Comparing time integrators for parabolic equations in two space dimensions with a mixed derivative

P. J. van der Houwen, B. P. Sommeijer
and J. G. Verwer (*)

ABSTRACT

A numerical comparison is made between three integration methods for semi-discrete parabolic partial differential equations in two space variables with a mixed derivative. Linear as well as non-linear equations are considered. The integration methods are the well-known one-step line hopscotch method, a four-step line hopscotch method, and a stabilized, explicit Runge-Kutta method.

1. INTRODUCTION

In the numerical integration of systems of ordinary differential equations

$$y' = f(t, y), \tag{1.1}$$

which are supposed to represent *semi-discrete* parabolic differential equations, that is, parabolic equations of which the space variables have been discretized and the time variable is left continuous, one usually distinguishes between three categories of integration formulas :

Explicit formulas [6, 10], such as forward Euler, which do not require the solution of any system of linear or non-linear equations;

Implicit and semi-implicit formulas [6], such as backward Euler or Rosenbrock-type methods, which always require the solution of systems of linear or non-linear equations containing the Jacobian matrix $\partial f / \partial y$;

Splitting formulas [1, 7], such as ADI or hopscotch, which are mostly partly implicit and partly explicit.

Each category has its advantages and its disadvantages. A clear advantage of an explicit formula is that it is easy to apply and applicable to a very general class of problems. A clear disadvantage of an explicit formula - at least of a standard one - is that it necessarily possesses conditional stability properties, in contrast with implicit formulas which may be unconditionally stable. The application of implicit formulas, however, requires the solution of large systems of equations, which, in case of more than one space variable, may be very time-consuming. Hence, the solution of these systems can annul the advantage of unrestricted stepsizes. To circumvent these time-consuming methods of solution, splitting formulas have been introduced. The advantage of this approach is that for relevant classes of problems a stepwise efficient process can be obtained, which, in addition, usually possesses unconditional stability properties. A disadvantage of splitting formulas is that they are not easily applicable to very general classes of problems.

From the foregoing it shall be clear that a sensible choice of an integration method for semi-discrete parabolic equations is difficult.

For example, if we consider systems of equations in 2 or 3 space variables, stabilized, explicit Runge-Kutta methods may be attractive [6, 9]. On the other hand, if we restrict ourselves to one space dimension fully implicit methods are obvious [4], because in this case the sets of linear or non-linear equations to be solved are not so complicated.

It is the purpose of this paper to contribute to such a choice for *scalar* parabolic equations in *two space dimensions* with a *mixed derivative*. We discuss test results of three time integrators, two of which are based on a splitting formula, namely on the *one-step line hopscotch* formula of Gourlay & McGuire [2], and on a *four-step line hopscotch* formula which is given in [8]. As pointed out by Gourlay & McKee [3], the choice of the second order line hopscotch method is, within the class of one-step splitting methods, self-evident. The four-step method, of which a second order and a fourth order implementation is considered, is chosen in order to investigate whether this improves the one-step method. The third time integrator is based on the *explicit, stabilized three-step Runge-Kutta* formulas from [9]. By comparing this explicit method

(*) P. J. van der Houwen, B. P. Sommeijer, J. G. Verwer, Stichting Mathematisch Centrum, 2de Boerhaavestraat 49, 1091 AL Amsterdam, Nederland.

with the two splitting methods - for our scalar problem class - we obtain at the same time an indication on its use for systems of parabolic equations. Because, for systems, the splitting formulas are more difficult to apply and are less efficient per integration step, whereas the stepwise efficiency of the stabilized, explicit formulas does not change with the number of parabolic equations, as these formulas do not require the solution of any set of equations. We did not consider any member from the class of implicit and semi-implicit formulas.

2. THE THREE TIME INTEGRATORS

In this section we describe the actual implementations - being used in our tests - of the three integration methods considered. For clarity, throughout the paper it is assumed that the semi-discrete parabolic equations, which are allowed to be *arbitrarily non-linear,* are in *explicit form,* that is, of type (1.1). In the next section we shall discuss the semi-discretization being used. In order to describe the line hopscotch formulas, we merely assume that the components of the vector functions y and f are 9-*point coupled.* To describe the stabilized Runge-Kutta formulas no special knowledge on (1.1) is needed.

2.1. The stabilized Runge-Kutta formulas

These formulas are the explicit, three-step Runge-Kutta formulas of *second order,* which are given in [9]. The main characteristic of them is that they have *stability regions* containing a *long narrow strip* around the negative axis of the complex plane.
Let y_n denote the approximation to $y(t_n)$, let $\tau = t_{n+1} - t_n$ denote the (constant) steplength. The approximation y_{n+1} at the next time point $t = t_{n+1}$ is defined by

$$y_{n+1}^{(0)} = y_n,$$

$$y_{n+1}^{(j)} = (1-b_j)\, y_n + b_j y_{n-1} + c_j \tau f(t_{n-1}, y_{n-1})$$

$$+ \lambda_j \tau f(t_n + \mu_{j-1}\tau, y_{n+1}^{(j-1)}), \qquad j = 1(1)m,$$

$$y_{n+1} = d\, y_{n+1}^{(m)} + (1-d)\, y_{n-2}, \tag{2.1}$$

where $\mu_0 = 0$, $\mu_j = -b_j + c_j + \lambda_j$, $j = 1(1)m-1$,

and m, the *degree* of the formula, that is, the number of f-evaluations, varies between 2 and 12. Thus (2.1) represents a family of 11 formulas. To be able to apply (2.1), two starting vectors should be given. In our experiments these starting vectors are obtained from the exact solution of the parabolic equation, which is always known (see section 3).
It shall be clear that, once second order is established, the greater part of the parameters b_j, c_j, λ_j, and d, is still free. For each m, $2 \leq m \leq 12$, they have been determined to obtain almost maximal *real stability boundaries,* say $\beta(m)$. There holds

$$\beta(m) \approx 2.29\, m^2, \tag{2.2}$$

hence the real stability boundaries of the family of formulas (2.1) approximately vary between 9.16 and 329.76. The *effective* boundaries $\beta(m)/m$ approximately vary between 4.58 and 27.48. In [9], also a set of first order formulas is given, for which $\beta(m) \approx 5.15m^2$. To save space, we do not give the expressions for the integration parameters b_j, c_j, λ_j and d.
An important observation is that, to exploit the large stability boundaries, the *eigenvalues* of $\partial f/\partial y$ should be *real or almost real.* If they are not, the stabilization is to no purpose. Fortunately, the greater part of the semi-discrete parabolic equations satisfy this requirement. Also of importance is to have an upper estimate of the *spectral radius,* say σ, of $\partial f/\partial y$ over the range of integration. Once σ is known, and τ is fixed beforehand, the degree m can then be minimized in the *stability condition*

$$\tau\sigma \leq \beta(m). \tag{2.3}$$

In all our experiments we specify an upper estimate of σ (valid for the whole range of integration) and always minimize m according to (2.3). These estimates are obtained automatically using the power method of the program M3RK discussed in [10]. The testing strategy shall be discussed in section 4.

