffusion to the OEH central difference scheme may render the process unstable. This observation is in clear

contrast to the common practice which teaches us that introducing artificial diffusion has a stabilizing effect. We note that this remark does not contradict the findings of Gourlay and Morris [6] in their investigation of the OEH scheme for nonlinear shock calculations as they restrict their attention to the one-dimensional case.

The second aspect of the OEH method considered in this paper is the drawback we refer to as the Du Fort–Frankel (DFF) accuracy deficiency. The consequence of the DFF deficiency is that convergence takes place for a smaller set of rules for refinement of the time-space mesh than allowed by stability [12]. To overcome this deficiency we discuss global Richardson extrapolation in time. This simple device can be placed on top of any OEH implementation without reducing feasibility. We present two examples to illustrate the technique.

2. The OEH method

In this section we briefly recall the OEH method which was first suggested by Gordon [2]. For an extensive discussion we refer to the work of Gourlay who invented the name hopscotch and made a thorough study of various techniques. Here we adopt his formulation.

Let the general form $u_t = Lu$ represent an evolutionary problem for a system of PDEs in d-space dimensions. Boundary conditions will not be specified here as we do not discuss their influence. So our study of the OEH method will be carried out as if we were studying the pure initial value problem. We let L_h be the finite difference replacement of the space operator L. Hence at the gridpoint x_j, where j represents a multi-index $(j_1, \ldots, j_d)$, $u_t = Lu$ is replaced by the continuous time ordinary differential equation

$$\dot{U}_j = L_h U_j. \tag{2.1}$$

In what follows it is supposed that in each coordinate direction L_h is based on second-order, three-point central differences on a uniform mesh. The restriction to uniform meshes in each coordinate direction is not essential. The OEH scheme allows a nonuniform mesh, but at most a three-point coupling in each coordinate direction. Hence one might also consider the use of simple one-sided spatial differencing. With regard to the convection-diffusion equation (1.1) we note that diffusion terms of the type $\nabla \cdot (\epsilon \cdot \nabla u)$, ϵ a $d \times d$ matrix [9], are not allowed because of the cross-derivatives.

According to Gourlay [3–5] the OEH scheme for problem (2.1) is given by

$$U_j^{n+1} = U_j^n + \tau\theta_j^n L_h U_j^n + \tau\theta_j^{n+1} L_h U_j^{n+1}, \tag{2.2}$$

where $\tau = t_{n+1} - t_n$, U_j^n approximates u at (x_j, t_n) and

$$\theta_j^n = \begin{cases} 1 & \text{if } \left(n + \sum_i j_i\right) \text{ is odd,} \\ 0 & \text{if } \left(n + \sum_i j_i\right) \text{ is even.} \end{cases} \tag{2.3}$$

If we take n fixed and consider only the odd points, for this n, (2.2) is just the forward Euler-central difference scheme. On the other hand, at the even points we recover the backward Euler-central difference scheme. Consequently, if we let n fixed and first apply the forward

scheme at all odd points and then the backward scheme at all remaining even points, we have carried out one step with the OEH scheme (2.2).

Due to the three-point coupling in each coordinate direction and the alternating use of forward and backward Euler methods, the process is only diagonally implicit. When applied to problem (1.1) it is even fully explicit (only division by scalars). This is also true for the nonlinear Burgers equation in divergence form

$$u_t + a(u) = \epsilon \Delta u, \qquad a(u) = \nabla \cdot (u; u). \tag{2.4}$$

The convective form, $a(u) = (u \cdot \nabla)u$, requires that per gridpoint a $d \times d$ system of linear algebraic equations must be solved. Of course this is still very cheap.

By writing down two successive steps of scheme (2.3),

$$\begin{aligned} U_j^{n+1} &= U_j^n + \tau\theta_j^n L_h U_j^n + \tau\theta_j^{n+1} L_h U_j^{n+1}, \\ U_j^{n+2} &= U_j^{n+1} + \tau\theta_j^{n+1} L_h U_j^{n+1} + \tau\theta_j^{n+2} L_h U_j^{n+2}, \end{aligned} \tag{2.5}$$

its connection to the Peaceman–Rachford method [3–5] is shown. In particular, if we let h fixed, (2.5) may be interpreted as a second-order integration formula using stepsize 2τ for the ODE system defined by (2.1) (see also [10]). From formulation (2.5) one can also derive the attractive fast form [3–5] which halves the computational work of the complete step $n \to n+2$. A particularly advantageous feature is that this fast form can be implemented such that only one array of storage is required. It is evident that this may be of considerable interest for multi-dimensional problems. Finally, when applied to the problems (1.1), (2.4) the fast form implementation requires roughly the same number of operations per step as the forward Euler scheme. However, the OEH method has much better stability properties. We discuss this in the next section.

3. Von Neumann stability for the linear convection-diffusion equation

In this section we derive the critical time step for von Neumann stability of the OEH central difference scheme for the convection-diffusion equation (1.1). Let h_k be the constant mesh width in the kth direction and H_k and δ_k^2 the corresponding finite difference operators for the first and second derivative, respectively. Then L_h can be written as

$$L_h U_j = \sum_k \left(-\frac{q_k}{2h_k} H_k + \frac{\epsilon}{h_k^2} \delta_k^2 \right) U_j,$$

where the summation is from 1 to d.

The stability analysis exploits the equivalence to the leapfrog-DFF scheme. This equivalence emerges by eliminating variables at the time level $n+1$ in (2.5). For the odd points we then get the relation

$$U_j^{n+2} = U_j^n + \tau L_h U_j^n + \tau L_h U_j^{n+2}, \tag{3.1}$$

and for the even ones

$$U_j^{n+2} = 2U_j^{n+1} - U_j^n. \tag{3.2}$$

By evaluating the linear expression $L_h U_j^{n+2}$ at the odd points and inserting (3.2) at the occurring even ones, relation (3.1) can be written as

$$\Big(1 + \sum_k 2\sigma_k\Big) U_j^{n+2} = \Big(1 - \sum_k 2\sigma_k\Big) U_j^n - \sum_k (c_k H_k - 4\sigma_k \mu_k) U_j^{n+1}, \tag{3.3}$$

where μ_k is the standard averaging operator in the kth direction and

$$\sigma_k = \epsilon\tau/h_k^2, \qquad c_k = q_k\tau/h_k. \tag{3.4}$$

Scheme (3.3) is the combination of the leapfrog and DFF scheme (the case $d = 1$ was studied earlier in [6]). Noteworthy is that (3.3) contains only grid values at the uncoupled set of odd-numbered points in space and time. Thus, if we ignore the start and completion of the OEH process and consider only the odd-numbered points, we may proceed with (3.3) for the investigation of linear stability. Note that U_j^{n+1} in (3.2) is a grid value at an odd point. Hence if the computation at the uncoupled set of odd-numbered points is stable, we have also stability at all even points.

We shall now examine the stability of scheme (3.3). For this purpose we employ the classical method of von Neumann [12]. So we introduce the Fourier mode

$$U_j^n = \xi^n e^{i\omega \cdot x}, \qquad \omega = (\omega_1, \ldots, \omega_d)^T \in \mathbb{R}^d, \quad \xi \in \mathbb{C}, \quad i^2 = -1,$$

and substitute into (3.3) to give

$$(1 + \sigma)\xi^2 + \Big(\sum_k 2c_k i \sin\theta_k - 4\sigma_k \cos\theta_k\Big)\xi - (1 - \sigma) = 0,$$

where $\theta_k = \omega_k h_k$ and $\sigma = 2(\sigma_1 + \cdots + \sigma_d)$. In what follows we demand von Neumann stability in the strict sense, that is

$$|\xi| \leqslant 1, \quad \text{all } |\theta_k| \leqslant \pi.$$

Bearing in mind that $\sigma_k > 0$ it then follows immediately from [11, Theorem 6.1] that we have stability iff the complex number λ given by

$$\lambda = \sum_k r_k \cos\theta_k - c_k i \sin\theta_k, \quad r_k = 2\sigma_k/\sigma,$$

satisfies $|\lambda| \leqslant 1$ for all $|\theta_k| \leqslant \pi$.

At this point we can make fruitful use of an interesting stability theorem due to Hindmarsh, Gresho and Griffiths [9] which they used in their stability analysis of the forward Euler-central difference scheme. As $\Sigma r_k = 1$, λ can be written as

$$\lambda = 1 - i\sum_k c_k \sin\theta_k + \sum_k r_k(\cos\theta_k - 1).$$

Their stability theorem then says that $|\lambda| \leqslant 1$ for all $|\theta_k| \leqslant \pi$ iff $\Sigma r_k \leqslant 1$ and

$$\sum_k c_k^2/r_k \leqslant 1. \tag{3.5}$$

Hence we can conclude immediately that this condition is sufficient and necessary for von Neumann stability (in the strict sense) of scheme (3.3).

Equation (3.5) may be rewritten as

$$\sum_k c_k^2/r_k = \sum_k \Big(\frac{\tau}{h_k}\Big)^2 \sum_k q_k^2 \leqslant 1, \tag{3.6}$$

and observe that the diffusion parameter ϵ is absent in this condition. This is plausible because the DFF scheme is unconditionally stable for the pure diffusion problem $u_t = \epsilon \Delta u$. Next an interesting situation arises if we put $\epsilon = 0$. Then scheme (3.3) reduces to the leapfrog scheme which is known to be stable in the strict sense of von Neumann iff the CFL condition holds:

$$\sum_k \frac{\tau}{h_k} |q_k| \leqslant 1. \tag{3.7}$$

Consequently, from the Cauchy inequality

$$\left(\sum_k \frac{\tau}{h_k} |q_k|\right)^2 \leqslant \sum_k \left(\frac{\tau}{h_k}\right)^2 \sum_k q_k^2, \tag{3.8}$$

it follows that the stability conditions are more restrictive for $\epsilon > 0$ than for $\epsilon = 0$. Hence if we add artificial diffusion to the OEH scheme for the pure convection problem we might destabilize the process. This observation is in clear contrast to the common practice which teaches us that introducing artificial diffusion has a stabilizing effect.

Observe that we have equality in (3.8) iff $h_k |q_k|$ is independent of k, so that only in this case the restrictions on τ and h_k in (3.6), (3.7) are identical. Of course, this is trivially so for $d = 1$ (see also [6]). If we put $h_k = h$, then (3.6), (3.7) lead to the time step restrictions

$$\tau^2 \leqslant h^2 / \left(d \sum_k q_k^2\right), \tag{3.6'}$$

$$\tau^2 \leqslant h^2 / \left(\sum_k |q_k|\right)^2. \tag{3.7'}$$

We see that when one of the velocities q_k dominates, the critical time step is approximately $\sqrt{d}$ times smaller than the critical time step imposed by the CFL condition.

We remark that the above pathological behaviour of the leapfrog-DFF scheme, and thus of the OEH scheme, has been observed earlier (see [13] and the references therein for numerical evidence). However, as far as we know, the expression for the critical time step implied by (3.6) is new.

Just for the sake of comparison we finally give the sufficient and necessary conditions for von Neumann stability of the forward Euler-central difference scheme for problem (1.1) [9]:

$$\sum_k \frac{2\epsilon\tau}{h_k^2} \leqslant 1, \qquad \sum_k \frac{q_k^2 \tau}{2\epsilon} \leqslant 1.$$

The second of these is known as the convection-diffusion barrier. It shows that the forward Euler-central difference scheme becomes unconditionally unstable as $\epsilon \to 0$. In contrast, the OEH central difference scheme is stable for all $\epsilon \geqslant 0$ under condition (3.6).

4. The DFF deficiency and global Richardson extrapolation

This section is devoted to the second aspect of the OEH method considered in this paper, viz. the DFF accuracy deficiency by which we mean that convergence takes place for a smaller set of rules for refinement of the space-time mesh than allowed by the stability condition [12]. For

example, if $\tau, h_k \to 0$ while satisfying (3.6) the solution of the leapfrog-DFF scheme (3.3) will converge to the solution of the problem

$$u_t + (q \cdot \nabla) u = \epsilon \Delta u - \epsilon a u_{tt}, \qquad a = \lim_{\tau,\, h_k \to 0} \sum_k \frac{\tau^2}{h_k^2}. \tag{4.1}$$

Hence for convergence to (1.1) it is necessary that $\tau^2 = o((\sum h_k^{-2})^{-1})$. Through the equivalence property the same conclusion is valid for the OEH scheme [2–5]. The equivalence to the leapfrog-DFF scheme cannot be derived for the nonlinear convection-diffusion equation. However, the DFF deficiency due to the viscous term $\epsilon \Delta u$ does still exist.

An immediate consequence of the DFF deficiency is that for a given space grid the OEH scheme may produce relatively inaccurate results unless τ is taken significantly smaller than necessary for stability. To a great extent, this disadvantage is compensated by the fact that per step the scheme is very cheap while on fixed space grids the fast form implementation (same costs as forward Euler) generates approximations which are second-order in τ. For those applications where the disadvantage is still pertinent we suggest the employment of global Richardson extrapolation in time. The idea is to eliminate the term $\tau^2 u_{tt}$ in (4.1). If we succeed in this elimination we have the usual convergence property because for stability τ must satisfy $\tau^2 = O((\sum h_k^{-2})^{-1})$.

The basis for global Richardson extrapolation is formed by the existence of asymptotic expansions for the global error. Without attempting full rigor we shall briefly sketch this. For this purpose we shall instead of examining the leapfrog-DFF scheme for linear problems directly consider the OEH scheme (2.5), but on a fixed space grid. Hence we follow the ODE approach and interpret (2.5) as a one-step, second-order integration formula using stepsize 2τ ($t_n \to t_{n+2}$). The advantage is that we then do not need to distinguish between linear and nonlinear problems since the theory of asymptotic expansions for one-step methods for ODEs applies generally. Numerical evidence which shows that our ideas are correct will be provided in Section 5.

For convenience of presentation we introduce the operator notation

$$U_j^{n+2} = S_h^{2\tau} U_j^n, \quad n = 0, 2, \ldots, \tag{4.2}$$

for the OEH scheme (2.5). Suppose that this scheme is applied from $t = 0$ up to $t_N = N(2\tau)$, N even, on a fixed spatial grid. Let

$$\epsilon_j^N = U_j^N - U_j(t_N) \tag{4.3}$$

be the global error at t_N for the intermediate ODE problem (2.1). Now if we let L_h be sufficiently smooth then it follows from numerical ODE theory [7, 8, 15] that grid functions $e^{(2)}$, $e^{(3)}, \ldots$ exist independently of τ such that

$$\epsilon_j^N = (2\tau)^2 e_j^{(2)}(t_N) + (2\tau)^3 e_j^{(3)}(t_N) + \cdots, \quad \tau \to 0. \tag{4.4}$$

The error term $(2\tau)^2 e_j^{(2)}(t_N)$ can be eliminated in the standard way by forming the combination

$$\tilde{U}_j^N = \tfrac{1}{3}\left(4U_j^N - V_j^N\right), \tag{4.5}$$

where U_j^N and V_j^N are obtained from the two different integrations

$$\begin{aligned} &U_j^{n+2} = S_h^{2\tau} U_j^n, \qquad n = o(2)(2N-2), \\ &V_j^0 = U_j^0, \quad V_j^{n+4} = S_h^{4\tau} V_j^n, \qquad n = o(4)(2N-4). \end{aligned} \tag{4.6}$$

Consequently,

$$\tilde{U}_j^N = U_j(t_N) - \tfrac{4}{3}(2\tau)^3 e_j^{(3)}(t_N) + \cdots . \tag{4.7}$$

Obviously, the error functions $e^{(2)}$, $e^{(3)}$,... do depend on the grid spacing. In fact, due to the DFF deficiency they grow beyond bound if the grid spacing is refined. Of course, this growth is annihilated by the simultaneous reduction of τ which is necessary for stability. In connection with the DFF deficiency we now hypothesize that the lack of convergence is entirely due to the first term $(2\tau)^2 e_j^{(2)}(t_N)$ in (4.4). If this hypothesis is true the elimination of this term by global Richardson extrapolation trivially implies that when the grid is refined and τ is reduced according to the requirement of stability, that then $\tilde{U}_j^N$ will converge to $u(x_j, t_N)$.

In contrast to its ease of implementation, the OEH scheme (4.2) is not very amenable to error analysis so that we have not attempted to analyze the expression (4.4). Numerical experimentation with a nonlinear problem (see Section 5) indicates that the hypothesis is true and even more, namely that by the simple extrapolation device we get a truly second-order behaviour upon simultaneous reduction of the time and space mesh.

Global extrapolation as formulated in (4.6) is easy to implement. Having computed an approximation at $t = t_N$, one simply repeats the process but now with a double stepsize. This can be done in parallel or after completion of the first integration. The additional computational effort is 50% and an extra array of storage is required. Above we tacitly assumed that τ is constant. However, the process can also be carried out for variable stepsizes [7, 8, 14–16] on the coarse grid without any additional difficulty. Finally we remark once more that a prerequisite for success of the extrapolation is that the asymptotics hold and thus that both integrations must be stable. Even marginal instability is not allowed because we then cannot count on a smooth global error. Lest we miss the obvious, the extrapolation makes no sense if the space error $U_j(t_N) - u(x_j, t_N)$ dominates the time error (4.3).

5. Numerical examples

We have applied the OEH central difference scheme to two initial/boundary value problems of the first type for the inhomogeneous Burgers equation in divergence form (cf. (2.4)). We recall that for this type of equation the OEH central difference scheme is essentially explicit (only division by scalars). Because so far nothing has been mentioned on the treatment of time-dependent boundary values and inhomogeneous terms we first give the necessary details on the actual implementation we applied. Consider the semi-discrete approximation (cf. (2.1))

$$\dot{U}_j = L_h U_j + F_j(t), \tag{5.1}$$

where $F_j(t)$ is the contribution of the inhomogeneous term at the gridpoint x_j. It is supposed that the prescribed solution values on the boundary are contained in $L_h U_j(t)$ (at the relevant places).

While omitting the fast form modifications our odd-even hopscotch implementation is then based on the one-step scheme ($t_n \to t_{n+2}$)

$$\begin{aligned} U_j^{n+1} &= U_j^n + \tau\theta_j^n L_h U_j^n + \tau\theta_j^{n+1} L_h U_j^{n+1} + \tau F_j(t_n + \tau), \\ U_j^{n+2} &= U_j^{n+1} + \tau\theta_j^{n+1} L_h U_j^{n+1} + \tau\theta_j^{n+2} L_h U_j^{n+2} + \tau F_j(t_n + \tau), \end{aligned} \tag{5.2}$$

where it is of importance to notice that any prescribed solution value from the boundary occurring in $L_h U_j^n$ has been set at the stage n. This can be motivated by the observation that each of the two stages is consistent with (5.1). The inhomogeneous term is computed at t_{n+1} in both stages in order to exploit fully the advantage of the fast form. When we apply the scheme only the points t_n, t_{n+2} are used for output.

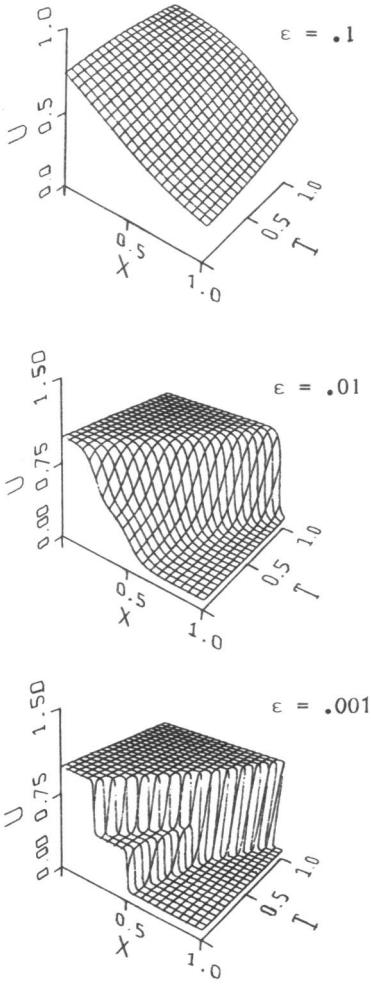


Fig. 1. Exact solutions (5.4) for three values of the parameter ϵ.

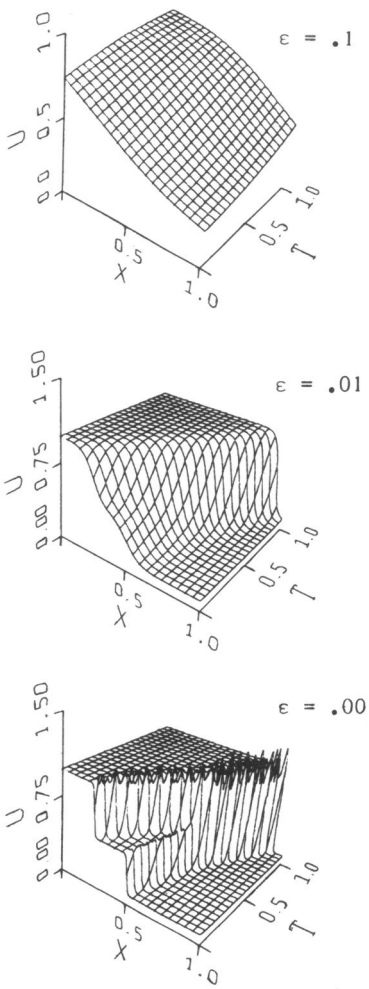


Fig. 2. Corresponding approximations generated by the OEH central difference scheme. We have used $h = \frac{1}{40}$, $\tau = \frac{1}{160}$ for $\epsilon = 0.1$ and $h = \frac{1}{80}$, $\tau = \frac{1}{320}$ for $\epsilon = 0.01$, 0.001.

Table 1
Number of significant digits of the OEH scheme

τ^{-1}	h^{-1}				
	20	40	80	160	320
20	2.04				
40	2.66	2.03			
80	3.24	2.63	2.03		
160	3.27	3.26	2.62	2.03	
320	3.27	3.84	3.23	2.62	2.03
640	3.26	3.88	3.86	3.23	2.62
1280	3.26	3.87	4.45	3.84	3.22
2560	3.26	3.87	4.48	4.47	3.83

Table 2
Number of significant digits of the extrapolated OEH scheme

τ^{-1}	h^{-1}				
	20	40	80	160	320
40	3.19				
80	3.30	2.98			
160	3.26	4.03	2.94		
320	3.26	3.87	4.66	2.93	
640	3.26	3.86	4.50	4.35	2.93
1280	3.26	3.86	4.47	5.21	4.29
2560	3.26	3.86	4.47	5.08	5.97

Problem 1. The one-space dimensional problem

$$u_t + \tfrac{1}{2}(u^2)_x = \epsilon u_{xx}, \qquad 0 < t \leqslant 1, \quad 0 < x < 1, \tag{5.3}$$

with the exact solution given by Whitham [17, Chapter 4]:

$$u(x, t) = 1 - 0.9\frac{r_1}{r_1 + r_2 + r_3} - 0.5\frac{r_2}{r_1 + r_2 + r_3}, \tag{5.4}$$

where

$$r_1 = \exp\left(-\frac{x - 0.5}{20\epsilon} - \frac{99t}{400\epsilon}\right), \quad r_2 = \exp\left(-\frac{x - 0.5}{4\epsilon} - \frac{3t}{16\epsilon}\right), \quad r_3 = \exp\left(-\frac{x - \frac{3}{8}}{2\epsilon}\right).$$

We have integrated this problem for $\epsilon = 0.1, 0.01, 0.001$. For these parameter values Fig. 1 and Fig. 2 show plots of the solution (5.4) and of corresponding numerical solutions, respectively. We see that when ϵ approaches zero the solution contains two shocks one of which is overtaken by the other. The plot for $\epsilon = 0.001$ in Fig. 1 clearly shows this. Noteworthy is that the corresponding OEH central difference approximation reproduces this behaviour in a very acceptable way. Of course, the wiggles in the approximation are due to the difficulty of fitting the shock on a too coarse grid.

For $\epsilon = 0.1$, Table 1 shows the minimum number of significant digits at $t = 1$, $\min_j(-{}_{10}\log(\text{abs. error at } (x_j, 1))$, for various values of τ and h. Note that since $\max(u(x, t)) = 1$ the critical time step for von Neumann stability is $\tau = h$. All computations with $\tau \leqslant h$ are indeed stable. Let us examine the results somewhat more closely. One then immediately recognizes the DFF deficiency: for a fixed mesh ratio τ/h the scheme fails to converge. Further, if we let $\tau \to 0$ and h fixed and sufficiently small, one can observe the second-order behaviour of the OEH integration formula (${}_{10}\log 4 \simeq 0.6$). For $\epsilon = 0.01, 0.001$ these phenomena remain hidden due to too large errors in space. Also recall that the DFF deficiency originates entirely from the second-order term ϵu_{xx}.

For $\epsilon = 0.1$ we present in Table 2 results of the extrapolated OEH scheme (4.5)–(4.6) (the meaning of the entries is the same as in Table 1). Let us examine the mesh ratio $\tau/h = \frac{1}{2}$. For this ratio the extrapolated scheme still suffers from the DFF deficiency, although the accuracy has improved. We think this is due to the fact that for the extrapolated scheme the mesh ratio $\tau/h = \frac{1}{2}$ is critical for von Neumann stability. Despite stability in such situations one may

encounter a nonsmooth global error so that Richardson extrapolation cannot be of much use. For the mesh ratios $\tau/h = \frac{1}{8}, \frac{1}{16}, \ldots$, the extrapolated scheme appears to be free of the DFF deficiency. More precisely, for these ratios the extrapolated scheme behaves as a second-order scheme as we expected. For the ratio $\tau/h = \frac{1}{4}$ the extrapolation works for $\tau = \frac{1}{80}, \frac{1}{160}, \frac{1}{320}$, but then yields no further improvement. We again think that this is due to a nonsmooth global error caused by a very weak instability. For clarity, we recall that the DFF deficiency may become visible at the moment that the space error becomes smaller than the time error. Of course, only then the extrapolation in time can be fruitful. Our tables illustrate this clearly. We also emphasize that in all these cases the outcome of the extrapolation is positive, including those where the DFF deficiency is still visible due to a too large mesh ratio.

Problem 2. The inhomogeneous, two-space dimensional Burgers equation

$$\begin{aligned} &u_t + (u^2)_x + (uv)_y = \epsilon\Delta u + f_1(x, y, t), \qquad 0 < t \leqslant 1, \quad 0 < x, y < 1, \\ &v_t + (uv)_x + (v^2)_y = \epsilon\Delta v + f_2(x, y, t), \end{aligned} \tag{5.5}$$

where

$$\begin{aligned} &u(x, y, t) = 2\pi \sin(2\pi x) \cos(y) z(x, y, t), \\ &v(x, y, t) = \cos(2\pi x) \sin(y) z(x, y, t), \\ &z(x, y, t) = 2\epsilon \exp\big(-(4\pi^2 + 1)\epsilon t\big)/\big(2 + \exp\big(-(4\pi^2 + 1)\epsilon t\big) \cos(2\pi x) \cos(y)\big). \end{aligned} \tag{5.6}$$

The functions u, v constitute an exact solution of the homogeneous Burgers equation in convective form [1]. Note that this solution is purely artificial as both u and v vanish as $\epsilon \to 0$ or $t \to \infty$. However, it is acceptable for our numerical illustration purpose which concerns the DFF deficiency and the Richardson extrapolation procedure. In Tables 3 and 4 we present results of the OEH central difference scheme ($h_1 = h_2 = h$) and of the extrapolation, respectively, for $\epsilon = 0.02$ and $t = 1$. The results concern the u-component. The numbers in the tables have the same meaning as in Table 1.

Noteworthy is that in Table 3 the DFF deficiency again manifests itself very clearly. On the other hand, the extrapolated OEH scheme turns out to be free of this deficiency as is shown in Table 4. It nicely shows the expected second-order behaviour, in particular for the mesh ratio $\tau/h = \frac{1}{4}$.

Our experience with the two numerical examples justifies the conclusion that the extrapolated OEH scheme possesses the normal convergence property. A very attractive feature of the global

Table 3
Number of significant digits of the OEH scheme

τ^{-1}	h^{-1}			
	10	20	40	80
20	2.71	2.78		
40	2.66	3.45	2.73	
80		3.32	3.39	2.71
160			4.06	3.34
320				4.00

Table 4
Number of significant digits of the extrapolated OEH scheme

τ^{-1}	h^{-1}			
	10	20	40	80
40	2.64	3.22		
80		3.23	3.80	
160			3.83	4.30
320				4.43

extrapolation in time is its simplicity. This classique technique can be placed on top of any OEH implementation, even for variable time steps. This allows the possibility of using it for stepsize control purposes, e.g. to avoid instabilities. In this section we have concentrated on convection-diffusion problems. The technique may also prove useful for pure diffusion problems. For such problems the OEH scheme has better stability properties so that the need for more accuracy in time will exist more frequently. Finally, we stress once more that the extrapolation idea assumes that the asymptotics hold. A prerequisite is thus that on both grids in time we have stability. In this respect, pure diffusion problems are even more attractive for application of the OEH extrapolation scheme.

References

[1] 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) 159–188.

[2] P. Gordon, Nonsymmetric difference equations, *SIAM J. Appl. Math. 13* (1965) 667–673.

[3] A.R. Gourlay, Hopscotch: a fast second order partial differential equation solver, *J. Inst. Math. Appl. 6* (1970) 375–390.

[4] A.R. Gourlay, Some recent methods for the numerical solution of time-dependent partial differential equations, *Proc. Roy. Soc. London A 323* (1971) 219–235.

[5] A.R. Gourlay, Splitting methods for time dependent partial differential equations, in: D. Jacobs, ed., *The State of the Art in Numerical Analysis* (Academic Press, London/New York/San Francisco, 1977) 757–791.

[6] A.R. Gourlay and J.LI. Morris, Hopscotch difference methods for nonlinear hyperbolic systems, *IBM J. Res. Develop. 16* (1972) 349–353.

[7] E. Hairer and C. Lubich, Asymptotic expansions of the global error of fixed-stepsize methods, *Numer. Math. 45* (1984) 345–360.

[8] P. Henrici, *Discrete Variable Methods in Ordinary Differential Equations* (Wiley, New York/London, 1962).

