stichting
mathematisch
centrum

MC

AFDELING NUMERIEKE WISKUNDE (DEPARTMENT OF NUMERICAL MATHEMATICS)

NN 13/77 MEI

J.G. VERWER

A COMPARISON BETWEEN THE ODD-EVEN HOPSCOTCH METHOD AND A CLASS OF RUNGE-KUTTA METHODS WITH EXTENDED REAL STABILITY INTERVALS

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS(MOS) subject classification scheme (1970): 65L05 65M99

A comparison between the odd-even hopscotch method and a class of Runge-Kutta methods with extended real stability intervals

by

J.G. Verwer

ABSTRACT

This report is written as a contribution to a project to develop numerical software for time dependent partial differential equations. The odd-even hopscotch method is formulated for systems of ordinary differential equations and applied to a number of semi-discretized parabolic problems. The results are compared with results obtained with explicit three-step Runge-Kutta methods.

KEYWORDS & PHRASES: *Numerical analysis, Parabolic partial differential equations, The odd-even hopscotch method, Stabilized Runge-Kutta methods.*

1. INTRODUCTION

This report is written as a contribution to a project of the Numerical Mathematics Department of the Mathematical Centre to develop numerical software for time-dependent partial differential equations. In particular, it is meant to contribute to part II of this project, viz. the selection of numerical algorithms suited for automatic packages for significant classes of semi-discretized partial equations ([3]). Stabilized, explicit Runge-Kutta methods, especially for two or higher dimensional problems, are known to be suited to this purpose. A completely different type of method, also suited to this purpose, is the odd-even hopscotch method ([4]). From a computational point of view this method is explicit, and unconditionally stable for a significant class of parabolic problems.

The main purpose of this report is now to get an indication about the mutual efficiency and accuracy of the odd-even hopscotch method and explicit Runge-Kutta methods, when applied to semi-discretized parabolic problems. To that end the odd-even hopscotch method is formulated as a second order integration method for systems of ordinary differential equations

$$\frac{d\vec{y}}{dt} = \vec{f}(t,\vec{y}), \tag{1.1}$$

and applied to a subset of the set of test examples used in [3], where explicit Runge-Kutta methods are compared. For the description of the test examples, as well as for the way of testing, we refer to [3]. Here we confine ourselves to giving the results of the odd-even hopscotch method and to comparing these with results obtained with first and second order three-step Runge-Kutta methods. For a description of the Runge-Kutta methods we we also refer to [3], where further references can be found.

2. THE ODD-EVEN HOPSCOTCH PROCESS

As already observed in the introduction, our starting point in the numerical solution of time-dependent partial differential equations is the method of semi-discretization. Contrary to the way of defining the hopscotch method as a direct grid method for a class of partial differential equations

([4,5]), we therefore prefer to define the method for the system of ordinary differential equations (1.1).

Let $\vec{y}_n$ denote the numerical approximation at $t = t_n$ and let τ_n denote the steplength. Further, let Λ_E and Λ_0 denote constant diagonal matrices such that $\Lambda_E + \Lambda_0 = I$, I denoting the unit matrix. According to the definition of Gourlay [4] the odd-even hopscotch method, when defined for system (1.1), then belongs to the class of methods given by

$$\vec{y}_{n+1} = \vec{y}_n + \tau_n \Lambda_E \vec{f}_n + \tau_n \Lambda_0 \vec{f}_{n+1},$$

(2.1)

$$\vec{y}_{n+2} = \vec{y}_n + \tau_n \Lambda_E \vec{f}_n + 2\tau_n \Lambda_0 \vec{f}_{n+1} + \tau_n \Lambda_E \vec{f}_{n+2},$$

where $\vec{f}_{n+i} = \vec{f}(t_{n+i}, \vec{y}_{n+i})$, $t_{n+i} = t_n + i\tau_n$ for $i = 1,2$, and $n = 0,2,\ldots$. Given $\vec{y}_n$ as initial vector, (2.1) generates approximations $\vec{y}_{n+i}$ at $t = t_{n+i}$ for $i = 1,2$ simultaneously. Thus (2.1) may be considered as a 2-block implicit one-step method (cf.[7]) with matrix parameters instead of scalars.

