An Implementation of a Class of Stabilized Explicit Methods for the Time Integration of Parabolic Equations

J. G. VERWER
Mathematical Center, Amsterdam, The Netherlands

An implementation of a class of explicit three-step Runge–Kutta methods is described for the numerical solution of initial value problems for systems of ordinary differential equations. These systems originate from parabolic partial differential equations by applying the semidiscretization method. The underlying schemes are stabilized and are of first and second order. The number of function evaluations per step varies between 2 and 12. The implementation is provided with step length, error, and order control. A Fortran version of the implementation is available. Numerical results of the Fortran program, applied to two semidiscretized problems, are reported.

Key Words and Phrases: numerical analysis, parabolic partial differential equations, semidiscretization, implementation of time integrators
CR Categories: 5.17
The Algorithm: M3RK, An Explicit Time Integrator for Semidiscrete Parabolic Equations. *ACM Trans. Math. Softw. 6*, 2 (June 1980), 236–239.

1. INTRODUCTION

The implementation of a class of explicit methods to be used for the time integration of semidiscretized parabolic partial differential equations is discussed. The semidiscretized system of ordinary differential equations is supposed to be in the autonomous form

$$y' = f(y) . \tag{1.1}$$

An important property, possessed by the majority of semidiscretized parabolic systems, is that the spectrum of the Jacobian matrix, say $J(y)$, is almost real, i.e., the eigenvalues are situated in a long narrow strip around the negative axis of the complex plane. This property is essential for the implemented class of methods. Therefore it must be assumed that the problems, to which our time integrator is applied, possess this property.

At the present time many numerical methods exist for solving time-dependent

Author's address: Mathematisch Centrum, 2e Boerhaavestraat 49, 1091 AL Amsterdam, The Netherlands.

partial differential equations (see [8]). When dealing with more than one dimension, the majority of these methods are not so easy to apply and can only efficiently be implemented for narrow classes of problems. As a consequence, the development of mathematical software for wide classes of linear, and also nonlinear, partial differential equations is still in a very early state (see [11] for a list of references). This is in direct contrast to the development in the field of ordinary differential equations (see, e.g. [10]). Very capable software exists for wide classes of nonlinear ordinary differential equations. By way of the method of semidiscretization, we can make use of the developments in this field (e.g., step length and error control) for the implementation of a time integrator.

In this connection, stabilized explicit integration formulas are suitable because of the fact that these formulas, when used in conjunction with semidiscretization, are easy to apply and can be implemented for wide classes of linear and nonlinear problems in one or more dimensions. The only mathematical restriction, to be posed for parabolic problems, is that the spectrum of the Jacobian of the semidiscretized system is almost real. A practical restriction, with respect to the application of such methods, may arise when the spectral radius of $J(y)$ is extremely large. In spite of the relatively large stability boundaries, the methods are then forced to integrate with very small steps, which may result in an excessive run time. In such a situation it may be preferable to use an implicit method, which is unconditionally stable (see [8]).

The integrator discussed is based on three-step Runge–Kutta formulas of orders 1 and 2. These formulas are stabilized with respect to the real boundary of absolute stability. In fact, the stability regions are long narrow strips around the negative axis of the complex plane. The stabilization of the formulas is achieved by using extra evaluations of the function f per integration step. The number of function evaluations may vary between 2 and 12. The analysis and construction of the formulas are given in [14]. In Section 2 of this paper we give a short review of the theoretical aspects. Section 3 is devoted to the actual implementation. In this section simple mechanisms for the step length, error, and order control are discussed. In Section 4 we discuss M3RK, a Fortran program based on the implementation discussed in Section 3. The last section of this paper is devoted to a discussion of some numerical results obtained with M3RK.

Finally it should be noted that an Algol 60 version of an implementation of stabilized explicit one-step Runge–Kutta methods (cf. van der Houwen [13]) has already been given by Beentjes [3]).

2. THE CLASS OF INTEGRATION FORMULAS

In this section we discuss the underlying class of methods. The analysis and construction of the formulas are extensively discussed in Verwer [14, 16, 17], where one can also find further references.

Our class of three-step Runge–Kutta formulas may be represented as follows:

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

$$y_{n+1}^{(j)} = (1 - b_j)\,y_n + b_j y_{n-1} + c_j h f(y_{n-1}) + \lambda_j h f(y_{n+1}^{(j-1)}), \qquad j = 1, \ldots, m, \tag{2.1}$$

$$y_{n+1} = d y_{n+1}^{(m)} + (1 - d)\,y_{n-2}, \qquad m \geq 2, \quad n = 2, 3, \ldots.$$

The vector y_n denotes the approximation to the analytical solution $y(x)$ at $x = x_n$. The points x_j, $j = n-2, \ldots, n+1$, denote the reference points of the three-step formula and h denotes the step length, i.e., $h = x_{n+1} - x_n$. In this section h is supposed to be constant. For the application of (2.1) the additional starting vectors y_1 and y_2 must be given. Equations (2.1) are called a three-step formula of degree m, m being the number of function evaluations per integration step.

When applied to the scalar test model

$$y' = \delta y, \tag{2.2}$$

scheme (2.1) yields the linear recurrence relation

$$y_{n+1} = dS(z)y_n + dP(z)y_{n-1} + (1-d)y_{n-2}, \tag{2.3}$$

where $S(z)$ and $P(z)$ are polynomials of degree m in $z = h\delta$. The stability of method (2.1) depends on the parameter d and the coefficients of the stability polynomials S and P. We have implemented schemes of order $p = 1$ and $p = 2$, with degree m satisfying $2 \leq m \leq 12$. The corresponding polynomials S and P are such that the absolute stability regions contain a long narrow strip around the negative axis. We have

$$\beta_1(m) \simeq 5.15m^2, \qquad \beta_2(m) \simeq 2.29m^2, \tag{2.4}$$

where $\beta_p(m)$ denotes the real boundary of absolute stability for the pth order scheme of degree m. For real eigenvalues, the extrema of the amplification factors of (2.3) are bounded by about 0.9 in the stability interval. Because of this we have a strong damping for the higher harmonics.