2.2. The one-step line hopscotch formula

To be able to give our formulation (see [2, 7]) of the one-step line hopscotch method in a compact way, we assume that each component of the 9-point coupled, semi-discrete equation is associated to an *internal grid point* of a two-dimensional grid. More precisely, we specifically use the fact that we consider scalar parabolic equations in two space dimensions with a mixed derivative, of which discrete boundary relations are always explicitly solvable.
Let Ω_h be the set of internal grid points. Divide Ω_h into 4 subsets, say Ω_o, $\Omega_\bullet$, Ω_+ and Ω_x, in a way as schematically shown in fig. 2.1. Now, related to these sets, we introduce the vector functions f_o, $f_\bullet$, f_+, f_x, satisfying

$$f = f_o + f_\bullet + f_+ + f_x \tag{2.4}$$

Thus, for grid points from Ω_o, the components of $f_o(t, y)$ are equal to the components of $f(t, y)$, while all other components of $f_o(t, y)$ are zero, and so on.

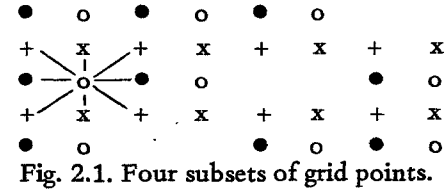


Fig. 2.1. Four subsets of grid points.

Next we define the vector functions

$$f_{\bullet o} = f_{\bullet} + f_{o},\quad f_{+x} = f_{+} + f_{x}. \tag{2.5}$$

Our one-step line hopscotch formula is now given by

$$y_{n+1}^{(1)} = y_n + \frac{\tau}{2} f_{+x}(t_n, y_n) + \frac{\tau}{2} f_{\bullet o}(t_n + \frac{\tau}{2}, y_{n+1}^{(1)}),$$

$$y_{n+1} = y_{n+1}^{(1)} + \frac{\tau}{2} f_{\bullet o}(t_n + \frac{\tau}{2}, y_{n+1}^{(1)}) + \frac{\tau}{2} f_{+x}(t_n + \tau, y_{n+1}). \tag{2.6}$$

In this formula $y_{n+1}^{(1)}$ is considered as an intermediate approximation. By computing $y_{n+1}^{(1)}$ first at the grid points $\in \Omega_{+} \cup \Omega_{x}$, and then at the points $\in \Omega_{\bullet} \cup \Omega_{o}$, only systems of (generally) non-linear equations with a *tridiagonal Jacobian* are to be solved. The same holds for y_{n+1}, but now in the reserved order, which serves as the approximation at the new time point $t_{n+1} = t_n + \tau$. It is easy to see that (2.6) is of *second order* consistency. Note that the splitting of f, defined by (2.5), is along grid lines in the $\bullet o$ or $+x$ direction. In a similar way the splitting may be defined in the other direction.

An attractive property of the hopscotch method is that the explicit f-evaluations can be saved by re-writing it in the so-called fast-form [2]. If we first rewrite (2.6) as (the notation is now self-evident)

$$(y_{n+1}^{(1)})_{+x} = (y_n)_{+x} + \frac{\tau}{2} f_{+x}(t_n, y_n),$$

$$(y_{n+1}^{(1)})_{\bullet o} = (y_n)_{\bullet o} + \frac{\tau}{2} f_{\bullet o}(t_n + \frac{\tau}{2}, y_{n+1}^{(1)}),$$

$$(y_{n+1})_{\bullet o} = (y_{n+1}^{(1)})_{\bullet o} + \frac{\tau}{2} f_{\bullet o}(t_n + \frac{\tau}{2}, y_{n+1}^{(1)}),$$

$$(y_{n+1})_{+x} = (y_{n+1}^{(1)})_{+x} + \frac{\tau}{2} f_{+x}(t_n + \tau, y_{n+1}), \tag{2.7}$$

the fast-form is easily recognized as

$$\begin{aligned}
&n = 0,\quad (y_{n+1}^{(1)})_{+x} = (y_n)_{+x} + \tfrac{1}{2}\tau f_{+x}(t_n, y_n),\\
&\quad (y_{n+1}^{(1)})_{\bullet o} = (y_n)_{\bullet o} + \tfrac{1}{2}\tau f_{\bullet o}(t_n + \tfrac{1}{2}\tau, y_{n+1}^{(1)}),\\
&\quad (y_{n+1})_{\bullet o} = 2(y_{n+1}^{(1)})_{\bullet o} - (y_n)_{\bullet o},\\
&\quad (y_{n+1})_{+x} = (y_{n+1}^{(1)})_{+x} + \tfrac{1}{2}\tau f_{+x}(t_n + \tau, y_{n+1}),\\
&\quad n = n + 1,\\
&\quad (y_{n+1}^{(1)})_{+x} = 2(y_n)_{+x} - (y_n^{(1)})_{+x}.
\end{aligned} \tag{2.8}$$