Substituting a sufficiently differentiable solution y(x), and expanding the right hand sides yields

$$\vec{y}_{n+1} = \vec{y}_n + \tau_n \vec{y}'_n + \Lambda_0 \tau_n^2 \vec{y}''_n + O(\tau_n^3),$$

(2.2)

$$\begin{aligned}\vec{y}_{n+2} &= \vec{y}_{n+1} + \tau_n \vec{y}'_{n+1} + \Lambda_E \tau_n^2 y''_{n+1} + O(\tau_n^3) \\ &= \vec{y}_n + 2\tau_n \vec{y}'_n + 2\tau^2 \vec{y}''_n + O(\tau_n^3).\end{aligned}$$

Thus we see that the local discretization error at $t = t_n$ of $\vec{y}_{n+1}$ and $\vec{y}_{n+2}$ is of order 2 and 3, respectively. That means that the hopscotch process, when formulated for a system of ordinary differential equations, has an order of convergence equal to 2, provided that only the second block solution is taken into account. If both block solutions are taken into account, the order of convergence is equal to 1.

A block method is in fact an ordinary one-step method of a special form (see e.g. [6]). From this point of view, (2.1) may be reformulated as the second order one-step method

$$
\begin{aligned}
\vec{y}_{n+\frac{1}{2}} &= \vec{y}_n + \tfrac{1}{2}\tau_n \Lambda_E \vec{f}_n + \tfrac{1}{2}\tau_n \Lambda_0 \vec{f}_{n+\frac{1}{2}}, \\
\vec{y}_{n+1} &= \vec{y}_{n+\frac{1}{2}} + \tfrac{1}{2}\tau_n \Lambda_0 \vec{f}_{n+\frac{1}{2}} + \tfrac{1}{2}\tau_n \Lambda_E \vec{f}_{n+1},
\end{aligned}
\tag{2.3}
$$

where $n = 0,1,2,\ldots$. We prefer this formulation to the block formulation (2.1). The only difference is that starting in t_n one application of (2.3) yields a solution at $t_n + \tau_n$, while (2.1) yields a solution at $t_n + 2\tau_n$.

Summarizing, the odd-even hopscotch method for systems of ordinary differential equations may be formulated as an implicit one-step Runge-Kutta method with matrix parameters instead of scalars. Integration formulas with matrix parameters for semi-discretized systems of hyperbolic equations are studied in [1] and [2]. In fact, formula (2.3) belongs to the class of formulas discussed in [2].

It is clear of course that the essential features of such a method are obtained after defining the matrix parameters. The parameters yielding the odd-even hopscotch method are defined by

$$
(i,i)\text{-element of } \Lambda_E = \begin{cases} 1 \text{ if } i \text{ is even}, \\ 0 \text{ if } i \text{ is odd}, \end{cases}
\tag{2.4}
$$

while $\Lambda_0 = I - \Lambda_E$. We thus see that the odd-even hopscotch process in formulation (2.3) consists of a componentwise application of the forward and backward Euler formula, with steplength $\frac{1}{2}\tau_n$, in an alternately manner.

We shall not discuss the computational aspects of the odd-even hopscotch process, as these are extensively discussed in [4]. We confine ourselves by enumerating these shortly:

1. If $\vec{f}$ satisfies the E-property, i.e. component f^i satisfies:

f^i, i odd, only depends on t, y^i, y^j, $j \neq i$, j even,

f^i, i even, only depends on t, y^i, y^j, $j \neq i$, j odd,

only scalar linear or non-linear equations must be solved. In this

case the method is effectively explicit, and suited for multi-dimensional problems.

2. If $\vec{f}$ is programmed componentwise, the whole process needs one array of storage.

3. By writing (2.3) in the fast form (componentwise)

$$
\begin{aligned}
&n = 0 \quad , \text{ even: } && y_{n+\frac{1}{2}} = y_n + \tfrac{1}{2}\tau_n f_n, \\
&\qquad\qquad \text{odd: } && y_{n+\frac{1}{2}} = y_n + \tfrac{1}{2}\tau_n f_{n+\frac{1}{2}}, \\
&\qquad\qquad \text{odd: } && y_{n+1} = 2y_{n+\frac{1}{2}} - y_n \\
&\qquad\qquad \text{even: } && y_{n+1} = y_{n+\frac{1}{2}} + \tfrac{1}{2}\tau_n f_{n+1} \\
&n = n+1, \text{ even: } && y_{n+\frac{1}{2}} = \left(1 + \frac{\tau_n}{\tau_{n-1}}\right) y_n + \frac{\tau_n}{\tau_{n-1}} y_{n-\frac{1}{2}},
\end{aligned}
\tag{2.5}
$$

the whole process costs one function evaluation (evaluations needed to solve the non-linear equations not counted).

From these points it is clear that the strength of the odd-even hopscotch method lies in its explicitness and simplicity. We therefore assume that the problems to which this method should be applied satisfy the E-property. A significant class of problems possess this property.