The integration parameters of (2.1) are expressed in the parameter d and the coefficients of S and P. They are determined in such a way that the principal local truncation error, LTE_p say, is given by (see Verwer [14])

$$\begin{aligned} \mathrm{LTE}_1 &= C_2 h^2 y^{(2)}(x_n), \qquad C_2 \simeq 1.27, \\ \mathrm{LTE}_2 &= C_3 h^3 y^{(3)}(x_n), \qquad C_3 \simeq 0.44. \end{aligned} \tag{2.5}$$

Observe that LTE_p does not depend on m. For $p = 1, 2$ and $m = 2, \ldots, 12$, the coefficients s_i of S and p_i of P are given in Verwer [14]. The parameters b_j, c_j, and λ_j are given by

$$\begin{aligned} b_m &= p_0, \\ c_m &= \frac{(1 - \tfrac{1}{2}p_0)(p_1 - 2p_2 + 2p_3 + 2s_3) - (\tfrac{1}{2} + \tfrac{1}{4}p_0)^2}{2 + p_1 - 2p_2 + 2p_3 + 2s_3}, \\ \lambda_m &= 1 - \tfrac{1}{2}p_0 - c_m, \\ b_j &= 0, \qquad j = 1, \ldots, m-2, \\ b_{m-1} &= \frac{p_1 - c_m}{\lambda_m}, \\ c_j &= \frac{p_{m+1-j}}{s_{m-j}}, \qquad j = 1, \ldots, m-2, \\ c_{m-1} &= \frac{p_2}{\lambda_m}, \end{aligned} \tag{2.6}$$

$$\lambda_j = \frac{s_{m+1-j}}{s_{m-j}}, \qquad j = 1, \ldots, m-2,$$

$$\lambda_{m-1} = \frac{s_2}{\lambda_m}.$$

The parameter d is independent of m and is given by

$$d = 1.375, \quad p = 1, \quad d = 0.775, \quad p = 2. \tag{2.7}$$

Finally we mention the concept of internal stability. Because of the relatively large degree and relatively large stability boundaries, we have to deal with an accumulation of rounding errors which appears per integration step. Especially for the higher degree formulas, it can easily reduce the local accuracy. For a formula of degree m this accumulation is approximately governed by a so-called internal stability function, say $Q_{m-1}^{[p]}(z)$, which is a strongly increasing polynomial of degree $m-1$. Let σ denote the spectral radius. Then the accumulation is generally under control if we adjust the step length h and the degree m to the so-called internal stability condition

$$Q_{m-1}^{[p]}(h\sigma(J(y_n)) \leq \frac{\text{maximal local truncation error}}{\text{arithmetic precision}}. \tag{2.8}$$

In the program we use the values $Q_{m-1}^{[p]}(\beta_p(m))$, which are listed in Table I. The values for the second-order schemes, which are used in the program, are defined by $Q_{m-1}^{[2]}(\beta_2(m)) = 10^2 Q_{m-1}^{[1]}(\beta_1(m))$.

Finally we note that scheme (2.1) needs six arrays of storage. We also use six arrays for the actual implementation discussed in Section 3.

Table I. The Values $Q_{m-1}^{[1]}(\beta_1(m))$

m										
2	3	4	5	6	7	8	9	10	11	12
$^{3}10^{1}$	$^{1}10^{2}$	$^{7}10^{2}$	$^{4}10^{3}$	$^{3}10^{4}$	$^{2}10^{5}$	$^{9}10^{5}$	$^{5}10^{6}$	$^{3}10^{7}$	$^{2}10^{8}$	$^{1}10^{9}$

3. THE IMPLEMENTATION OF THREE-STEP RUNGE–KUTTA FORMULAS

When integrating time-dependent partial differential equations by using the semidiscretization method, there arise two types of discretization errors, viz., the error due to the spatial discretization and the error due to the time integration. In general, the first error cannot be controlled. In our opinion it is nevertheless useful to supply a method for the time integration with various control mechanisms if possible. By doing this one relieves the task of the user of such an integrator. To support this opinion we make the following observation. Our methods are conditionally stable. When applied to a nonlinear system, it may then happen that a sudden instability arises because of an increase of the spectral radius. When a method is supplied with error and step length control, such a sudden instability is immediately detected and the step length is decreased.

The most widely applied implementation technique for linear multistep methods is the Nordsieck technique (see Gear [5]). This technique makes it very

easy to realize error, step length, and order control. Compared with a Lagrange implementation, i.e., an implementation where the y and y' values are stored, a Nordsieck implementation is less efficient for large systems because of the higher overhead costs. Therefore we prefer the Lagrange implementation for our formulas. As we have to deal with a low order and with three-step formulas, this causes no particular problems.

The greater part of the ideas we apply are well known and extensively discussed in the literature (see, e.g., Gear [5] and Shampine and Gordon [10]). We therefore omit details where possible, but still observe that (as usual) most of the ideas we apply are based partly on theoretical arguments and partly on heuristics.