[9] A.C. Hindmarsh, P.M. Gresho and D.F. Griffiths, The stability of explicit Euler time-integration for certain finite-difference approximations of the multi-dimensional advection-diffusion equation, *Internat. J. Numer. Meth. Fluids 4* (1984) 853–897.

[10] P.J. van der Houwen and J.G. Verwer, One-step splitting methods for semi-discrete parabolic equations, *Computing 22* (1979) 291–309.

[11] J.J.H. Miller, On the location of zeros of certain classes of polynomials with applications to numerical analysis, *J. Inst. Maths. Appl. 8* (1971) 397–406.

[12] R.D. Richtmyer and K.W. Morton, *Difference Methods for Initial-Value Problems* (Interscience, New York/London/Sydney, 1967).

[13] U. Schumann, Linear stability of finite difference equations for three dimensional flow problems, *J. Comput. Phys. 18* (1975) 465–470.

[14] L.F. Shampine and H.A. Watts, Global error estimation for ordinary differential equations, *ACM Trans. Math. Software 2* (1976) 172–186.

[15] H.J. Stetter, *Analysis of Discretization Methods for Ordinary Differential Equations* (Springer, Berlin/Heidelberg/New York, 1973).

[16] J.G. Verwer and H.B. de Vries, Global extrapolation of a first order splitting method, *SIAM J. Sci. Statist. Comp. 6* (3) (1985).

[17] G.B. Whitham, *Linear and Nonlinear Waves* (Wiley-Interscience, New York, 1974).

28 Applied Numerical Mathematics 3 (1987) 361–362
North-Holland

CORRIGENDUM

J.G. BLOM, J.H.M. TEN THIJE BOONKKAMP and J.G. VERWER
Centre for Mathematics and Computer Science, Amsterdam, The Netherlands

"On the odd-even hopscotch scheme for the numerical integration of time-dependent partial differential equations", J.H.M. ten Thije Boonkkamp and J.G. Verwer [*Applied Numerical Mathematics 3* (1, 2) (1987) 183–193]

In Section 4 global Richardson extrapolation is suggested as a means for eliminating the Du Fort–Frankel deficiency. Then, in Section 5, on the basis of two numerical examples the conclusion is drawn that the deficiency is truly absent in the extrapolated scheme. This corrigendum serves to show that this conclusion has turned out to be incorrect.

First consider the linear heat flow equation $u_t = u_{xx}$ which is a special case of (4.1). Suppose that we compute a solution on successively finer grids, using the stepsizes $\tau = \tau_0, \frac{1}{2}\tau_0, \frac{1}{4}\tau_0, \ldots$ and $h = h_0, \frac{1}{2}h_0, \frac{1}{4}h_0, \ldots$, with either the hopscotch scheme or the Du Fort–Frankel scheme. It is thus assumed, through the equivalence property, that the two schemes generate the same approximate values. As $\tau, h \to 0$, these approximations converge to the solution values of the related problem $v_t = v_{xx} - (\tau^2/h^2)v_{tt}$. Let (x, t) be a point shared by all grids. Obviously $v(x, t) = v_{\tau/h}(x, t)$, that is, apart from initial and boundary data, $v(x, t)$ is determined exclusively bh the ratio of τ and h.

Let us now consider the suggested extrapolation procedure. In the limit, that is $\tau, h \to 0$ as above, we herewith form the values

$$w(x, t) = w_{\tau/h}(x, t) = \tfrac{4}{3}v_{\tau/h}(x, t) - \tfrac{1}{3}v_{2\tau/h}(x, t).$$

A trivial calculation shows that $w(x, t)$ is a solution of the differential equation

$$w_t = w_{xx} + \tfrac{4}{3}(\tau^2/h^2)(v_{2\tau/h} - v_{\tau/h})_{tt},$$

which still contains a (τ^2/h^2)-term. This implies that the extrapolation cannot have the effect which was aimed at. Also observe that $w_{\tau/h}(x, t) = w_{(\tau/2)/(h/2)}(x, t)$.

Next we consider the numerical example Problem 1 of Section 5 with the aim of presenting the correct interpretation of Table 2 and illustrating more comprehensively the effect of the extrapolation. For this purpose we show, in addition to Tables 1 and 2, the new Tables 1′, 2′ and 5. Their entries have the following meaning. Table 5 gives the accuracy obtained in the spatial discretization (5.1). Its entries contain the minimum of the number of significant digits in the space errors, i.e., $\min_j(-_{10}\log|u(x_j, 1) - U_j(1)|)$. Tables 1′ and 2′ give the accuracy obtained in the time integration of (5.1) by means of scheme (5.2) and the extrapolation thereof, respectively. Their entries contain the minimum of the number of significant digits in the time errors, i.e., $\min_j(-_{10}\log|U_j(1) - U_j^N|)$ for Table 1′ and $\min_j(-_{10}\log|U_j(1) - \tilde{U}_j^N|)$ for Table 2′. Recall that the old Tables 1, 2 refer to the full error, being the sum of the space error and the time error.

Table 1′

τ^{-1}	h^{-1}				
	20	40	80	160	320
20	2.03				
40	2.63	2.03			
80	3.23	2.62	2.03		
160	3.84	3.22	2.62	2.03	
320	4.44	3.83	3.22	2.62	2.03
640	5.04	4.43	3.83	3.22	2.62
1280	5.64	5.03	4.43	3.83	3.22
2560	6.24	5.63	5.03	4.43	3.83

Table 2′

τ^{-1}	h^{-1}				
	20	40	80	160	320
40	2.93				
80	4.28	2.93			
160	5.53	4.28	2.93		
320	6.74	5.53	4.28	2.93	
640	7.95	6.74	5.52	4.28	2.93
1280	9.15	7.95	6.74	5.52	4.28
2560	10.37	9.15	7.95	6.74	5.52

Table 5

h^{-1}	20	40	80	160	320
	3.26	3.86	4.47	5.07	5.67

Inspection of Table 1′ reveals two relevant features of the hopscotch scheme (5.2), namely its second order in time for fixed h (N.B. ${}_{10}\log 2 \approx 0.3$) and the h^{-2}-dependence of the error function $e^{(2)}$ occurring in the global error expansion (4.4). Of course, this h^{-2}-dependence is due to the Du Fort–Frankel deficiency. In passing we note that when the numbers in Table 1 are slightly larger than the minima of the corresponding numbers in Tables 1′ and 5, that is due to cancellation of time and space errors. We next consider the more interesting Table 2′ of the extrapolated scheme. Surprisingly, this table shows fourth order in time for fixed h (the entries increase approximately with 1.2 upon halving of τ), which means that the error function $e^{(3)}$ in (4.4) is absent. However, it also shows a h^{-4}-dependence of the next error function $e^{(4)}$, which in turn implies that the deficiency is still there. This observation illustrates our proof above for the heat equation.

Comparison of Tables 2′ and 1′ clearly shows that the extrapolation does reduce the time integration errors. In fact, the decrease is so large that for many of the entries the spatial error becomes dominant. This explains why in the greater part of the full error Table 2 second order shows up upon simultaneously halving τ and h and, consequently, why the extrapolation in connection to the Du Fort–Frankel deficiency was misinterpreted.

SIAM J. SCI. STAT. COMPUT.
Vol. 9, No. 2, March 1988

003

THE ODD-EVEN HOPSCOTCH PRESSURE CORRECTION SCHEME FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS*

J. H. M. TEN THIJE BOONKKAMP†

Abstract. The odd-even hopscotch (OEH) scheme is a time-integration technique for time-dependent partial differential equations. In this paper we apply the OEH scheme to the incompressible Navier-Stokes equations in conservative form. In order to decouple the computation of the velocity and the pressure, the OEH scheme is applied in combination with the pressure correction technique. The resulting scheme is referred to as the odd-even hopscotch pressure correction (OEH-PC) scheme. This scheme requires per time step the solution of a Poisson equation for the computation of the pressure. For space discretization we use standard central differences. We applied the OEH-PC scheme to the Navier-Stokes equations for the computation of an exact solution, with the purpose of testing the (order of) accuracy of the scheme in time as well as in space. Furthermore we applied the OEH-PC scheme for the computation of a model problem. Finally, a comparison between two Poisson solvers for the computation of the pressure is presented.

Key words. Navier-Stokes equations, odd-even hopscotch method, pressure correction method

AMS(MOS) subject classifications. 65M20, 76D05

1. The OEH-PC scheme: time-integration. In this section we consider the odd-even hopscotch (OEH) scheme applied to the incompressible Navier-Stokes equations in conservative form. The OEH scheme is an integration scheme for time-dependent partial differential equations (PDEs), and it is applicable to wide classes of problems. In addition, it possesses attractive computational properties which make the scheme relatively easy to implement. For a detailed discussion of the OEH scheme the reader is referred to [5] and [6]. Application to the compressible Navier-Stokes equations of a scheme related to the OEH scheme is discussed in [16] and [17].

We adopt the pressure correction approach, which means that during the time stepping process the computation of the velocity and the pressure is decoupled in a predictor-corrector fashion. In what follows, the resulting scheme will be referred to as the odd-even hopscotch pressure correction scheme (OEH-PC scheme). A discussion of the pressure correction approach can be found in [1], [2] and [12].

Consider the incompressible Navier-Stokes equations in conservative form in d space dimensions ($d=2$ or $d=3$) [15]

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}) - \nabla p, \quad \text{with } \mathbf{f}(\mathbf{u}) = -\nabla \cdot (\mathbf{uu}) + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}, \quad t>0, \quad \mathbf{x} \in \Omega, \tag{1.1}$$

$$\nabla \cdot \mathbf{u} = 0, \quad t>0, \quad \mathbf{x} \in \Omega, \tag{1.2}$$

where $\mathbf{u}$ is the (scaled) velocity, p the (scaled) pressure, and Re the Reynolds number. Boundary conditions, to be specified for the velocity field $\mathbf{u}$ on the boundary Γ of the connected space domain Ω, will be introduced later. We shall present the OEH-PC scheme for (1.1), (1.2) by following the method of lines approach [11]. Thus we suppose first that by an appropriate finite difference space discretization the PDE problem (1.1), (1.2) is replaced by a system of (time-continuous) ordinary differential equations (ODEs) coupled with a set of (time-continuous) algebraic equations

$$\dot{\mathbf{U}} = \mathbf{F}(\mathbf{U}) - GP, \tag{1.3}$$

$$DU = B. \tag{1.4}$$

* Received by the editors October 22, 1986; accepted for publication (in revised form) May 6, 1987.
† Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, the Netherlands.

In (1.3) and (1.4), $\mathbf{F}(\mathbf{U})$ is the finite difference replacement of $\mathbf{f}(\mathbf{u})$, G and D are the finite difference replacements of the gradient- and divergence-operator, respectively, and B is a term containing boundary values for the velocity $\mathbf{u}$.

At this stage of development of the OEH-PC scheme, there is no need to be precise on the form of (1.3), (1.4). It suffices to mention that $\mathbf{U}$, $\mathbf{F}$ and P are grid functions (vectors) defined on a space grid covering Ω. G and D are (nonsquare) constant matrices and B is a vector. In what follows, $j=(j_1,\cdots,j_d)$ is a multi-index connected to the grid point $\mathbf{x}_j$ of the space grid under consideration and $\mathbf{U}_j$ the component of $\mathbf{U}$ in the $\mathbf{x}_j$-direction (and likewise for $P, \mathbf{F}, B$).

We are now ready to define the OEH-PC scheme for the semidiscrete PDE problem (1.3), (1.4). First we consider only the ODE system (1.3). (Suppose for the time being that P is a known forcing term.) For this system the OEH scheme is given by the numerical integration formula

$$\mathbf{U}_j^{n+1}-\tau\theta_j^{n+1}(\mathbf{F}(\mathbf{U})_j^{n+1}-(GP)_j^{n+1})=\mathbf{U}_j^n+\tau\theta_j^n(\mathbf{F}(\mathbf{U})_j^n-(GP)_j^n). \tag{1.5}$$

Here $\tau=t_{n+1}-t_n$ is the time step, $\mathbf{U}_j^n$ stands for the fully discrete approximation to $\mathbf{U}_j(t_n)$, and θ is a grid function whose components θ_j^n are defined by [5], [6]

$$\theta_j^n=\begin{cases}1 & \text{if } n+\sum_i j_i \text{ is odd (odd points),}\\ 0 & \text{if } n+\sum_i j_i \text{ is even (even points).}\end{cases} \tag{1.6}$$

Note that if we keep n fixed, then (1.5) is just the explicit Euler rule at the odd points and the implicit Euler rule at the even ones. Alternating between the explicit and implicit Euler rules over the time-space grid is the essential feature of the OEH scheme. We return to this point later in the paper.

Writing down two successive steps of scheme (1.5) yields

$$\mathbf{U}_j^{n+1}=\mathbf{U}_j^n+\tau\theta_j^n\mathbf{F}(\mathbf{U})_j^n+\tau\theta_j^{n+1}\mathbf{F}(\mathbf{U})_j^{n+1}-\tau(GP)_j^n, \tag{1.7a}$$

$$\mathbf{U}_j^{n+2}=\mathbf{U}_j^{n+1}+\tau\theta_j^{n+1}\mathbf{F}(\mathbf{U})_j^{n+1}+\tau\theta_j^{n+2}\mathbf{F}(\mathbf{U})_j^{n+2}-\tau(GP)_j^{n+2}. \tag{1.7b}$$

Notice that in (1.7a) P is set at time level $t_n=n\tau$ and in (1.7b) at level $t_{n+2}=(n+2)\tau$. On a fixed space grid (1.7) may be interpreted as a second order integration formula, using stepsize 2τ, for the ODE system (1.3). A somewhat more convenient form of (1.7) is, using stepsize τ instead of 2τ,

$$\tilde{\mathbf{U}}=\mathbf{U}^n+\tfrac{1}{2}\tau\mathbf{F}_O(\mathbf{U}^n)+\tfrac{1}{2}\tau\mathbf{F}_E(\tilde{\mathbf{U}})-\tfrac{1}{2}\tau GP^n, \tag{1.8a}$$

$$\mathbf{U}^{n+1}=\tilde{\mathbf{U}}+\tfrac{1}{2}\tau\mathbf{F}_E(\tilde{\mathbf{U}})+\tfrac{1}{2}\tau\mathbf{F}_O(\mathbf{U}^{n+1})-\tfrac{1}{2}\tau GP^{n+1}, \tag{1.8b}$$

where $\mathbf{F}_O$ is the restriction of $\mathbf{F}$ to the odd points, etc. Note that $\mathbf{F}_O+\mathbf{F}_E=\mathbf{F}$. In (1.8) $\tilde{\mathbf{U}}$ is interpreted as a result from an intermediate time level like in a Runge-Kutta formula. We shall use this formulation in the remainder of the section.

Consider (1.8a), (1.8b) coupled with the (time-discretized) set of algebraic equations

$$D\mathbf{U}^{n+1}=B^{n+1}. \tag{1.8c}$$

The computation of $\mathbf{U}^{n+1}$ and P^{n+1} requires the simultaneous solution of (1.8b) and (1.8c). We compute on approximation to $\mathbf{U}^{n+1}$ and P^{n+1}, by following the well-known pressure correction approach [1], [2], [12], in which the computation of the velocity and pressure at the new time level is decoupled in the predictor-corrector fashion.

Substitution of P^n for P^{n+1} in (1.8b) defines the predicted velocity $\tilde{\tilde{\mathbf{U}}}$:

$$\tilde{\tilde{\mathbf{U}}} = \tilde{\mathbf{U}} + \tfrac{1}{2}\tau \mathbf{F}_E(\tilde{\mathbf{U}}) + \tfrac{1}{2}\tau \mathbf{F}_O(\tilde{\tilde{\mathbf{U}}}) - \tfrac{1}{2}\tau G P^n. \tag{1.9}$$

The corrected velocity and pressure (which we hereafter also denote by $\mathbf{U}^{n+1}$ and P^{n+1} and hence should not be mixed up with the approximations in (1.8a), (1.8b) and (1.8c)) are then defined by replacing $\mathbf{F}_O(\mathbf{U}^{n+1})$ in (1.8b) by $\mathbf{F}_O(\tilde{\tilde{\mathbf{U}}})$:

$$\mathbf{U}^{n+1} = \tilde{\mathbf{U}} + \tfrac{1}{2}\tau \mathbf{F}_E(\tilde{\mathbf{U}}) + \tfrac{1}{2}\tau \mathbf{F}_O(\tilde{\tilde{\mathbf{U}}}) - \tfrac{1}{2}\tau G P^{n+1}, \tag{1.10}$$

together with the discrete continuity equation (1.8c). From (1.9) and (1.10) we trivially obtain

$$\mathbf{U}^{n+1} - \tilde{\tilde{\mathbf{U}}} = -\tfrac{1}{2}\tau G Q^n, \qquad Q^n = P^{n+1} - P^n. \tag{1.11}$$

The trick of the pressure correction approach is now to multiply (1.11) by D and to write, using (1.8c),

$$LQ^n = \frac{2}{\tau}(D\tilde{\tilde{\mathbf{U}}} - B^{n+1}), \qquad L = DG. \tag{1.12}$$

Note that (1.12) is a Poisson equation for Q^n ($L = DG$ is a discretization of the Laplace operator $\nabla \cdot (\nabla)$). The correction Q^n for the pressure can be computed from (1.12), and once Q^n is known, the new velocity $\mathbf{U}^{n+1}$ can be directly determined from (1.11).

To sum up, the OEH-PC scheme for the semidiscrete Navier-Stokes problem (1.3), (1.4) reads

$$\tilde{\mathbf{U}} = \mathbf{U}^n + \frac{1}{2}\tau \mathbf{F}_O(\mathbf{U}^n) + \frac{1}{2}\tau \mathbf{F}_E(\tilde{\mathbf{U}}) - \frac{1}{2}\tau G P^n, \tag{1.13a}$$

$$\tilde{\tilde{\mathbf{U}}} = \tilde{\mathbf{U}} + \frac{1}{2}\tau \mathbf{F}_E(\tilde{\mathbf{U}}) + \frac{1}{2}\tau \mathbf{F}_O(\tilde{\tilde{\mathbf{U}}}) - \frac{1}{2}\tau G P^n, \tag{1.13b}$$

$$LQ^n = \frac{2}{\tau}(D\tilde{\tilde{\mathbf{U}}} - B^{n+1}), \qquad P^{n+1} = P^n + Q^n, \tag{1.13c}$$

$$\mathbf{U}^{n+1} = \tilde{\tilde{\mathbf{U}}} - \frac{1}{2}\tau G Q^n. \tag{1.13d}$$

When combined with a suitable space discretization, the OEH-PC scheme possesses various advantageous features. We shall discuss this in greater detail in the next section for symmetric finite differences on a staggered grid.

For the boundary conditions for the intermediate velocities we take the first order approximations $\tilde{\mathbf{U}} = \mathbf{u}(t_{n+1/2})$ and $\tilde{\tilde{\mathbf{U}}} = \mathbf{u}(t_{n+1})$ on Γ, where $\mathbf{u}$ is the exact boundary value for the velocity. The initial pressure P^0 is computed from the Poisson equation

$$LP^0 = D\mathbf{F}(\mathbf{U}^0) - \dot{B}^0, \tag{1.14}$$

which can be easily derived from (1.3) and (1.4).

We conclude this section with some remarks. First, the second stage (1.13b) can be economized by using its equivalent fast form (cf. [5], [6])

$$\tilde{\tilde{\mathbf{U}}}_E = 2\tilde{\mathbf{U}}_E - \mathbf{U}_E^n, \qquad \tilde{\tilde{\mathbf{U}}}_O = \tilde{\mathbf{U}}_O + \frac{1}{2}\tau \mathbf{F}_O(\tilde{\tilde{\mathbf{U}}}) - \frac{1}{2}\tau (GP^n)_O. \tag{1.13b'}$$

Our implementation is based on this fast form. Second, in the derivation of scheme (1.13) no use has been made of the particular definition of $\mathbf{F}_O$ and $\mathbf{F}_E$, except that $\mathbf{F}_O + \mathbf{F}_E = \mathbf{F}$. Consequently, in the spirit of the method of lines formulation [11], pressure

correction schemes using other splittings of **F**, such as ADI, can also be described by (1.13) (see e.g., [12], where an ADI splitting is used). It is of further interest to note that when considered as a solver for the ODE system (1.3) coupled with the set (1.4), the OEH-PC scheme is of second order, for the computation of **U** and of first order for the computation of P.

Finally, the OEH-PC scheme requires roughly the same number of operations per time-step as the forward Euler scheme (we will demonstrate this in § 2.1), but has much better stability properties. To illustrate this, consider the convection-diffusion equation which models the convective and viscous effects of the Navier-Stokes equations

$$u_t + (\mathbf{q} \cdot \nabla)u = \varepsilon \nabla^2 u, \quad t > 0, \quad \mathbf{x} \in \mathbb{R}^d. \tag{1.15}$$

Here $u(\mathbf{x}, t)$ represents the convected and diffused variable, the vector $\mathbf{q} = (q_1, \cdots, q_d)^T$ the (constant) velocity, and $\varepsilon > 0$ a viscosity parameter. Suppose that for space discretization we use standard central differences, with constant grid size h in all space-directions. If the OEH scheme is formulated like in (1.8), then von Neumann stability analysis applied to this scheme yields the following necessary and sufficient time step restriction [21]

$$d\left(\frac{\tau}{h}\right)^2 \sum_{k=1}^{d} q_k^2 \leqq 4. \tag{1.16}$$

For the forward Euler-central difference scheme, the time step restrictions for von Neumann stability are [10], [21]

$$\frac{2d\varepsilon\tau}{h^2} \leqq 1, \qquad \sum_{k=1}^{d} \frac{q_k^2 \tau}{2\varepsilon} \leqq 1. \tag{1.17}$$

The second inequality of (1.17) (convection-diffusion barrier) shows that the forward Euler-central difference scheme becomes unconditionally unstable as $\varepsilon \to 0$, whereas the OEH scheme is conditionally stable uniformly in ε, i.e., $\tau = O(h)$ independent of ε. Observe that the first inequality for the forward Euler-central difference scheme implies $\tau = O(\varepsilon^{-1}h^2)$, which is disadvantageous for larger values of ε. It is fair to say that, in general, a disadvantage of the OEH-central difference scheme is the so-called Du Fort-Frankel deficiency [5], [21]. However, as we will point out in § 3.2, in the present application this disadvantage is of minor importance.

2. The OEH-PC scheme: space discretization. In § 2.1 we will discuss the space discretization on a staggered grid of the Navier-Stokes problem, which defines the fully discrete OEH-PC scheme. We will show that due to the conservative form, our fully discrete OEH-PC scheme is in fact an explicit scheme, which needs only one array of storage for the computation of the velocity. In § 2.2 we will discuss the Poisson equation for the pressure, and in § 2.3 we will discuss (briefly) the space discretization on two other grids. For the sake of presentation, we restrict ourselves to two-dimensional rectangular domains.

2.1. Space discretization on a staggered grid. Consider the two-dimensional incompressible Navier-Stokes equations in conservative form

$$u_t = f_1(u, v) - p_x \quad \text{with } f_1(u, v) = -(u^2)_x - (uv)_y + \frac{1}{\mathrm{Re}}(u_{xx} + u_{yy}), \tag{2.1a}$$

$$v_t = f_2(u, v) - p_y \quad \text{with } f_2(u, v) = -(uv)_x - (v^2)_y + \frac{1}{\mathrm{Re}}(v_{xx} + v_{yy}), \tag{2.1b}$$

$$u_x + v_y = 0, \tag{2.2}$$

with boundary conditions

$$u = u_\Gamma, \qquad v = v_\Gamma \quad \text{on } \Gamma = \partial\Omega. \tag{2.3}$$

Note that there are no pressure boundary conditions available, although we have to solve a Poisson equation for the pressure. We will return to this point later in the section.

For the space discretization, we use the staggered grid first introduced by Harlow and Welch [8], see Fig. 1. The application of standard, second order central differences on this grid converts (2.1a) and (2.1b) into (cf. (1.3))

$$\dot{U}_{ij} = F_{1,ij}(U, V) - d_x P_{ij}, \qquad i = 1(1)N-1, \quad j = 1(1)M \quad \text{(interior } \times\text{-points)}, \tag{2.4a}$$

$$\dot{V}_{ij} = F_{2,ij}(U, V) - d_y P_{ij}, \qquad i = 1(1)N, \quad j = 1(1)M-1 \quad \text{(interior } \bigcirc\text{-points)}, \tag{2.4b}$$

where

$$F_{1,ij}(U, V) = -\frac{1}{2h}(U_{i+1,j}^2 - U_{i-1,j}^2) - \frac{1}{2k}(U_{i,j+1}\bar{V}_{i,j+1} - U_{i,j-1}\bar{V}_{i,j-1}) + \frac{1}{\mathrm{Re}\,h^2}\cdot(U_{i+1,j} - 2U_{ij} + U_{i-1,j}) + \frac{1}{\mathrm{Re}\,k^2}(U_{i,j+1} - 2U_{ij} + U_{i,j-1}), \tag{2.5a}$$

$$F_{2,ij}(U, V) = -\frac{1}{2h}(\bar{U}_{i+1,j}V_{i+1,j} - \bar{U}_{i-1,j}V_{i-1,j}) - \frac{1}{2k}(V_{i,j+1}^2 - V_{i,j-1}^2) + \frac{1}{\mathrm{Re}\,h^2}\cdot(V_{i+1,j} - 2V_{ij} + V_{i-1,j}) + \frac{1}{\mathrm{Re}\,k^2}\cdot(V_{i,j+1} - 2V_{ij} + V_{i,j-1}), \tag{2.5b}$$

$$d_x P_{ij} = \frac{1}{h}(P_{i+1,j} - P_{ij}), \tag{2.5c}$$

$$d_y P_{ij} = \frac{1}{k}(P_{i,j+1} - P_{ij}). \tag{2.5d}$$

Note that in the above formulation U, V and P are time-continuous grid functions whose components U_{ij}, V_{ij} and P_{ij} approximate the velocities u, v and the pressure p,

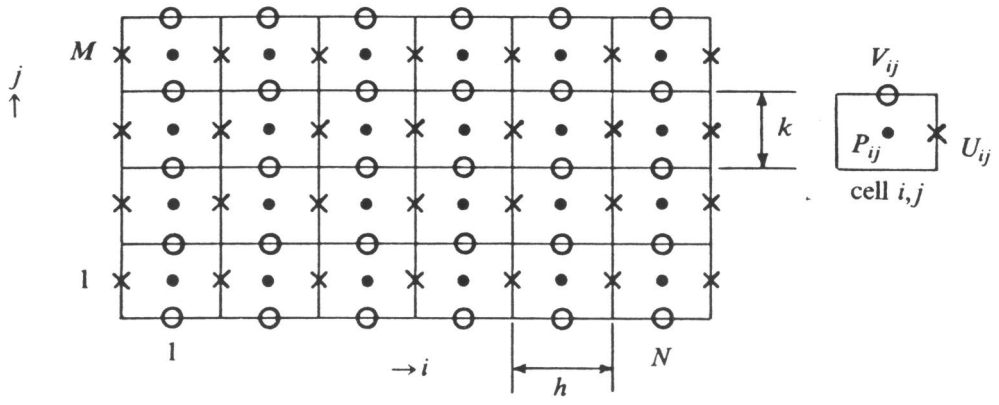


FIG. 1. *The staggered grid.*

respectively, at the corresponding gridpoints. In (2.5a) $\bar{V}_{ij}$ represents an approximation to V in the $\times$-points (points where U is defined); likewise $\bar{U}_{ij}$ represents an approximation to U in the $\bigcirc$-points. The values of $\bar{V}_{ij}$ and $\bar{U}_{ij}$ are determined by averaging over neighbouring values of V_{ij} and U_{ij} respectively, in such a way that the odd-even coupling between the variables is preserved. This means that a variable in an odd point is only coupled with variables in even points and vice versa. This leads to

$$\bar{U}_{ij} = \tfrac{1}{2}(U_{ij} + U_{i-1,j+1}), \qquad \bar{V}_{ij} = \tfrac{1}{2}(V_{ij} + V_{i+1,j-1}). \tag{2.6}$$

The space discretization of (2.1), as defined in (2.4), (2.5) determines the vector-function $\mathbf{F}(\mathbf{U})$ and the operator G in (1.3). Let $\mathbf{U} = (U, V)^T$, then $\mathbf{F}_{ij}(\mathbf{U}) = (F_{1,ij}(U, V), F_{2,ij}(U, V))^T$ and $GP_{ij} = (d_x P_{ij}, d_y P_{ij})^T$.

Concerning the boundary conditions for the velocity we note the following. Consider, for example, (2.1a) in the $\times$-points $(i, 1)(i = 1(1)N-1)$. Discretization of the derivatives $(uv)_y$ and u_{yy} would require values outside the computational domain. Therefore we replace the central difference approximations to $(uv)_y$ and u_{yy} by the following noncentered first order differences [15], which preserve the odd-even coupling between the variables

$$((uv)_y)_{i1} = \frac{2}{3k}(U_{i2}\bar{V}_{i2} - u(ih, 0)v(ih, 0)), \tag{2.7a}$$

$$(u_{yy})_{i1} = \frac{4}{3k^2}(U_{i2} - 3U_{i1} + 2u(ih, 0)). \tag{2.7b}$$

Second order noncentered approximations to $(uv)_y$ and u_{yy} would destroy the odd-even coupling.