The stability of general hopscotch processes is extensively discussed in [4]. For our case, the most important result can be stated as follows: let $\tau_n = \tau$, τ constant, and apply the odd-even hopscotch process (2.3) to the stable linear system

$$\frac{d\vec{y}}{dt} = J\vec{y}, \tag{2.6}$$

yielding the linear recurrence relation

$$\vec{y}_{n+1} = T\vec{y}_n, \qquad n = 0,1,\ldots, \tag{2.9}$$

where

$$(2.8) \qquad T = (I - \tfrac{1}{2}\tau\Lambda_E J)^{-1}(I + \tfrac{1}{2}\tau\Lambda_0 J)(I - \tfrac{1}{2}\tau\Lambda_0 J)^{-1}(I + \tfrac{1}{2}\tau\Lambda_E J).$$

Then powers of T are uniformly bounded in n, if J is non-singular, has a full set of eigenvectors and satisfies the diagonal dominance conditions. After semi-discretization, a lot of parabolic problems give rise to vector functions $\vec{f}(t,\vec{y})$ possessing a Jacobian matrix with these properties, while also possessing the E-property.

3. NUMERICAL RESULTS

In figures 1 - 5 we give results of (2.3) when applied to a set of 5 test examples from [3] with a constant steplength. In the figures these results are compared with results of first and second order 3-step Runge-Kutta methods. The test examples are one-dimensional. We emphasize that the benefits of both methods are lying in more-dimensional problems. As the main purpose of this report is to get an indication about the mutual accuracy and efficiency of both methods, one-dimensional problems are suited. For a description of the Runge-Kutta methods and of the test examples the reader is referred to [3]. As a consequence, to be able to interpret the results the reader has to read parts of that report, in particular those parts where the test strategy is given.

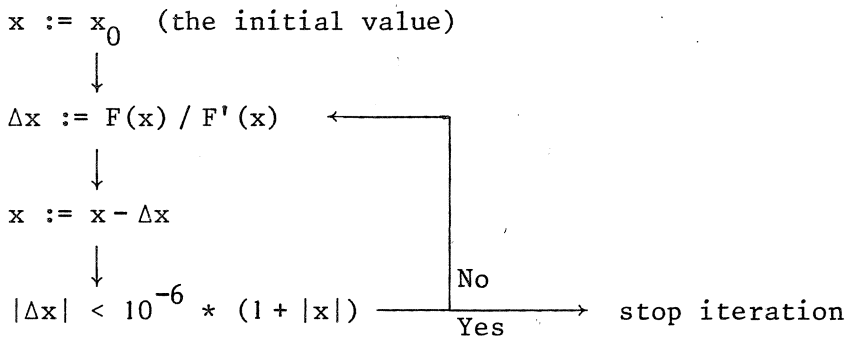
The test examples were integrated using algorithm (2.5), i.e. the derivative functions $\vec{f}(t,\vec{y})$ were programmed componentwise. The number of derivative evaluations fe used in [3], is here defined by

$$(3.1) \qquad fe = \text{entier}(cfe / N+1),$$

where cfe denotes the total number of component evaluations, and where N denotes the number of components.

To solve the non-linear scalar equations, say F(x), the test program was provided with the following Newton-Raphson algorithm:

(3.2)



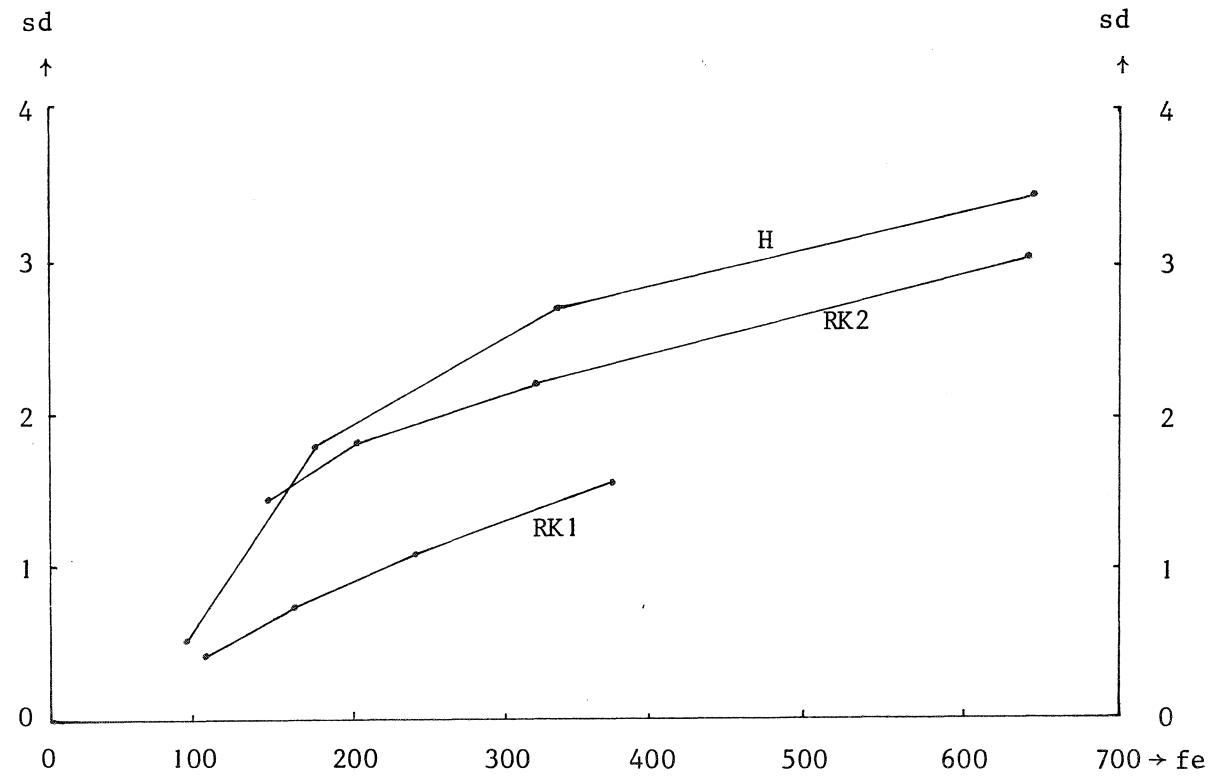
The initial value was chosen equal to the last calculated component value. Thus according to (2.5), $x_0 = y_n$ and $x_0 = y_{n+\frac{1}{2}}$ for odd and even components, respectively. For convenience, convergence problems are left out of consideration.