3.1 The Start of the Process

The two additional starting vectors y_1 and y_2 are computed by means of a one-step Runge–Kutta scheme of order $p = 2$, which is also formulated as a three-step scheme by introducing zero-parameters. The scheme is obtained from (2.1) by putting

$$d = 1, \qquad b_j = c_j = 0, \qquad \lambda_j = r_{m+1-j}/r_{m-j}, \qquad j = 1, \ldots, m, \quad 2 \le m \le 12, \tag{3.1}$$

where r_j, $j = 0, \ldots, m$, denote the coefficients of the corresponding mth degree stability polynomial, say R_m. For $m = 2$ this polynomial is given by $R_2(z) = 1 + z + \frac{1}{2}z^2$. For $3 \le m \le 12$ this polynomial is chosen equal to the stabilized polynomial $\tilde{R}_m^{(2)}$ given by van der Houwen [13, Table 2.6.7′]. The extrema of $\tilde{R}_m^{(2)}$ in its real interval of absolute stability, say $(-\tilde{\beta}_2(m), 0)$, are bounded by 0.95. Observe that for $m = 2$ no specific damping properties are imposed. For $m = 2$ the absolute stability boundary is 2. For convenience we approximate the boundaries $\tilde{\beta}_2(m)$ with

$$\tilde{\beta}_2(m) \simeq 0.44m^2 + 0.03m^3. \tag{3.2}$$

If $m \neq 12$, these approximations are slightly smaller than the true boundaries of $\tilde{R}_m^{(2)}$.

The internal stability behavior of the starting schemes is roughly the same as that of the three-step schemes. In particular, the values given in Table I hold for the starting schemes.

In order to start the process we need an initial step length, say h_{start}. This initial step length should be related to the local tolerance, say TOL, which is specified by the user. We estimate h_{start} as follows. Let $\sigma_0 = \sigma(J(y_0))$, σ denoting the spectral radius, be given (if σ_0 is not available, it is estimated by the program as outlined in Section 3.5). Let $\|\cdot\|$ denote the divided Euclidean norm (i.e., Euclidean norm divided by the square root of the number of components). The idea is now to estimate $\frac{1}{2}\sigma_0^{-2}y^{(2)}(x_0)$ which represents the last Taylor term taken into account by the actual start formula, obtained with step size σ_0^{-1}. By relating this conservative estimation of the principal local truncation error of the start formula with TOL, we reasonably obtain a safe estimation of h_{start}. Following this idea h_{start} is then defined by

$$h_{\text{start}} = \sigma_0^{-1}(\eta_t/\eta_e)^{1/2}/10, \tag{3.3}$$

where

$$\eta_t = \mathrm{TOL} + \mathrm{TOL} * \|y_0\|,$$
$$\eta_e = \sigma_0^{-1} \| f(y_0 + \sigma_0^{-1} f(y_0)) - f(y_0) \|. \tag{3.4}$$

It should be observed that in the root formula (3.3), which is used to extrapolate a new step size (cf. Gear [5, p. 156]), the estimation of $\frac{1}{2}\sigma_0^{-2} y^{(2)}(x_0)$ is multiplied by 200 to obtain an extra safety margin.

In order to obtain absolute stability at the start of the process, the initial step length must satisfy the stability condition

$$h_{\mathrm{start}}\, \sigma_0 \leq \tilde{\beta}_2(m_{\max}), \tag{3.5}$$

where $m_{\max}$ denotes the maximal degree allowed with respect to internal stability. The estimation of σ_0 and $m_{\max}$ is discussed in Section 3.5. If h_{start} does not satisfy (3.5), we put $h_{\mathrm{start}} = \tilde{\beta}_2(m_{\max})/\sigma_0$.

3.2 Estimation and Control of the Local Error

For the error control we use the local truncation error which is estimated by LTE_p (see (2.5)). For the estimation of LTE_p, $p = 1, 2$, we apply the simple interpolation formulas ($n > 2$)

$$\mathrm{LTE}_1 \simeq \frac{C_2}{1 - C_2} [y_{n+1} - 2y_n + y_{n-1}], \tag{3.6}$$

$$\mathrm{LTE}_2 \simeq \frac{C_3}{1 - C_3} [y_{n+1} - 3(y_n - y_{n-1}) - y_{n-2}],$$

where, according to the definition of LTE_p, y_{n-1} and y_{n-2} are assumed to be approximations of a sufficiently high order to a local analytical solution at the points x_{n-1} and x_{n-2}, respectively.

The error criterion which is to be performed after each nth integration step, $n > 2$, is the mixed criterion

$$\| \mathrm{LTE}_p \| \leq \mathrm{TOL} + \mathrm{TOL} * \| y_{n+1} \|, \tag{3.7}$$

where TOL stands for a user-specified tolerance parameter and $\| \cdot \|$ denotes the divided Euclidean norm. If (3.7) is satisfied, the integration step is accepted; otherwise it is rejected.

During the start of the process, i.e. , if $n = 1, 2$, no error control is performed. This is justified by the conservative estimation of the initial step length. If the third step fails, however, all results are rejected and the process is restarted with $h = h/10$.

3.3 Changing the Step Length

Before discussing the estimation and control of the step length we first mention how we realize the change. Our integration formulas (2.1) are developed for a fixed step length. This means changing the step size must be handled separately. In a Nordsieck implementation, changing the step size is an interpolation–extrapolation process (see Gear [5]). We also use interpolation–extrapolation to determine the new y values.

Let h and αh denote the old and new step length, respectively. The new values of y_{n-1} and y_{n-2} are then interpolated or extrapolated by means of the quadratic formula

$$y(x-\bar{\alpha}h) \simeq \tfrac{1}{2}\bar{\alpha}(\bar{\alpha}-1)y(x-2h) + \bar{\alpha}(2-\bar{\alpha})y(x-h) + \tfrac{1}{2}(2-\bar{\alpha})(1-\bar{\alpha})y(x), \tag{3.8}$$

where $\bar{\alpha} = \alpha$ or $\bar{\alpha} = 2\alpha$, respectively. Equation (3.8) is applied for $p = 1$ and $p = 2$. The error introduced by (3.8) is of order 3 and is ignored at the estimation of LTE_2.

Applying (3.8) too frequently may lead to severe instabilities. Therefore we also use the rule of thumb: after a change of h at least four steps are performed with h fixed, provided a step is not rejected. We return to this point in Section 3.4. The new values of y'_{n-1} are not computed by means of interpolation or extrapolation, but by using the derivative function $f(y)$. In our experience this leads to a more stable process of step-length changing.