This is the line hopscotch formula we implemented. In our tests with this implementation (see section 4 for a discussion of the testing strategy) the systems of non-linear equations are always solved, approximately, by performing 2 Newton-type iterations. Consequently, in case of *non-linear problems* $dy/dt = f(t, y)$ each integration step ($n > 0$) costs 2 *complete f-evaluations*, provided the tridiagonal Jacobian matrices are available. This is easy to verify by inspection of formula (2.8). In case of linear problems, say $dy/dt = Jy + g(t)$, J a constant matrix, we perform 1 iteration using the same Newton-type process. Hence, in case of *linear problems*, each integration step costs *1 complete f-evaluation*. In the Newton-type process for the unknowns $(y_{n+1}^{(1)})_{\bullet o}$ we use y_n as predictor. This is sensible, as $y_{n+1}^{(1)}$ is of first order at the point $t = t_n + \frac{\tau}{2}$. Similarly, in the process for $(y_{n+1})_{+x}$ we use $y_{n+1}^{(1)}$ as predictor. All tridiagonal Jacobian matrices, needed in the Newton-type process, are approximately evaluated using the finite difference relation

$$\frac{F(z + \Delta z) - F(z)}{\Delta z},\quad \Delta z = 10^{-6}(1 + |z|), \tag{2.9}$$

$F(z)$ denoting a scalar function. The matrices, including those for the calculation of $(y_{n+1})_{+x}$, are always evaluated at points (t_n, y_n). In our implementation the *updating* of the whole set of matrices costs *4 complete f-evaluations*. The frequency of updating shall be discussed in section 4. The reader who is interested in more precise details concerning the implementation aspects is referred to [5], in which a similar implementation of two splitting methods, including line hopscotch, is discussed.

2.3. The four-step line hopscotch formula

The four-step line hopscotch formula used in our tests is based on the fourth order backward differentiation formula for ordinary differential equations, i.e. the formula

$$y_{n+1} = \frac{1}{25}[48y_n - 36y_{n-1} + 16y_{n-2} - 3y_{n-3}] + \frac{12}{25}\tau f(t_n + \tau, y_{n+1}), \tag{2.10}$$

and on the line hopscotch method to solve y_{n+1} iteratively from (2.10). Performing m (m even) iterations and using the fast-form mentioned in the preceding section, the computational scheme reads as follows (for a derivation we refer to [8]) :

$$n = 3,\ \Sigma_n = \frac{1}{25}(48y_n - 36y_{n-1} + 16y_{n-2} - 3y_{n-3}),$$

$$(y_{n+1}^{(1)})_{+x} = (\Sigma_n)_{+x} + \frac{12}{25}\tau f_{+x}(t_n + \tau, y_n),$$

$$(y_{n+1}^{(1)})_{\bullet o} = (\Sigma_n)_{\bullet o} + \frac{12}{25}\tau f_{\bullet o}(t_n + \tau, y_{n+1}^{(1)}),$$

$$(y_{n+1}^{(j)})_{\bullet o} = (\Sigma_n)_{\bullet o} + \frac{12}{25}\tau f_{\bullet o}(t_n + \tau, y_{n+1}^{(j-1)}) = (y_{n+1}^{(j-1)})_{\bullet o},$$

$$(y_{n+1}^{(j)})_{+x} = (\Sigma_n)_{+x} + \frac{12}{25}\tau f_{+x}(t_n + \tau, y_{n+1}^{(j)}),$$

$$(y_{n+1}^{(j+1)})_{+x} = (\Sigma_n)_{+x} + \frac{12}{25}\tau f_{+x}(t_n + \tau, y_{n+1}^{(j)}) = (y_{n+1}^{(j)})_{+x},$$

$$(y_{n+1}^{(j+1)})_{\bullet o} = (\Sigma_n)_{\bullet o} + \frac{12}{25}\tau f_{\bullet o}(t_n + \tau, y_{n+1}^{(j+1)}),$$

$$j = 2, 4, \ldots, m-2$$

$$(y_{n+1})_{\bullet o} = (\Sigma_n)_{\bullet o} + \frac{12}{25}\tau f_{\bullet o}(t_n + \tau, y_{n+1}^{(m-1)}) = (y_{n+1}^{(m-1)})_{\bullet o},$$

$$(y_{n+1})_{+x} = (\Sigma_n)_{+x} + \frac{12}{25}\tau f_{+x}(t_n + \tau, y_{n+1}),$$

$$n = n + 1,$$

$$\Sigma_n = \frac{1}{25}(48y_n - 36y_{n-1} + 16y_{n-2} - 3y_{n-3}),$$

$$(y_{n+1}^{(1)})_{+x} = (\Sigma_n)_{+x} + (y_n)_{+x} - (\Sigma_{n-1})_{+x}. \qquad (2.11)$$

The non-linear systems in (2.11) are approximately solved by performing one Newton-type iteration (see the preceding section). Thus, one integration step requires m/2 complete f-evaluations, irrespective whether the problem is linear or non-linear. Therefore, for non-linear problems, (2.11) is m/4 times more expensive than the one-step implementation, whereas for linear problems this factor is m/2.
In [8] it is proved that scheme (2.11) is fourth order accurate for $m \geqslant 4$ and m-th order accurate for $m < 4$, and that we have stability in cases where the Jacobian matrices of the systems occurring in (2.11) have negative eigenvalues, provided these matrices share the same eigensystem.
In our experiments we choose $m = 2$ and $m = 4$ leading to a second and fourth order method, respectively. Both methods were applied with exact starting values for y_0, y_1, y_2 and y_3. For further details of the implementation we refer to the preceding section.

3. THE SET OF TEST PROBLEMS

One of the major difficulties in the performance evaluation of algorithms is the choice of a *representative* set of test problems from the problem class under consideration. Desirable is, anyhow, that one uses the same set to simplify the comparison of results presented in other papers. As we do not know of any existing collection of test problems from the class under consideration, we constructed a number of (hopefully non-trivial) problems with a prescribed exact solution. In the problems only a limited number of difficulties are included, which, in our opinion, should be adequately handled by any algorithm that is passed to a possibly more severe test set. They include : arbitrary non-linearities to test the stability of the formulas, ill-balancedness of space derivatives and arbitrary coupling between space derivatives, which are unpleasant properties for splitting formulas. The degree of difficulty in the equations can be varied by one or more parameters.