For testing convenience linear and non-linear equations were treated in the same way, i.e. algorithm (3.2) was also used to solve linear equations. From (3.2) it is clear that each component evaluation involves an evaluation of the corresponding element of the diagonal Jacobian matrix. For our type of problems the additional costs to evaluate F'(x) are generally small when compared with the costs to evaluate F(x). These additional costs are expressed in the costs to evaluate F(x). The estimated ratio between the costs, viz.

$$r = \frac{\text{costs to evaluate } F'(x)}{\text{costs to evaluate } F(x)} \quad , \tag{3.3}$$

is given at the figures. In these figures, sd denotes the number of significant digists of the least accurate component. Below each figure a small table is given containing the actual data inserted in the figure. In these tables we also give the number of integration steps, say step, needed to integrate the corresponding interval. Thus fe / step equals the average number of function evaluations per integration step. The symbols H, RK1 and RK2 refer to the hopscotch method and to the Runge-Kutta method of order 1 and 2, respectively.

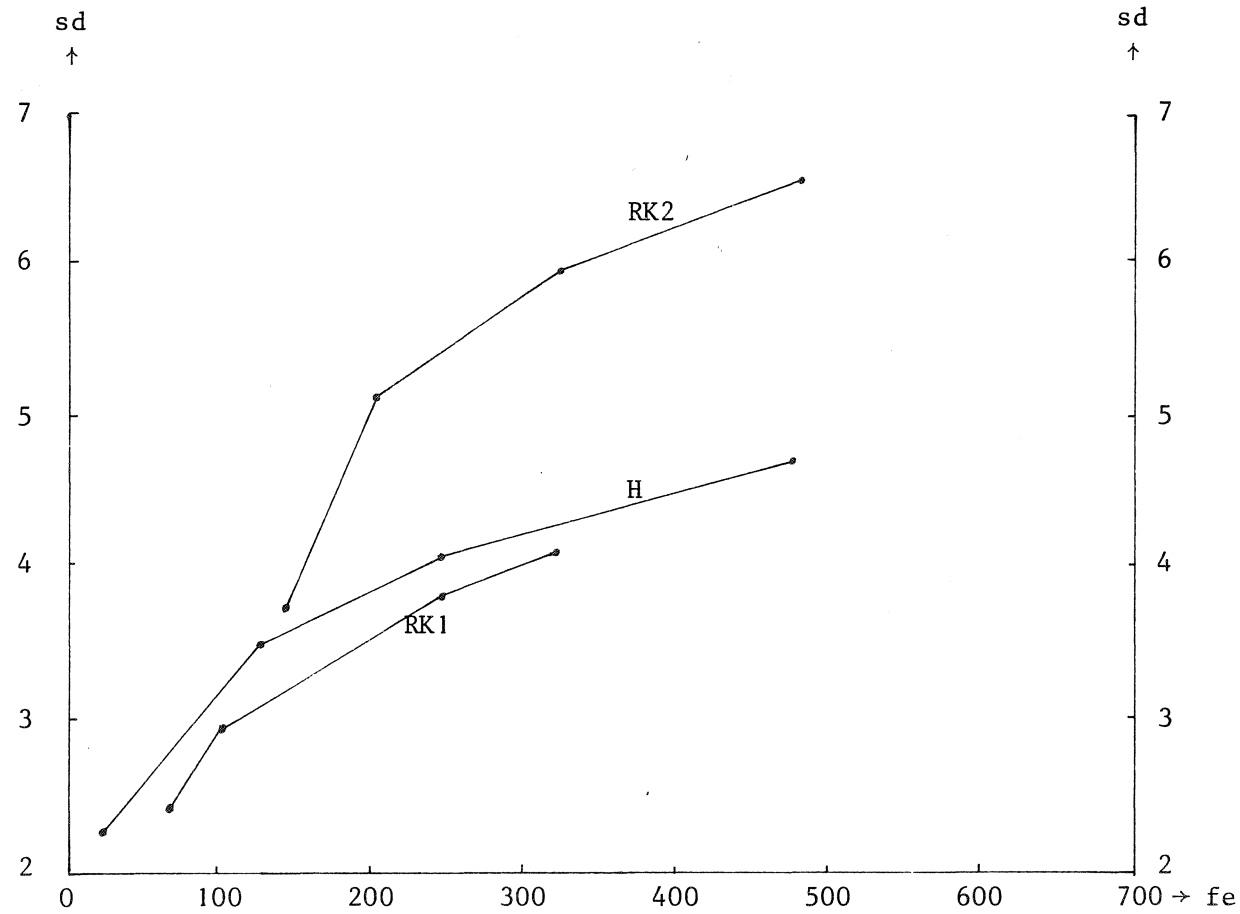
Figure 1 Problem I from [3], r = 0.