Space discretization of (2.2) in all $\cdot$-points (using central differences) yields

$$(D\mathbf{U})_{ij} := \frac{1}{h}(U_{ij} - U_{i-1,j} + \beta(V_{ij} - V_{i,j-1})) = 0, \tag{2.8}$$

where $\beta = h/k$. Note that boundary values for U or V occurring in (2.8) are written in the right-hand side B (cf. (1.4)). For example, for $j = 1$, (2.2) is discretized as

$$(D\mathbf{U})_{i1} := \frac{1}{h}(U_{i1} - U_{i-1,1} + \beta V_{i1}) = B_{i1} = \frac{1}{k} V_{i0}. \tag{2.8'}$$

Having defined the operators G and D, one can easily deduce the following expression for the operator L

$$\begin{aligned}(LQ)_{ij} = D(GQ)_{ij} &= \frac{1}{h}(d_x Q_{ij} - d_x Q_{i-1,j} + \beta(d_y Q_{ij} - d_y Q_{i,j-1})) \\ &= \frac{1}{h^2}(\beta^2 Q_{i,j-1} + Q_{i-1,j} - (2 + 2\beta^2)Q_{ij} + Q_{i+1,j} + \beta^2 Q_{i,j+1}),\end{aligned} \tag{2.9}$$

which is the standard 5-point molecule for the Laplace operator. Near a boundary (2.9) takes a different form, because of the different definition of the operator D. For example for $j = 1$, one finds

$$\begin{aligned}(LQ)_{i1} = D(GQ)_{i1} &= \frac{1}{h}(d_x Q_{i1} - d_x Q_{i-1,1} + \beta d_y Q_{i1}) \\ &= \frac{1}{h^2}(Q_{i-1,1} - (2 + \beta^2)Q_{i1} + Q_{i+1,1} + \beta^2 Q_{i2}).\end{aligned} \tag{2.9'}$$

Now, consider (1.13c) at the ·-points $(i, 1)(i = 1(1)N)$. Using (2.8), (2.8′), (2.9) and (2.9′), it is easy to see that $(Q^n_{i0} - Q^n_{i1})/k = 2(V^{n+1}_{i0} - \tilde{V}_{i0})/\tau = 0$, which is the (central difference) approximation of $(\partial Q^n/\partial n)_{i0} = 0$, where n is the outward unit normal on $x = 0$. A similar argument applied to the Poisson equation for the initial pressure P^0 ((1.14)) leads for $j = 1$ to the boundary condition $(\partial P^0/\partial n)_{i0} = \dot{V}^0_{i0} - F_{2i0}(U^0, V^0)$, which is in accordance with the Navier–Stokes equations. Hence we see that a Neumann condition for the pressure (-increment) is automatically involved in the scheme.

Thus the scheme implies $\partial P^n/\partial n = \partial P^0/\partial n$ on Γ at every time level $t_n = n\tau$, although the exact pressure does not in general satisfy this condition. Most methods involve artificial pressure boundary conditions, like, for example, the projection method [2], [4], [15], [20]. In [20] Temam defines a projection method, which is a predictor corrector method like the pressure correction method, in which in the predictor step the pressure term is completely omitted. This scheme implies the "unphysical" boundary condition $\partial P/\partial n = 0$. Nevertheless, he proves the convergence of his scheme. Therefore, it is believed that our OEH-PC scheme does converge too, although the artificial pressure condition will lead to some loss of accuracy. This is demonstrated with a numerical example in § 3.1. A proof of convergence is out of the scope of the present paper.

Having defined the space discretization, we now discuss in some detail the merits of the resulting fully discrete OEH-PC scheme. Consider (1.13a) and (1.13b) of the OEH-PC scheme. The order of computation is

$$\tilde{\mathbf{U}}_O = \mathbf{U}^n_O + \tfrac{1}{2}\tau \mathbf{F}_O(\mathbf{U}^n) - \tfrac{1}{2}\tau(GP^n)_O, \tag{2.10a}$$

$$\tilde{\mathbf{U}}_E = \mathbf{U}^n_E + \tfrac{1}{2}\tau \mathbf{F}_E(\tilde{\mathbf{U}}) - \tfrac{1}{2}\tau(GP^n)_E, \tag{2.10b}$$

$$\tilde{\tilde{\mathbf{U}}}_E = \tilde{\mathbf{U}}_E + \tfrac{1}{2}\tau \mathbf{F}_E(\tilde{\mathbf{U}}) - \tfrac{1}{2}\tau(GP^n)_E = 2\tilde{\mathbf{U}}_E - \mathbf{U}^n_E, \tag{2.10c}$$

$$\tilde{\tilde{\mathbf{U}}}_O = \tilde{\mathbf{U}}_O + \tfrac{1}{2}\tau \mathbf{F}_O(\tilde{\tilde{\mathbf{U}}}) - \tfrac{1}{2}\tau(GP^n)_O. \tag{2.10d}$$

This scheme is in fact an explicit scheme. To demonstrate this, consider the computation of $\tilde{\mathbf{U}}$. Clearly the computation of $\tilde{\mathbf{U}}_O$ is explicit. Equation (2.10b) for the computation of $\tilde{\mathbf{U}}_E$ reads for the U-component in an even point (i, j) (substitute (2.5a), (2.5c) and (2.6))

$$\begin{aligned}\tilde{U}_{ij} = U^n_{ij} &- \frac{\tau}{4h}(\tilde{U}^2_{i+1,j} - \tilde{U}^2_{i-1,j}) - \frac{\tau}{8k}(\tilde{U}_{i,j+1}(\tilde{V}_{i,j+1} + \tilde{V}_{i+1,j}) \\ &- \tilde{U}_{i,j-1}(\tilde{V}_{i,j-1} + \tilde{V}_{i+1,j-2})) + \frac{\tau}{2\,\mathrm{Re}\,h^2}(\tilde{U}_{i+1,j} - 2\tilde{U}_{ij} + \tilde{U}_{i-1,j}) \\ &+ \frac{\tau}{2\,\mathrm{Re}\,k^2}(\tilde{U}_{i,j+1} - 2\tilde{U}_{ij} + \tilde{U}_{i,j-1}) - \frac{\tau}{2h}(P^n_{i+1,j} - P^n_{ij}).\end{aligned} \tag{2.11}$$

The values of $\tilde{U}_{i\pm1,j}$, $\tilde{U}_{i,j\pm1}$, $\tilde{V}_{i,j\pm1}$ and $\tilde{V}_{i+1,j-2}$ are odd numbered values which were already computed with (2.10a). This means that (2.11) is only diagonally implicit, since $\tilde{U}_{ij}$ is the only unknown, and hence explicit. In the same way, the computation of $\tilde{V}_E$ is explicit. A similar argument applies to the computation of $\tilde{\tilde{\mathbf{U}}}$.

In scheme (2.10a)–(2.10d) the steps (2.10b) and (2.10c) are considered as one computational step; first compute $\tilde{\mathbf{U}}_E$ in a point, store this value in a dummy-variable, and then compute $\tilde{\tilde{\mathbf{U}}}_E$ in the same point, using the fast form. Taking this into consideration, we easily see that only one array of storage is required for the computation of $\tilde{\tilde{\mathbf{U}}}$, which is especially advantageous for multidimensional problems.

2.2. The Poisson equation for the pressure. The pressure increment Q^n is computed from (1.13c), where the operator L is defined as in (2.9) and (2.9′). In fact

L is the 5-point discretization of the Laplace operator with Neumann boundary conditions. Considered as a matrix, L has a few attractive properties, such as symmetry, negative definiteness and a pentadiagonal structure. There are many methods available for the solution of a set of equations with matrix L. Since the OEH scheme is very cheap per step, it is essential that we combine it with a fast Poisson solver in order to obtain a fast OEH-PC scheme. In our computations, we used the incomplete Choleski conjugate gradient (ICCG) method [13], [14] and a multigrid (MG) method [9], [19]. A comparison between these two methods will be presented in § 3.3.

2.3. Space discretizations on other grids. For the space discretization, one can also use the ordinary grid or the half-staggered grid, see Fig. 2; cf. [15]. In the ordinary grid, the components of the velocity and the pressure are all defined at the nodes of the grid. The advantage of this grid is its simplicity, especially treatment of the boundary conditions for the velocity is straightforward.

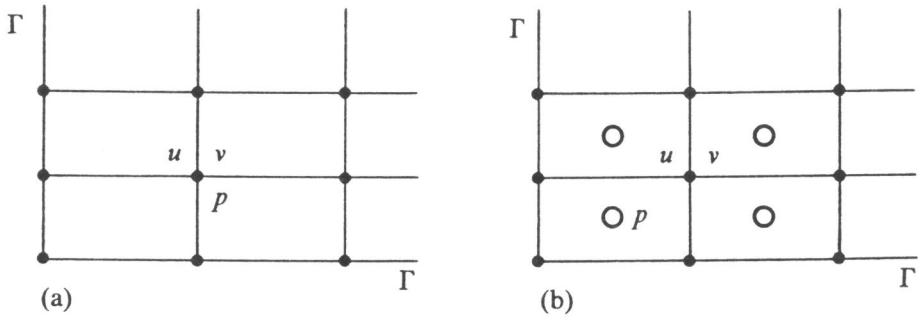


FIG. 2. *The ordinary grid* (a) *and the half-staggered grid* (b).

However a disadvantage of this grid is the fact that the pressure is defined at nodes on the boundary. Therefore, computation of the pressure in a pressure correction fashion requires pressure boundary conditions, which are generally not available. In the half-staggered grid, the components of the velocity are defined at the nodes of the grid and the pressure is defined at the centre of each cell of the grid, cf. [4], [15]. The pressure is not prescribed on the boundary anymore, and hence no pressure boundary conditions are required. A disadvantage of this grid is the fact that the discretization of the gradient- and divergence-operator is slightly more difficult than on the ordinary grid or on the staggered grid.

The major drawback of the ordinary grid and the half-staggered grid is the fact that these grids are not suitable for the computation of the pressure in a pressure correction fashion. To make this plausible, consider the molecule for the operator $L = DG$ on the ordinary grid and the half-staggered grid, respectively, when standard central differences are used for the discretization of the gradient- and divergence-operator; see Fig. 3. The operator L on the ordinary grid is again the usual 5-point discretization of the Laplacian, but now on a grid with double gridsize. The consequence is that there exist four uncoupled networks of pressure points (see Fig. 3). This leads to the existence of four independent solutions for the pressure, which differ from each other by arbitrary constants. Furthermore, due to the double gridsize, the pressure on the ordinary grid will be less accurate than on the staggered grid. The operator L on the half-staggered grid is a 9-point discretization of the Laplacian unless $\beta = 1 (h = k)$, then L is a 5-point discretization of the Laplacian denoted by the solid lines. In the latter case, the pressure field is decoupled in two independent pressure fields, which

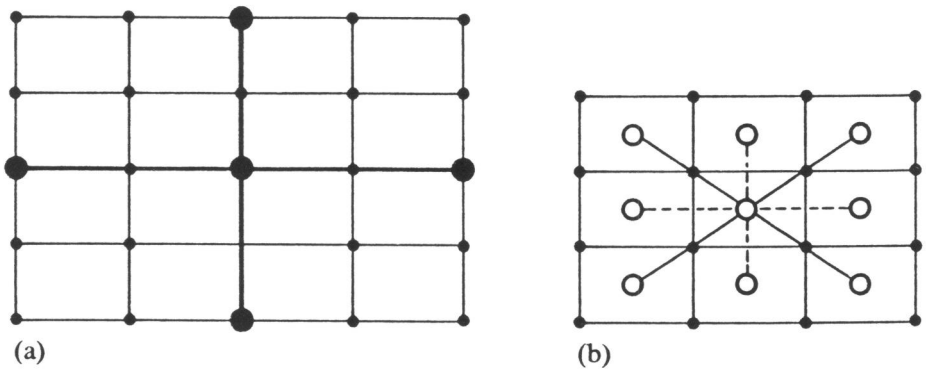


FIG. 3. *Molecule of the operator L, on the ordinary grid* (a) *and on the half-staggered grid* (b).

differ from each other by an arbitrary constant. Because of this decoupling, the ordinary grid and the half-staggered grid are not suitable for the computation of the pressure using a pressure correction scheme. However, the pressure gradient is not affected by this decoupling, and therefore one can still use these grids for the computation of the velocity. In § 3.1 we will present a numerical illustration which clearly favours the staggered grid.

3. Numerical examples. Combined with the ICCG method and an MG method for the solution of the Poisson equation, we have applied our OEH-PC scheme to two Navier-Stokes problems. The first is a simple test problem, of which the exact solution is known [2]. We used this problem to test the accuracy and the order of accuracy of the OEH-PC scheme, in time and in space (see § 3.1). Our second problem is a model problem that comes closer to practical applications. It concerns the flow through a reservoir [12] (see § 3.2). In § 3.3 we will present a comparison, based on our experiences, between the two Poisson solvers.

3.1. Accuracy and order test. In this section we discuss results of the OEH-PC scheme applied to the incompressible Navier-Stokes problem with the exact solution

$$
\begin{aligned}
u(x, y, t) &= -\cos \lambda(x-a) \cdot \sin \lambda(y-a) \cdot e^{-2\lambda^2 t/\mathrm{Re}}, \\
v(x, y, t) &= \sin \lambda(x-a) \cdot \cos \lambda(y-a) \cdot e^{-2\lambda^2 t/\mathrm{Re}}, \\
p(x, y, t) &= -\tfrac{1}{4} \cdot (\cos 2\lambda(x-a) + \cos 2\lambda(y-a)) \cdot e^{-4\lambda^2 t/\mathrm{Re}}.
\end{aligned} \tag{3.1}
$$

In our computation we prescribed Dirichlet boundary conditions for u, v and for the parameters λ, a and Re we took: $\lambda = \pi$, $a = 0$, 0.25 and $\mathrm{Re} = 100$. The velocity field and the isobars for these values of λ, a and Re are displayed in Fig. 4. The computational domain is $\Omega = (0, 1) \times (0, 1)$ and the time-integration interval is $[0, 1]$. While referring to our comments on the artificial pressure boundary condition, we notice that for $a = 0$, $\partial p/\partial n = 0$ on the boundary Γ and for $a = 0.25$, $\partial p/\partial n \neq 0$ and a function of t on Γ. Computations were performed on a staggered grid as well as on an ordinary grid, with grid sizes $h = k = \frac{1}{10}, \frac{1}{20}, \frac{1}{40}$ and stepsizes $\tau = \frac{1}{10}, \frac{1}{20}, \cdots, \frac{1}{160} (\tau \leqq h)$. Since $\max(u(x, y, t)) = 1$ and $\max(v(x, y, t)) = 1$, the critical time step for von Neumann stability for the related convection-diffusion equation is $\tau = h$ (cf. (1.16)).

With the purpose of testing the (order of) accuracy of the OEH-PC scheme in time, as well as in space, we compare the numerical solution to the exact solution (3.1). Let $\varepsilon_f(h, \tau)$ be the l_1-norm of the absolute error in $f (f = u, v \text{ or } p)$ at $t = 1$, obtained for gridsize $h = k$ and stepsize τ. Then the number of significant

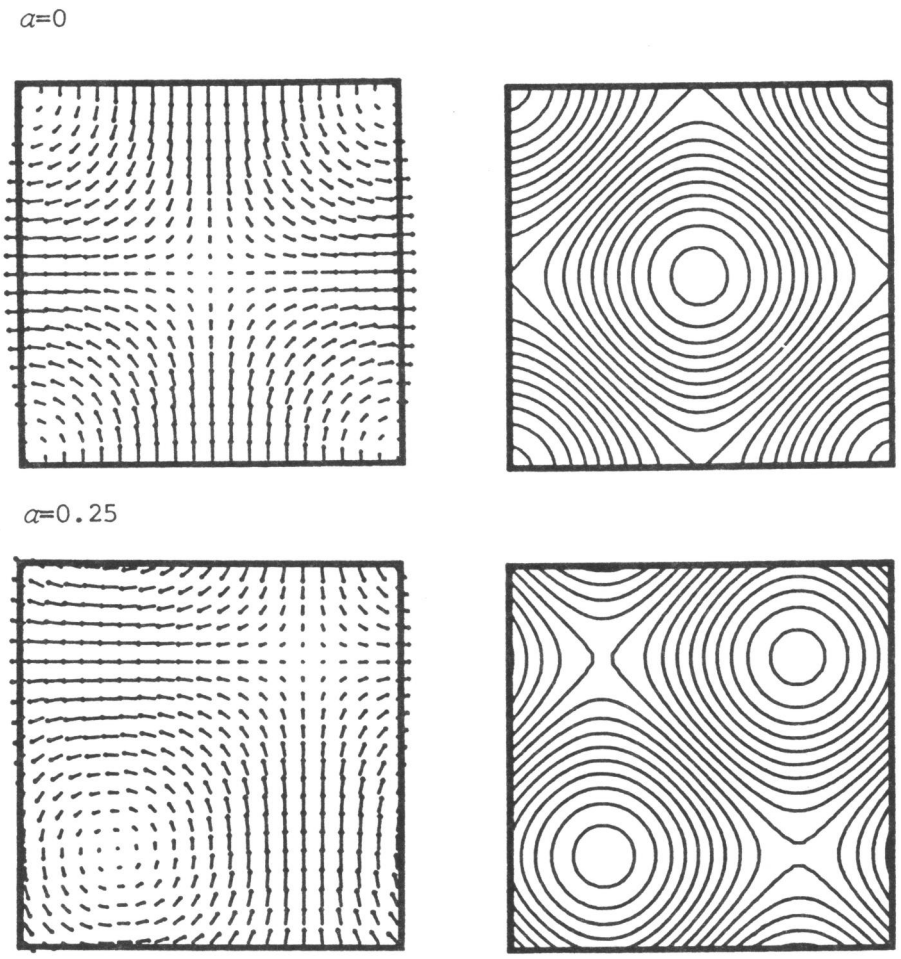


FIG. 4. *Velocity field and isobars.*

digits in f, $\lambda_f(h, \tau)$, is defined as: $\lambda_f(h, \tau) := -\log_{10}(\varepsilon_f(h, \tau))$. Table 1 displays $\lambda_u(h, \tau)$, $\lambda_v(h, \tau)$ and $\lambda_p(h, \tau)$ for the numerical solution computed on the staggered grid. For $a = 0$, when looking along rows (τ fixed, $h \to 0$), one can observe second order behaviour in space ($\log_{10}(4) \approx 0.6$), and when looking along diagonals (τ/h fixed, $\tau \to 0$), one observes second order behaviour in time and space of the OEH-PC scheme. Note that the error in the solution is dominated by the space error. In the same way, one observes that for $a = 0.25$ the velocity components u and v behave second order in space and time, and the pressure p at least first order. Comparing both solutions, we see that the solution for $a = 0.25$ is less accurate than the solution for $a = 0$, this due to the artificial pressure boundary condition for $a = 0.25$ (see § 2.1). However, the OEH-PC scheme clearly does converge for $a = 0.25$.

The same computations were performed on the ordinary grid, the results of which can be found in Table 2 (we only present results for the velocity for $a = 0$). The same conclusions concerning the order of accuracy of the OEH-PC scheme apply to this case. Comparing the results on the ordinary grid and the staggered grid, one sees that the velocity on the staggered grid is approximately ten times more accurate than on the ordinary grid. The reason for this is the inaccurate computation of the pressure (-gradient) on the ordinary grid. This clearly demonstrates that the staggered grid is

TABLE 1
$\lambda_u(h, \tau)$, $\lambda_v(h, \tau)$ *and* $\lambda_p(h, \tau)$ *for the staggered grid.*

$a = 0.0$

	$\lambda_u(h, \tau)$				$\lambda_v(h, \tau)$				$\lambda_p(h, \tau)$		
h^{-1} / τ^{-1}	10	20	40	h^{-1} / τ^{-1}	10	20	40	h^{-1} / τ^{-1}	10	20	40
10	2.53			10	2.47			10	1.93		
20	2.53	3.22		20	2.47	3.16		20	1.91	2.63	
40	2.53	3.23	3.85	40	2.47	3.16	3.79	40	1.91	2.56	3.45
80	2.53	3.23	3.86	80	2.47	3.15	3.79	80	1.91	2.55	3.26
160	2.53	3.23	3.86	160	2.47	3.15	3.79	160	1.91	2.54	3.19

$a = 0.25$

	$\lambda_u(h, \tau)$				$\lambda_v(h, \tau)$				$\lambda_p(h, \tau)$		
h^{-1} / τ^{-1}	10	20	40	h^{-1} / τ^{-1}	10	20	40	h^{-1} / τ^{-1}	10	20	40
10	2.01			10	1.81			10	1.72		
20	2.01	2.79		20	1.81	2.60		20	1.74	2.09	
40	2.01	2.79	3.50	40	1.81	2.60	3.36	40	1.75	2.11	2.62
80	2.01	2.79	3.50	80	1.81	2.60	3.35	80	1.75	2.12	2.64
160	2.01	2.79	3.50	160	1.81	2.60	3.35	160	1.75	2.12	2.65

TABLE 2
$\lambda_u(h, \tau)$ *and* $\lambda_v(h, \tau)$ *for the ordinary grid,* $a = 0$.

	$\lambda_u(h, \tau)$				$\lambda_v(h, \tau)$		
h^{-1} / τ^{-1}	10	20	40	h^{-1} / τ^{-1}	10	20	40
10	1.52			10	1.52		
20	1.52	2.19		20	1.52	2.20	
40	1.52	2.19	2.81	40	1.52	2.20	2.82
80	1.52	2.19	2.81	80	1.52	2.20	2.82
160	1.52	2.19	2.81	160	1.52	2.20	2.82

to be preferred to the ordinary grid, when solving the Navier-Stokes equations in a pressure correction fashion.

In order to test the accuracy of the OEH-PC scheme when considered purely as a time-integrator, it is convenient to compare the numerical solution to the exact solution of the system of ODEs which results after space discretization. As an approximation to this exact solution, we take the numerical solution computed with stepsize $\tau = 1/1280$. The l_1-norm of the absolute time error, $\varepsilon_f^*(h, \tau)$, is defined with respect to this solution, and $\lambda_f^*(h, \tau) := -\log_{10}(\varepsilon_f^*(h, \tau))$ ($f = u$, v or p). We only present results on the staggered grid for $a = 0.25$, which can be found in Table 3. The experiment clearly demonstrates the second order behaviour of the OEH-PC scheme for the computation of the velocity, when considered as an ODE time-integrator. For the computation of the pressure, the OEH-PC scheme is first order in time, though this is not quite clear for $h = 1/20$, $1/40$. This is probably due to the fact that for these values

TABLE 3
$\lambda_u^*(h,\tau)$, $\lambda_v^*(h,\tau)$ *and* $\lambda_p^*(h,\tau)$ *for the staggered grid,* $a = 0.25$.

	$\lambda_u^*(h,\tau)$				$\lambda_v^*(h,\tau)$				$\lambda_p^*(h,\tau)$		
h^{-1} / τ^{-1}	10	20	40	h^{-1} / τ^{-1}	10	20	40	h^{-1} / τ^{-1}	10	20	40
10	3.20			10	3.23			10	2.60		
20	3.90	3.97		20	3.99	4.03		20	3.31	3.27	
40	4.59	4.51	4.68	40	4.59	4.57	4.74	40	3.74	3.79	3.59
80	5.10	5.08	5.15	80	5.19	5.14	5.25	80	4.09	4.26	4.16
160	5.70	5.67	5.68	160	5.79	5.73	5.78	160	4.42	4.70	4.74

the asymptotics still do not hold (τ too large). We emphasize that columnwise the number of digits found correspond to different ODE systems.

Next we discuss briefly the DuFort-Frankel (DFF) deficiency [6], [21]. Consider the convection-diffusion equation (1.15), which models the convective and viscous effects of the Navier-Stokes equations. The OEH scheme for this equation is equivalent to the leapfrog-DFF scheme at the odd points, cf. [21]. Let H_k be the central difference approximation to the first space derivative in the kth direction and μ_k the standard averaging operator in the kth direction, then the leapfrog-DFF scheme for problem (1.15) reads

$$(1+2d\sigma)U_j^{n+2} = (1-2d\sigma)U_j^n - \sum_{k=1}^{d} (c_k H_k - 4\sigma\mu_k)U_j^{n+1}, \tag{3.2}$$

where $\sigma = \varepsilon\tau/h^2$, $c_k = q_k\tau/h$ and h is the constant gridsize in all space directions. By the DFF deficiency we mean that for $\tau, h \to 0$ the solution of scheme (3.2) will converge to the solution of the problem

$$u_t + (\mathbf{q}\cdot\nabla)u = \varepsilon\nabla^2 u - \varepsilon d\left(\frac{\tau}{h}\right)^2 u_{tt}. \tag{3.3}$$

In general, for convergence it is thus necessary that $\tau = o(h)$. Through the equivalence property, the same conclusion is valid for the OEH scheme. The DFF deficiency also exists for the nonlinear Burgers equation, although the equivalence to the leapfrog-DFF scheme cannot be derived in this case. Experiments in [21] showed that the OEH scheme applied to the nonlinear Burgers equation failed to converge for a fixed ratio τ/h when $\tau, h \to 0$. In our example, however, the OEH scheme does not suffer from this deficiency. The reason for this is that the term $\varepsilon d u_{tt}$ is very small, and hence the DFF deficiency is practically absent. In general the DFF deficiency will have some negative influence on the accuracy. Fortunately, there is clear practical evidence (see also [21]) that in most cases this will be of only minor importance.

3.2. Flow through a reservoir. In this section we discuss results of the OEH-PC scheme when used to compute the flow in a reservoir [12] (see Fig. 5). Computations were performed subject to the following initial conditions and boundary conditions:

initial conditions: $u = v = 0$ for $t = 0$
boundary conditions:
no slip: $u = 0$, $v = 0$

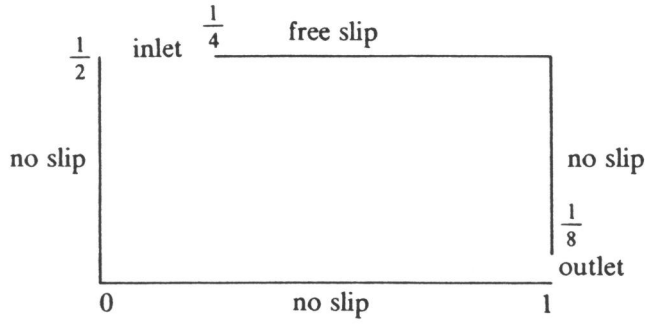


FIG. 5. *The reservoir.*

free slip: $u_y = 0$, $v = 0$
inlet: $u = 0$, $v = -432(x - \frac{1}{4})^2 x(1 - e^{-t})$
outlet: $u = 432(\frac{1}{8} - y)y(1 - e^{-t})$, $v = 0$.

Notice that the boundary conditions satisfy

$$\int_{\partial\Omega} \mathbf{u} \cdot \mathbf{n} ds = \iint_{\Omega} \nabla \cdot \mathbf{u} \, dS = 0,$$

where $\mathbf{n}$ is the unit normal on $\partial\Omega$ (conservation of mass). The outlet boundary condition, which is a Poisseuille profile, is not very realistic, especially not for high Re-numbers since it causes an artificial numerical boundary layer at the outlet. This boundary layer may cause oscillations in the solution in the interior domain [18]. Therefore, we have to look for other outlet boundary conditions with minimal influence on the interior flow field. A very suitable outlet boundary condition is the so-called traction-free boundary condition. This means that there are no viscous normal and tangential stresses at the outlet (cf. [7]), i.e.,

$$\tau_{xx} = -p + \frac{2}{\mathrm{Re}} u_x = 0, \qquad \tau_{xy} = \frac{1}{\mathrm{Re}} (u_y + v_x) = 0. \tag{3.4}$$

However, these boundary conditions do not easily fit in the OEH-PC scheme. Another possibility we adopt is to extend the computational domain with a horizontal pipe connected at the outlet (extended domain). The assumption here is that the flow has fully developed into a Poisseuille flow at the end of the pipe, which is a realistic assumption, provided the pipe is long enough. In our computations we did not bother about the length of the pipe, and took it equal to 1. The horizontal walls of the pipe are no-slip walls.

We have computed the solution for = 100(100)800 on the original domain as well as on the extended domain, on a staggered grid with gridsize $h = k = 1/32$. Time-integration was performed from $t = 0$ to $t = 4$. The time step τ was bounded by the linearized stability restriction $\tau/h \leqq \sqrt{2}/u_{\max}$, where $u_{\max}$ is the (modulus of the) maximum velocity (cf. (1.16)). Consequently we have chosen $\tau = \frac{1}{4}h$ for Re = 100(100)700 and $\tau = \frac{1}{8}h$ for Re = 800, although these values for τ are not the optimal ones. However, especially for increasing Re, we prefer to remain on the safe side in order to prevent nonlinear instabilities. Another reason to be careful is the fact that we use the pressure correction method, the influence of which on stability is not yet fully clear. The Poisson solver we used is the MG algorithm MGD5V (see § 3.3). Figs. 6 and 7 present the velocity and the isobars for, respectively, Re = 100, 500 and

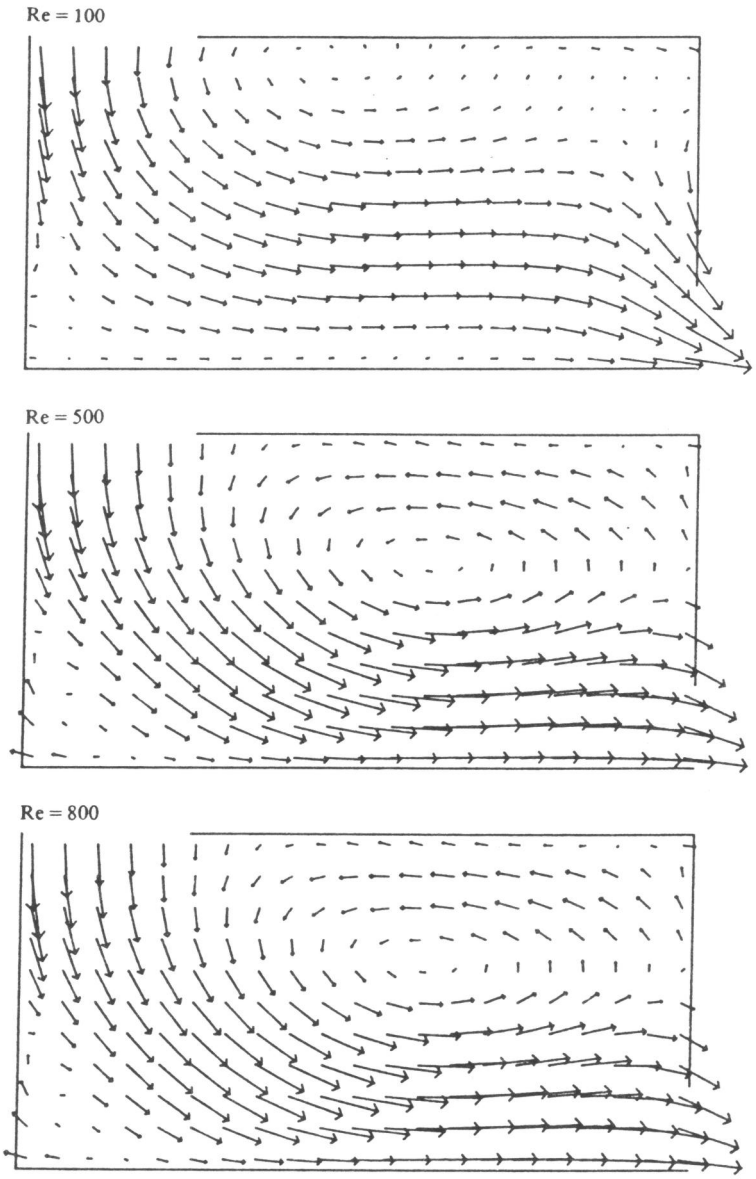


FIG. 6. *Velocity field at* $t = 4$ *for* Re = 100, 500 *and* 800.