3.1. The actual test examples

The equations are *scalar* equations and belong to the general class

$$u_t = G(t, x_1, x_2, u, u_{x_1}, u_{x_2}, u_{x_1x_1}, u_{x_1x_2}, u_{x_2x_2}) \qquad (3.1)$$

defined on the product set $\{(t, x_1, x_2) \mid 0 < t \leqslant 1, (x_1, x_2) \in \Omega\}$, Ω being the two-dimensional region

$$\Omega = \{(x_1, x_2) \mid (0 < x_1 < 1,\ 0 < x_2 \leqslant \tfrac{3}{7}) \cup (0 < x_1 < \tfrac{4}{7},\ \tfrac{3}{7} \leqslant x_2 < 1)\}. \qquad (3.2)$$

The initial condition is obtained from the exact solution. The boundary conditions, which are assumed to be of the form

$$\eta(t, x_1, x_2)u + \chi(t, x_1, x_2)u_n = \xi(t, x_1, x_2), \qquad (3.3)$$

u_n outward normal derivative, will be specified with the examples. In case of Dirichlet conditions, that is $\chi = 0$, the exact solution values can be used.
The parabolic equations themselves belong to four families containing one or more parameters. The exact solutions are specified with the families. First we list these families, and then specify the actual test examples. The space-discretization of these test examples is discussed in section 3.2.

First family

$$u_t = u^{2\nu}(u_{x_1x_1} - \theta u_{x_1x_2} + u_{x_2x_2}) + a(t, x_1, x_2)u^{2\nu} + g(t, x_1, x_2), \qquad (3.4)$$

where

$$a(t,x_1,x_2) = 2t^{\omega_1}[\theta x_2 - x_1 - \sin(\omega_2 \pi t)],$$

$$g(t,x_1,x_2) = t^{\omega_1}\{(x_1^2 + x_2)[\omega_1 t^{-1}\sin(\omega_2\pi t) + \omega_2 \pi \cos(\omega_2 \pi t)] + \omega_1 t^{-1} x_1 x_2^2\}.$$

For all θ and ν the solution is given by

$$u(t,x_1,x_2) = a + t^{\omega_1}[(x_1^2 + x_2)\sin(\omega_2 \pi t) + x_1 x_2^2].$$

The parameters a, ω_1 and ω_2 can be used to control the variation of the solution.

Second family

$$u_t = (1+t)^{-1}(\beta u_{x_1 x_1} + u_{x_2 x_2}) + \delta u_{x_1} u_{x_2} + \theta(1-u)u_{x_1 x_2} + g(t,x_1,x_2), \qquad (3.5)$$

where

$$g(t,x_1,x_2) = -\frac{2\beta(1+\theta x_2^2) + 2(\theta x_1^2 - 1) + 4\delta x_1 x_2 (1+\theta x_2^2)(\theta x_1^2 - 1)}{(1+t)^2} + \frac{(4\theta^2 x_1 x_2 - 1)(x_1^2 + \theta x_1^2 x_2^2 - x_2^2)}{(1+t)^2}.$$

For all β and δ the solution is given by

$$u(t,x_1,x_2) = 1 + \frac{x_1^2 + \theta x_1^2 x_2^2 - x_2^2}{1+t},$$

where θ is still free.

Third family

$$u_t = [d(t,x_1,x_2)(1+u)]^{2\nu}[a(x_1,x_2)u_{x_1 x_1} + 2b(x_1,x_2)u_{x_1 x_2} + c(x_1,x_2)u_{x_2 x_2}], \qquad (3.6)$$

where

$$d(t,x_1,x_2) = [1 + x_1 x_2 (x_1 + x_2)e^{-t}]^{-1},$$

$$a(x_1,x_2) = \frac{1}{2}x_1^2 + x_2^2,$$

$$b(x_1,x_2) = -\frac{1}{2}(x_1^2 + x_2^2),$$

$$c(x_1,x_2) = x_1^2 + \frac{1}{2}x_2^2.$$

Equation (3.6) was constructed by "non-linearizing" the equation which arises for $\nu = 0$ and which is used by Gourlay and McKee [3]. For all values of ν we have

$$u(t,x_1,x_2) = x_1 x_2 (x_1 + x_2)\, e^{-t}.$$

Fourth family

$$u_t = \sqrt{u}\, u_{x_1 x_1} - \theta(1+t)^{-1} u_{x_1 x_2} + \sqrt{u}\, u_{x_2 x_2} + \frac{1}{2}(1+t)^{-1}(2\theta - 1)u - 2u\sqrt{u}, \qquad (3.7)$$

which has for all θ the solution

$$u(t,x_1,x_2) = e^{-x_1 - x_2}/\sqrt{1+t}.$$

To be able to specify the actual examples in a compact way, we finally introduce

$$(\delta\Omega)_1 = \{(x_1,x_2) \mid (0 \leqslant x_1 < \tfrac{4}{7}, x_2 = 1) \cup (x_1 = \tfrac{4}{7}, \tfrac{3}{7} < x_2 \leqslant 1) \cup (\tfrac{4}{7} \leqslant x_1 < 1, x_2 = \tfrac{3}{7}) \cup (x_1 = 1, 0 \leqslant x_2 \leqslant \tfrac{3}{7})\},$$

$$(\delta\Omega)_2 = \{(x_1,x_2) \mid x_1 = 0, 0 < x_2 < 1) \cup (0 \leqslant x_1 < 1, x_2 = 0)\},$$

Boundary condition 1 : Dirichlet condition on $(\delta\Omega)_1 \cup (\delta\Omega)_2$,

Boundary condition 2 : Dirichlet condition on $(\delta\Omega)_1$ and von Neumann condition [$\eta = 0$ in (3.3)] on $(\delta\Omega)_2$.

The test examples are now given by :

Example 1

Equation (3.4) with $(\nu, \theta, \omega_1, \omega_2, a) = (1, 1.5, 2, 2, 1)$ and boundary condition 1. The problem is non-linear.

Example 2

Like example 1, but boundary condition 2 instead of boundary condition 1.

Example 3

Equation (3.5) with $(\beta, \delta, \theta) = (2, 5, 2)$ and boundary condition 1. The problem is non-linear.

Example 4

Equation (3.6) with $\nu = 0$ and boundary condition 1. This problem is linear.

Example 5

Like example 4 with boundary condition 1 replaced by boundary condition 2.

Example 6

Like example 4 with $\nu = 0$ replaced by $\nu = 5$. Hence this problem is strongly non-linear.