H			RK1			RK2		
step	fe	sd	step	fe	sd	step	fe	sd
40	88	0.48	20	100	0.39	20	140	1.41
80	168	1.76	40	160	0.73	40	200	1.79
160	327	2.70	80	240	1.13	80	320	2.19
320	646	3.43	160	320	1.54	320	640	3.05

REMARKS. The hopscotch method is clearly more efficient than the first order Runge-Kutta method. For the higher accuracies it is also more efficient than the second order Runge-Kutta method, while for low accuracy these methods are comparable. The inaccuracy of the Runge-Kutta formulas is probably due to the severe non-linearities of the problem.

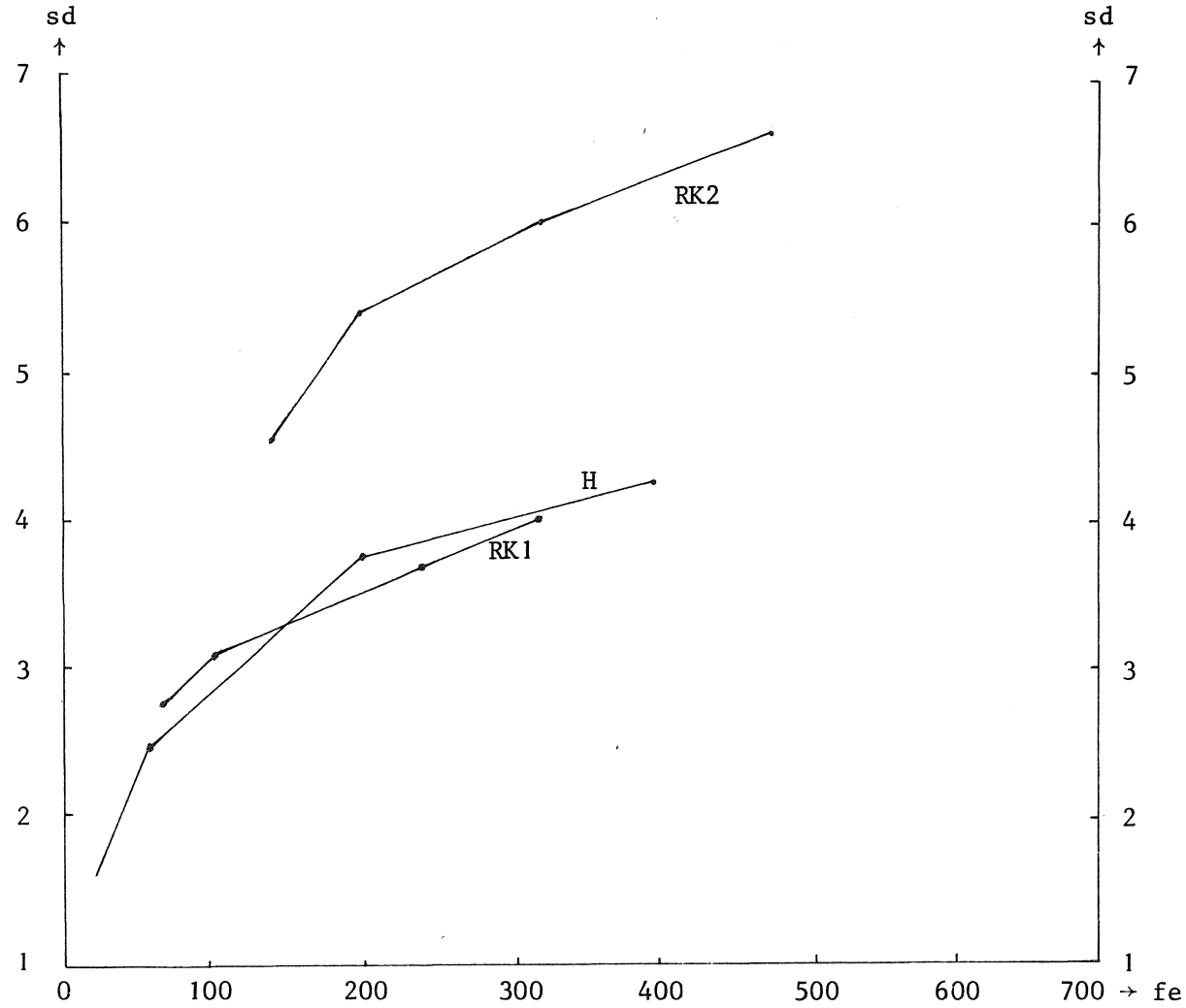
Figure 2 Problem II from [3], r = 0.