800 at $t = 4$ computed on the extended domain (the pipe of the extended domain is not shown in these figures).

From our numerical experiments we can draw the following conclusions. For small Re-numbers ($\mathrm{Re} \leqq 200$), there is hardly any difference between the velocity field and the isobars computed on the original domain and on the extended domain. The velocity fields computed on both domains are almost free of oscillations. However, oscillations do occur in the velocity field for $\mathrm{Re} > 200$. In this case, the velocity field computed on the extended domain is slightly better (smaller oscillations) than the velocity field computed on the original domain. The isobars computed on the original

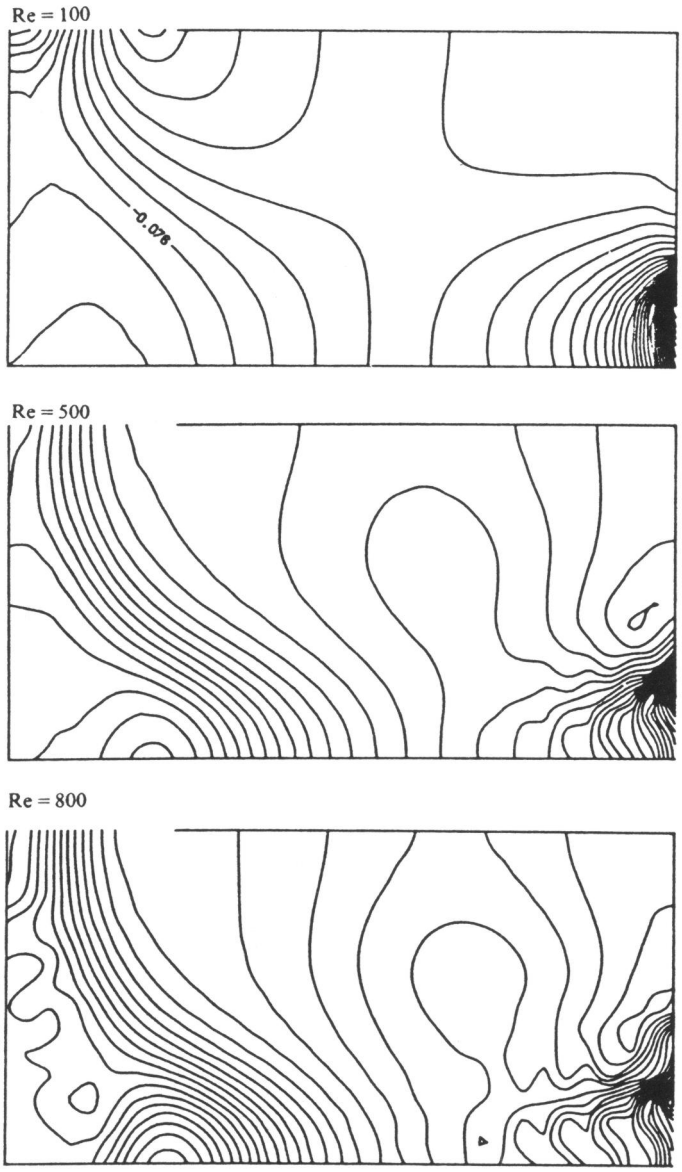


FIG. 7. *Isobars at* $t = 4$ *for* Re = 100, 500 *and* 800.

domain for $Re > 200$ are not correct, whereas the isobars computed on the extended domain are much more realistic.

We borrowed this model problem from van Kan [12]. He computes the flow (without pipe) using a pressure correction Crank–Nicolson ADI scheme (ADI-PC scheme). The outflow boundary conditions he uses are a Poisseuille profile and the traction-free boundary conditions. Comparing his results with ours, we can conclude the following. Our velocity fields are in good agreement with the corresponding ones computed by van Kan. However, his results are more disturbed by oscillations than ours, and this is due to the numerical boundary layer at the outlet occurring in his computations. We note that for the corresponding linear convection-diffusion problem (1.15) the ADI scheme is unconditionally stable, whereas the OEH scheme is only

conditionally stable. In practice the ADI scheme is often only conditionally stable, especially for high Re-numbers, because of the nonlinearity of the convective terms [12]. Nevertheless, the ADI scheme possesses better stability properties than the OEH scheme, so that with respect to stability he normally can take larger time steps. The computational costs per time step for the OEH scheme are less than those for the ADI scheme, since the OEH scheme is in fact an explicit scheme and the ADI scheme requires the solution of a number of tridiagonal linear systems every time step. Therefore, it is not clear which scheme is to be favoured regarding the computational time required. Another point is that extension of the computational domain is rather tedious using an ADI technique, whereas for the OEH scheme this extension is straightforward to implement.

3.3. A comparison between the Poisson solvers. The OEH scheme is a fast scheme per time step. Therefore, in order to construct a fast OEH-PC scheme per time step, one needs a fast Poisson solver. In this section we will compare the ICCG method [13], [14] with the MG method MGD5V [9], [19] we employed for the model problem. This comparison is focussed on the computational time required for both methods.

The storage requirements for both methods are approximately the same, and are substantial compared to the storage requirements for the OEH scheme. With respect to the storage requirements, an excellent candidate to combine with the OEH scheme is the MG method MG00 [3]. Unfortunately, at the time of carrying out this research, MG00 was not available in our computer centre, so we decided to compare ICCG with MGD5V.

The ICCG method is an iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, and hence this method can be used for the computation of the pressure. It is an incomplete decomposition method, combined with the conjugate gradient method (cf. [13] and [14]). We used the ICCG (1, 3) method from [14]. The MG method MDG5V is a sawtooth multigrid iterative process (i.e., one relaxation-sweep after each coarse grid correction) for the solution of linear second order elliptic boundary value problems, cf. [9] and [19]. This multigrid method uses incomplete line LU-decomposition as relaxation method, a 7-point prolongation and restriction, and a Galerkin approximation for the coarse grid matrices. The ICCG (1, 3) process and the MG process were repeated, until the l_2-norm of the residual was less than 10^{-6}.

Using both Poisson solvers, the computations of § 3.1 (for $a = 0$) were repeated on a staggered grid with gridsizes $h = k = 1/8$, $1/16$, $1/32$ and stepsizes $\tau = h^{-1}$, $\frac{1}{2}h^{-1}, \cdots, 1/1024$. Computations were performed on a Cyber 170-750 computer, and all codes were in standard Fortran 77, except the code for the ICCG method which is written in standard Fortran 66. Parameters of interest in our comparison are: the (CPU-) time (in seconds) needed for the OEH scheme (TOEH), the time needed for the ICCG method (TICCG), the time needed for the MG method (TMG), the ratios $\alpha_1 = \text{TICCG/TOEH}$ and $\alpha_2 = \text{TMG/TOEH}$, and the average number of iteration steps (average over the number of time steps) for either the ICCG method or the MG method (ANIT). In Table 4 we present the results for $h^{-1} = k^{-1} = 8, 16, 32$.

From this table, we can draw the following conclusions. For the ICCG method ANIT (and hence α_1) is approximately proportional to $h^{-1} = k^{-1}$, whereas for the MG method ANIT (and hence α_2) is approximately constant. One iteration step of the ICCG method is faster than one iteration step of the MG method, and therefore the ICCG method is faster on coarser grids and the MG method is faster on finer grids. It should be noted that in the ICCG method, the decomposition of the matrix L is

TABLE 4
Comparison between the ICCG *method and the* MG *method.*

ICCG method												
	$h^{-1}=k^{-1}=8$				$h^{-1}=k^{-1}=16$				$h^{-1}=k^{-1}=32$			
τ^{-1}	TOEH	TICCG	α_1	ANIT	TOEH	TICCG	α_1	ANIT	TOEH	TICCG	α_1	ANIT
8	0.035	0.090	2.57	7.00								
16	0.070	0.159	2.27	6.06	0.191	1.026	5.37	11.38				
32	0.131	0.313	2.39	5.91	0.410	1.847	4.50	10.09	1.291	14.329	11.10	21.07
64	0.273	0.555	2.03	5.02	0.784	3.401	4.34	9.14	2.543	25.901	10.19	19.17
128	0.563	0.973	1.73	4.27	1.561	6.144	3.94	8.29	5.093	47.019	9.23	17.21
256	1.070	1.582	1.48	3.16	3.100	9.247	2.98	5.96	10.324	84.165	8.22	15.16
512	2.161	2.956	1.37	2.82	6.234	15.991	2.57	4.87	20.558	143.051	6.96	12.68
1024	4.348	4.819	1.11	2.00	12.566	27.302	2.17	3.96	40.657	201.660	4.96	8.59

MG method												
	$h^{-1}=k^{-1}=8$				$h^{-1}=k^{-1}=16$				$h^{-1}=k^{-1}=32$			
τ^{-1}	TOEH	TMG	α_2	ANIT	TOEH	TMG	α_2	ANIT	TOEH	TMG	α_2	ANIT
8	0.029	0.207	7.14	5.00								
16	0.071	0.389	5.48	4.69	0.197	0.936	4.75	5.06				
32	0.137	0.669	4.88	4.03	0.384	1.785	4.65	4.78	1.274	5.246	4.12	5.00
64	0.279	1.281	4.59	3.77	0.770	3.044	3.95	4.02	2.546	10.049	3.95	4.78
128	0.543	2.099	3.87	3.01	1.543	5.801	3.76	3.81	5.046	17.075	3.38	4.00
256	1.084	3.161	2.92	2.13	3.143	9.588	3.05	3.00	10.099	32.406	3.21	3.77
512	2.154	6.078	2.82	2.00	6.286	18.024	2.87	2.82	19.874	52.385	2.64	3.01
1024	4.448	12.209	2.74	2.00	12.595	27.252	2.16	2.00	40.210	97.593	2.43	2.71

computed at every time step, whereas in the MG method this is done only once. This will not affect our conclusions seriously, since the computational time required for this decomposition is negligible compared to the computational time needed even for a small number of iterations [13]. Therefore, we may conclude that the MG method is to be preferred to the ICCG method. Also observe that ANIT (and hence the computational time per time step) decreases if we take smaller time steps. The obvious reason for this is that the initial guess of the pressure increment Q^n, for which we use Q^n from the previous time step, improves if we take smaller time steps τ. Finally, although the ICCG method and the MG method are generally considered as fast Poisson solvers, they still require a considerable part of the computational time in the OEH-PC scheme. In our test problem this varies from 53–92 percent for the ICCG method and from 68–88 percent for the MG method (see the columns under α_1 or α_2 in Table 4). This clearly demonstrates that it is very important to use a fast Poisson solver for the construction of a fast OEH-PC scheme.

4. Concluding remarks. In this paper we have constructed the OEH-PC scheme, and demonstrated by two examples that it is a feasible scheme for the computation of incompressible fluid flow. The scheme has a few attractive properties. First, the scheme is very simple as it is (almost) explicit. Therefore, extension to arbitrary domains (also three-dimensional) and to nonuniform grids is straightforward. Also the use of standard upwind differencing renders no problem. Second, the scheme is fast per time step, provided we have a fast Poisson solver for the computation of the pressure. Since the scheme is in fact an explicit scheme, it is easy to vectorize for applications on a

supercomputer. Finally, the scheme requires only one array of storage for the computation of the velocity, which is especially advantageous for three-dimensional problems. A drawback of the scheme is the so-called DFF deficiency (see (3.3)), which has in general a negative influence on the accuracy. For many flow problems however, this deficiency will be of only minor importance.

Considered as an ODE time-integration technique, the OEH-PC scheme is second order accurate for the computation of the velocity and first order accurate for the computation of the pressure. Comparing the underlying OEH scheme with the ADI scheme, we note the following. The ADI scheme has in general better stability properties than the OEH scheme. However, the ADI scheme is stepwise more expensive than the OEH scheme, and it would also require more memory. Therefore, we believe that the OEH-PC scheme is a simple alternative to the ADI-PC scheme for many flow problems.

Acknowledgment. The author would like to thank J. van Kan for reading the manuscript.

REFERENCES

[1] T. Cebeci, R. S. Hirsch, H. B. Keller and P. G. Williams, *Studies of numerical methods for the plane Navier-Stokes equations*, Comput. Meth. Appl. Mech. Engrg., 27 (1981), pp. 13-44.

[2] A. J. Chorin, *Numerical solution of the Navier-Stokes equations*, Math. Comp., 22 (1968), pp. 745-762.

[3] H. Foerster and K. Witsch, *Multigrid software for the solution of elliptic problems on rectangular domains*: MG00 (*release* 1), in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, 1981, pp. 427-460.

[4] M. Fortin, R. Peyret and R. Temam, *Résolution numérique des équations de Navier-Stokes pour un fluide incompressible*, J. Méc., 10 (1971), pp. 357-390.

[5] A. R. Gourlay, *Hopscotch: a fast second-order partial differential equation solver*, J. Inst. Maths. Applics., 6 (1970), pp. 375-390.

[6] A. R. Gourlay and G. R. McGuire, *General hopscotch algorithm for the numerical solution of partial differential equations*, J. Inst. Maths. Applics., 7 (1971), pp. 216-227.

[7] P. M. Gresho, R. L. Lee and R. L. Sani, *On the time-dependent solution of the incompressible Navier-Stokes equations in two and three dimensions*, in Recent Advances in Numerical Methods in Fluids, Vol. 1, C. Taylor and R. Morgan, eds., Pineridge Press, Swansea, Wales, 1981, pp. 27-79.

[8] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluids with free surface*, Phys. Fluids, 8 (1965), pp. 2182-2189.

[9] P. W. Hemker and P. M. de Zeeuw, *Some implementations of multigrid linear system solvers*, in Multigrid Methods For Integral And Differential Equations, The Institute of Mathematics and Its Applications Conference Series, D. J. Paddon and H. Holstein, eds., Oxford University Press, New York, pp. 85-116.

[10] A. C. Hindmarsh, P. M. Gresho and D. F. Griffith, *The stability of explicit Euler time-integration for certain finite difference approximations of the multi-dimensional advection-diffusion equation*, Internat. J. Numer. Meth. Fluids, 4 (1984), pp. 853-897.

[11] P. J. van der Houwen and J. G. Verwer, *One-step splitting methods for semi-discrete parabolic equations*, Computing, 22 (1979), pp. 291-309.

[12] J. van Kan, *A second-order pressure correction method for viscous incompressible flow*, this Journal, 7 (1986), pp. 870-891.

[13] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148-162.

[14] ———, *Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems*, J. Comput. Phys., 44 (1981), pp. 134-155.

[15] R. Peyret and T. D. Taylor, *Computational Methods for Fluid Flow*, Springer-Verlag, New York, 1983.

[16] S. M. Scala and P. Gordon, *Reflection of a shock wave at a surface*, Phys. Fluids, 9 (1966), pp. 1158-1166.

[17] ———, *Solution of the time-dependent Navier-Stokes equations for the flow around a circular cylinder*, AIAA J., 6 (1968), pp. 815-822.

[18] A. SEGAL, *Aspects of numerical methods for elliptic singular perturbation problems*, this Journal, 3 (1982), pp. 327-349.

[19] P. SONNEVELD, P. WESSELING AND P. M. DE ZEEUW, *Multigrid and conjugate gradient methods as convergence acceleration techniques*, in Multigrid Methods For Integral And Differential Equations, The Institute Of Mathematics And Its Applications Conference Series, D. J. Paddon and H. Holstein, eds., Oxford University Press, New York, 1985, pp. 117-167.

[20] R. TEMAM, *Navier-Stokes Equations*, North-Holland, Amsterdam, 1977.

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

The Odd-Even Hopscotch Pressure Correction Scheme for the Computation of Free Convection in a Square Cavity

J.H.M. ten Thije Boonkkamp
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

The odd-even hopscotch scheme is a time-integration technique applicable to large classes of time-dependent partial differential equations. In this paper the scheme is applied to the incompressible Navier-Stokes equations in Boussinesq approximation. The odd-even hopscotch scheme is combined with the pressure correction method in order to decouple the computation of the pressure from that of the velocity and temperature. The resulting scheme is called the odd-even hopscotch pressure correction scheme, and when combined with a suitable space discretization this scheme proves to be efficient regarding computing time and storage requirements. The scheme is used for the computation of steady and periodic free convection in a square cavity.

1980 Mathematics Subject Classification: 65M20, 65N05, 76D05
Keywords & Phrases: free convection, Navier-Stokes equations in Boussinesq approximation, odd-even hopscotch method, pressure correction method.

1. Introduction

In this paper we consider the free convection of a fluid in a square cavity, i.e. the flow in a cavity caused by a temperature gradient. For this problem we use the primitive variable formulation (velocity, pressure, temperature) and the governing equations are the Navier-Stokes equations in Boussinesq approximation [3].

We consider the fully transient Navier-Stokes equations in Boussinesq approximation. For the time-integration of these equations one can choose an explicit scheme, an implicit scheme or a combination of both. Explicit schemes are very cheap (per time step), but stability of these schemes is subject to severe time step restrictions. Implicit schemes are usually unconditionally stable, but are much more expensive since they require the solution of large sets of algebraic equations at each time step. The time-integration technique we use is the odd-even hopscotch (OEH) method, which is a combination of the explicit and implicit Euler rule [5, 6, 10]. When combined with a suitable space

discretization, the OEH scheme is almost as cheap (stepwise) as the explicit Euler rule, but has much better stability properties.

In order to decouple the computation of the pressure from the computation of the velocity and the temperature, the OEH scheme is combined with the pressure correction method [2, 11]. The scheme thus obtained is called the odd-even hopscotch pressure correction (OEH-PC) scheme. The OEH-PC scheme we will describe in this paper is an extension of the OEH-PC scheme for the incompressible Navier-Stokes equations described in [16].

The purpose of this paper is to give a description of the OEH-PC scheme for the free convection problem and to demonstrate that it is a feasible time-integration technique for this problem. To that end we apply the OEH-PC scheme to a steady free convection problem and a periodic free convection problem.

The contents of the paper is the following. In Section 2 a description of the free convection problem is given. The OEH-PC scheme is introduced in Section 3 and the space discretization combined with this scheme is given in Section 4. Section 5 is devoted to the pressure computation and Section 6 gives a survey of the stability results for the OEH scheme based on linear stability theory. In Section 7 the computational results are presented for both the steady and periodic free convection problem. Finally, some conclusions are formulated in Section 8.

2. Physical Problem and Equations

Consider the free convection of a viscous fluid in a two-dimensional square cavity of width l as shown in Fig. 1. [13, 18]. The direction of the gravitational acceleration g is along the negative y-axis and the physical boundary conditions are:

- no slip conditions for the velocity on all 4 walls
- constant temperatures T_1 and $T_2(T_1 > T_2)$ on respectively the left and right vertical walls
- adiabatic horizontal walls or a linearly varying temperature at the horizontal walls.

The equations governing the fluid motion, in the Boussinesq approximation [3], are:
equation of continuity

$$u_x + v_y = 0 \tag{2.1}$$

equations of motion (Navier-Stokes equations)

$$u_t + uu_x + vu_y = -\frac{1}{\rho_0} p_x + \nu \nabla^2 u \tag{2.2a}$$

$$v_t + uv_x + vv_y = -\frac{1}{\rho_0} p_y + \nu \nabla^2 v - g(1 - \alpha(T - T_0)) \tag{2.2b}$$

temperature equation

$$T_t + uT_x + vT_y = a\nabla^2 T. \tag{2.3}$$

In the above equations u and v are the components of the velocity in x- and y-direction, respectively, p is the pressure and T the temperature. The unknown quantities u,v,p and T are all functions of x,y and t, whereas ρ_0 is the (constant) density at some properly chosen mean temperature T_0. In this problem $T_0 = \frac{1}{2}(T_1 + T_2)$. The coefficients ν,α and a are respectively the kinematic viscosity, the coefficient of volume expansion and the coefficient of thermal conductivity.

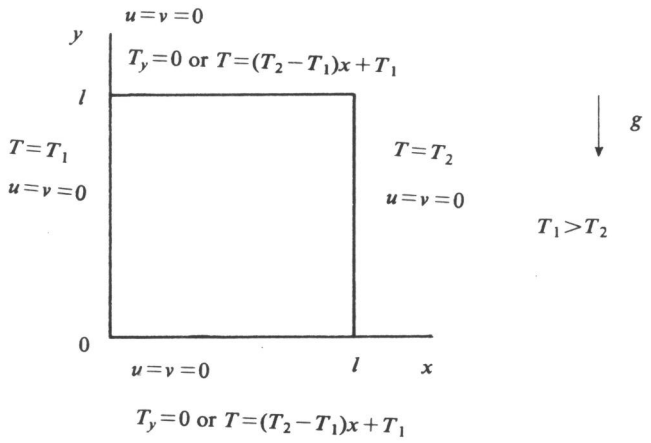


Fig.1. Geometry and boundary conditions.

In order to make the above equations dimensionless, we introduce the following (dimensionless) quantities:

$$x' = \frac{x}{l},\ y' = \frac{y}{l},\ t' = \frac{a}{l^2}t,\ u' = \frac{l}{a}u,\ v' = \frac{l}{a}v, \tag{2.4}$$

$$p' = \frac{l^2}{\rho_0 a^2}(p - p_{hydr}),\ T' = \frac{T - T_2}{T_1 - T_2}.$$

Note that p' is the dimensionless deviation from the hydrostatic pressure $p_{hydr} = const. - \rho_0 g y$. After substitution of these variables, the governing equations take the following form (after dropping the primes)

$$u_x + v_y = 0 \tag{2.5}$$

$$u_t + uu_x + vu_y = -p_x + Pr\nabla^2 u \tag{2.6a}$$

$$v_t + uv_x + vv_y = -p_y + Pr\nabla^2 v + RaPr(T - \tfrac{1}{2}) \tag{2.6b}$$

$$T_t + uT_x + vT_y = \nabla^2 T, \tag{2.7}$$

where Pr and Ra are the Prandtl number and the Rayleigh number respectively, defined by

$$Pr = \frac{\nu}{a},\ Ra = \frac{g\alpha l^3(T_1 - T_2)}{\nu a}. \tag{2.8}$$

In the dimensionless variables the computational space domain is $\Omega = [0,1]\times[0,1]$ and the boundary conditions we consider read

$$x = 0\colon u = v = 0,\ T = 1 \tag{2.9}$$

$$x = 1\colon u = v = 0,\ T = 0$$

$$y = 0,1\colon u = v = 0,\ T_y = 0 \text{ or } T = -x + 1.$$

Initial conditions will be specified later.

In what follows we will consider the dimensionless equations and refer to them, for convenience, as the Navier-Stokes equations in Boussinesq approximation. Finally we note that the equations (2.6a)-(2.7) can be rewritten as

$$u_t + (u^2)_x + (uv)_y = -p_x + Pr\nabla^2 u \tag{2.6a'}$$

$$v_t + (uv)_x + (v^2)_y = -p_y + Pr\nabla^2 v + RaPr(T - \tfrac{1}{2}) \tag{2.6b'}$$

$$T_t+(uT)_x+(vT)_y = \nabla^2 T. \tag{2.7'}$$

The equations (2.6a)-(2.7) are written in the so-called convective form and the equations (2.6a')-(2.7') in the conservative form.

3. The Odd-Even Hopscotch Pressure Correction Scheme

In this section we consider the odd-even hopscotch (OEH) scheme for the time-integration of the Navier-Stokes equations in Boussinesq approximation. For a detailed discussion of the OEH scheme, the reader is referred to [5,6,10,17]. The OEH scheme is combined with the pressure correction method, which is a predictor-corrector method for the decoupling of the pressure computation. The resulting scheme will be referred to as the odd-even hopscotch pressure correction (OEH-PC) scheme, and is an extension of the scheme given in [16]. A description of the pressure correction method can be found in [2, 11].

The Navier-Stokes equations in Boussinesq approximation in d space dimensions ($d=2$ or $d=3$) can in general be written as:

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u},T)-\nabla p \tag{3.1}$$

$$T_t = h(\mathbf{u},T) \tag{3.2}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{3.3}$$

where $\mathbf{u}$ is the velocity, p the pressure and T the temperature. For the time being, the exact form of $\mathbf{f}(\mathbf{u},T)$ and $h(\mathbf{u},T)$ (convective/conservative) is not important. The partial differential equations (PDEs) (3.1)-(3.3) are defined on a connected space domain Ω with boundary Γ, on which conditions for the velocity $\mathbf{u}$ and the temperature T are specified. Notice that the boundary conditions for $\mathbf{u}$ must satisfy

$$\oint_\Gamma \mathbf{u}\cdot\mathbf{n}ds = \iint_\Omega \nabla\cdot\mathbf{u}dS = 0, \tag{3.4}$$

where $\mathbf{n}$ is the unit outward normal on Γ (conservation of mass).

We present the OEH-PC scheme for the PDEs (3.1)-(3.3) following the method of lines approach [10]. Thus suppose first that by a suitable finite difference space discretization the PDEs (3.1)-(3.3) are replaced by the following set of ordinary differential equations (ODEs) and algebraic equations

$$\dot{\mathbf{U}} = \mathbf{F}(\mathbf{U},T)-GP \tag{3.5}$$

$$\dot{T} = H(\mathbf{U},T) \tag{3.6}$$

$$D\mathbf{U} = B. \tag{3.7}$$

In (3.5)-(3.7) the variables $\mathbf{U},P$ and T are grid functions defined on a space grid covering Ω, and $\mathbf{F}(\mathbf{U},T)$ and $H(\mathbf{U},T)$ are the finite difference replacements of $\mathbf{f}(\mathbf{u},T)$ and $h(\mathbf{u},T)$, respectively. The operators G and D are the finite difference replacements of the gradient- and divergence-operators and B is a term containing boundary values for the velocity $\mathbf{u}$.

First consider the ODEs (3.5), (3.6) and suppose for the time being that GP is a known forcing term. Let $\mathbf{x}_j$ be a gridpoint corresponding to the multi-index $j = (j_1,...,j_d)$ and let $\mathbf{U}_j^n$ denote the approximation to $\mathbf{u}(\mathbf{x}_j,t_n)$ (and likewise for $P,T,\mathbf{F}$ and H), then the OEH scheme for (3.5), (3.6) reads [6]

$$\mathbf{U}_j^{n+1} - \tau\theta_j^{n+1}(\mathbf{F}_j(\mathbf{U}^{n+1},T^{n+1}) - (GP)_j^{n+1}) = \mathbf{U}_j^n + \tau\theta_j^n(\mathbf{F}_j(\mathbf{U}^n,T^n) - (GP)_j^n) \tag{3.8a}$$

$$T_j^{n+1} - \tau\theta_j^{n+1}H_j(\mathbf{U}^{n+1},T^{n+1}) = T_j^n + \tau\theta_j^n H_j(\mathbf{U}^n,T^n). \tag{3.8b}$$

Here $\tau = t_{n+1} - t_n$ is the time step and θ_j^n is the so-called odd-even function defined by

$$\theta_j^n = \begin{cases} 1 \text{ if } n + \sum_i j_i \text{ is odd (odd points)} \\ 0 \text{ if } n + \sum_i j_i \text{ is even (even points).} \end{cases} \tag{3.9}$$

Note that in the odd points the OEH scheme reduces to the forward Euler rule and in the even points to the backward Eules rule.

An alternative form of (3.8a), (3.8b) is

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \tau\mathbf{F}_O(\mathbf{U}^n,T^n) + \tau\mathbf{F}_E(\mathbf{U}^{n+1},T^{n+1}) - \tau(GP^n)_O - \tau(GP^{n+1})_E \tag{3.10a}$$

$$T^{n+1} = T^n + \tau H_O(\mathbf{U}^n,T^n) + \tau H_E(\mathbf{U}^{n+1},T^{n+1}), \tag{3.10b}$$

where $\mathbf{F}_O$ is the restriction of $\mathbf{F}$ to the odd points (etc.). Note that $\mathbf{F}_O + \mathbf{F}_E = \mathbf{F}$ and $H_O + H_E = H$. We shall use this formulation in the remainder of the section. It is customary to write down two successive steps of (3.10a), (3.10b) with stepsize $\tau/2$, where the order of implicit and explicit calculations alternate [17]

$$\tilde{\mathbf{U}} = \mathbf{U}^n + \tfrac{1}{2}\tau\mathbf{F}_O(\mathbf{U}^n,T^n) + \tfrac{1}{2}\tau\mathbf{F}_E(\tilde{\mathbf{U}},\tilde{T}) - \tfrac{1}{2}\tau GP^n \tag{3.11a}$$

$$\tilde{T} = T^n + \tfrac{1}{2}\tau H_O(\mathbf{U}^n,T^n) + \tfrac{1}{2}\tau H_E(\tilde{\mathbf{U}},\tilde{T}) \tag{3.11b}$$

$$\mathbf{U}^{n+1} = \tilde{\mathbf{U}} + \tfrac{1}{2}\tau \mathbf{F}_E(\tilde{\mathbf{U}}, \tilde{T}) + \tfrac{1}{2}\tau \mathbf{F}_O(\mathbf{U}^{n+1}, T^{n+1}) - \tfrac{1}{2}\tau G P^{n+1} \tag{3.11c}$$

$$T^{n+1} = \tilde{T} + \tfrac{1}{2}\tau H_E(\tilde{\mathbf{U}}, \tilde{T}) + \tfrac{1}{2}\tau H_O(\mathbf{U}^{n+1}, T^{n+1}). \tag{3.11d}$$

This is a second order accurate integration formula for the numerical integration of the ODE systems (3.5), (3.6) using stepsize τ. The variables $\tilde{\mathbf{U}}$ and $\tilde{T}$ can be interpreted as results from the intermediate time level $(n+\frac{1}{2})\tau$, like in a Runge-Kutta formula. Note that in (3.11a) P is set at time level $t_n = n\tau$ and in (3.11c) at time level $t_{n+1} = (n+1)\tau$. An alternative for maintaining second order is to compute P at time level $(n+1/2)$ in both stages. However the choice made in (3.11a), (3.11c) is better adapted to the pressure correction approach.