3.4 Estimation and Control of the Step Length and Order

The new step length αh is estimated using the well-known root formula. Let $\bar{\alpha}$ be defined by

$$\bar{\alpha} = \left(\frac{\mathrm{TOL} + \mathrm{TOL} * \|y_{n+1}\|}{\|\mathrm{LTE}_p\|}\right)^{(p+1)^{-1}}. \tag{3.9}$$

Then we put

$$\alpha = \begin{cases} \bar{\alpha}/2.0, & p = 1, \\ \bar{\alpha}/1.6, & p = 2. \end{cases} \tag{3.10}$$

The factors 2.0 and 1.6 are to provide a conservative estimate. In order to prevent marginal changes, the change is not performed when $0.9 < \alpha < 1.1$ Moreover, in order to prevent an excessive decrease or increase of the step length, α is bounded by 0.1 and 3.0, respectively. Because of the factors in (3.10), a decrease of the step length is not necessarily due to a step failure.

The estimate of α is made when a step fails or at least four steps have been performed after the last change. Because of the fact that repeated rejections may be caused by severe errors, the process is interrupted if this happens three times in succession. After this interruption we make a restart as described in Section 3.1.

Our formulas are explicit and thus conditionally stable. Therefore the step length h is always bounded by

$$h_{\max}(p) = \beta(m_{\max})/\sigma, \tag{3.11}$$

$h_{\max}(p)$ being the maximal step length with respect to absolute stability. In (3.11), $\beta(m_{\max})$ stands for $\beta_1(m_{\max})$, $\beta_2(m_{\max})$, or $\tilde{\beta}_2(m_{\max})$.

Next we mention the implemented order control which is very simple and based on the fact that, in general, the step length is bounded by stability requirements. As already observed, the process is always started with a second-order one-step scheme and a second-order three-step scheme. During the process

h normally increases until $h = h_{\max}(2)$. If h reaches this value, four steps are performed with the second-order scheme and $h = h_{\max}(2)$, provided no step failure occurs. Then, α is estimated for $p = 1$. If this particular $\alpha < 1.1$, the process is continued with $h = h_{\max}(2)$ and the current second-order scheme. Otherwise the process is continued with a first-order scheme, but, as a matter of caution, with $h = h_{\max}(2)$. Then the next time α is estimated, h is allowed to increase while $p = 1$. This specific check for an order decrease is made every four steps, provided $h = h_{\max}(2)$.

If during a first-order integration h becomes smaller than $h_{\max}(2)$, p is reset to 2. It is observed that an order increase is not necessarily due to a step failure.

3.5 Estimation and Control of the Degree and Spectral Radius

In order to control the propagation of local errors per integration step, we want to satisfy condition (2.8). This is achieved by putting $m \leq m_{\max}$, $m_{\max}$ being the maximal degree of the schemes, which satisfies (see Table I)

$$Q_{m-1}^{[p]}(\beta_p(m)) \leq \frac{\text{TOL}}{\text{arithmetic precision}}. \tag{3.12}$$

For the three-step schemes $m_{\max}$ thus depends on p; there holds $m_{\max}(2) \leq m_{\max}(1)$. The maximal degree for the starting scheme is chosen equal to $m_{\max}(2)$ of the three-step scheme. If the exceptional situation arises that $m_{\max} < 2$ (the quotient of TOL and arithmetic precision is too small), the process is discontinued.

A property of stabilized methods is that the local truncation error of the formulas is approximately independent of m. Thus it is useful to minimize m with respect to the stability condition

$$h\sigma \leq \beta(m) \tag{3.13}$$

for given h and σ, while $\beta(m)$ represents $\beta_1(m)$, $\beta_2(m)$, or $\tilde{\beta}_2(m)$, respectively. The degree m is computed in this way at the start of the process, at the changeover from a one-step to a three-step scheme, and further every time h or p is changed.

From the foregoing it is clear that we need an estimation of σ. Because σ is used to determine $h_{\max}(p)$ and to select m minimal with respect to (3.13), it must always be an upper estimation. Once σ is estimated and the system to be integrated is nonlinear, it may be necessary to control the variation of σ. Thus we also need a control mechanism for the spectral radius. With respect to the estimation and control of σ we distinguish between three options; which option is chosen has to be specified by the user.

Option I. The user provides an estimation of σ. This is often easy to do, especially for linear problems.

Option II. The user does not provide an estimation. In this case σ is estimated by means of a power method which is adapted for general nonlinear vector functions f (cf. Lindberg [6]). This method may be described as follows.

Suppose $\sigma_0 = \sigma(J(y_0))$ is to be estimated. Let r_i be a random number from $[-\epsilon, \epsilon]$, $\epsilon > 0$, and let v_0 be defined componentwise by

$$v_{0,i} = \begin{cases} y_{0,i}(1 + r_i), & y_{0,i} \neq 0, \\ r_i, & y_{0,i} = 0. \end{cases} \tag{3.14}$$

Let $\epsilon_{\max} = \max(\epsilon, \epsilon \| v_0 \|_2)$. The adapted power method is then defined by the iteration

$$v_{j+1} = v_0 + \epsilon_{\max} \frac{f(v_j) - f(v_0)}{\| f(v_j) - f(v_0) \|_2}, \tag{3.15}$$

$$\rho_{j+1} = \frac{\| f(v_{j+1}) - f(v_0) \|_2}{\epsilon_{\max}},$$

where $v_1 = y_0$ and $j = 1, 2, \ldots$. If f is linear, i.e., $J(y)$ constant, then $\rho_j \to \sigma_0$. If f is nonlinear, then choosing ϵ sufficiently small, $J(y)$ is approximately constant in $S(v_0, \epsilon_{\max}) = \{v \mid \| v - v_0 \|_2 \leq \epsilon_{\max}\}$. Thus for ϵ sufficiently small, ρ_j will converge to an accurate estimation of σ_0.