H			RK1			RK2		
step	fe	sd	step	fe	sd	step	fe	sd
20	25	2.25	10	70	2.41	20	140	3.74
80	124	3.49	20	100	2.93	40	200	5.13
160	245	4.09	80	240	3.81	80	320	5.97
320	476	4.69	160	320	4.11	160	480	6.56

REMARKS. The problem is non-linear in only one component, viz. the component corresponding to the right boundary of the original problem. The hopscotch method is more efficient than the first order Runge-Kutta method. For higher accuracies the hopscotch method is clearly less efficient than the second order Runge-Kutta method. For low accuracies they are comparable.

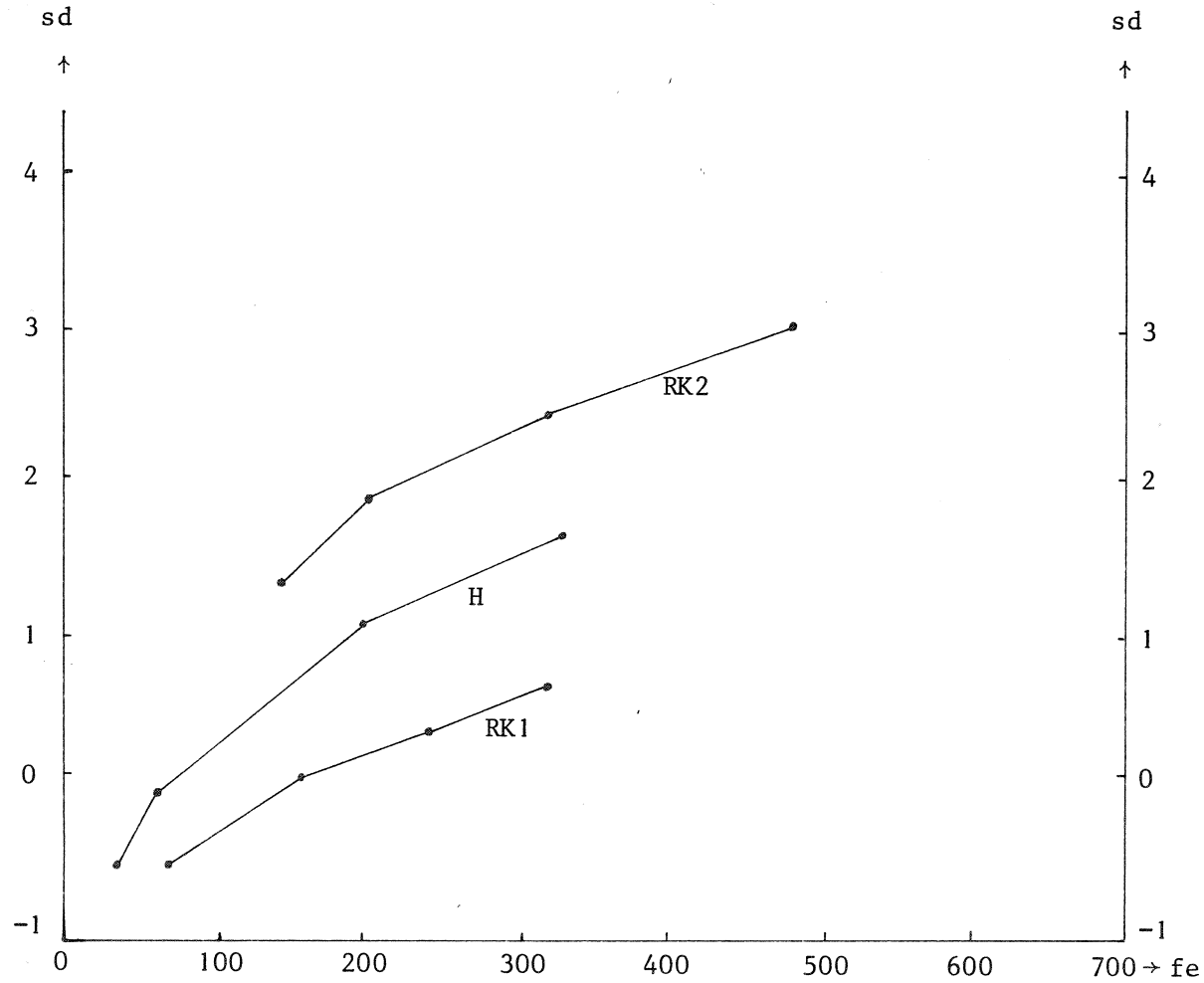
Figure 3 Problem III from [3], r = $\frac{1}{4}$.