Consider (3.11a)-(3.11d) coupled with the (time discretized) set of algebraic equations

$$D\mathbf{U}^{n+1} = B^{n+1}. \tag{3.11e}$$

The computation of $\mathbf{U}^{n+1}, P^{n+1}$ and T^{n+1} requires the simultaneous solution of (3.11c)-(3.11e). In order to avoid this, we follow the known pressure correction approach [11] in which the computation of P^{n+1} is decoupled in the predictor-corrector fashion. Substitution of P^n for P^{n+1} in (3.11c) defines the predicted velocity $\tilde{\tilde{\mathbf{U}}}$ and the predicted temperature $\tilde{\tilde{T}}$. The corrected velocity, pressure and temperature (which we hereafter also denote by $\mathbf{U}^{n+1}, P^{n+1}$ and T^{n+1} and hence should not be mixed up with the approximations in (3.11c), (3.11d)) are then defined by replacing $\mathbf{F}_O(\mathbf{U}^{n+1}, T^{n+1})$ in (3.11c) and $H_O(\mathbf{U}^{n+1}, T^{n+1})$ in (3.11d) by respectively $\mathbf{F}_O(\tilde{\tilde{\mathbf{U}}}, \tilde{\tilde{T}})$ and $H_O(\tilde{\tilde{\mathbf{U}}}, \tilde{\tilde{T}})$:

$$\mathbf{U}^{n+1} = \tilde{\mathbf{U}} + \tfrac{1}{2}\tau \mathbf{F}_E(\tilde{\mathbf{U}}, \tilde{T}) + \tfrac{1}{2}\tau \mathbf{F}_O(\tilde{\tilde{\mathbf{U}}}, \tilde{\tilde{T}}) - \tfrac{1}{2}\tau G P^{n+1} \tag{3.12a}$$

$$T^{n+1} = \tilde{T} + \tfrac{1}{2}\tau H_E(\tilde{\mathbf{U}}, \tilde{T}) + \tfrac{1}{2}\tau H_O(\tilde{\tilde{\mathbf{U}}}, \tilde{\tilde{T}}), \tag{3.12b}$$

together with the discrete continuity equation (3.11e). From (3.12a), (3.12b) and the modified equations (3.11c), (3.11d) one can easily see that

$$\mathbf{U}^{n+1} - \tilde{\tilde{\mathbf{U}}} = -\tfrac{1}{2}\tau G Q^n, \; Q^n = P^{n+1} - P^n \tag{3.13a}$$

$$T^{n+1} - \tilde{\tilde{T}} = 0. \tag{3.13b}$$

The trick of the pressure correction method is now to multiply (3.13a) by D and to write, using (3.11e),

$$LQ^n = \frac{2}{\tau}(D\tilde{\tilde{\mathbf{U}}} - B^{n+1}), L = DG. \tag{3.14}$$

Since $L = DG$ is a discretization of the Laplace operator $\nabla\cdot(\nabla)$, the correction Q^n for the pressure can be obtained by applying a Poisson solver. Once Q^n is known, the new velocity can be directly determined from (3.13a).

To sum up, the OEH-PC scheme for the semi-discrete system (3.5)-(3.7), reads

$$\tilde{\mathbf{U}} = \mathbf{U}^n + \tfrac{1}{2}\tau\mathbf{F}_O(\mathbf{U}^n, T^n) + \tfrac{1}{2}\tau\mathbf{F}_E(\tilde{\mathbf{U}}, \tilde{T}) - \tfrac{1}{2}\tau GP^n \tag{3.15a}$$

$$\tilde{T} = T^n + \tfrac{1}{2}\tau H_O(\mathbf{U}^n, T^n) + \tfrac{1}{2}\tau H_E(\tilde{\mathbf{U}}, \tilde{T}) \tag{3.15b}$$

$$\tilde{\tilde{\mathbf{U}}} = \tilde{\mathbf{U}} + \tfrac{1}{2}\tau\mathbf{F}_E(\tilde{\mathbf{U}}, \tilde{T}) + \tfrac{1}{2}\tau\mathbf{F}_O(\tilde{\tilde{\mathbf{U}}}, T^{n+1}) - \tfrac{1}{2}\tau GP^n \tag{3.15c}$$

$$T^{n+1} = \tilde{T} + \tfrac{1}{2}\tau H_E(\tilde{\mathbf{U}}, \tilde{T}) + \tfrac{1}{2}\tau H_O(\tilde{\tilde{\mathbf{U}}}, T^{n+1}) \tag{3.15d}$$

$$LQ^n = \frac{2}{\tau}(D\tilde{\tilde{\mathbf{U}}} - B^{n+1}), P^{n+1} = P^n + Q^n \tag{3.15e}$$

$$\mathbf{U}^{n+1} = \tilde{\tilde{\mathbf{U}}} - \tfrac{1}{2}\tau GQ^n. \tag{3.15f}$$

When combined with a suitable space discretization, the OEH-PC scheme possesses various advantageous features. We shall discuss this in greater detail in the next section for symmetric finite differences on a staggered grid.

We conclude this section with two remarks. Firstly, the second stage (3.15c), (3.15d) can be economized using its equivalent fast form (cf. [5, 6])

$$\tilde{\tilde{\mathbf{U}}}_E = 2\tilde{\mathbf{U}}_E - \mathbf{U}_E^n,\ T_E^{n+1} = 2\tilde{T}_E - T_E^n \tag{3.16a}$$

$$\tilde{\tilde{\mathbf{U}}}_O = \tilde{\mathbf{U}}_O + \tfrac{1}{2}\tau\mathbf{F}_O(\tilde{\tilde{\mathbf{U}}},\ T^{n+1}) - \tfrac{1}{2}\tau(GP^n)_O,\ T_O^{n+1} = \tilde{T}_O + \tfrac{1}{2}\tau H_O(\tilde{\tilde{\mathbf{U}}}, T^{n+1}). \tag{3.16b}$$

Our implementation is based on this fast form. Secondly, in the derivation of scheme (3.15) no use has been made of the particular definition of $\mathbf{F}_O, \mathbf{F}_E, H_O$ and H_E, except that $\mathbf{F}_O + \mathbf{F}_E = \mathbf{F}$ and $H_O + H_E = H$. Consequently, pressure correction schemes using other splittings of $\mathbf{F}$ and H, such as ADI, can also be described by (3.15).

4. Space Discretization

Consider the two-dimensional Navier-Stokes equations in Boussinesq approximation (see Section 2)

$$u_t = f_1(u,v,T) - p_x, \text{ with } f_1(u,v,T) = -(u^2)_x - (uv)_y + Pr(u_{xx} + u_{yy}) \tag{4.1a}$$

$$v_t = f_2(u,v,T) - p_y, \text{ with } f_2(u,v,T) = -(uv)_x - (v^2)_y + Pr(v_{xx} + v_{yy}) + RaPr(T - \tfrac{1}{2}) \tag{4.1b}$$

$$T_t = h(u,v,T), \text{ with } h(u,v,T) = -uT_x - vT_y + T_{xx} + T_{yy} \tag{4.2}$$

$$u_x + v_y = 0. \tag{4.3}$$

Note that the equations of motion are written in conservative form while the temperature equation is written in convective form. The reason for this will become clear later. Boundary conditions for the velocity and the temperature are specified. There are no pressure boundary conditions available for (4.1)-(4.3). In [7] it is shown that the semi-discrete system (3.5)-(3.7) combined with the central difference space discretization on a staggered grid, to be discussed below, automatically involves pressure boundary conditions. Furthermore, in [2] it is shown that for the pressure correction method no pressure boundary conditions are needed. This relieves us from specifying extra numerical pressure boundary conditions.

For the space discretization we use the staggered grid of Fig.2. (see also [12,16]). The application of standard, second order central differences converts (4.1a)-(4.2) into (Cf.(3.5), (3.6))

$$\dot{U}_{ij} = F_{1,ij}(U,V,T) - d_x P_{ij} \quad i = 1(1)N-1,\ j = 1(1)M \ (\textit{interior } \times - \textit{points}) \tag{4.4.a}$$

$$\dot{V}_{ij} = F_{2,ij}(U,V,T) - d_y P_{ij} \quad i = 1(1)N,\ j = 1(1)M-1 \ (\textit{interior } \circ - \textit{points}) \tag{4.4b}$$

$$\dot{T}_{ij} = H_{ij}(U,V,T) \quad i = 1(1)N-1,\ j = 1(1)M-1 \ (\textit{interior } \star - \textit{points}) \tag{4.5}$$

where

$$F_{1,ij}(U,V,T) = -\frac{1}{2h}(U_{i+1,j}^2 - U_{i-1,j}^2) - \frac{1}{2k}(U_{i,j+1}\overline{V}_{i,j+1} - U_{i,j-1}\overline{V}_{i,j-1}) + \tag{4.6a}$$

$$\frac{Pr}{h^2}(U_{i+1,j} - 2U_{ij} + U_{i-1,j}) + \frac{Pr}{k^2}(U_{i,j+1} - 2U_{ij} + U_{i,j-1})$$

$$F_{2,ij}(U,V,T) = \frac{-1}{2h}(\overline{U}_{i+1,j}V_{i+1,j} - \overline{U}_{i-1,j}V_{i-1,j}) - \frac{1}{2k}(V_{i,j+1}^2 - V_{i,j-1}^2) + \tag{4.6b}$$

$$\frac{Pr}{h^2}(V_{i+1,j} - 2V_{ij} + V_{i-1,j}) + \frac{Pr}{k^2}(V_{i,j+1} - 2V_{ij} + V_{i,j-1}) + RaPr(\overline{T}_{ij} - \tfrac{1}{2})$$

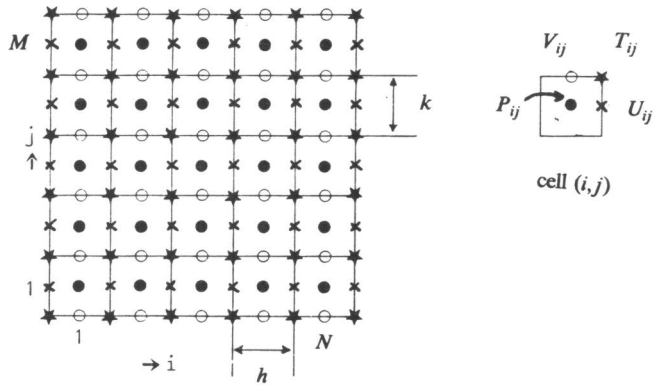


Fig.2. The staggered grid.

$$H_{ij}(U,V,T) = \frac{-1}{2h}\overline{U}_{ij}(T_{i+1,j}-T_{i-1,j})-\frac{1}{2k}\overline{V}_{ij}(T_{i,j+1}-T_{i,j-1}) + \tag{4.6c}$$

$$\frac{1}{h^2}(T_{i+1,j}-2T_{ij}+T_{i-1,j})+\frac{1}{k^2}(T_{i,j+1}-2T_{ij}+T_{i,j-1})$$

$$d_xP_{ij} = \frac{1}{h}(P_{i+1,j}-P_{ij}) \tag{4.6d}$$

$$d_yP_{ij} = \frac{1}{k}(P_{i,j+1}-P_{ij}). \tag{4.6e}$$

In (4.6a) $\overline{V}_{ij}$ represents an approximation to V in the $\times$-points (points where U is defined); likewise $\overline{U}_{ij}$ and $\overline{T}_{ij}$ represent approximations to U and T in the O-points. The values of $\overline{U}_{ij}, \overline{V}_{ij}$ and $\overline{T}_{ij}$ are determined by averaging over neighbouring values of respectively U_{ij}, V_{ij} and T_{ij} in such a way that the odd-even coupling between the variables is preserved. This means that the variables in an odd cell are only coupled with variables in even cells and vice versa. This leads to

$$\overline{U}_{ij} = \tfrac{1}{2}(U_{ij}+U_{i-1,j+1}), \overline{V}_{ij} = \tfrac{1}{2}(V_{ij}+V_{i+1,j-1}), \overline{T}_{ij} = \tfrac{1}{2}(T_{ij}+T_{i-1,j}). \tag{4.7}$$

The same argument leads to the definitions

$$\bar{\bar{U}}_{ij} = \tfrac{1}{2}(U_{ij}+U_{i,j+1}), \bar{\bar{V}}_{ij} = \tfrac{1}{2}(V_{ij}+V_{i+1,j}), \tag{4.8}$$

where $\bar{\bar{U}}_{ij}$ and $\bar{\bar{V}}_{ij}$ are approximations to U and V in the $\star$-points. In case we use the convective form of the equations of motion or the conservative form of the temperature equation, one can see that the odd-even coupling between the variables is lost. This explains our choice of writing the equations of motion in conservative form and the temperature equation in convective form. Observe that the gradient operator G (Cf.(3.5)) is defined by $GP_{ij} = (d_x P_{ij}, d_y P_{ij})^T$.

Concerning the boundary conditions for the velocity we note the following. Consider e.g. equation (4.1a) in the $\times$-points $(i, 1)(i = 1(1)N-1)$. Discretization of the derivatives $(uv)_y$ and u_{yy} would require values outside the computational domain. Therefore we replace the central difference approximations to $(uv)_y$ and u_{yy} by the following noncentered first order differences [12], which preserve the odd-even coupling between the variables

$$((uv)_y)_{i1} = \frac{2}{3k}(U_{i2}\bar{V}_{i2} - u(ih, 0)v(ih, 0)) \tag{4.9a}$$

$$(u_{yy})_{i1} = \frac{4}{3k^2}(U_{i2} - 3U_{i1} + 2u(ih, 0)). \tag{4.9b}$$

Second order non-centered approximations to $(uv)_y$ and u_{yy} would destroy the odd-even coupling.

Space discretization of equation (4.3) in all $\cdot$-points (using central differences) yields

$$(D\mathbf{U})_{ij} := \frac{1}{h}(U_{ij} - U_{i-1,j} + \beta(V_{ij} - V_{i,j-1})) = 0, \tag{4.10}$$

where $\beta = h/k$. One should note that boundary values for U or V occurring in (4.10) are written in the right hand side B (Cf. (3.7)). For example for $j = 1$, equation (4.3) is discretized as

$$(D\mathbf{U})_{i1} := \frac{1}{h}(U_{i1} - U_{i-1,1} + \beta V_{i1}) = B_{i1} = \frac{1}{k}V_{i0}. \tag{4.10'}$$

Having defined the operators G and D, one can easily deduce the following expression for the operator L

$$(LQ)_{ij} = D(GQ)_{ij} = \frac{1}{h}(d_x Q_{ij} - d_x Q_{i-1,j} + \beta(d_y Q_{ij} - d_y Q_{i,j-1})) = \tag{4.11}$$

$$\frac{1}{h^2}(\beta^2 Q_{i,j-1} + Q_{i-1,j} - (2+2\beta^2)Q_{ij} + Q_{i+1,j} + \beta^2 Q_{i,j+1}),$$

which is the standard 5-point molecule for the Laplace operator. Near a boundary (4.11) takes a different form, because of the different definition of the operator D. For example, for $j = 1$ one finds

$$(LQ)_{i1} = D(GQ)_{i1} = \frac{1}{h}(d_x Q_{i1} - d_x Q_{i-1,1} + \beta d_y Q_{i1}) = \tag{4.11'}$$

$$\frac{1}{h^2}(Q_{i-1,1} - (2+\beta^2)Q_{i1} + Q_{i+1,1} + \beta^2 Q_{i2}).$$

Now consider equation (3.15e) at the ·-points $(i, 1)$ $(i = 1(1)N)$. Using (4.10), (4.10′), (4.11) and (4.11′) it is easy to see that $\frac{1}{k}(Q_{i0} - Q_{i1}) = \frac{2}{\tau}(V_{i0}^{n+1} - \tilde{\tilde{V}}_{i0}) = 0$, which is the (central difference) approximation to $\frac{\partial Q}{\partial n}((i-\frac{1}{2})h, 0) = 0$, where n is the outward unit normal on $x = 0$. Hence we see that a Neumann condition for the pressure increment is automatically involved in the scheme (see also [2]).

We conclude this section with a few remarks. The essential feature of the OEH scheme is the alternating use of the forward and backward Euler rule. Consider the equations (3.15a)-(3.15d) of the OEH-PC scheme (3.15). The order of computation is:

$$\tilde{\mathbf{U}}_O = \mathbf{U}_O^n + \tfrac{1}{2}\tau \mathbf{F}_O(\mathbf{U}^n, T^n) - \tfrac{1}{2}\tau(GP^n)_O \tag{4.12a}$$

$$\tilde{T}_O = T_O^n + \tfrac{1}{2}\tau H_O(\mathbf{U}^n, T^n) \tag{4.12b}$$

$$\tilde{\mathbf{U}}_E = \mathbf{U}_E^n + \tfrac{1}{2}\tau \mathbf{F}_E(\tilde{\mathbf{U}}, \tilde{T}) - \tfrac{1}{2}\tau(GP^n)_E \tag{4.12c}$$

$$\tilde{T}_E = T_E^n + \tfrac{1}{2}\tau H_E(\tilde{\mathbf{U}}, \tilde{T}) \tag{4.12d}$$

$$\tilde{\tilde{\mathbf{U}}}_E = \tilde{\mathbf{U}}_E + \tfrac{1}{2}\tau \mathbf{F}_E(\tilde{\mathbf{U}}, \tilde{T}) - \tfrac{1}{2}\tau(GP^n)_E = 2\tilde{\mathbf{U}}_E - \mathbf{U}_E^n \tag{4.12e}$$

$$T_E^{n+1} = \tilde{T}_E + \tfrac{1}{2}\tau H_E(\tilde{\mathbf{U}}, \tilde{T}) = 2\tilde{T}_E - T_E^n \tag{4.12f}$$

$$\tilde{\tilde{\mathbf{U}}}_O = \tilde{\mathbf{U}}_O + \tfrac{1}{2}\tau \mathbf{F}_O(\tilde{\tilde{\mathbf{U}}}, T^{n+1}) - \tfrac{1}{2}\tau(GP^n)_O \tag{4.12g}$$

$$T_O^{n+1} = \tilde{T}_O + \tfrac{1}{2}\tau H_O(\tilde{\tilde{\mathbf{U}}}, T^{n+1}). \tag{4.12h}$$

Because of the odd-even coupling between the variables and the alternating use of the forward and backward Euler rule, the algorithm (4.12) is only diagonally implicit. This means that per cell a 3×3 system of linear equations has to be solved, which is of course very cheap. Hence the OEH-PC

scheme is almost as fast as the explicit Euler rule, but has much better stability properties as we shall see in Section 6. From (4.12) one can also see that only one array of storage is required for each variable (a known feature of the OEH scheme [5, 6]), which is especially of interest for multi-dimensional problems.

5. Computation of the Pressure

For the computation of the pressure (-increment) we have to solve the Poisson equation

$$LQ^n = \frac{2}{\tau}c, \; c = D\tilde{\tilde{\mathbf{U}}} - B^{n+1}, \tag{5.1}$$

where L is the operator defined in (4.11) and (4.11′). Considered as a matrix, L has a few attractive properties such as symmetry, non-positive definiteness and a pentadiagonal structure. However, L is singular with $Le = 0$, where $e = (1,...,1)^T$, and therefore the set of equations (5.1) has only a solution if $(e,c) = 0$. The condition $(e,c) = 0$ is equivalent to

$$\sum_{j=1}^{M} k(U_{Nj}^{n+1} - U_{0j}^{n+1}) + \sum_{i=1}^{N} h(V_{iM}^{n+1} - V_{i0}^{n+1}) = 0, \tag{5.2}$$

which is a second order approximation to (3.4) at time level $t_{n+1} = (n+1)\tau$. In our case (5.2) is trivially satisfied due to the zero boundary values for the velocity. For arbitrary boundary values $\mathbf{u}_\Gamma$, it may be necessary to make small adjustments in the right hand side c in order to satisfy (5.2).

There are many methods available for the solution of (5.1). Since the OEH scheme is very cheap per time step, it is essential that we combine it with a fast Poisson solver in order to obtain a fast OEH-PC scheme. In our computations we used the multigrid method MGD5V [8, 15]. The multigrid method MGD5V is a saw tooth multigrid iterative process (i.e. one relaxation sweep after each coarse grid correction) for the solution of linear second order elliptic boundary value problems. It uses incomplete line LU-decomposition as relaxation method, a 7-point prolongation and restriction, and a Galerkin approximation for the coarse grid matrices. The multigrid process is repeated until the l_2-norm of the residual is less than 10^{-4}. We wish to emphasize that MGD5V was designed for more general elliptic problems than our simple Poisson equation. Consequently, the computation of the pressure-increment, which is considerable anyhow, can probably be done faster with a solver specifically designed for the Poisson equation.

6. Linear Stability Analysis

In this section we present the conditions for stability based on von Neumann analysis [14]. Consider the linearized equations (Cf.(2.6a)-(2.7))

$$u_t + q_1 u_x + q_2 u_y = -p_x + Pr\nabla^2 u \tag{6.1a}$$

$$v_t + q_1 v_x + q_2 v_y = -p_y + Pr\nabla^2 v + RaPr(T - \tfrac{1}{2}) \tag{6.1b}$$

$$T_t + q_1 T_x + q_2 T_y = \nabla^2 T, \tag{6.2}$$

where q_1 and q_2 are properly chosen approximations to u and v. Note that the computation of T is now decoupled from the computation of u and v, and hence the term $RaPr(T - \frac{1}{2})$ in (6.1b) can be considered as a source term, which has no influence on stability. Therefore we can leave out this term in our analysis. For the sake of simplicity we also leave out the terms $-p_x$ and $-p_y$, and thus consider the equation

$$f_t + (\mathbf{q}\cdot\nabla)f = \epsilon\nabla^2 f, \quad t>0, \ \mathbf{x}\in\mathbb{R}^d, \tag{6.3}$$

where $f = u,v$ or T, $\mathbf{q} = (q_1,\ldots,q_d)^T$ is the constant velocity and $\epsilon>0$ the viscosity parameter. This equation models the convective and viscous effects in (6.1a)-(6.2).

Suppose that for space discretization of (6.3) we use standard central differences with mesh size h in all space directions. Then von Neumann stability analysis yields the following necessary and sufficient time step restrictions [17].

$$d.(\frac{\tau}{h})^2 \sum_{k=1}^{d} q_k^2 \leqslant 4. \tag{6.4}$$

Observe that the time step restriction is independent of the viscosity parameter ϵ. In our actual computations ($d=2$), the value of τ is based on the choice $q_1 = u_{\max}, q_2 = v_{\max}$, where $u_{\max}$ and $v_{\max}$ are estimates of the maximum values of respectively u and v.

For the sake of comparison, we give the necessary conditions for von Neumann stability of the forward Euler central difference scheme for (6.3)[9]

$$\frac{2d\epsilon\tau}{h^2} \leqslant 1, \sum_{k=1}^{d} \frac{q_k^2\tau}{2\epsilon} \leqslant 1. \tag{6.5}$$

The second inequality (convection-diffusion barrier) shows that the forward Euler central difference scheme becomes unconditionally unstable as $\epsilon \to 0$, whereas the OEH scheme is conditionally stable uniformly in ϵ, i.e. $\tau = O(h)$ independent of ϵ. The first inequality of (6.5) implies that

$\tau = O(\epsilon^{-1}h^2)$ for stability, which is disadvantageous for large values of ϵ. From the above it is clear that the OEH scheme has much better stability properties than the forward Euler central difference scheme.

7. Computational Results

We have computed the solution of the free convection problem (2.5)-(2.9) with adiabatic horizontal walls and with perfectly conducting horizontal walls, i.e. walls with a linear temperature profile. In all computations, the Prantl number is set equal to 0.71 (air). The first problem, with adiabatic horizontal walls, has a steady solution for at least $Ra \leqslant 10^7$ [13] and is referred to as the steady free convection problem. The second problem, with a linear temperature profile at the horizontal walls, has a solution which undergoes a Hopf bifurcation at approximately $Ra = 3*10^6$ and beyond this value the flow is periodic [1]. This problem is referred to as the periodic free convection problem.

7.1 Steady free convection

We have computed the solution of this problem for $Ra = 10^3, 10^4, 10^5$ and 10^6. Computations are performed on a 20*20 and a 40*40 grid, until the steady state is reached. We assume that the solution has reached its steady state if $\|U^{n+1} - U^n\|_1 < \delta\tau$, $\|V^{n+1} - V^n\|_1 < \delta\tau$ and $\|T^{n+1} - T^n\|_1 < \delta\tau$, where δ is a sufficiently small number. In our computations we take $\delta = 10^{-2}$. As initial condition, the steady solution at the next lower Ra-number is used, except for $Ra = 10^3$ for which we take the trivial solution $u = v = 0$, $p = 0$ and $T = 0.5$. Figure 3 and 4 show the steady state streamlines and isotherms computed on the 40*40 grid.

At this point we emphasize that the OEH-PC scheme is a solution technique for the fully time-dependent Navier-Stokles equations. Here we use it to compute the solution of the steady free convection problem which enables us to test the accuracy of the scheme in space by means of a comparison with a very accurate benchmark solution, computed by de Vahl Davis [18]. For that purpose we introduce the following characteristic values of the flow. In Table 1 we compare these for the benchmark solution of the Vahl Davis and for our solution.

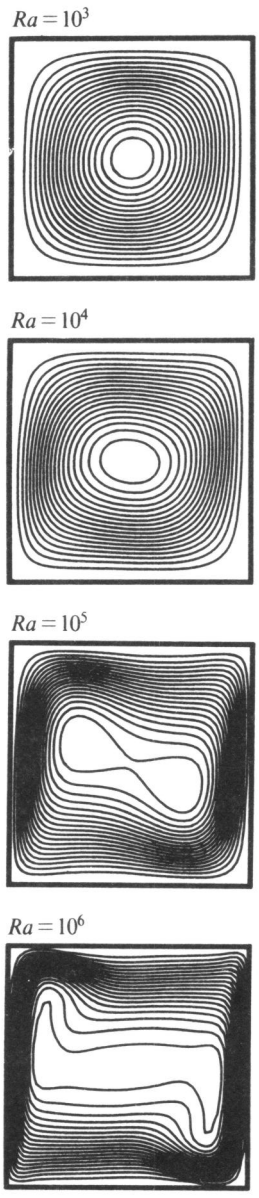


Fig. 3. Streamlines

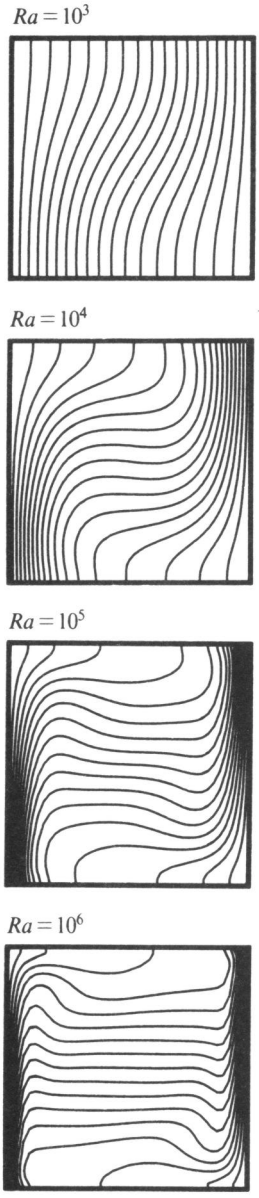


Fig. 4. Isotherms

$U_{\max}$ the maximum horizontal velocity on the line $x=0.5$ (together with its location)

$V_{\max}$ the maximum vertical velocity on the line $y=0.5$ (together with its location)

ψ_{mid} the absolute value of the stream function at the point (0.5,0.5)

$\psi_{\max}$ the maximum absolute value of the streamfunction (together with its location)

$\overline{Nu}$ the average Nusselt number throughout the cavity

$Nu_{\frac{1}{2}}$ the average Nusselt number on the line $x=0.5$

Nu_0 the average Nusselt number on the line $x=0$

$Nu_{\max}$ the maximum absolute value of the local Nusselt number on the line $x=0$ (together with its location)

$Nu_{\min}$ the minimum absolute value of the local Nusselt number on the line $x=0$ (together with its location).

Concerning the characteristic values in Table 1 we note the following. The streamfunction $\psi(x,y)$ is computed from the Poisson equation

$$\nabla^2\psi = u_y - v_x, \tag{7.1}$$

subject to the boundary condition $\psi = const.(=0)$ on $\partial\Omega$ (the boundary is a streamline). The local Nusselt number is the local heat flux in the horizontal direction, and is given by [18]

$$Q(x,y) = uT - T_x. \tag{7.2}$$

Through any vertical line $x = x_0$, the total heat flux is given by

$$Nu_{x_0} = \int_0^1 Q(x_0,y)dy. \tag{7.3}$$

Nu_{x_0} is called the average Nusselt number on the line $x = x_0$. In a cavity with adiabatic horizontal walls, Nu_{x_0} must be independent of x_0. Finally, the average Nusselt number $\overline{Nu}$ throughout the cavity is given by

$$\overline{Nu} = \int_0^1 Nu_x dx. \tag{7.4}$$

The integrals in (7.3) and (7.4) have been computed using Simpson's rule, and the term T_x in (7.2) is evaluated using a second order finite difference approximation (also on the boundary $x=0$). The

maximum and minimum values in Table 1 (and their locations) are computed by numerical differentiation, using a least squares polynomial of appropriate degree.

From Table 1 we can conclude the following. The 40*40 solution is in close agreement with the benchmark solution, except for the characteristic values in the boundary layer ($V_{max}, Nu_0, Nu_{max}, Nu_{min}$) occurring for $Ra = 10^6$ (see Fig. 3 and 4). Thus, the 40*40 solution is an accurate representation of the flow, except in the boundary layers for $Ra = 10^6$. The 20*20 solution is still a fairly accurate solution for $Ra = 10^3, 10^4$, but not for $Ra = 10^5, 10^6$.