In our program we have set $\epsilon = 10^4$ APR, APR denoting the arithmetic precision (e.g., for CDC Cyber $\epsilon = 10^{-10}$). The iteration (3.15) is stopped as soon as $| \rho_{j+1} - \rho_j | \leq 10^{-3} \rho_{j+1}$, provided $j \geq 4$. If this inequality is not satisfied within 50 iterations, the whole process is discontinued. In general, the process converges slowly because of the absence of a dominant eigenvalue. As a matter of safety we therefore put $\sigma_0 = 1.1\rho$, ρ being the last iterate.

In case of the second option, the variation of the spectral radius is also controlled. We distinguish between two situations. First, σ increases during the course of the integration and the process becomes unstable. The instability is immediately detected by the error control and results in a step failure. After a step failure we therefore simply reestimate σ, provided the failure was not in succession. Second, σ decreases during the course of the integration. To detect a decrease of σ we use an inaccurate estimation of σ, say σ^*, which is given by $\sigma^* = \rho_3$. The estimation σ^* is computed every 25 steps since the last estimation of σ or σ^*, and we decide to reestimate σ if σ^* has been decreased more than 10 percent.

We note that ρ_3 is in most cases a very rough approximation to σ. In our purpose this rough approximation suffices. At this place it is emphasized that random values are used in formula (3.14). Though these values are small, they may slightly influence σ and, in particular, ρ_3. As a consequence, one should use a random number generator using a fixed generative value to be able to recover earlier obtained results (see also Section 4).

Option III. The user does not provide an estimation, but decides that an initial estimation by means of the power method at the start suffices. Thus in this case no control on the variation of σ is performed. For linear problems it is clear that one chooses between option I and option III.

4. A FORTRAN PROGRAM

A Fortran version of the implementation is available (see [15]). This program consists of the driver M3RK and 11 auxiliary subroutines, of which each performs a special task as discussed in Section 3. For specific information about input requirements and output we refer to the prologue of comments of M3RK, which further explains how to use it. Here we give a detailed flowchart which should clarify the algorithmic connection between the mechanisms for controlling the order p, the degree m, the step size h, and the spectral radius σ, and which should

facilitate the reading of M3RK. To study the flowchart (see Figure 1) the following information suffices:

(1) The program integrates a given problem from an initial value of x to a value of x which may be slightly beyond a user-specified output point, say x_e. The desired solution at the output point x_e is always determined by means of quadratic interpolation. After a normal return of M3RK the parameters in the call list are ready to continue the integration. This means that when the user decides to continue the integration, he only needs to define a new output point x_e and call again.

(2) The names in the flowchart are the names of the auxiliary subroutines. They perform the following tasks: HSTART computes the initial step length according to Section 3.1. PARAM delivers the integration parameters (2.6) and (3.1). POWERM estimates σ as given in Section 3.5; if the estimation fails, the error flag IFLAG := 3. MAXDEG computes the maximal degree $m_{\max}$ (cf. (3.12)); if $m_{\max} < 2$, IFLAG := 2. MINDEG computes the minimal degree satisfying (3.13). STEP computes the approximation y_{n+1} defined by (2.1). ESTIMA estimates the local error according to Section 3.2. NEWH delivers a new step size and the factor α of (3.10). INTER1 performs the interpolation (3.8). INTER2 performs the interpolation at x_e. SHIFT shifts the data for a next step.

(3) In the flowchart, r, n, and s, respectively, denote the number of successive rejected steps, the number of steps after start or restart, and the number of steps after an estimation or check for an estimation of σ. Further, $k = 1$ for a one-step formula and $k = 3$ for a three-step one. Finally, at the first call of M3RK the user has to specify a maximum number of f evaluations to be spent. If this number is exceeded, IFLAG := 1.

There remains to make some comments about the portability of the program. The whole package has been tested on a CDC 73/28 using 14 digits. It has been accepted by the PFORT verifier (see Ryder [9]). The PFORT verifier is a program which checks a Fortran program for adherence to PFORT, a portable subset of American National Standard Fortran. M3RK uses one machine-dependent constant, namely, the arithmetic precision represented by the internal variable APR. POWERM contains the CDC system subprograms RANSET and RANF, constituting a random number generator. RANF is the actual generator, while RANSET initializes the generative value of RANF. It is emphasized that replacing the CDC random number generator slightly influences the results given in Section 5.

5. NUMERICAL EXAMPLES

In order to test subroutine M3RK, it was applied to several problems. Two of these problems are discussed in this section. Both problems are nonlinear and arise in practice. For both problems the semidiscretization has been performed by means of a continuous time Galerkin method (see, e.g., Douglas and Dupont [4]).