H			RK1			RK2		
step	fe	sd	step	fe	sd	step	fe	sd
5	20	1.62	10	70	2.78	20	140	4.58
20	51	2.45	20	100	3.10	40	200	5.43
80	201	3.68	80	240	3.71	80	320	6.05
160	401	4.29	160	320	4.00	160	480	6.65

REMARKS. The problem is non-linear. The second order Runge-Kutta method gives much better results than the hopscotch method. For this problem the first order Runge-Kutta method is comparable with the hopscotch method.

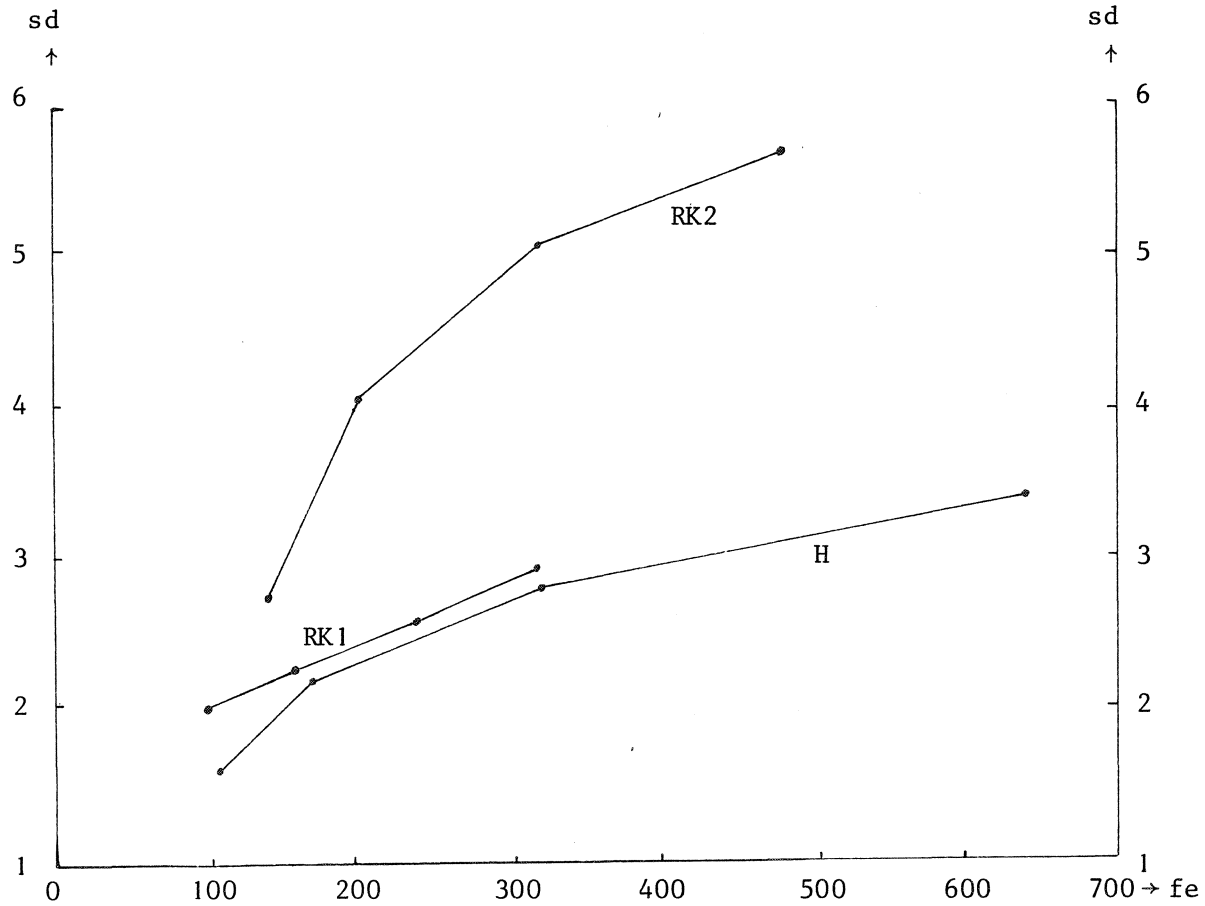
Figure 4 Problem V from [3], r = 0.