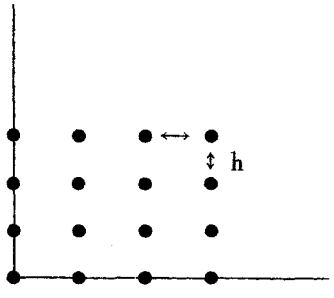
Example 7

The non-linear equation (3.7) with $\theta = 1$ and boundary condition 1.

As can be observed from the above examples, difficulties which are not included are, e.g., non-well behaviour of the function G, singularities in the solution, curved boundaries, non-linear boundary conditions, large convective terms. The equations are, in fact, chosen in such a way that the solutions are smooth functions of x_1 and x_2 and, in addition, most equations are chosen such that discretization of the space variables on a uniform grid by standard finite differences does not give a space discretization error. This means that the time integration aspect of the whole algorithm can be tested more or less separately from the effects of space discretization.

3.2. The space discretization of the test examples

All examples are semi-discretized on a uniform grid using finite differences with grid size h = 1/21. Consequently, the number of *internal* grid points lying in the region Ω, given by (3.2), comes to 292. At these points we use *second order, symmetrical differences,* that is, u_{x_i} is replaced by the standard 2-point difference, $u_{x_i x_i}$ by the standard 3-point difference, and $u_{x_1 x_2}$ by the standard 4-point difference. The expressions obtained at grid points nearest to the boundary thus contain values at boundary points. To point out how we calculate these values, that is, how we discretize the boundary relation (3.3), we consider the following picture and assume that on the line $x_2 = 0$ relation (3.3) is to be discretized.



Let $U_{cr}(t)$ denote the semi-discrete approximation to the exact solution $u(t, x_1, x_2)$ at the point $(x_1, x_2) = (ch, rh)$. Along the line $x_2 = 0$ relation (3.3) is then replaced by the *second order,* 3-point difference relation

$$\frac{\xi(t,ch,0) - \eta(t,ch,0)\, U_{c0}(t)}{\chi(t,ch,0)}$$

$$= h^{-1}\left[-\frac{1}{2}U_{c2}(t) - 2U_{c1}(t) + \frac{3}{2}U_{c0}(t)\right], \qquad (3.8)$$

of which the right member approximates the outward normal derivative of $u(t, x_1, x_2)$ at the point (ch, 0). Next, $U_{c0}(t)$ is expressed in its two neighbours $U_{c1}(t)$ and $U_{c2}(t)$. This expression is then substituted in the difference expressions, at internal grid points, in which $U_{c0}(t)$ is asked for. Consequently, each component of the resulting system of ordinary differential equations is related to an internal grid point. Note that the space discretization errors for problems belonging to the first three families are equal to zero. It should also be observed that in case of boundary condition 2 the approximation U_{00} is expressed in U_{11}, U_{12}, U_{21} and U_{22}. The expression can be obtained in two ways, viz. by using the normal of the line $x_1 = 0$, or the normal of the line $x_2 = 0$. We apply the second possibility.

4. TESTING STRATEGY AND PRESENTATION OF RESULTS

To be able to obtain a clear indication on the merits of the formulas we are testing we keep it as simple as possible. To each test example, their implementations, as given in section 2, are applied for a sequence of constant stepsizes (which may differ per implementation). We thus do not use any strategy to estimate errors and to vary the stepsize. Neither a strategy to control the updating of the tridiagonal matrices used by the hopscotch methods, nor a strategy to control the variation of the spectral radius used for the application of the explicit method, is included. In case of non-linear problems the updating of the tridiagonal matrices is performed every integration step, while an estimate of the spectral radius is specified beforehand. This estimate is usually a constant. From the results reported in [10], we know that additional costs due to an automatic control on the variation of the spectral radius, as well as its updating, are relatively small (in practice not more than about 10 %).
Per example, all results are embodied in one accuracy-efficiency diagram and an accompanying table. The accuracy is measured by

$$A = \min_{c,\,r}\left[-^{10}\log\left|\tilde{U}_{cr}(1) - u(1, ch, rh)\right|\right],$$

where $\tilde{U}_{cr}(1)$ denotes the final approximation at $(t, x_1, x_2) = (1, ch, rh)$, with (ch, rh) running through the whole set of internal grid points and boundary grid points. The efficiency is measured by

C_I = the number of evaluations of the semi-discrete system, where evaluations for the Jacobian matrices are not taken into account.

and

C_J = the number of evaluations of the semi-discrete system used in updating the Jacobian matrices.

The measure C_I merely counts the number of evalua-

tions required by the integrators themselves. Only this measure is, in fact, embodied in the diagrams. For the explicit formulas this is obvious. In case of linear problems (constant Jacobian matrices) this is also obvious for the hopscotch methods. For non-linear problems, one should, in fact, also take into account some fraction of C_J. Such a fraction, however, is rather problem dependent. If the Jacobians are slowly varying, it will be small; if the Jacobians are strongly varying, it will be large. The numbers C_J are given in the accompanying tables, so that the reader can judge the results himself. We shortly return to this point at the end of this section. Other computations, such as LU-decompositions and forward-backward substitutions needed in the hopscotch methods, are also not taken into account in our efficiency measure. This slightly favours the hopscotch methods in our comparisons.
The diagrams and accompanying tables are collected in figures 4.1 - 4.7 and tables 4.1 - 4.7. For linear problems the C_J-column is left empty.

Large negative A-values must be interpreted as unstable results. We use the following abbreviations :

1 - for the explicit integrator,
2 - for the one-step hopscotch integrator,
3 - for the second order four-step hopscotch integrator,
4 - for the fourth order four-step hopscotch integrator.

5. CONCLUSIONS

The results of the experiments lead us to the following observations :

1) As a rule, the three hopscotch integrators behave similarly and are competitive to each other, the one-step version being somewhat more stable when integrating non-linear problems. Because, in addition, the one-step method is somewhat easier to implement, it is to be preferred to the four-step versions.

2) The hopscotch methods are particularly suited if one is satisfied with low accuracy results, say A between 2 and 3. It appears that such results can be obtained with a relatively small amount of computational effort.

3) The fourth order hopscotch method does not stand out clearly. This is due to the fact that its effective order, when considering realistic stepsizes, is much lower than four. Its theoretical order only appears in the results for unrealistic values of $\tau\sigma$.