We conclude this section with a few remarks about the time step restriction and the computing time. The time step restriction for von Neumann stability of the OEH scheme is given by (6.4). Especially for large values of Ra, this inequality implies a rather small time step due to the large velocity values. To be more specific, the time steps used in the computation of the 40*40 solution are the following: $\tau = 10^{-3}$ for $Ra = 10^3$, $\tau = 5*10^{-4}$ for $Ra = 10^4$, $\tau = 2*10^{-4}$ for $Ra = 10^5$ and $\tau = 2.5*10^{-5}$ for $Ra = 10^6$. Thus, especially for large values of Ra, the computation of the steady solution requires a lot of time steps. However, since one time step of the OEH-PC scheme is cheap, this is not very trouble some. To get an impression, the computation of the steady solution for $Ra = 10^6$ on a 40*40 grid requires approximately 7200 time steps and one hour CPU time on a Cyber 170-750 computer. However, this CPU time can be reduced considerably if we use an optimal Poisson solver.

Ra	$U_{\max}$ y	$V_{\max}$ x	ψ_{mid}	$\psi_{\max}$ x/y	$\overline{Nu}$	$Nu_{\frac{1}{2}}$	Nu_0	$Nu_{\max}$ y	$Nu_{\min}$ y
10^3	3.649 0.813	3.697 0.178	1.174		1.118	1.118	1.117	1.505 0.092	0.692 1.0
10^4	16.178 0.823	19.617 0.119	5.071		2.243	2.243	2.238	3.528 0.143	0.586 1.0
10^5	34.73 0.855	68.59 0.066	9.111	9.612 0.285/0.601	4.519	4.519	4.509	7.717 0.081	0.729 1.0
10^6	64.63 0.850	219.36 0.0379	16.32	16.750 0.151/0.547	8.800	8.799	8.817	17.925 0.0378	0.989 1.0

Solution on 40×40 grid

Ra	$U_{\max}$ y	$V_{\max}$ x	ψ_{mid}	$\psi_{\max}$ x/y	$\overline{Nu}$	$Nu_{\frac{1}{2}}$	Nu_0	$Nu_{\max}$ y	$Nu_{\min}$ y
10^3	3.647 0.813	3.697 0.178	1.176	1.176 0.5/0.5	1.117	1.117	1.119	1.504 0.090	0.695 1.0
10^4	16.172 0.823	19.624 0.118	5.080	5.080 0.5/0.5	2.240	2.237	2.245	3.531 0.147	0.590 1.0
10^5	34.564 0.856	67.925 0.065	9.077	9.581 0.281/0.600	4.519	4.511	4.604	8.000 0.088	0.744 1.0
10^6	64.656 0.867	213.575 0.036	16.106	16.608 0.157/0.553	8.927	8.919	9.858	21.437 0.055	1.070 1.0

Solution on 20×20 grid

Ra	$U_{\max}$ y	$V_{\max}$ x	ψ_{mid}	$\psi_{\max}$ x/y	$\overline{Nu}$	$Nu_{\frac{1}{2}}$	Nu_0	$Nu_{\max}$ y	$Nu_{\min}$ y
10^3	3.642 0.813	3.694 0.178	1.180	1.180 0.5/0.5	1.114	1.115	1.117	1.495 0.098	0.702 1.0
10^4	16.144 0.823	19.645 0.116	5.104	5.104 0.5/0.5	2.231	2.218	2.273	3.612 0.152	0.601 1.0
10^5	34.507 0.861	66.940 0.060	9.070	9.528 0.291/0.613	4.565	4.513	5.063	9.236 0.110	0.777 0.996
10^6	62.978 0.891	219.216 0.030	15.153	15.772 0.147/0.559	9.425	9.096	12.635	24.260 0.086	0.648 0.981

Table 1. Some characteristic values of the free convection flow.

7.2 Periodic free convection

We have computed the solution of this problem for $Ra = 2.5*10^6$, $5*10^6$ and $8.5*10^6$. Because of accuracy considerations, we used a 40*40 grid (see Section 7.3). For $t \to \infty$, the solution for $Ra = 2.5*10^6$ tends to a steady state and the solutions for the other two Ra-numbers become periodic [1]. The initial conditions are given by the trivial solution $u = v = 0$, $p = 0$ and $T = 0.5$. Thus, at $t = 0$ the fluid is at rest with a pressure equal to the hydrostatic pressure and a temperature equal to the mean wall temperature.

For illustrating the transient flow behaviour, we shall present plots of the (dimensionless) kinetic energy defined by

$$K = \tfrac{1}{2} \int\int_{\Omega} (u^2 + v^2) dS, \tag{7.5}$$

and the average Nusselt number throughout the cavity $\overline{Nu}$, defined by (7.4). First, consider the case $Ra = 2.5*10^6$. Figure 5 shows the kinetic energy K and the average Nusselt number $\overline{Nu}$ as a function of the (dimensionless) time. We see that both K and $\overline{Nu}$ show a damped oscillatory behaviour. The steady solution is reached for approximately $t = 0.05$. The small oscillations for $t > 0.05$ are due to space discretization errors. The steady state streamlines and isotherms are plotted in Figure 6. A usable time step is in this case $\tau = 10^{-5}$ and the total CPU time required to step to $t = 0.08$ (8,000 time steps) is approximately 80 min.

The kinetic energy and the average Nusselt number as a function of time, for time, for $Ra = 5*10^6$, are shown in Figure 7. From this figure we see that both K and $\overline{Nu}$ rapidly tend to a periodic behaviour for increasing t, with a period of approximately $t_0 = 7*10^{-3}$. Thus, the oscillatory behaviour already observed for $Ra = 2.5*10^6$ has now become undamped. In [1] a very regular periodic flow with constant frequency was observed for $Ra = 5*10^6$, which is in agreement with our results. Note that the periodic pattern for K and $\overline{Nu}$ is disturbed by space discretization errors. Especially in the boundary layers these are still rather large. Also in this case $\tau = 10^{-5}$ and the total CPU time needed to compute the solution for $t \leqslant 0.1$ (10,000 time steps) is approximately 105 min. Note that the time step τ has to be chosen rather small in order to satisfy the stability constraint. However, since the solution is rapidly varying in time, the time step τ has to be chosen small anyhow for accuracy. Figure 8 and 9 show the streamlines and isotherms during two periods of the flow. The flow contains one main vortex and a few secondary vortices which alternate in size and position. The isotherms are nearly horizontal, except in a small region near the walls. Their shape oscillates with the periodic growth and decay of the vortices. These two figures clearly demonstrate the periodic nature of the flow.

Finally, we consider the case $Ra = 8.5*10^6$. The kinetic energy and the average Nusselt number are

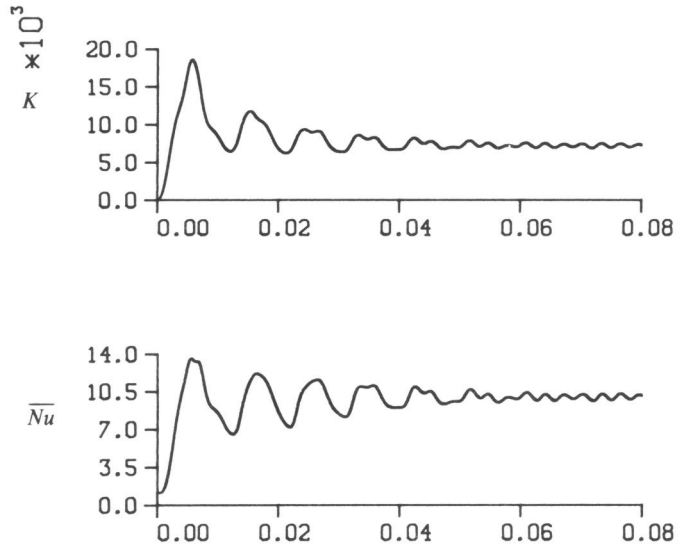


Fig. 5. The kinetic energy and the average Nusselt number as a function of time for $Ra = 2.5 * 10^6$.

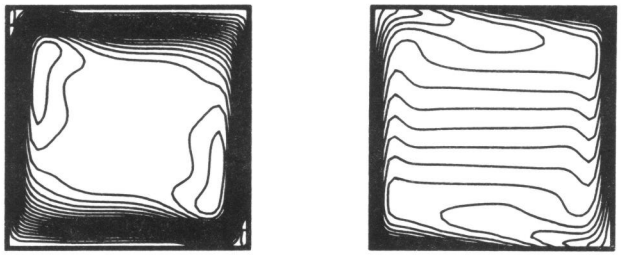


Fig. 6. Steady state streamlines and isotherms for $Ra = 2.5 * 10^6$.

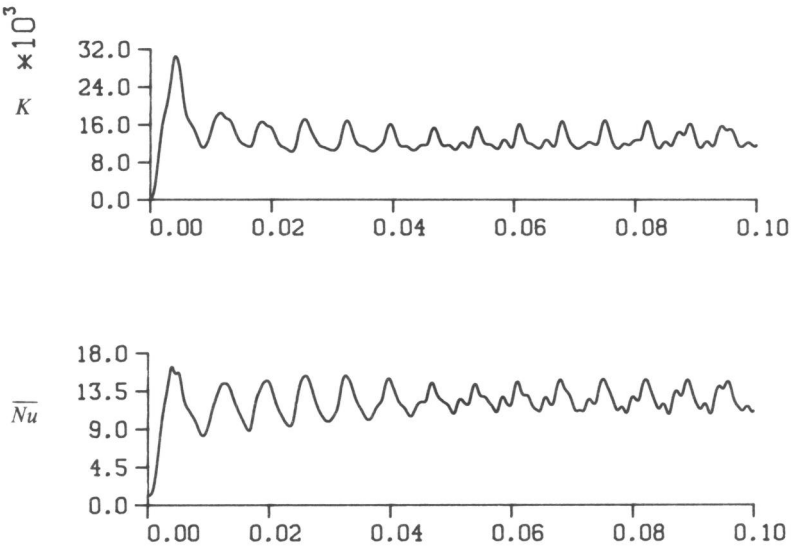


Fig. 7. The kinetic energy and the average Nusselt number as a function of time for $Ra = 5 * 10^6$.

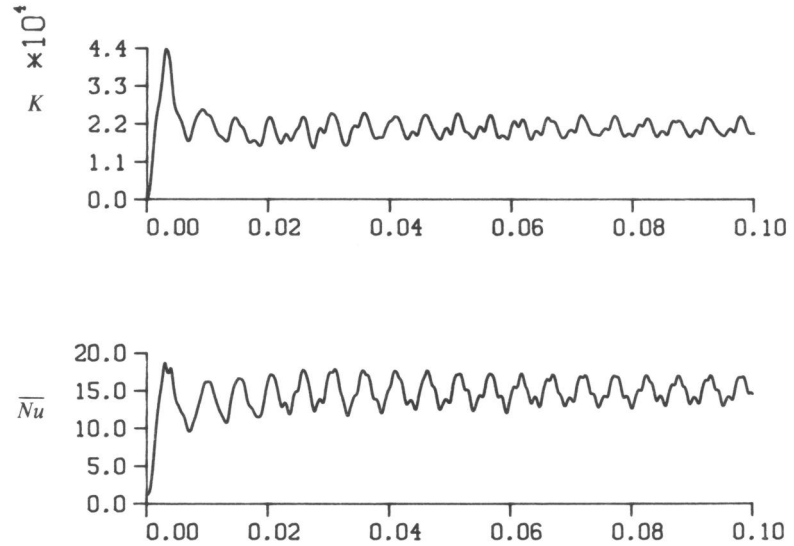


Fig. 10 The kinetic energy and the average Nusselt number as a function of time for $Ra = 8.5 * 10^6$.

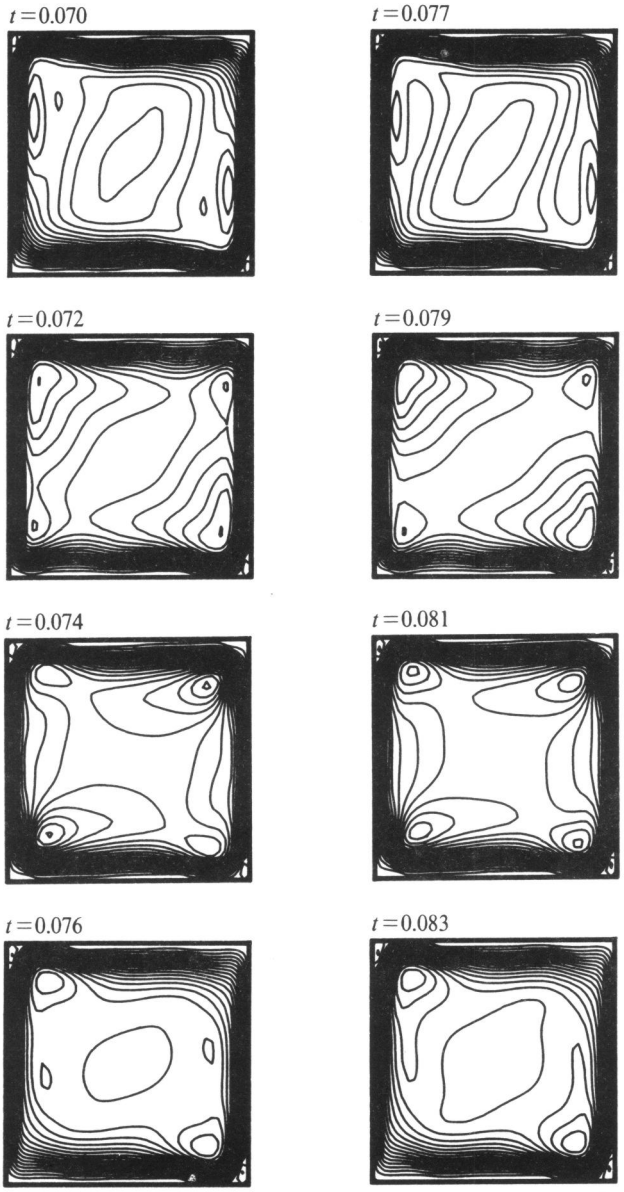


Fig. 8. Streamlines during two periods of the flow for $Ra = 5*10^6$.

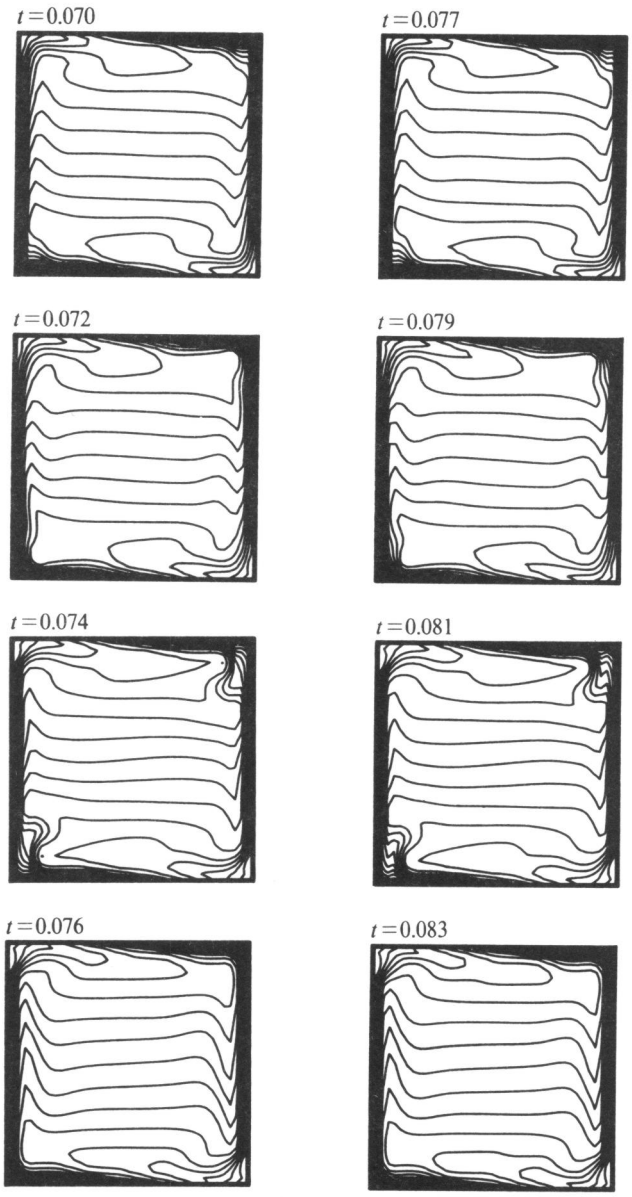
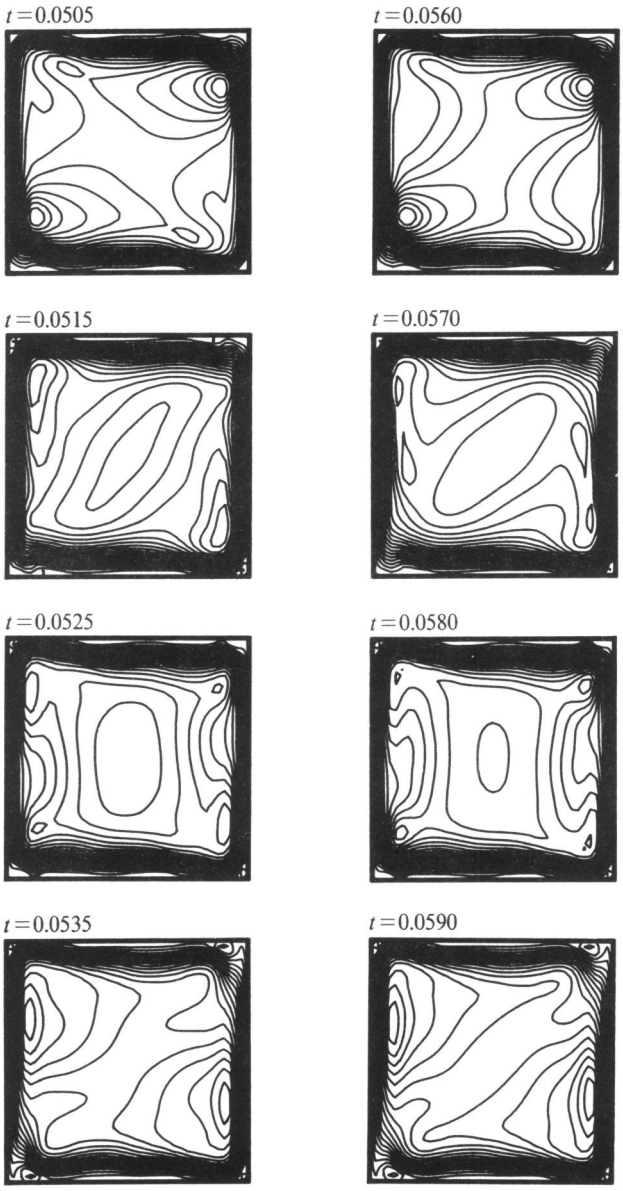


Fig. 9. Isotherms during two periods of the flow for $Ra = 5*10^6$


t=0.0505
t=0.0560
t=0.0515
t=0.0570
t=0.0525
t=0.0580
t=0.0535
t=0.0590

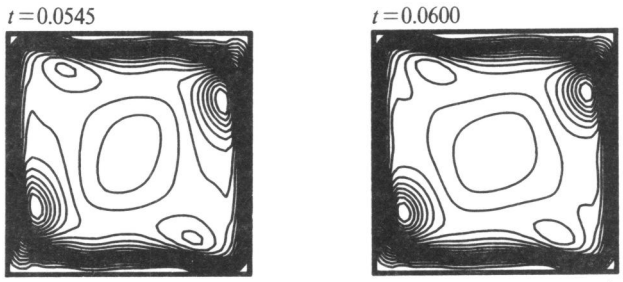
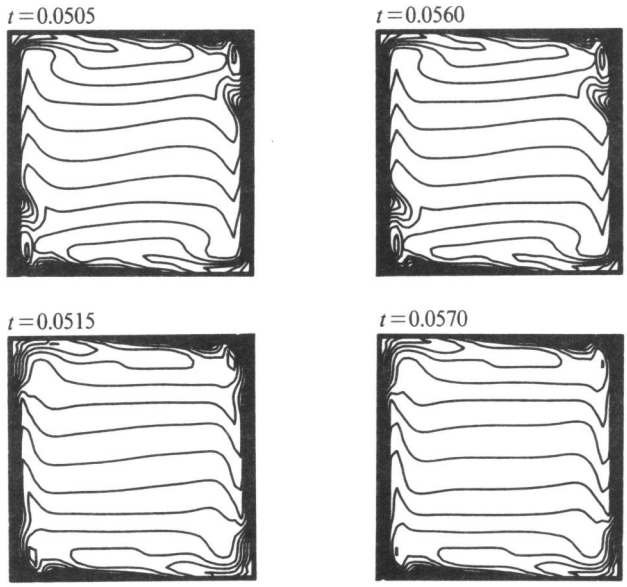


Fig. 11. Streamlines during two periods of the flow for $Ra = 8.5 * 10^6$.



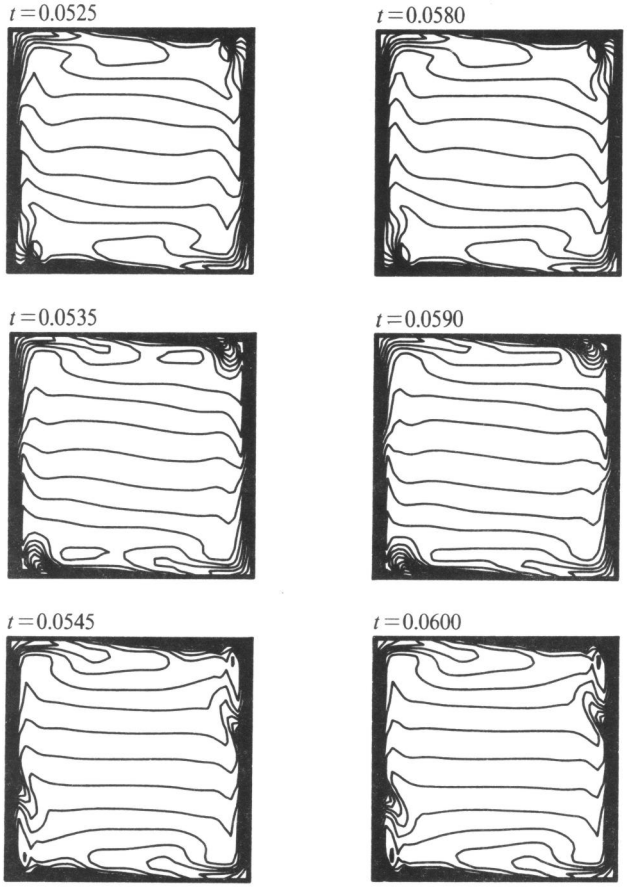


Fig. 12. Isotherms during two periods of the flow for $Ra = 8.5 * 10^6$.

shown in Figure 10. Like in the previous case, K and $\overline{Nu}$ rapidly tend to a periodic behaviour. In [1], for $Ra=8.5*10^6$, two stable modes of flow were observed with a slightly different frequency. From Figure 10 we see, however, that both K and $\overline{Nu}$ are periodic functions of time with a constant frequency (period). The period t_0 is approximately $t_0=5.5*10^{-3}$. The reason for this discrepancy is probably the space discretization error. Thus, in order to find the two modes of the flow with different frequencies, it is necessary to compute the solution much more accurately. Observe, that for increasing Ra the period decreases (or equivalently: the frequency increases). The time step is in this case $\tau=5*10^{-6}$ and the total CPU time for the computation of the solution for $t\leqslant 0.1$ (20,000 time steps) is roughly 3h 15min. The streamlines and isotherms during two periods of the flow are shown in Figure 11 and 12. These figures clearly show the periodic behaviour of the flow.

For time-dependent problems, the OEH scheme suffers from the so-called DuFort-Frankel (DFF) deficiency [17]. We close this section with a short discussion of this deficiency. Consider to this purpose the convection-diffusion equation (6.3). The OEH scheme for this equation is equivalent to the leapfrog-DFF scheme at the odd points [17]. Suppose that for space discretization we use standard central differences with gridsize h in all space directions. By the DFF deficiency we mean that for $\tau,h\to 0$ the solution of the leapfrog-DFF scheme converges to the solution of the problem.

$$f_t+(\mathbf{q}\cdot\nabla)f = \epsilon\nabla^2 f-\epsilon d(\frac{\tau}{h})^2 f_{tt}. \tag{7.6}$$

In general, for convergence it thus is necessary that $\tau=o(h)$. Through the equivalence property, the same conclusion is valid for the OEH scheme. In our computations, however, $\frac{\tau}{h}$ and the viscosity parameter ϵ are relatively small, so that the DFF deficiency has only a minor influence on the accuracy.

8. Conclusions

In this paper we have discussed the OEH-PC scheme for the computation of incompressible fluid flow. The OEH-PC scheme is a combination of the OEH scheme for the time integration of the Navier-Stokes equations and the pressure correction method for the computation of the pressure. The scheme has a few attractive properties. First, the scheme is very simple as it is (almost) explicit. Therefore, extension to arbitrary domains (also 3-dimensional) and to non-uniform grids is straightforward. The scheme is also easy to vertorize for use on a vectorcomputer. In [4] it is shown that a vectorized OEH scheme for the Burgers' equations is indeed very fast. Second, the scheme is fast per time step, provided we have a fast Poisson solver for the computation of the pressure. Finaly, the storage requirements of the scheme are very modest. A drawback of the scheme is the DFF-deficiency,

which has in general a negative influence on the accuracy. For many flow problems, however, this deficiency is only of minor importance. Furthermore, the scheme is only conditionally stable, which can be rather restrictive for highly convection dominated flows.

We have applied the OEH-PC scheme to two free convection problems, viz. steady free convection and periodic free convection. The results for the steady free convection problem are in good agreement with a very accurate benchmark solution for this problem. For the periodic free convection problem, there is (to our knowledge) no benchmark solution available. Therefore, we have compared our results with experimental results in [1], showing that the qualitative behaviour of our solution agrees with the experimental data. These two examples demonstrate the feasibility of the OEH-PC scheme for the computation of incompressible fluid flow.

References

[1] D.G. Briggs and D.N. Jones, J. Heat Transf. 107 (1985), 850-854.

[2] T. Cebeci, R.S. Hirsch, H.B. Keller and P.G. Williams, Comp. Meth. Appl. Mech. Eng. 27 (1981), 13-44.

[3] S. Chandrasekhar, "Hydrodynamic and Hydromagnetic Stability," (The Clarendon Press, Oxford, 1961).

[4] E.D. de Goede and J.H.M. ten Thije Boonkkamp, Centre for Mathematics and Computer Science Report NM-8720, 1987 (submitted for publication).

[5] A.R. Gourlay, J. Inst. Maths. Applics. 6 (1970), 375-390.

[6] A.R. Gourlay, and G.R. McGuire, J. Inst. Maths. Applics. 7 (1971) 216-227.

[7] P.H. Gresho and R.L. Sani, Int. J. Num. Methods Fluids 7 (1987), 1111-1145.

[8] P.W. Hemker, and P.M. de Zeeuw, in: "Multigrid methods for integral and differential equations," The institute of mathematics and its applications conference series, edited by D.J. Paddon and H. Holstein, Oxford University Press, New York (1985) p.85-116.

[9] A.C. Hindmarsh, P.M. Gresho and D.F. Griffith, Int. J. Num. Meth. Fluids 4, (1984) 853-897.

[10] P.J. van der Houwen, and J.G. Verwer, Computing 22, (1979) 291-309.

[11] J. van Kan, SIAM J. Sci. Stat. Comput. 7, (1986) 870-891.

[12] R. Peyret, and T.D. Taylor, "Computational methods for fluid flow," (Springer-Verlag, New York, 1983).

[13] P. le Quere, and T. Alziary de Roquefort, J. Comput. Phys. 57, (1985) 210-228.

[14] R.D. Richtmeyer, and K.W. Morton, "Difference methods for initial value problems,"

(Interscience Publishers, New York, 1967).

[15] P. SONNEVELD, P. WESSELING and P.M. DE ZEEUW, in: "Multigrid methods for integral and differential equations," The institute of Mathematics and its applications conference series, edited by D.J. Paddon and H. Holstein, (Oxford University Press, New York (1985)), p.117-167.

[16] J.H.M. TEN THIJE BOONKKAMP, SIAM J. Sci. Stat. Comput. 9, (1988), 252-270.

[17] J.H.M. TEN THIJE BOONKKAMP, and J.G. VERWER, Applied Numerical Mathematics 3 (1987), 183-193.

[18] G. DE VAHL DAVIS, Int. J. Num. Meth. Fluids 3, (1983) 249-264.

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 [16], 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-

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.

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, since they 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 single pair of data requires one single instruction (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. A Cray-computer accepts vectors for which the number of memory locations between consecutive elements (the so-called stride) is constant.

The time required for completing a vector instruction has two components. The first is the start-up time, that is the time elapsed before the first result is computed. The second component is proportional to the vector length and is called the streamtime. For example, for a (2-pipe) CDC Cyber 205 the result rate for an add or multiply instruction is two results per cycle (clock period). Due to the start-up time it is beneficial to work with longer vectors [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 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. 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

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. 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

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

with u and v the velocities in x- and y-directions and Re denoting the Reynolds number. On the boundary Γ of the connected space domain Ω, 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)

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

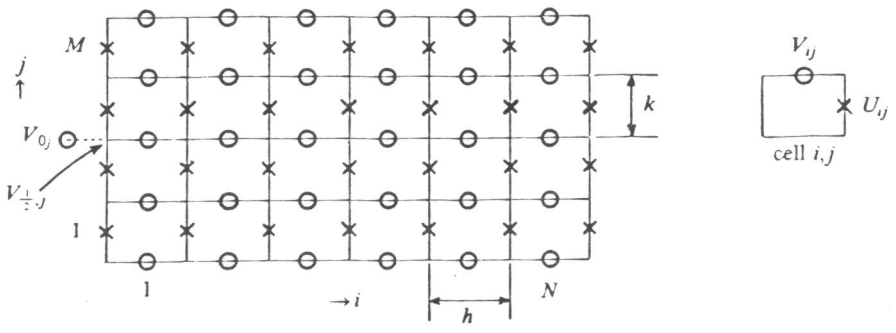


Fig 1. The staggered grid

where

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

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

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 $\circ$-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

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

However, for the OEH scheme we choose

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

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 the boundary value $V_{\frac{1}{2},j}$ 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 two-stage formula

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

with τ 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]

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

where $\boldsymbol{U}_{ij}^n = (U_{ij}^n, V_{ij}^n)^T$ (likewise for $\boldsymbol{F}_{ij}(\mathbf{U}^n)$). 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

$$\boldsymbol{U}_{ij}^{n+1} = \boldsymbol{U}_{ij}^n + \tau\theta_{ij}^n\boldsymbol{F}_{ij}(\mathbf{U}^n) + \tau\theta_{ij}^{n+1}\boldsymbol{F}_{ij}(\mathbf{U}^{n+1}) \tag{3.9a}$$

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

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

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

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

The order of computation for the OEH scheme is

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

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

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

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

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]

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

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

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

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

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

are linearized as follows :

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

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

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

$$V^* = 2V^{n+\frac{1}{2}} - V^n.$$

Now, the ADI scheme requires the solution of linear tridiagonal systems. In Section 4.2 we will discuss some algorithms for the solution of 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)).