H			RK1			RK2		
step	fe	sd	step	fe	sd	step	fe	sd
20	32	-0.48	10	70	-0.41	20	140	1.36
40	62	-0.05	40	160	0.06	40	200	1.91
160	194	1.07	80	240	0.34	80	320	2.48
320	328	1.67	160	320	0.63	160	480	3.06

REMARKS. For this linear problem the second order Runge-Kutta method is clearly more effective than the hopscotch method, while the latter is clearly more effective than the first order Runge-Kutta method. The negative values of sd are due to the fact that the components of this problem are very small.

Figure 5 Problem VI from [3], r = 1.



H			RK1			RK2		
step	fe	sd	step	fe	sd	step	fe	sd
20	110	1.60	20	100	2.01	20	140	2.74
40	168	2.21	40	160	2.28	40	200	4.07
80	322	2.81	80	240	2.58	80	320	5.08
160	642	3.41	160	320	2.88	160	480	5.69

REMARKS. For the present non-linear problem the hopscotch method is slightly less efficient than the first order Runge-Kutta method. This is due to the fact that the factor r, given by (3.3), equals 1. The second order Runge-Kutta method is again significantly better.

4. CONCLUDING REMARKS

The main purpose of this report was to get an indication about the efficiency and accuracy of the odd-even hopscotch method when compared with stabilized Runge-Kutta methods. In this connection we make the following observations (bearing in mind the fact that the hopscotch method was not programmed optimally with respect to the solution of the scalar equations):

1. The second order Runge-Kutta method is clearly more efficient than the odd-even hopscotch method for most of the problems (for an exception to this rule see figure 1).
2. The first order Runge-Kutta method is less efficient than the odd-even hopscotch method.
3. The results of the odd-even hopscotch scheme (2.3) warrant further investigations to this type of schemes when formulated as Runge-Kutta schemes with matrix parameters. The main objective of such an investigation should be the development of more accurate schemes, while retaining the stability properties and computational simplicity of the odd-even hopscotch scheme.
4. The explicit Runge-Kutta schemes are conditionally stable, whereas the hopscotch schemes are unconditionally stable for an important class of problems. This fact is of importance when the steplength for the explicit schemes is determined by stability restrictions. In such a situation the hopscotch schemes may become more efficient than the stabilized Runge-Kutta schemes. This situation is left out of consideration in the previous tests.
5. Hopscotch type schemes, when formulated as Runge-Kutta schemes with matrix parameters, can be provided with steplength and error control in a similar manner as the stabilized Runge-Kutta schemes (cf. [8]).

REFERENCES

[1] DEKKER, K., *Generalized Runge-Kutta methods for coupled systems of hyperbolic differential equations*, Report NW 41/77, Mathematisch Centrum, Amsterdam, 1977.

[2] DEKKER, K., *A note on implicit generalized Runge-Kutta methods*, Mathematisch Centrum, Amsterdam (to appear).

[3] DEKKER, K., VAN DER HOUWEN, P.J., VERWER, J.G. & P.H.M. WOLKENFELT, *Comparing stabilized Runge-Kutta methods for semi-discretized parabolic and hyperbolic equations*, Mathematisch Centrum, Amsterdam (to appear).

[4] GOURLAY, A.R., *Hopscotch, a fast second order partial differential equation solver*, Jour. Inst. Math. Applics. 6, pp. 375-390, 1970.

[5] GOURLAY, A.R. & G.R. McGUIRE, *General hopscotch methods for partial differential equations*, Jour. Inst. Math. Applics. 7, pp. 216-227, 1971.

[6] LAMBERT, J.D., *Computational methods in ordinary differential equations*, John Wiley & Sons, 1973.

[7] SHAMPINE, L.F. & M.A. WATTS, *Block implicit one-step methods*, Math. Comp. 23, pp. 731-740, 1969.

[8] 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, 1977.