4) In all diagrams the stabilized Runge-Kutta method appears to be the most expensive integrator. It should be observed, however, that this method is prejudiced in the comparisons, because of the fact that C_J is not taken into account (this observation does not apply in case of linear problems). Therefore, by way of trial, we performed two other experiments. We again integrated examples 1 and 7 with the three hopscotch methods, but now updated the tridiagonal Jacobians every 10 integration steps. The results for example 7, given in table 4.8, are almost identical to those given in table 4.7. For example 1, however, the results become worse. In this case, the exact Jacobian matrices are varying rather strongly over the integration interval, so that, in the comparison with the explicit method, a large fraction of C_J should be taken into account.

Concluding, from the four methods considered, the one-step line hopscotch method appears to be the most efficient one for the numerical solution of scalar parabolic equations in two space dimensions with a mixed derivative. We expect that, if one is satisfied with low accuracy, this method is also more efficient than any fully implicit or semi-implicit one. It seems worth pursuing this point further. A slight disadvantage of the line hopscotch method is, that it is sensitive to ill-balancedness of space derivatives. For example, if in equation (3.5) the parameter β is large, the method has to be implicit in the x_1-direction, otherwise the results become worse. Thus, in case of non-linear problems where the ill-balancedness may vary, one has to be careful in choosing the implicit direction.

REFERENCES

1. GOURLAY, A. R. : "Splitting methods for time-dependent partial differential equations", in the proceedings of the 1976 conference : *The state of the art in numerical analysis,* ed. Jacobs, D. A. H., Academic Press, 1977.
2. GOURLAY, A. R. and McGUIRE, G. R. : "General hopscotch methods for partial differential equations", JIMA 7, 216-227, 1971.
3. GOURLAY, A. R. and McKEE, S. : "The construction of hopscotch methods for parabolic and elliptic equations in two space dimensions with a mixed derivative", J. of Comp. and Appl. Math. 3, 201-206, 1977.
4. SINCOVEC, R. F. and MADSEN, N. K. : "Software for non-linear partial differential equations", TOMS 1, 232-260, 1975.
5. SOMMEIJER, B. P. : "An ALGOL 68 implementation of two splitting methods for semi-discretized parabolic differential equations", Note NN 15/77, Mathematisch Centrum, Amsterdam, 1977.
6. VAN DER HOUWEN, P. J. : *"Construction of integration formulas for initial value problems"*, North-Holland Publishing Company, Amsterdam, 1977.
7. VAN DER HOUWEN, P. J. and VERWER, J. G. : "One-step splitting methods formulated for semi-discrete parabolic equations", Report NW 55/78, Mathematisch Centrum, Amsterdam (prepublication) 1978.
8. VAN DER HOUWEN, P. J. : "Multistep splitting methods of high order for initial value problems", Report NW 49/77, Mathematisch Centrum, Amsterdam, 1977.
9. VERWER, J. G. : "A class of stabilized three-step Runge-Kutta methods for the numerical integration of parabolic equations", J. of Comp. and Appl. Math. 3, 155-166, 1977.
10. VERWER, J. G. : "An implementation of a class of stabilized, explicit methods for the time integration of parabolic equations", Report NW 38/77, Mathematisch Centrum, Amsterdam (prepublication) 1977.

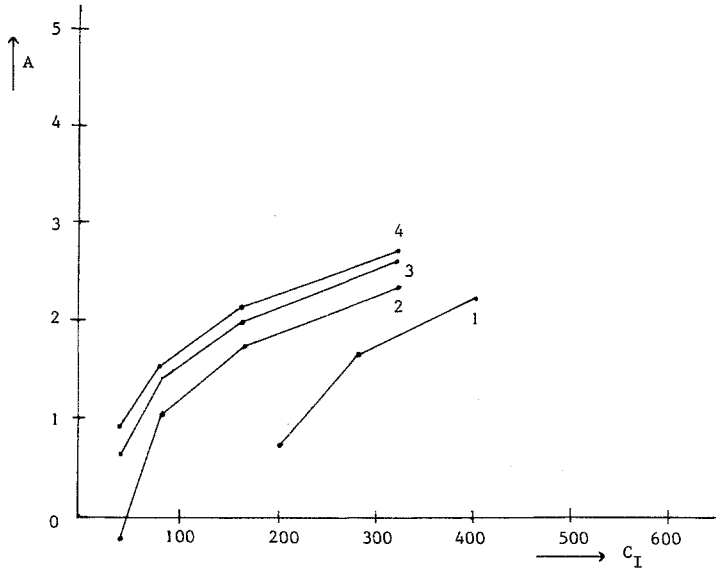


Fig. 4.1 The A-C_I diagram for the non-linear example 1. For all methods the accuracy is low because of the oscillating solution. The hopscotch methods are comparable and more efficient than the explicit method.

	τ	C_I	C_J	A
1	1/20	200		0.72
	1/40	280		1.67
	1/80	400		2.26
2	1/20	40	80	-0.25
	1/40	80	160	1.05
	1/80	160	320	1.72
	1/160	320	640	2.37
3	1/40	40	160	0.92
	1/80	80	320	1.55
	1/160	160	640	2.14
	1/320	320	1280	2.73
4	1/20	40	80	0.61
	1/40	80	160	1.41
	1/80	160	320	2.00
	1/160	320	640	2.67

Table 4.1 Example 1, σ = 4200.

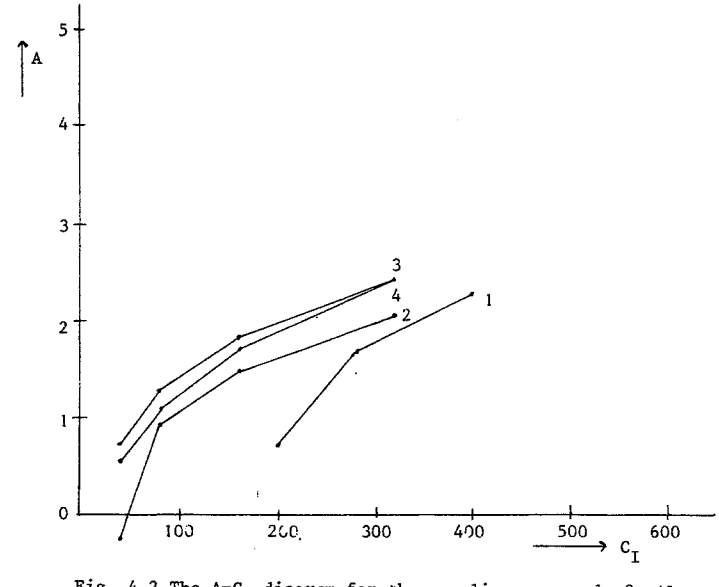


Fig. 4.2 The A-C_I diagram for the non-linear example 2. Almost the same picture as in fig. 4.1. The replacement of Dirichlet conditions by von Neumann conditions slightly reduces the accuracy of the hopscotch formulas.