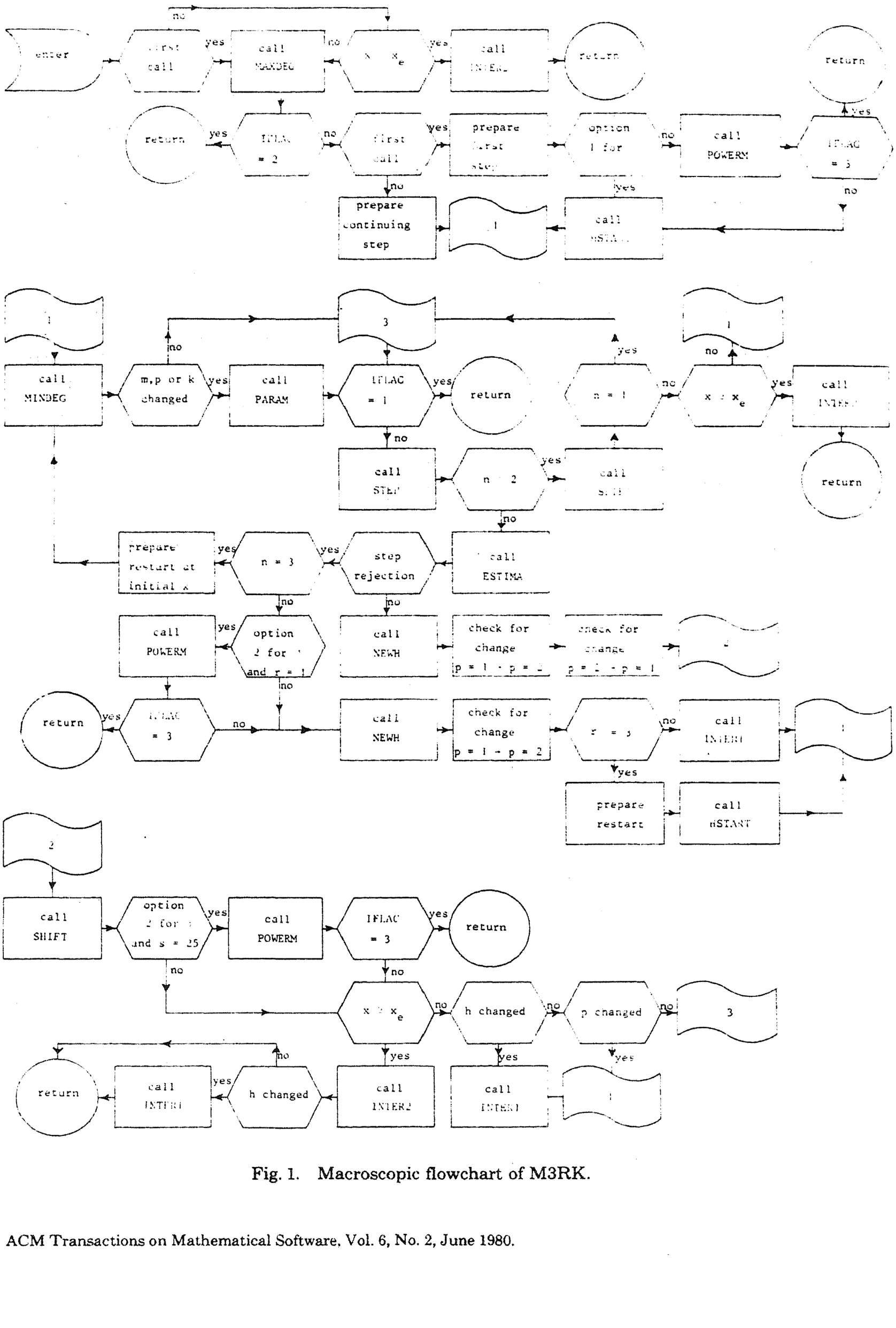

Fig. 1. Macroscopic flowchart of M3RK.

5.1 A One-Dimensional System of Two Nonlinear Equations

The first problem we consider is a one-dimensional system of two nonlinear equations from electricity theory (cf. Te Riele [12, Sec. 3.2.6]):

$$\begin{aligned} \frac{\partial u}{\partial t} &= \epsilon\rho \frac{\partial^2 u}{\partial x^2} - g(u - v), \\ \frac{\partial v}{\partial t} &= \rho \frac{\partial^2 v}{\partial x^2} + g(u - v), \end{aligned} \tag{5.1}$$

where $0 \le x \le 1$, $g(z) = \exp(\frac{1}{3}\mu z) - \exp(-\frac{2}{3}\mu z)$, $\mu = 17.19$, $\epsilon = 0.143$, $\rho = 0.1743$. The corresponding initial and boundary conditions read

$$\begin{aligned} &u = 1, \quad v = 0, && 0 \le x \le 1, \quad t = 0, \\ &\frac{\partial u}{\partial x} = v = 0, && x = 0, \quad t > 0, \\ &u = 1, \quad \frac{\partial v}{\partial x} = 0, && x = 1, \quad t > 0. \end{aligned} \tag{5.2}$$

Equation (5.1) was semidiscretized by means of a Galerkin method based on piecewise quadratic polynomials and implemented to yield a purely explicit system of ordinary differential equations (cf. Bakker [1]). It is beyond the scope of this paper to discuss this discretization technique. We confine ourselves therefore to the definition of the semidiscretized system obtained for the equidistant grid $\{x_i \mid x_i = (i-1)/(M-1), i = 1, \ldots, M; M \text{ odd}\}$. Let $u_i(t) \simeq u(t, x_i)$ and $v_i(t) \simeq v(t, x_i)$. The equations for the u_i components then are

$$\begin{aligned} \dot{u}_1 &= -\tfrac{1}{2}\epsilon\rho(M-1)^2[7u_1 - 8u_2 + u_3] - g(u_1 - v_1), \\ \dot{u}_{2i} &= -\epsilon\rho(M-1)^2[2u_{2i} - u_{2i-1} - u_{2i+1}] - g(u_{2i} - v_{2i}), \\ &\qquad i = 1, \ldots, \frac{M-1}{2}, \\ \dot{u}_{2i+1} &= -\tfrac{1}{4}\epsilon\rho(M-1)^2[14u_{2i+1} - 8(u_{2i+2} + u_{2i}) + u_{2i+3} + u_{2i-1}] \\ &\quad - g(u_{2i+1} - v_{2i+1}), \qquad i = 1, \ldots, \frac{M-3}{2}, \\ \dot{u}_M &= 0. \end{aligned} \tag{5.3}$$

We do not give the equations for the v_i components, because they are now easy to find.