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

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

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)

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

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 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 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 be subdivided accordingly 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 in general an advantage on

vector computers. Moreover, no data reorderings have to be performed.

Notice 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 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 (3.12) we have M tridiagonal (linear) systems of order N. However, the M individual systems can be combined to a single tridiagonal system of order NM. 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 and the Cray X-MP/24 the maximal memory size 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. Following Van der Vorst [15] and Wubs and De Goede [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 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 in which 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 in which Gaussian elimination is used for the solution of the tridiagonal systems,

ADEI the Peaceman-Rachford scheme in which 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) (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, 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 ϕ is the solution of

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

In our test problem we choose $\phi = f_1 + f_2$ [17], 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 = 100,1000,10000.

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 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.

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

Table 5.1. cd_∞-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 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) one observes 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.

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 129×129 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	1.8	1.0	15.4 (8.6)
ADIW	27.3	8.7	181.6 (6.6)
ADIW1	18.6	8.9	187.1 (10.0)
ADIGE	54.1	16.6	121.4 (2.2)
ADEI	22.5	6.1	176.6 (7.8)
ADEI1	12.7	6.8	182.2 (14.4)

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

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 considerably faster than the (2-pipe) CDC Cyber 205. This is due to a smaller clock cycle and 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_∞-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 time step as large as possible such that $\mathrm{cd}_1 \geqslant 3$. As soon as $\mathrm{cd}_1 < 3$ 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.

Re	correct digits for u-field (1- norm)							
	OEH scheme				ADIW scheme			
	$h=1/33$ $\tau=1/40$	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.

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).

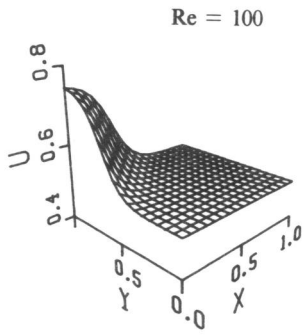
First, let us consider the advantages of the OEH scheme over the ADI-type schemes :

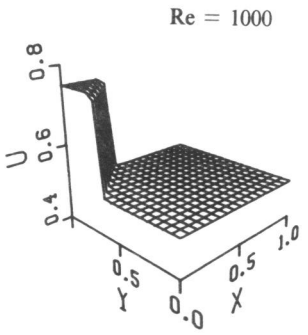
(i) On both vector computers the OEH scheme is considerably faster than the ADI-type schemes, due to the near-explicitness of the OEH scheme.

(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 :





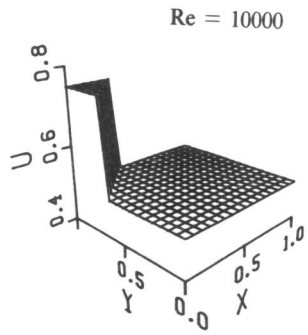
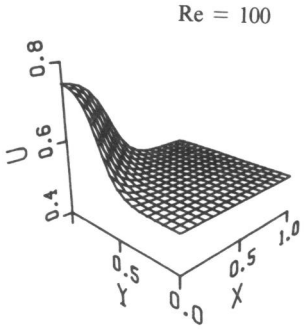
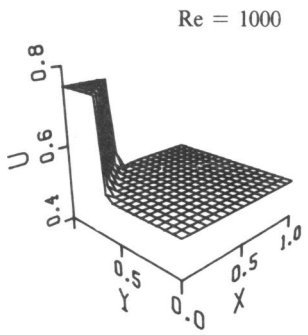


Fig. 2. Exact solutions (5.3a) for Re = 100, 1000, 10000.





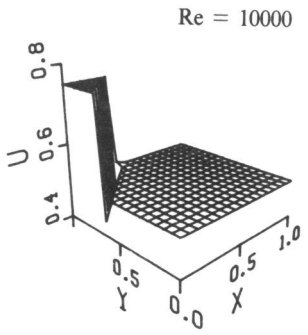


Fig. 3. Corresponding numerical solutions

(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, *SIAM J. Sci. Stat. Comput.*, 9 (1988), pp.252-270.
[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.A. VAN DER VORST, *Large tridiagonal and block tridiagonal linear systems on vector and parallel computers*, Report 86-25, Delft University of Technology, Delft, 1986.
[16] H.H. WANG, A parallel method for tridiagonal system equations, *ACM Trans. on Math. Softw.*, 7 (1981), pp. 170-183.
[17] G.B. WHITHAM, *Linear and nonlinear waves*, Wiley, New York, 1974.

Residual smoothing for accelerating the ADI iteration method for elliptic difference equations

J.H.M. ten Thije Boonkkamp
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Residual smoothing is a simple technique to increase the rate of convergence of iterative methods for elliptic difference equations. In this paper, we combine residual smoothing with the ADI iteration method, which can be done in several ways. When applied in the proper way, residual smoothing can considerably reduce the number of iterations and thus the computing time of the ADI scheme. The parameter values of the smoothed ADI scheme are chosen such that the high- and low frequency components in the iteration error are damped very well. Due to the residual smoothing, the other components in the error are also properly damped. Numerical examples demonstrate the performance results of the ADI scheme and the smoothed ADI scheme.

1980 Mathematics Subject Classification: 65F10, 65N20.
Key Words & Phrases: elliptic difference equation, ADI iteration, residual smoothing, smoothed ADI iteration.
Note: This report will be submitted for publication elsewhere.

1. Introduction

We consider the first boundary-value problem for the two-dimensional elliptic partial differential equation (PDE)

$$(p(x,y)u_x)_x + (q(x,y)u_y)_y - w(x,y)u = f(x,y),\ (x,y)\in\Omega = [0,1]\times[0.1], \tag{1.1}$$

where $p(x,y)>0$, $q(x,y)>0$ and $w(x,y)\geqslant 0$. As a special case of (1.1) we employ the Poisson equation

$$u_{xx} + u_{yy} = f(x,y) \tag{1.2}$$

as a model problem.

For space discretization, we cover Ω with a uniform space grid with gridsize h, where $h=1/(M+1)$ and M is the number of internal gridpoints in x-and y-direction. Space discretization of (1.1), using standard central differences, yields a difference system

$$D_{xx}U + D_{yy}U = B. \tag{1.3}$$

In (1.3) U is a vector, with components U_{ij}, and B is a vector originating from the right hand side f and the boundary conditions for u. The component U_{ij} is the finite difference approximation to $u(ih,jh)$. The matrices D_{xx} and D_{yy} in (1.3) are the finite difference replacements of respectively $\frac{\partial}{\partial x}(p(x,y)\frac{\partial}{\partial x}) - \frac{1}{2}w(x,y)$ and $\frac{\partial}{\partial y}(q(x,y)\frac{\partial}{\partial y}) - \frac{1}{2}w(x,y)$ and are defined by

$$(D_{xx}U)_{ij} := \frac{1}{h^2}(p_{i-\frac{1}{2},j}U_{i-1,j} - (p_{i-\frac{1}{2},j} + p_{i+\frac{1}{2},j})U_{ij} + p_{i+\frac{1}{2},j}U_{i+1,j}) - \tfrac{1}{2}w_{ij}U_{ij}, \tag{1.4a}$$

$$(D_{yy}U)_{ij} := \frac{1}{h^2}(q_{i,j-\frac{1}{2}}U_{i,j-1} - (q_{i,j-\frac{1}{2}} + q_{i,j+\frac{1}{2}})U_{ij} + q_{i,j+\frac{1}{2}}U_{i,j+1}) - \tfrac{1}{2}w_{ij}U_{ij}, \tag{1.4b}$$

with $p_{i\pm\frac{1}{2},j} = p((i\pm\frac{1}{2})h,jh)$ (analogous definitions for $q_{i,j\pm\frac{1}{2}}$ and w_{ij}). The matrices D_{xx} and D_{yy} are tridiagonal, symmetric and negative definite.

For the iterative solution of (1.3) we examine the ADI scheme of Peaceman and Rachford [3,5]. For the model problem, the ADI scheme is known to be a fast scheme if one chooses its parameter values in the right way. However, the scheme is very sensitive to the parameter values used, i.e., the iteration count grows rapidly when the computation is carried out away from the optimal parameter values. Therefore, the ADI scheme is in general not a fast iteration technique. It is the purpose of this paper to apply residual smoothing for improving the rate of convergence of the ADI scheme and, most importantly, to make the scheme less sensitive to the choice of the parameter values. This paper is inspired by [2], where residual smoothing is applied to Jacobi iteration.

The contents of the paper is the following. In Section 2 a short outline of the theory of residual smoothing is given. The ADI scheme and the smoothed ADI scheme are discussed in Section 3 and parameter values for both schemes are given in Section 4. Section 5 is devoted to a numerical comparison between the ADI scheme and the smoothed ADI scheme. This comparison also involves a nonlinear example. In Section 6, an alternative smoothed ADI scheme is briefly discussed. Some conclusions are formulated in Section 7.

2. Residual smoothing

In this section we give a short outline of the theory of residual smoothing as a means of accelerating the convergence of iterative methods for elliptic difference equations. For a more extensive treatment of the special type of explicit residual smoothing used here, the reader is referred to [2].

Consider the linear system

$$AU=B, \tag{2.1}$$

obtained by discretizing a linear elliptic boundary value problem. We assume that A has negative eigenvalues. Iterative methods for solving (2.1) are based upon the splitting $A=P-Q$, where P is a non-singular and easily invertible matrix [1,5]. The iteration scheme thus takes the form

$$PU^{n+1}=QU^n+B, \tag{2.2}$$

or equivalently, in residual form,

$$PU^{n+1}=PU^n-(AU^n-B). \tag{2.2'}$$

The idea of residual smoothing is now to multiply the residual in (2.2') by a matrix S such that the condition number of SA is much smaller that the condition number of A. The iteration scheme then reads

$$PU^{n+1}=PU^n-S(AU^n-B). \tag{2.3}$$

Thus, instead of solving (2.1), we solve the preconditioned system $SAU=SB$ with the original iteration method.

Following [2], S is taken of the form $S=P_k(D)$, where $P_k(z)$ is a polynomial of degree k satisfying $P_k(0)=1$ and D is a scaled difference matrix with eigenvalues in the interval [-1,0]. In order to analyse the residual smoothing technique we choose

$$D=\frac{1}{\rho}A, \tag{2.4}$$

where $\rho=\rho(A)$ is the spectral radius of A. In [2], for this choice, an optimal smoothing matrix $S=P_k(D)$ is derived, in the sense that SA has negative eigenvalues and the smallest possible condition number. The condition number $\gamma(A)$ of a matrix A is defined as $\gamma(A)=\rho(A)/\delta(A)$, where $\delta(A)$ is the in absolute value smallest eigenvalue of A. The polynomial $P_k(z)$ is given by

$$P_k(z)=\frac{T_{k+1}(1+2z)-1}{2(k+1)^2 z}, \tag{2.5}$$

where $T_k(z)$ is the *kth* degree Chebyshev polynomial of the first kind. Because of the factorization

properties of the Chebyshev polynomials, the smoothing matrix S can be computed very efficiently if $k=2^q-1$ for some positive integer q.

We emphasize, however, that in actual computations we do not use the difference matrix D defined by (2.4), because it is much too expensive in the general case. Instead, for one-dimensional problems, the matrix D is given by

$$D=\frac{1}{4}\begin{bmatrix} 0 & & & & & & & \\ 1 & -2 & 1 & & & & & \\ & 1 & -2 & 1 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & \ddots & & \ddots \\ & & & & & 1 & -2 & 1 \\ & & & & & & & 0 \end{bmatrix} \tag{2.6}$$

For two-dimensional computations we do not use the two-dimensional analogue of (2.6) because the computation of $S=P_k(D)$ is then not attractive [2]. Therefore, we consider an alternative which only uses one-dimensional smoothing matrices. The residual $r=AU^n-B$ in (2.2′) can in the two-dimensional case be written as $r=D_{xx}U^n+D_{yy}U^n-B$ (Cf(1.3)). The residual is then smoothed by applying the one-dimensional smoothing matrix to D_{xx} and/or D_{yy}. In other words, let the residual r be arranged in a two-dimensional array in the natural way, then r is smoothed by applying the one-dimensional smoothing matrix to all rows and/or columns of r.

3. ADI- AND SMOOTHED ADI ITERATION

Consider equation (1.3)

$$AU=B,\ A=D_{xx}+D_{yy}. \tag{3.1}$$

The ADI scheme for (3.1) can be written in residual form as [5]

$$(D_{xx}-\nu_1 I)U^*=(D_{xx}-\nu_1 I)U^n-(AU^n-B) \tag{3.2a}$$

$$(D_{yy}-\nu_2 I)U^{n+1}=(D_{yy}-\nu_2 I)U^*-(AU^*-B), \tag{3.2b}$$

where $\nu_1,\nu_2>0$ and are supposed to be independent of n.

The first stage (3.2a) of the ADI scheme is implicit in x-direction and explicit in y-direction. This suggests to apply in (3.2a) a smoothing matrix $\tilde{S}_y$ for the preconditioning of D_{yy}. In other words, we multiply each column of the residual in (3.2a) by a (one-dimensional) smoothing matrix S_y, where S_y is such that S_yD_{yy} has the smallest possible condition number. In the same way, we apply a smoothing matrix S_x at the second stage (3.2b) for the preconditioning of D_{xx}. Each row of the residual in (3.2b) is then multiplied by a (one-dimensional) smoothing matrix S_x. The smoothed ADI (SADI) scheme then reads

$$(D_{xx}-\nu_1 I)U^*=(D_{xx}-\nu_1 I)U^n-\tilde{S}_y(AU^n-B) \tag{3.3a}$$

$$(D_{yy}-\nu_2 I)U^{n+1}=(D_{yy}-\nu_2 I)U^*-\tilde{S}_x(AU^*-B). \tag{3.3b}$$

In the analysis, the operators S_x and S_y are defined by $S_x=P_k(\frac{1}{\rho_1}D_{xx})$ and $S_y=P_k(\frac{1}{\rho_2}D_{yy})$ where $\rho_1=\rho(D_{xx})$ and $\rho_2=\rho(D_{yy})$. We emphasize once more, that in practice the matrices $\frac{1}{\rho_1}D_{xx}$ and $\frac{1}{\rho_2}D_{yy}$ are replaced by difference matrices like the one defined in (2.6). The degree k of the

polynomial $P_k(z)$ will be specified later.

If $D_{xx} - \frac{1}{2}(\nu_1 - \nu_2)I$ and $D_{yy} + \frac{1}{2}(\nu_1 - \nu_2)I$ are negative definite then the ADI scheme is convergent [5]. Likewise, the SADI scheme is convergent if $D_{xx} - D_{yy} + S_x A - (\nu_1 - \nu_2)I$ and $-D_{xx} + D_{yy} + S_y A + (\nu_1 - \nu_2)I$ are negative definite. The proof is along the same lines as the proof for ADI.

In order to get an indication about the performance of both the ADI scheme and the SADI scheme, we consider the eigenvalues of the iteration matrix of both schemes. These eigenvalues are called the damping factors of the iteration scheme. In the remainder of the paper we consider the following two cases:

$$\text{case1: } \rho(D_{xx}) = \rho(D_{yy}) = \rho,\ \delta(D_{xx}) = \delta(D_{yy}) = \delta,$$

$$\text{case 2: } \rho_1 = \rho(D_{xx}) \neq \rho_2 = \rho(D_{yy}),\ \delta_1 = \delta(D_{xx}) \neq \delta_2 = \delta(D_{yy}).$$

For simplicity, we take $\nu_1 = \nu_2 = \nu$, unless stated otherwise, and assume that D_{xx} and D_{yy} commute.

First, we restrict ourselves to case 1. The damping factor of the ADI scheme is given by

$$\xi = \xi(\lambda_x, \lambda_y; \nu) = \frac{(\lambda_x + \nu)(\lambda_y + \nu)}{(\lambda_x - \nu)(\lambda_y - \nu)}, \tag{3.4}$$

where λ_x and λ_y are the eigenvalues of D_{xx} and D_{yy} respectively ($\lambda_x, \lambda_y < 0$). It is convenient to write ξ as a function of the scaled eigenvalues $\mu_x := \lambda_x/\rho$ and $\mu_y := \lambda_y/\rho$, so that

$$\xi = \xi(\mu_x, \mu_y; \omega) = \frac{(\mu_x + \omega)(\mu_y + \omega)}{(\mu_x - \omega)(\mu_y - \omega)}, \tag{3.5}$$

where $\omega := \nu/\rho$. The parameter ω should be chosen in the range $0 < \omega \leqslant 1$ [5]. In Fig. 1. $\xi(\mu_x, \mu_y; \omega)$ is plotted for $\mu_x = \mu_y$ and for $\omega = 1, 10^{-1}, 10^{-2}, 10^{-3}$. For $\mu_y = a\mu_x (a \neq 1)$ the graph of $|\xi(\mu_x, \mu_y; \omega)|$ displays a similar behaviour.

From (3.3) one can easily see that the damping factor of the SADI scheme is given by

$$\xi = \xi(\lambda_x, \lambda_y; \nu) = \frac{\lambda_y - \nu - P_k(\lambda_x/\rho)(\lambda_x + \lambda_y)}{\lambda_x - \nu} \cdot \frac{\lambda_x - \nu - P_k(\lambda_y/\rho)(\lambda_x + \lambda_y)}{\lambda_y - \nu}, \tag{3.6a}$$

or equivalently as a function of μ_x and μ_y

$$\xi = \xi(\mu_x, \mu_y; \omega) = \frac{\mu_y - \omega - P_k(\mu_x)(\mu_x + \mu_y)}{\mu_x - \omega} \cdot \frac{\mu_x - \omega - P_k(\mu_y)(\mu_x + \mu_y)}{\mu_y - \omega}. \tag{3.6b}$$

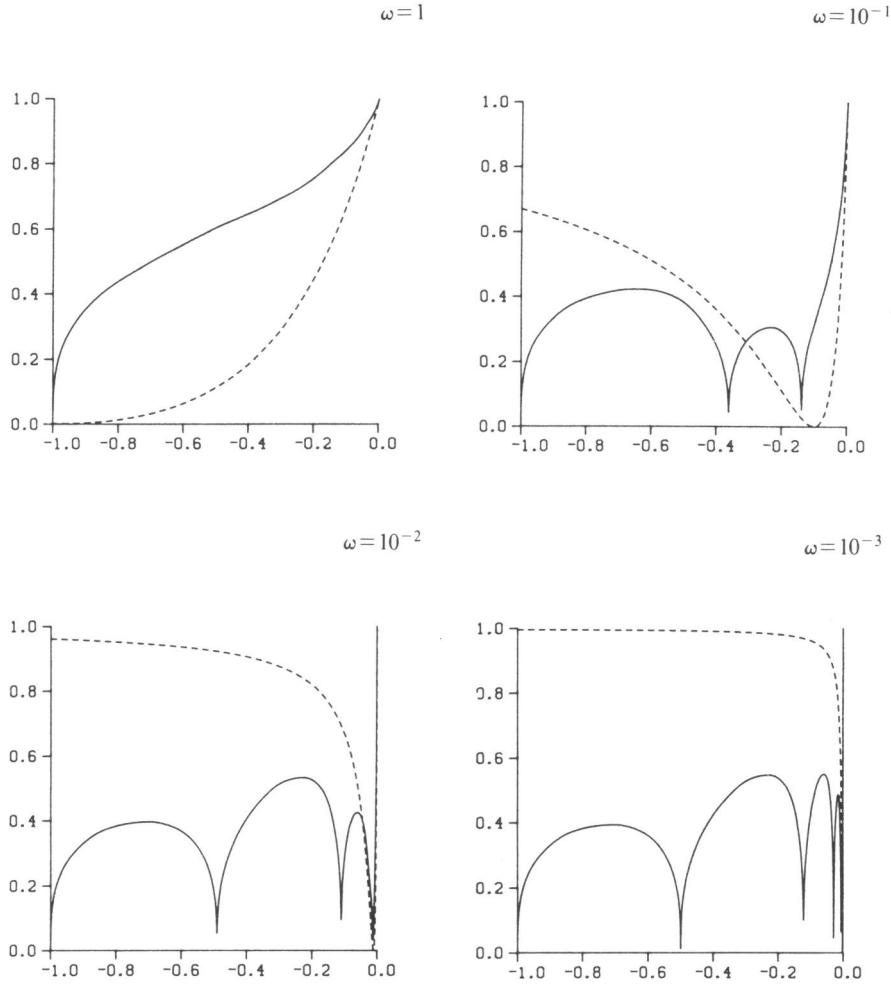


Fig. 1.a. The damping factors for the ADI scheme and the average damping factors for the SADI scheme on the interval $-1 \leqslant \mu \leqslant 0$ for $\omega = 1, 10^{-1}, 10^{-2}, 10^{-3}$.
----- ADI, ——— SADI.

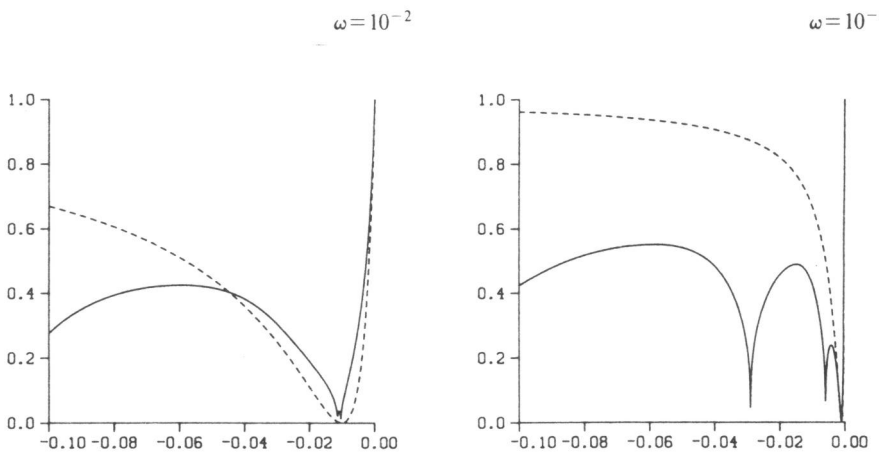


Fig. 1.b. The damping factors for the ADI scheme and the average damping factors for the SADI scheme on the interval $-0.1 \leqslant \mu \leqslant 0$ for $\omega = 10^{-2}, 10^{-3}$.
----- ADI, ——— SADI.

Note that $\xi(\mu_x, \mu_y; \omega) = 1$ in all points where $P_k(\mu_x) = P_k(\mu_y) = 0$. This implies that we should not iterate with a fixed value of k and ω. Therefore, we consider cyclic methods where $k = k_q$ and $\omega = \omega_q$, k_q and ω_q being periodic functions of q: $k_q = k_{q+N}$, $\omega_q = \omega_{q+N}$ with N fixed. In our experiments we choose $k_q = 2^q - 1 (q = 0(1)N-1)$ since then the smoothing matrices can be computed very efficiently [2]. The integer N will be specified later. In stead of $\xi = \xi_q(\mu_x, \mu_y; \omega_q)$ we thus consider the average damping factor

$$\alpha = \alpha(\mu_x, \mu_y; \omega_0, \ldots, \omega_{N-1}) := \left[\prod_{q=0}^{N-1} |\xi_q(\mu_x, \mu_y; \omega_q)| \right]^{1/N}. \tag{3.7}$$

Since $\xi_0(-1, -1; \omega_0) = \left(\dfrac{\omega_0 - 1}{\omega_0 + 1} \right)^2$ and $\xi_q(-1, -1; \omega_q) = 1$ for $q > 0$, we choose $\omega_0 = 1$ in order to damp the eigenvector components in the iteration error which correspond to values of μ_x, μ_y close to -1. These components are the high frequency components. Likewise, the low frequency components correspond to values of μ_x, μ_y close to 0. The other ω_q values are chosen equal: $\omega_q = \omega$ for $q > 0$. The average damping factor $\alpha = \alpha(\mu_x, \mu_y; \omega) := \alpha(\mu_x, \mu_y; 1, \omega \ldots \omega)$ of the SADI scheme is also plotted in Fig. 1. for $\mu_x = \mu_y, N = 6$ and $\omega = 1, 10^{-1}, 10^{-2}, 10^{-3}$. Also in this case, the graph of $\alpha(\mu_x, a\mu_x; \omega)$ $(a \neq 1)$ is very similar to the graph of $\alpha(\mu_x, \mu_y; \omega)$.

Comparing both damping factors, we see that for small ω-values $(10^{-3} \leqslant \omega \leqslant 10^{-2})$ the SADI scheme has substantial better damping properties than the ADI scheme. In particular, with the exception of the lowest ones $(\mu \approx 0)$, SADI damps all error components with a factor of a least 0.6.

4. CHOICE OF THE PARAMETER VALUES

In this section we derive parameter values for the SADI scheme. The derivation of parameter values for the ADI scheme (3.2) is extensively described in [5], therefore we only present the results.

The damping factor $\xi(\lambda_x,\lambda_y;\nu)$ of the ADI scheme in case 1 is given by (3.4). We choose the ν-parameter to minimize the function

$$\psi = \psi(\nu;\rho,\delta) := \max_{-\rho\leqslant\lambda_x,\lambda_y\leqslant-\delta} |\xi(\lambda_x,\lambda_y;\nu)|. \tag{4.1}$$

Asymptotically, the eigenvector corresponding to the maximum damping factor dominates the error. Therefore, in order to minimize the number of iterations, we have to minimize $\psi(\nu;\rho,\delta)$. We emphasize, however, that this only applies if we compute the solution sufficiently accurate. For moderate accurate computations, the ν-value thus obtained can be far from optimal, i.e., the corresponding number of iterations is far from minimal. A simple analysis gives that the optimal parameter is given by $\nu^* = (\delta\rho)^{\frac{1}{2}}$ [5].

EXAMPLE 1. Consider the Poisson equation. The eigenvalues λ_x and λ_y of D_{xx} and D_{yy} are given by $\lambda_{x,i} = \lambda_{y,i} = -\frac{4}{h^2}\sin^2(\frac{\pi}{2}ih), i = 1(1)M$, with $h = 1/(M+1)$. In this case $\rho(D_{xx}) = \rho(D_{xx}) = \rho \approx \frac{4}{h^2}$ and $\delta(D_{xx}) = \delta(D_{yy}) = \delta \approx \pi^2$, so that $\nu^* \approx \frac{2\pi}{h}$.

In case 2, the function ψ to be minimized is defined by

$$\psi = \psi(\nu;\rho_1,\delta_1,\rho_2,\delta_2) := \max_{\substack{-\rho_1\leqslant\lambda_x\leqslant-\delta_1\\ -\rho_2\leqslant\lambda_y\leqslant-\delta_2}} |\xi(\lambda_x,\lambda_y;\nu)|. \tag{4.1'}$$

Assume that $\rho_1\delta_1 \leqslant \rho_2\delta_2$. Then one can prove the following result for the ADI scheme [5]: if $\delta_1 \geqslant \delta_2$ or $\delta_1 \leqslant \delta_2$ and $\delta_1\rho_2 \geqslant \delta_2\rho_1$ then $\nu^* = (\delta_1\rho_1)^{\frac{1}{2}}$, and if $\rho_1 \geqslant \rho_2$ or $\rho_1 \leqslant \rho_2$ and $\delta_1\rho_2 \leqslant \delta_2\rho_1$ then $\nu^* = (\delta_2\rho_2)^{\frac{1}{2}}$.

Consider the SADI scheme. In case 1, the damping factor $\xi(\lambda_x,\lambda_y;\nu)$ is given by (3.6a). Since $\xi(\lambda_x,\lambda_y;\nu) = 1$ for all λ_x,λ_y for which $P_k(\lambda_x/\rho) = P_k(\lambda_y/\rho) = 0$, we have to iterate with varying $k = k_q$ and $\nu = \nu_q$ (see Section 3). Therefore, instead of $\xi = \xi_q(\lambda_x,\lambda_y;\nu_q)$ we consider the average damping factor α defined by (Cf. (3.7))

$$\alpha = \alpha(\lambda_x,\lambda_y;\nu_0,\ldots,\nu_{N-1}) := \left|\prod_{q=0}^{N-1} |\xi(\lambda_x,\lambda_y;\nu_q)|\right|^{1/N}. \tag{4.2}$$

In order to damp the high frequency components, we require $\xi_0(-\rho,\lambda_y;\nu_0) = \xi_0(\lambda_x,-\rho;\nu_0) = 0$, which gives $\nu_0 = \rho$. For the other ν_q-values we choose $\nu_q = \nu, q > 0$. This ν-value is chosen to minimize $\alpha(-\delta,-\delta;\nu) := \alpha(-\delta,-\delta;\rho,\nu,\ldots,\nu)$ because of the following reasons

(i) the lowest frequency eigenvector corresponding to $\lambda_x = \lambda_y = -\delta$ has often a large weight in the error

(ii) the eigenvalue $\lambda_x = \lambda_y = -\delta$ is either known or can be approximated.

In this way we construct a SADI scheme which damps the high- and low frequency components in the iteration error very well. It turns out that a SADI scheme constructed this way also damps the remaining error components very well, as illustrated before in Fig. 1.