	τ	C_I	C_J	A
1	1/20	200		0.74
	1/40	280		1.67
	1/80	400		2.26
2	1/20	40	80	-0.25
	1/40	80	160	0.91
	1/80	160	320	1.48
	1/160	320	640	2.08
3	1/40	40	160	0.72
	1/80	80	320	1.26
	1/160	160	640	1.83
	1/320	320	1280	2.42
4	1/20	40	80	0.54
	1/40	80	160	1.09
	1/80	160	320	1.72
	1/160	320	640	2.42

Table 4.2 Example 2, σ = 4200.

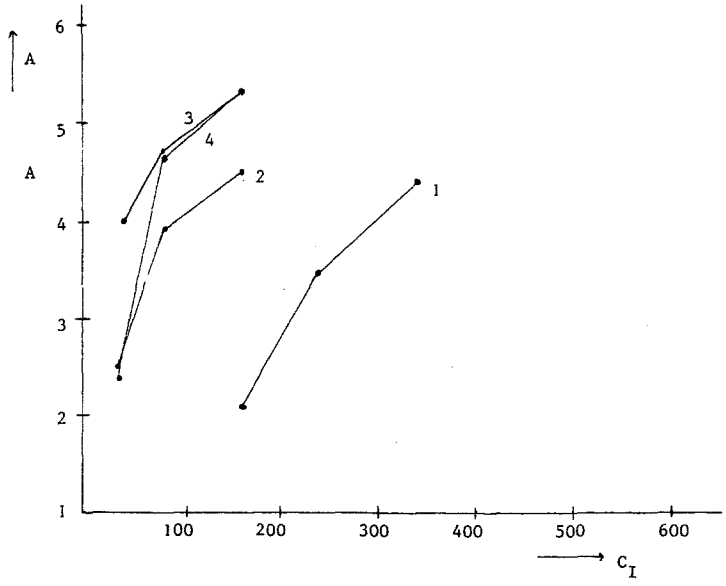


Fig. 4.3 The A-C_I diagram for the non-linear example 2. Again the hopscotch methods are comparable and more efficient than the explicit method.

	τ	C_I	C_J	A
1	1/15	164		2.08
	1/30	238		3.49
	1/60	339		4.42
2	1/20	40	80	2.46
	1/40	80	160	3.92
	1/80	160	320	4.52
3	1/40	40	160	4.00
	1/80	80	320	4.72
	1/160	160	640	5.33
4	1/20	40	80	2.37
	1/40	80	160	4.67
	1/80	160	320	5.35

Table 4.3 Example 3, $\sigma = 4930 - 2340t$.

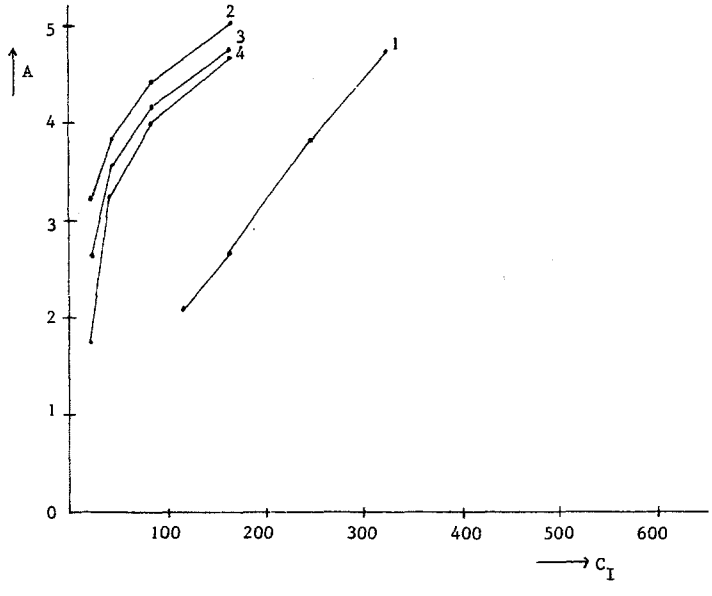


Fig. 4.4 The A-C_I diagram for the linear example 4. Roughly the same picture as before.

	τ	C_I	C_J	A
1	1/10	110		2.07
	1/20	160		2.64
	1/40	240		3.79
	1/80	320		4.77
2	1/20	20		3.21
	1/40	40		3.81
	1/80	80		4.41
	1/160	160		5.01
3	1/20	20		2.62
	1/40	40		3.53
	1/80	80		4.14
	1/160	160		4.74
4	1/10	20		1.73
	1/20	40		3.26
	1/40	80		3.98
	1/80	160		4.68

Table 4.4 Example 4, $\sigma = 2740$.

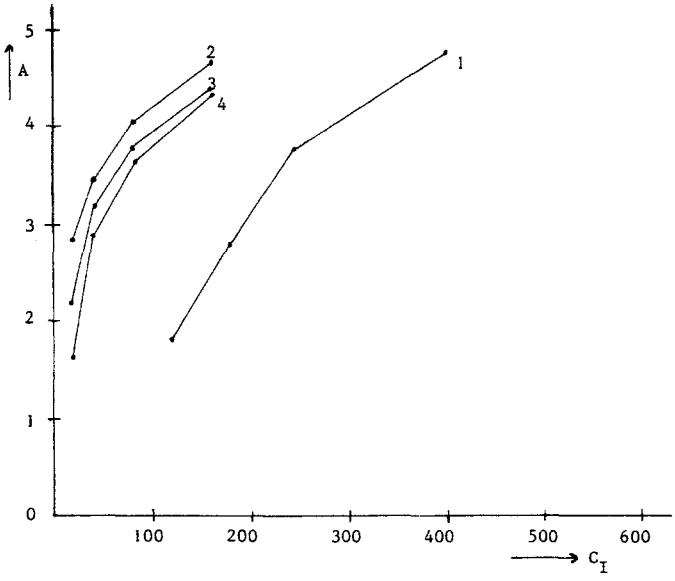


Fig. 4.5 The A-C_I diagram for the linear example 5. When compared with example 4, the hopscotch results are slightly less accurate.