We integrated two systems, viz., system I and II obtained by setting $M = 31$ and $M = 61$, respectively. Both systems were integrated over the interval [0, 20] by calling M3RK for $x_e = 10^{-2}$ (first call), and for $x_e = 10^{-1}$, 1, 5, 10, 20 (subsequent calls). They were integrated for three values of TOL, viz., 10^{-3}, 10^{-4}, and 10^{-5}, using the second option for the spectral radius.

Results of the integrations are listed in Tables II and III. Table II gives the system I and system II approximations to $u(t, x)$ for the specified number of

Table II. Approximations to $u(t, x)$ of Equations (5.1) at Several Times and Points

	System I						System II					
$t \backslash x$	0	0.2	0.4	0.6	0.8	0.9	0	0.2	0.4	0.6	0.8	0.9
						TOL = 10^{-3}						
10^{-2}	0.5279	0.6875	0.6875	0.6875	0.6875	0.6880	0.5495	0.6876	0.6876	0.6876	0.6876	0.6880
10^{-1}	0.2191	0.4741	0.5103	0.5122	0.5240	0.5771	0.2215	0.4742	0.5103	0.5121	0.5223	0.5714
1	0.0421	0.1985	0.3675	0.5122	0.6518	0.7378	0.0424	0.1980	0.3663	0.5102	0.6489	0.7345
5	0.0330	0.1637	0.3227	0.4812	0.6398	0.7319	0.0332	0.1627	0.3208	0.4785	0.6365	0.7283
10	0.0327	0.1623	0.3203	0.4785	0.6374	0.7300	0.0329	0.1617	0.3190	0.4764	0.6347	0.7269
20	0.0327	0.1623	0.3204	0.4785	0.6375	0.7301	0.0329	0.1617	0.3190	0.4764	0.6347	0.7269
						TOL = 10^{-4}						
10^{-2}	0.5270	0.6869	0.6869	0.6869	0.6869	0.6873	0.5487	0.6869	0.6869	0.6869	0.6869	0.6873
10^{-1}	0.2183	0.4737	0.5099	0.5118	0.5235	0.5767	0.2208	0.4739	0.5099	0.5117	0.5219	0.5710
1	0.0419	0.1980	0.3671	0.5123	0.6522	0.7382	0.0422	0.1976	0.3660	0.5103	0.6492	0.7348
5	0.0330	0.1637	0.3226	0.4811	0.6398	0.7318	0.0332	0.1629	0.3211	0.4789	0.6369	0.7286
10	0.0327	0.1624	0.3204	0.4786	0.6376	0.7301	0.0329	0.1617	0.3190	0.4765	0.6348	0.7270
20	0.0327	0.1623	0.3204	0.4785	0.6375	0.7301	0.0329	0.1617	0.3190	0.4764	0.6347	0.7269
						TOL = 10^{-5}						
10^{-2}	0.5267	0.6867	0.6867	0.6867	0.6867	0.6871	0.5485	0.6867	0.6867	0.6867	0.6867	0.6871
10^{-1}	0.2181	0.4737	0.5097	0.5117	0.5234	0.5766	0.2206	0.4738	0.5097	0.5116	0.5217	0.5709
1	0.0419	0.1979	0.3670	0.5124	0.6523	0.7383	0.0422	0.1975	0.3659	0.5104	0.6493	0.7349
5	0.0330	0.1637	0.3226	0.4811	0.6397	0.7318	0.0332	0.1630	0.3213	0.4791	0.6370	0.7287
10	0.0328	0.1624	0.3205	0.4786	0.6376	0.7302	0.0329	0.1617	0.3190	0.4765	0.6348	0.7270
20	0.0327	0.1623	0.3204	0.4785	0.6375	0.7301	0.0329	0.1617	0.3190	0.4764	0.6347	0.7269

output times and some grid values x_i. The approximations were rounded to 4 decimal places. Comparing the results for a system for several values of TOL yields an indication about the accuracy of the time integration. Comparing the results for both systems for several values of TOL yields an indication about the accuracy of the space discretization. From this table we thus have an indication about the accuracy of the approximations to $u(t, x)$, i.e., the solution of one of the components of the partial differential equation, at several times and points.

To get some insight into the course of the time integrations, and thus into the behavior of M3RK, Table III gives for all integrations for each output time the following information: step = the total number of steps, restep = the number of rejected steps, fev = the total of $f(y)$ evaluations, sig = the number of $f(y)$ evaluations needed for the estimation and control of the spectral radius, sigma = the estimation of the spectral radius used by M3RK. We observe that for both systems no step rejections occurred. Among others, Table III clearly shows the increase of the step size during the process. Because of the fact that the problem possesses a steady state solution, such an increase must occur. From this table we can also calculate the average number of function evaluations per step at the given output times. We also see that the average number of function evaluations decreases as TOL becomes smaller. This is due to the fact that for the present

Table III. Information About M3RK Concerning Equations (5.3)

	System I					System II				
t	step	restep	fev	sig	sigma	step	restep	fev	sig	sigma
				TOL = 10^{-3}						
0	0	0	21	21	4462.2	0	0	21	21	6871.4
10^{-2}	41	0	137	43	1360.7	42	0	136	40	4406.4
10^{-1}	64	0	198	48	1360.7	64	0	215	45	4406.4
1	88	0	337	53	1360.7	88	0	429	50	4406.4
5	108	0	568	58	1360.7	127	0	876	60	4406.4
10	120	0	698	58	1360.7	157	0	1241	65	4406.4
20	139	0	931	63	1360.7	217	0	1971	75	4406.4
				TOL = 10^{-4}						
0	0	0	21	21	4462.2	0	0	21	21	6871.4
10^{-2}	73	0	231	65	1240.7	74	0	212	44	4992.3
10^{-1}	116	0	338	75	1240.7	117	0	345	54	4992.3
1	161	0	525	85	1240.7	162	0	674	64	4992.3
5	193	0	803	90	1240.7	237	0	1394	79	4992.3
10	219	0	1047	95	1240.7	275	0	1847	89	4992.3
20	238	0	1272	100	1240.7	342	0	2661	99	4992.3
				TOL = 10^{-5}						
0	0	0	21	21	4462.2	0	0	21	21	6871.4
10^{-2}	139	0	423	109	1422.2	139	0	386	72	4819.5
10^{-1}	224	0	637	133	1053.4	224	0	605	87	4819.5
1	314	0	914	153	1053.4	314	0	1080	107	4819.5
5	369	0	1275	163	1053.4	422	0	2066	127	4819.5
10	397	0	1532	168	1053.4	506	0	2832	147	4819.5
20	428	0	1829	178	1053.4	571	0	3623	157	4819.5