So Consider $\alpha(-\delta,-\delta;\nu)$, which can be written as

$$\alpha(-\delta,-\delta;\nu) = \left((\frac{\rho-\delta}{\rho+\delta})^2 \prod_{q=1}^{N-1} \xi_q(-\delta,-\delta;\nu)\right)^{1/N}, \tag{4.3a}$$

with

$$\xi_q(-\delta,-\delta;\nu) = (1 - P_k(-\frac{\delta}{\rho})\frac{2\delta}{\delta+\nu})^2, k = 2^q - 1, q = 1(1)N-1. \tag{4.3b}$$

If $\xi_q(-\delta,-\delta;\nu) = 0$ for some $q > 0$ then $\alpha(-\delta,-\delta;\nu) = 0$, and thus $\alpha(-\delta,-\delta;\nu)$ is minimal. From

(4.3b) one can easily see that $\xi_q(-\delta,-\delta;\nu)=0$ if $\nu=\nu_k=\delta(2P_k(-\frac{\delta}{\rho})-1)$, provided $P_k(-\frac{\delta}{\rho})>\frac{1}{2}$. A Taylor series expansion yields

$$P_k(-\frac{\delta}{\rho})\approx 1-a_k\frac{\delta}{\rho},\ a_k:=\tfrac{1}{3}k(k+2), \tag{4.4}$$

if $b_k:=\frac{2}{45}(\frac{\delta}{\rho})^2(k+1)^4\ll 1$. For k sufficiently small, this condition is fulfilled and ν_k is approximately given by $\nu_k=c_k\delta$, $c_k:=1-2a_k\frac{\delta}{\rho}$. In our numerical experiments we take $\nu^*=\nu_1\approx\delta$ (see Table 1).

EXAMPLE 2. Consider again the Poisson equation for which $\rho\approx\frac{4}{h^2}$ and $\delta\approx\pi^2$. In this case $P_k(-\frac{\delta}{\rho})\approx 1-\frac{\pi^2}{12}(\frac{k+1}{M+1})^2$, $b_k=\frac{\pi^4}{360}(\frac{k+1}{M+1})^4$ and $c_k\approx 1-\frac{\pi^2}{6}(\frac{k+1}{M+1})^2$. These values for $k=2^q-1(q=1(1)5)$ and for $M=39$ are given in Table 1. Note that the value c_{31} does not make sense since $P_{31}(-\frac{\delta}{\rho})<\frac{1}{2}$. For the general elliptic case one finds similar results since the ratio $\frac{\delta}{\rho}=\mathcal{O}(h^2)$ just as for the Poisson equation.

k	$P_k(-\delta/\rho)$	b_k	c_k
1	0.9979	$1.69*10^{-6}$	0.9959
3	0.9918	$2.71*10^{-5}$	0.9836
7	0.9671	$4.33*10^{-4}$	0.9342
15	0.8684	$6.93*10^{-3}$	0.7368
31	0.4736	$1.11*10^{-1}$	-0.0528

Table 1. $P_k(-\frac{\delta}{\rho})-$, b_k- and c_k-values for the Poisson equation for $k=2^q-1(q=1(1)5)$ and $M=39$.

In case 2, the damping factor of the SADI scheme can be written as (Cf. (3.6a))

$$\xi=\xi(\lambda_x,\lambda_y;\nu_1,\nu_2)=\frac{\lambda_y-\nu_2-P_k(\lambda_x/\rho_1)(\lambda_x+\lambda_y)}{\lambda_x-\nu_1}\cdot\frac{\lambda_x-\nu_1-P_k(\lambda_y/\rho_2)(\lambda_x+\lambda_y)}{\lambda_y-\nu_2}. \tag{4.5}$$

Note that in (4.5) we assume that $\nu_1\neq\nu_2$. The corresponding average damping factor is given by (4.2) with $\xi=\xi_q(\lambda_x,\lambda_y;\nu_{1q},\nu_{2q})$ defined by (4.5). For the damping of the high frequency components we require $\xi_0(-\rho_1,\lambda_y;\nu_{10},\nu_{20})=\xi_0(\lambda_x,-\rho_2;\nu_{10},\nu_{20})=0$, which implies that we indeed should iterate with two different ν-values (Cf. (3.3)). This gives $\nu_{10}=\rho_2$ and $\nu_{20}=\rho_1$. For $q>0$ we choose $\nu_{1q}=\nu_1$ and $\nu_{2q}=\nu_2$. These two values are chosen to minimize $\alpha(-\delta_1,-\delta_2;\nu_1,\nu_2)$, which can be written as

$$\alpha(-\delta_1,-\delta_2;\nu_1,\nu_2)=\left(\frac{\rho_1-\delta_1}{\rho_2+\delta_1}\cdot\frac{\rho_2-\delta_2}{\rho_1+\delta_2}\prod_{q=1}^{N-1}|\xi_q(-\delta_1,-\delta_2;\nu_1,\nu_2)|\right)^{1/N}, \tag{4.6a}$$

with

$$\xi_q(-\delta_1,-\delta_2;\nu_1,\nu_2)=(1-P_k(-\frac{\delta_1}{\rho_1})\cdot\frac{\delta_1+\delta_2}{\delta_2+\nu_2})\cdot(1-P_k(-\frac{\delta_2}{\rho_2})\cdot\frac{\delta_1+\delta_2}{\delta_1+\nu_1})\cdot \tag{4.6b}$$

Also in this case, if $\xi_q(-\delta_1,-\delta_2;\nu_1,\nu_2)=0$ for some $q>0$ then $\alpha(-\delta_1,-\delta_2;\nu_1,\nu_2)$ is minimal as a function of ν_1 and ν_2. From (4.6b) one can readily see that this condition is fulfilled if $\nu_1=\nu_{1,k}=P_k(-\delta_2/\rho_2)(\delta_1+\delta_2)-\delta_1$ or $\nu_2=\nu_{2,k}=P_k(-\delta_1/\rho_1)(\delta_1+\delta_2)-\delta_2$ provided that $P_k(-\delta_2/\rho_2)>\delta_1/(\delta_1+\delta_2)$ or $P_k(-\delta_1/\rho_1)>\delta_2/(\delta_1+\delta_2)$. Substitution of the approximation $P_k(-\delta_i/\rho_i)=1-a_k\delta_i/\rho_i(i=1,2)$ (see (4.4)) then yields the following expression for $\nu_{1,k}$ and $\nu_{2,k}$: $\nu_{1,k}=\delta_2-a_k(\delta_2/\rho_2)(\delta_1+\delta_2)$ and $\nu_{2,k}=\delta_1-a_k(\delta_1/\rho_1)(\delta_1+\delta_2)$.

As in case 1, we choose the following approximation: $\nu_1^* = \nu_{1,1} \approx \delta_2$ and $\nu_2^* = \nu_{2,1} \approx \delta_1$.

For the computation of the parameter values for both schemes the values of $\delta(D_{xx}), \delta(D_{yy}), \rho(D_{xx})$ and $\rho(D_{yy})$ are required. As we have seen, for the Poisson equation $\rho(D_{xx}) = \rho(D_{yy}) \approx \frac{4}{h^2}$ and $\delta(D_{xx}) = \delta(D_{yy}) \approx \pi^2$. For the general elliptic equation (1.3) these values can only be approximated as follows. Consider the general matrix D_{xx} defined by (1.4a). Let $\bar{p} := \max_{0 \leqslant x,y \leqslant 1} p(x,y)$, $\underline{p} := \min_{0 \leqslant x,y \leqslant 1} p(x,y)$ and analogous definitions for $\bar{q}, \underline{q}, \bar{w}$ and $\underline{w}$. Let the matrices $\bar{D}_{xx}$ and $\underline{D}_{xx}$ be defined by replacing $p_{i \pm \frac{1}{2}, j}$ and w_{ij} in (1.4a) by $\bar{p}$ and $\bar{w}$ respectively $\underline{p}$ and $\underline{w}$. In other words, $\bar{D}_{xx} = \bar{p}\delta_{xx} - \frac{1}{2}\bar{w}I$ and $\underline{D}_{xx} = \underline{p}\delta_{xx} - \frac{1}{2}\underline{w}I$, with δ_{xx} denoting the standard central difference approximation to $\frac{\partial^2}{\partial x^2}$. Then one can easily show that $\rho(\underline{D}_{xx}) \leqslant \rho(D_{xx}) \leqslant \rho(\bar{D}_{xx})$ and $\delta(\underline{D}_{xx}) \leqslant \delta(D_{xx}) \leqslant (\bar{D}_{xx})$. The values $\rho(D_{xx})$ and $\delta(D_{xx})$ can then be approximated by $\rho(D_{xx}) \approx \frac{1}{2}(\rho(\bar{D}_{xx}) + \rho(\underline{D}_{xx})) = \frac{2}{h^2}(\bar{p} + \underline{p}) + \frac{1}{4}(\bar{w} + \underline{w})$ and $\delta(D_{xx}) \approx \frac{1}{2}(\delta(\bar{D}_{xx}) + \delta(\underline{D}_{xx})) = \frac{\pi^2}{2}(\bar{p} + \underline{p}) + \frac{1}{4}(\bar{w} + \underline{w})$. In the same way one finds $\rho(D_{yy}) \approx \frac{2}{h^2}(\bar{q} + \underline{q}) + \frac{1}{4}(\bar{w} + \underline{w})$ and $\delta(D_{yy}) \approx \frac{\pi^2}{2}(\bar{q} + \underline{q}) + \frac{1}{4}(\bar{w} + \underline{w})$.

5. Numerical examples

In this section we present a few numerical examples, in order to compare the ADI scheme and the SADI scheme. We restrict ourselves to Dirichlet problems. The solution is computed for $h = \frac{1}{20}, \frac{1}{40}, \frac{1}{80}$ with the parameter values derived in Section 4. In addition, we compute the solution for $h = \frac{1}{40}$ for several ν-values, in order to check whether the ν-values derived in Section 4 are good enough. Further, to demonstrate the power of residual smoothing, we apply the SADI scheme to a nonlinear problem.

For the degree k of the smoothing matrices we choose $k = k_q = 2^q - 1$, $q = 0(1)N - 1$, such that k_{N-1} is the largest k_q smaller than $M = h^{-1} - 1$. The reason for this is, that for $k_q > M$ for some q, the computation of the smoothing matrices becomes cumbersome. Thus for $h = \frac{1}{20}, \frac{1}{40}, \frac{1}{80}$ we choose, respectively, $N = 5,6,7$. We emphasize once more that the choice $k_q = 2^q - 1$ admits an efficient computation of the smoothing matrices [2], which is a prerequisite for accelerating the ADI scheme. In all computations, the initial approximation is defined by forming linear interpolations of the boundary values on $x = 0$, $x = 1$ and on $y = 0$, $y = 1$, respectively, and by taking the average value of these functions. The iteration is stopped if the scaled residual

$$r(n) := \frac{\|AU^n - B\|_1}{\|AU^0 - B\|_1} \tag{5.1}$$

has dropped below a certain tolerance TOL.

The examples we consider are the following.

Example 1 [4, p. 427]

$$u_{xx} + u_{yy} = f(x,y)$$

$$u(x,y) = 3e^{x+y}(x - x^2)(y - y^2), f(x,y) = 6xye^{x+y}(xy + x + y - 3)$$

$$\rho = \rho(D_{xx}) = \rho(D_{yy}) = \frac{4}{h^2}, \ \delta = \delta(D_{xx}) = \delta(D_{yy}) = \pi^2.$$

Example 2

$(e^x u_x)_x + (e^y u_y)_y = f(x,y)$

$u(x,y) = (xy)^3$, $f(x,y) = 3xy((2+x)y^2 e^x + x^2(2+y)e^y)$

$\rho = \rho(D_{xx}) = \rho(D_{yy}) = \frac{2}{h^2}(e+1)$, $\delta = \delta(D_{xx}) = \delta(D_{yy}) = \frac{\pi^2}{2}(e+1)$

Example 3

$(e^{-xy} u_x)_x + (e^{xy} u_y)_y - (x+y)u = f(x,y)$

$u(x,y) = (xy)^3$, $f(x,y) = 3xy^3(2-xy)e^{-xy} + 3x^3 y(2+xy)e^{xy} - (x+y)(xy)^3$.

$\rho_1 = \rho(D_{xx}) = \frac{1}{e}\cdot\frac{2}{h^2}(e+1) + \frac{1}{2}$, $\delta_1 = \delta(D_{xx}) = \frac{1}{e}\frac{\pi^2}{2}(e+1) + \frac{1}{2}$, $\rho_2 = \rho(D_{yy}) = \frac{2}{h^2}(e+1) + \frac{1}{2}$,

$\delta_2 = \delta(D_{yy}) = \frac{\pi^2}{2}(e+1) + \frac{1}{2}$.

Example 4

$(e^u u_x)_x + (e^u u_y)_y - w(x,y,u) = 0$

$u(x,y) = (xy)^2$, $w(x,y,u) = 2(x^2+y^2)(1+2x^2y^2)e^u$.

Note that the matrices D_{xx} and D_{yy} commute for the first two examples but not for the third one. Note that Example 4 is a nonlinear problem. Like the ADI scheme, the SADI scheme can be applied to nonlinear problems in a straightforward manner. We have included this example, in order to show the power of the residual smoothing technique.

Consider the first three examples. First we present results for $h = \frac{1}{20}, \frac{1}{40}, \frac{1}{80}$ obtained with the ν-values derived in Section 4. The results are collected in Table 2, which contains the following values: the total number of iterations n_0, the average reduction factor $\bar{r}$ defined by $\bar{r} := r(n_0)^{1/n_0}$ (Cf. (5.1)) and the computing time (CT) in seconds needed for the iteration process. For the tolerance we take $TOL = 10^{-8}$; similar results are obtained for larger values of TOL. From Table 2 we see that, especially on the finer grids, the SADI scheme needs much less iterations than the ADI scheme, which results in a considerable reduction of CT.

Next we present results obtained on a 40*40 grid for several ν-values, with the purpose of testing the ν-parameter values derived in Section 4. Case 1 ($\rho = \rho(D_{xx}) = \rho(D_{yy})$, $\delta = \delta(D_{xx}) = \delta(D_{yy})$) applies to the first two examples. Instead of ν, consider for these two examples the scaled parameter $\omega = \nu/\rho$. One can readily see that $\omega^* = \frac{\nu^*}{\rho} = (\frac{\delta}{\rho})^{\frac{1}{2}} = 0.039269908$ for the ADI scheme and $\omega^* = \frac{\delta}{\rho} = 0.001542126$ for the SADI scheme. Case 2 ($\rho_1 = \rho(D_{xx}) \neq \rho_2 = \rho(D_{yy})$, $\delta_1 = \delta(D_{xx}) \neq \delta_2 = \delta(D_{yy})$) applies to Example 3. Let in this case $\omega := \nu/\rho_1$, then one can easily see that for the ADI scheme $\omega^* = (\frac{\delta_1}{\rho_1})^{\frac{1}{2}} = 0.040696$. Since $\rho_2 \approx e\rho_1$ and $\delta_2 \approx e\delta_1$, it is obvious to choose $\nu_1 = e\nu$ and $\nu_2 = \nu$ for the SADI scheme. The ω^*-value is then given by $\omega^* = \frac{\delta_1}{\rho_1} = 0.001656164$. The number of iterations, for $TOL = 10^{-8}$, are presented in Table 3. We may conclude that the parameter values derived in Section 4 are fairly good since the corresponding number of iterations is nearly minimal. Furthermore, we see that in the range $10^{-3} \leq \omega \leq 10^{-2}$, the SADI scheme is less

ADI

	example 1			example 2			example 3		
h^{-1}	n_0	$\bar{r}$	CT	n_0	$\bar{r}$	CT	n_0	$\bar{r}$	CT
20	58	0.73	0.702	67	0.76	1.263	76	0.78	1.397
40	116	0.85	5.301	138	0.87	11.069	155	0.89	11.042
80	231	0.92	41.196	279	0.94	76.486	312	0.94	86.092

SADI

	example 1			example 2			example 3		
h^{-1}	n_0	$\bar{r}$	CT	n_0	$\bar{r}$	CT	n_0	$\bar{r}$	CT
20	18	0.33	0.369	21	0.42	0.512	26	0.49	0.747
40	21	0.40	1.781	27	0.49	3.306	34	0.58	4.080
80	25	0.45	9.219	31	0.55	15.490	43	0.64	17.712

Table 2. The n_0-,$\bar{r}-$ and CT-values for the first three examples.

	example 1		example 2		example 3	
ω	ADI	SADI	ADI	SADI	ADI	SADI
$5*10^{-2}$	147	200	143	188	166	159
10^{-2}	100	41	267	39	220	34
$5*10^{-3}$	199	21	>500	26	440	31
10^{-3}	>500	22	>500	27	>500	37
ω^*	116	21	138	27	155	34

Table 3. The n_0-values for $h=\frac{1}{40}$ and various values of ω, for the first three examples.

	ADI			SADI		
h^{-1}	n_0	$\bar{r}$	CT	n_0	$\bar{r}$	CT
20	27	0.71	13.013	12	0.45	5.961
40	95	0.91	194.378	14	0.51	30.030

Table 4. The n_0-,$\bar{r}-$ and CT-values for Example 4.

sensitive to the choice of the parameter values than the ADI scheme. Thus, a ω-value which differs a little from the ω^*-value can lead to considerably extra computing time for the ADI scheme, but not so for the SADI scheme.

Consider Example 4. Application of the ADI scheme or the SADI scheme to this nonlinear problem requires at each iteration the solution of a set of nonlinear tridiagonal systems, for which we use Newton iteration. Results for $h=\frac{1}{20}, \frac{1}{40}$ and for $\mathrm{TOL}=10^{-4}$ are presented in Table 4. The best ω-values are experimentally found to be $\omega^*=10^{-1}$ for the ADI scheme and $\omega^*=10^{-2}$ for the SADI scheme. From this table we see that residual smoothing leads to a considerable saving of the number of iterations and hence also of the computing time. Note that in this case the gain in computing time is even more than for the first three examples, since one ADI iteration is now very expensive compared to the computation of the smoothing matrices.

6. An alternative smoothed ADI scheme

In this section we briefly consider an alternative to the SADI scheme (3.3). For this purpose, we rewrite the ADI scheme (3.2) in the one-stage form

$$(D_{xx}-\nu_1 I)(D_{yy}-\nu_2 I)U^{n+1} = (D_{xx}-\nu_1 I)(D_{yy}-\nu_2 I)U^n + (\nu_1+\nu_2)(AU^n - B). \tag{6.1}$$

The idea is now to multiply the residual in (6.1) by the smoothing matrices $\tilde{S}_x$ and $\tilde{S}_y$ (see Section 3):

$$(D_{xx}-\nu_1 I)(D_{yy}-\nu_2 I)U^{n+1} = (D_{xx}-\nu_1 I)(D_{yy}-\nu_2 I)U^n + (\nu_1+\nu_2)\tilde{S}_y\tilde{S}_x(AU^n - B). \tag{6.2}$$

For brevity, we restrict ourselves to case 1 and assume that $\nu_1=\nu_2=\nu$. The damping factor of scheme (6.2), as a function of μ_x and μ_y can then be written as

$$\xi = \xi(\mu_x,\mu_y;\omega) = 1+\frac{2\omega(\mu_x+\mu_y)}{(\mu_x-\omega)(\mu_y-\omega)}\cdot P_k(\mu_x)P_k(\mu_y), \tag{6.3}$$

where $\omega=\nu/\rho$. The corresponding average damping factor α is then given by (3.7) with $\xi=\xi_q(\mu_x,\mu_y;\omega_q)$ defined in (6.3). In order to damp the high frequency error components, we choose $\omega_0=1$ and $\omega_q=\omega$ for $q=1(1)N-1$ (see Section 3). The average damping factor $\alpha=\alpha(\mu_x,\mu_y;\omega)$ is plotted in Fig. 2. for $\mu_x=\mu_y, N=6$ and $\omega=1,10^{-1},10^{-2},10^{-3}$. Comparing Fig. 1. and Fig. 2. it is apparent that the SADI scheme gives a much better "overall" damping of the iteration error than the alternative scheme.

As an illustration, we apply the alternative scheme (6.2) to Example 1 for $h=\frac{1}{40}$ and for various values of the parameter ω. For the tolorance TOL we take TOL$=10^{-8}$. The results are presented in Table 5. From Table 3 and Table 5 one can readily see that scheme (6.2) is slightly faster than the ADI scheme, however, much slower than the SADI scheme. Thus, the SADI scheme is clearly to be preferred to the alternative scheme (6.2).

ω	$5*10^{-2}$	10^{-2}	$5*10^{-3}$	10^{-3}
n_0	219	79	105	229

Table 5: n_0-values for $h=\frac{1}{40}$ and various ω-values for Example 1.

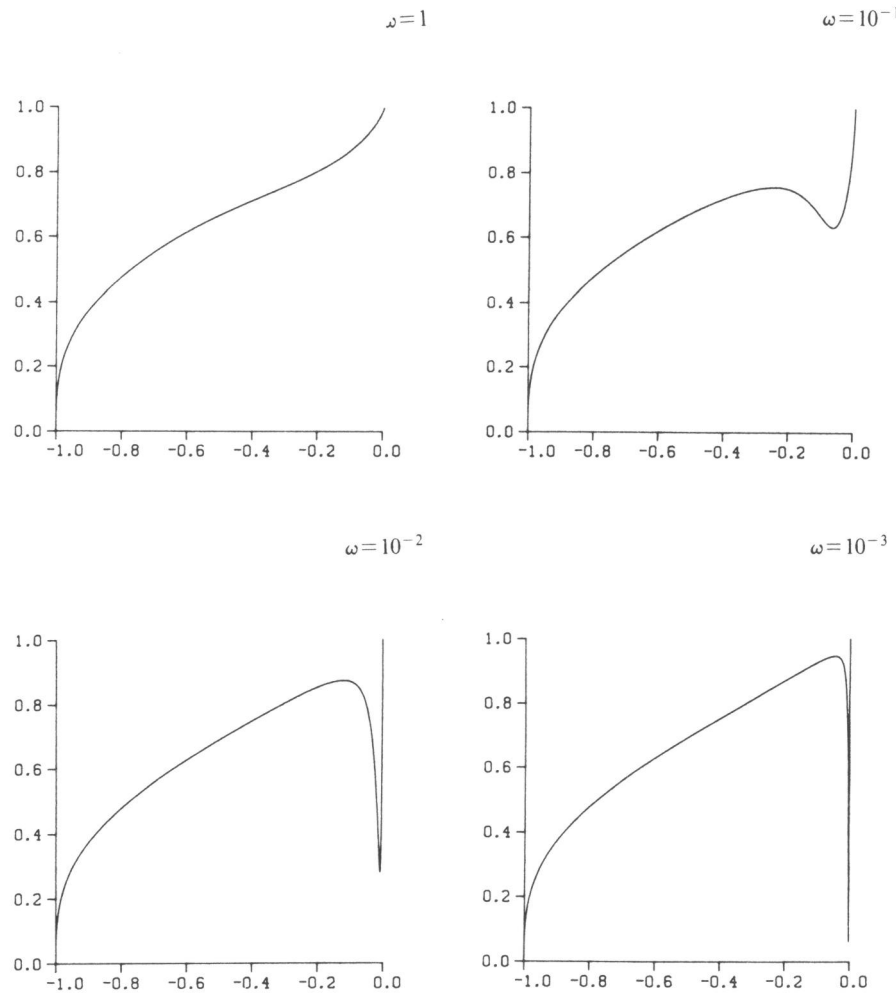


Fig. 2. The average damping factor for scheme (6.2) for $\omega = 1, 10^{-1}, 10^{-2}, 10^{-3}$.

7. Concluding remarks

In this paper we considered residual smoothing as a means to accelerate the convergence of the ADI scheme for elliptic difference equations. Concerning this technique we note the following.

(i) Residual smoothing can be easily applied to general elliptic problems, even to nonlinear problems, to speed up iterative methods such as the ADI method.

(ii) For a proper choice of the degree of smoothing k ($k = 2^q - 1$ for some integer $q \geq 0$), residual smoothing can be implemented very efficiently.

(iii) Residual smoothing can be combined with the ADI scheme in several ways. When it is applied in the right way, as is done for the SADI scheme (3.3), residual smoothing can lead to a considerable reduction of the number of iterations and the computing time for the ADI scheme.

(iv) The parameters for the SADI scheme are chosen such that the high- and low frequency components in the iteration error are rapidly damped. Due to the residual smoothing, the other components in the error are also properly damped.

(v) For a certain range of the parameter values, the SADI scheme is much less sensitive to the choice of these values that the ADI scheme.

References

[1] G. Birkhoff & R.E. Lynch, Numerical solution of elliptic problems, SIAM, Philadelphia (1984).

[2] P.J. van der Houwen, C. Boon & F.W. Wubs, Analysis of smoothing matrices for the preconditioning of elliptic difference equations, Z. Angew. Math. Mech 68, 3-10 (1988).

[3] D.W. Peaceman & H.H. Rachford, The numerical solution of parabolic and elliptic differential equations, J. Soc. Ind. Appl. Math. 3, 28-41 (1955).

[4] J.R. Rice & R.F. Boisvert, Solving elliptic problems using ELLPACK, Springer-Verlag, New York (1984).

[5] D.M. Young, Iterative solution of large linear systems, Academic Press, New York (1971).

Samenvatting

In dit proefschrift beschouwen we numerieke methoden voor de berekening van tijdsafhankelijke, onsamendrukbare vloeistof- stromingen. Het proefschrift bestaat uit twee delen. Het eerste deel geeft een korte wiskundige beschrijving van onsamendrukbare vloeistofstromingen, en dient als achtergrond voor de artikelen welke in het tweede deel worden gepresenteerd. Deze artikelen bevatten een gedetailleerde beschrijving van enkele numerieke technieken, welke een rol spelen in de berekening van tiids- afhankelijke, onsamendrukbare vloeistofstromingen.

Deel I

Voor de stroming van een onsamendrukbare vloeistof kan men de volgende drie beschrijvingswijzen onderscheiden: de primitieve- variabelen-formulering, de vorticiteit-stroomfunctie-formulering en de biharmonische-stroomfunctie-formulering. Het verschil tussen deze drie formuleringen is gelegen in de keuze van de variabelen. De primitieve-variabelen-formulering maakt gebruik van de snelheid en de druk in de stroming als beschrijvende variabelen. Deze formulering is, zeker voor drie-dimensionale berekeningen, de meest gebruikte beschrijvingswijze voor onsamendrukbare stromingen. In het proefschrift beperken we ons tot deze formulering.

Voor de primitieve-variabelen-formulering kan men ruwweg twee klassen van oplossingsmethoden onderscheiden, nl. de impliciete methoden en de zogenaamde druk-correctie-methoden. Het kenmerk van impliciete methoden is dat het snelheidsveld en de druk simultaan worden uitgerekend m.b.v. een iteratieve procedure. Druk-correctie-methoden daarentegen, zijn methoden waarbij de berekening van de snelheid en de druk zijn ontkoppeld. De druk wordt nu rechtstreeks berekend uit een Poisson-vergelijking. In dit proefschrift beperken we ons tot de druk- correctie-methoden.

Deel II

In de druk-correctie-methoden kan men globaal de volgende drie stappen onderscheiden:

— plaatsdiscretisatie
— tijdsintegratie
— oplossen van een Poisson-vergelijking voor de druk.

Er bestaan verschillende plaatsdiscretisatie-technieken, zoals bijvoorbeeld de eindige differentiemethode en de eindige elementenmethode. In dit proefschrift beschouwen we uitsluitend de eindige differentiemethode.

Er bestaan vele tijdsintegratie-technieken welke men kan gebruiken in een druk-correctie-methode. Men kan de voldende gevallen onderscheiden: expliciete methoden, impliciete methoden en splitmethoden. In dit proefschrift beperken we ons tot de belangrijke klasse van splitmethoden. Voorbeelden hiervan zijn het 'odd-even hopscotch' (OEH)- schema en het 'alternating

direction implicit' (ADI)-schema.

In het eerste artikel wordt het OEH-schema besproken voor meerdimensionale convectie-diffusievergelijkingen. Met name de von Neumann-stabiliteit van het schema voor een klasse van lineaire convectie-diffusie- vergelijkingen wordt onderzocht. Het schema wordt getest aan de hand van de Burgers-vergelijkingen, welke een eenvoudig model vormen voor de vergelijkingen van een onsamendrukbare vloeistofstroming (in primitieve-variabelen-formulering).

Het vierde artikel bespreekt vectorizatie-aspekten van zowel het OEH-schema als het ADI-schema, voor het oplossen van de Burgers-vergelijkingen. Beide schemas zijn geimplementeerd op de vectorcomputers Cyber 205 en Cray X-MP/24. Datastructuren en oplossingstechnieken welke worden gebruikt voor het vectorizeren van beide schemas, worden kort besproken. Een evaluatie van beide schemas wordt gegeven.

In het tweede artikel wordt het OEH-schema gecombineerd met een snelle oplossingsmethode voor de Poisson-vergelijking, voor het berekenen van onsamendrukbare vloeistofstromingen. Het totale schema wordt het 'odd-even hopscotch pressure-correction' (OEH-PC)-schema genoemd. Nauwkeurigheid en efficiëntie van het schema worden onderzocht aan de hand van een eenvoudig testprobleem. Een meer praktisch testprobleem heeft betrekking op de stroming door een reservoir. Het derde artikel is een uitbreiding van het tweede artikel. In dit artikel wordt het OEH-PC schema toegepast voor de berekening van vrije convectie, d.w.z. de stroming veroorzaakt door een temperatuurgradient in de vloeistof. In dit geval moet het stelsel vergelijkingen voor een onsamendrukbare vloeistofstroming worden uitgebreid met een convectie-diffusievergelijking voor de temperatuur.

Er bestaan tegenwoordig vele snelle oplossingsmethoden voor de Poisson-vergelijking, zoals bijvoorbeeld de 'Fast Fourier Transform'-methode en vele multigrid methoden. In het vijfde artikel wordt een nieuwe methode besproken voor het efficiënt oplossen van elliptische vergelijkingen (waaronder de Poisson-vergelijking). In dit artikel wordt de klassieke ADI- iteratiemethode gecombineerd met het zogenaamde 'residual smoothing'. Dit is een eenvoudige techniek voor het versnellen van de convergentie van iteratieve methoden voor elliptische differentievergelijkingen. Wanneer 'residual smoothing' op de juiste manier wordt gecombineerd met de ADI- iteratiemethode, leidt dit tot een aanzienlijke reductie van het aantal iteraties en de rekentijd van de ADI-methode.