	τ	C_I	C_J	A
1	1/10	120		1.80
	1/20	180		2.82
	1/40	240		3.78
	1/80	400		4.76
2	1/20	20		2.83
	1/40	40		3.45
	1/80	80		4.06
	1/160	160		4.66
3	1/20	20		2.18
	1/40	40		3.19
	1/80	80		3.79
	1/160	160		4.39
4	1/10	20		1.64
	1/20	40		2.90
	1/40	80		3.64
	1/80	160		4.36

Table 4.5 Example 5, σ = 3000.

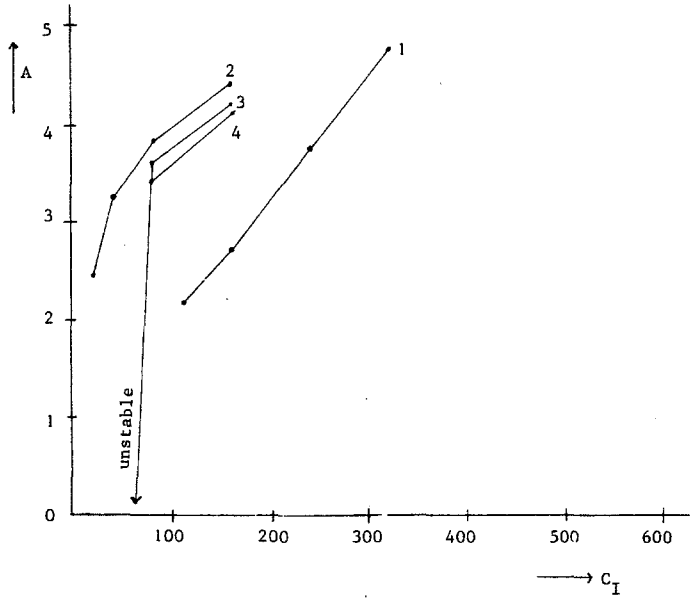


Fig. 4.6 The A-C_I diagram for the strongly non-linear example 6. The four-step hopscotch methods become unstable for the larger stepsizes.

	τ	C_I	C_J	A
1	1/10	110		2.18
	1/20	160		2.71
	1/40	240		3.79
	1/80	320		4.77
2	1/10	20	40	2.46
	1/20	40	80	3.23
	1/40	80	160	3.84
	1/80	160	320	4.44
3	1/20	20	80	- 50
	1/40	40	160	- 100
	1/80	80	320	3.63
	1/160	160	640	4.24
4	1/10	20	40	- 100
	1/20	40	80	- 200
	1/40	80	160	3.45
	1/80	160	320	4.15

Table 4.6 Example 6, σ = 2740.

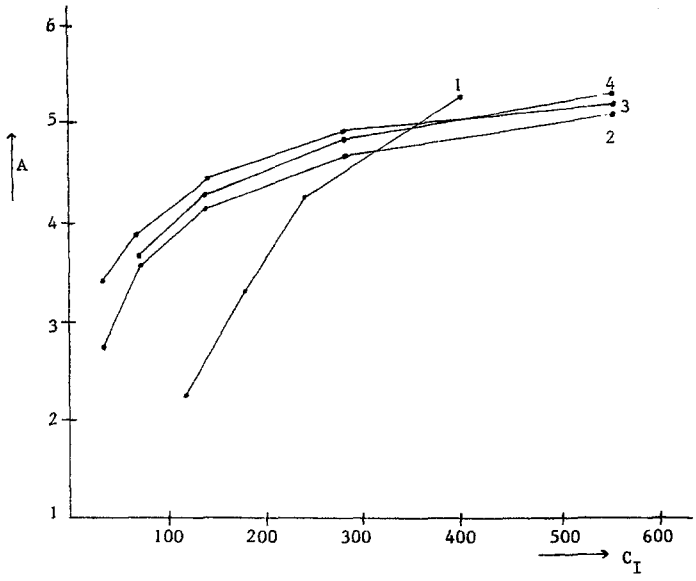


Fig. 4.7 The A-C_I diagram for the non-linear example 7. In this case the explicit method becomes competitive if high accuracy is desired.

	τ	C_I	C_J	A
1	1/10	120		2.24
	1/20	180		3.27
	1/40	240		4.25
	1/80	400		5.26
2	1/17	34	68	2.74
	1/35	70	140	3.57
	1/70	140	280	4.15
	1/140	280	560	4.68
3	1/35	35	140	3.39
	1/70	70	280	3.88
	1/140	140	560	4.45
	1/280	280	1120	4.92
4	1/35	70	140	3.65
	1/70	140	280	4.29
	1/140	280	560	4.88
	1/280	560	1120	5.26

Table 4.7 Example 7, σ = 3000.

	Example 1				Example 7			
	τ	C_I	C_J	A	τ	C_I	C_J	A
2	1/20	40	8	0.29	1/17	34	8	2.62
	1/40	80	16	-0.37	1/35	70	16	3.58
	1/80	160	32	-0.19	1/70	140	28	4.17
	1/160	320	64	0.73	1/140	280	56	4.69
3	1/40	40	16	-4.82	1/35	35	16	3.13
	1/80	80	32	-2.83	1/70	70	28	4.41
	1/160	160	64	1.11	1/140	140	56	4.96
	1/320	320	128	1.66	1/280	280	112	5.98
4	1/20	40	8	-0.31	1/35	70	16	3.51
	1/40	80	16	-68.31	1/70	140	28	4.78
	1/80	160	32	-5.42	1/140	280	56	5.83
	1/160	320	64	1.34	1/280	560	112	5.47

Table 4.8 Results of two experiments with inaccurate Jacobian matrices.