problem the step size is mostly restricted by accuracy requirements. If this is the case the degree is minimized. As a consequence, a significantly larger number of steps, due to a smaller value of TOL, does not always yield a significantly larger number of function evaluations. This fact is clearly illustrated by Table III. It may thus be preferred to choose TOL not too large. Moreover, the various control mechanisms work better for smaller values of TOL. We conclude this example by noting that all integrations were performed without the necessity of making restarts.

5.2 A Nonlinear, Two-Dimensional Problem

The second problem we consider is a two-dimensional diffusion problem:

$$\frac{\partial u}{\partial t} = \beta(u)\left[\frac{\partial^2 u}{\partial z^2} + \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right] + \gamma(u), \quad 0 \le r \le r_e, \quad 0 \le z \le z_e,$$

$$u(0, r, z) = 500, \qquad 0 \le r \le r_e, \quad 0 \le z \le z_e, \tag{5.4}$$

$$u(t, r, 0) = u(t, r, z_e) = 500, \qquad 0 \le r \le r_e, \quad t > 0,$$

$$u_r(t, 0, z) = 0, \quad u_r(t, r_e, z) = \mu(u(t, r_e, z)), \qquad 0 \le z \le z_e, \quad t > 0,$$

where $r_e = 10^{-4}$, $z_e = 0.15$, and $\beta(u) = \lambda(u)/\rho(u)$, $\gamma(u) = \eta(u)/\rho(u)$, $\mu(u) = \psi(u)/\lambda(u)$. The functions λ, ρ, η, and ψ are defined by

$$\begin{aligned}
\lambda(u) &= 418.4 * \{0.1287496_{10} - 13u^4 - 0.116873_{10} - 9u^3 \\
&\qquad + 0.384891_{10} - 6u^2 - 0.569812_{10} - 3u + 0.57185303\}, \\
\rho(u) &= 19300 * \{0.14644_{10}3 + 0.18513_{10} - 1 * (u - 0.104_{10}4)\}, \\
\eta(u) &= 7.1486^2 * \pi^{-2} * 10^{10} * \{-0.378808_{10} - 1 + 0.267688_{10} - 3u \\
&\qquad + 0.1724588_{10} - 7u^2\}, \\
\psi(u) &= -0.56696_{10} - 7u^4 * \{-0.270491_{10} - 17u^5 + 0.3438691_{10} - 13u^4 \\
&\qquad - 0.163905_{10} - 9u^3 + 0.3305647_{10} \\
&\qquad - 6u^2 - 0.1194_{10} - 3u + 0.2667112_{10} - 1\}.
\end{aligned} \tag{5.5}$$

Problem (5.4) describes the temperature in a filament and was communicated to us by Polak [7], who is gratefully acknowledged for his cooperation.

For the semidiscretization of (5.4) we have applied a Galerkin method based on piecewise bilinear functions. We have made use of an implementation, written by Bakker [2], yielding a purely explicit initial value problem. Again we shall confine ourselves to the definition of the resulting system of ordinary differential equations.

Let $\{r_i \mid r_1 = 0, r_i < r_{i+1}, i = 1, \ldots, M-1, r_M = r_e\}$ and $\{z_j \mid z_1 = 0, z_j < z_{j+1}, j = 1, \ldots, N-1, z_N = z_e\}$ be partitions of the r-axis and z-axis, respectively. Let $p_i(r)$, $i = 1, \ldots, M$, and $q_j(z)$, $j = 1, \ldots, N$, be piecewise linear functions, i.e., $p_i(r_k) = \delta_{ik}$ and $q_j(z_1) = \delta_{j1}$, where δ denotes the Kronecker symbol. Further, let $u_{ij}(t) \simeq u(t, r_i, z_j)$. The resulting system of ordinary differential equations is then defined by

$$\begin{aligned}
&\dot{u}_{ij} = 0, \qquad i = 1, \ldots, M, \quad j = 1, N, \\
&\dot{u}_{ij} = v_i^{-1} w_j^{-1} \beta(u_{ij}) \{r_e w_j \mu(u_{Mj}) \delta_{iM} - a_{ij} u_{ij} - a_{ij-1} u_{ij-1} \\
&\qquad - a_{ij+1} u_{ij+1} - a_{i-1j} u_{i-1j} - a_{i+1j} u_{i+1j}\} + \gamma(u_{ij}), \\
&\qquad i = 1, \ldots, M, \quad j = 2, \ldots, N-1,
\end{aligned} \tag{5.6}$$

where

$$\begin{aligned}
v_i &= \int_0^{r_e} r p_i(r)\, dr, \qquad w_j = \int_0^{z_e} q_j(z)\, dz, \\
a_{kl} &= \int_0^{r_e} \int_0^{z_e} r[\dot{p}_i(r)\dot{p}_k(r) q_j(z) q_1(z) + p_i(r) p_k(r) \dot{q}_j(z) \dot{q}_1(z)]\, dr\, dz.
\end{aligned} \tag{5.7}$$

Again, M3RK was applied to two systems. For both systems, $r_i = (i-1)r_e/(M-1)$, $i = 1, \ldots, M$, and $z_j = 0.06 * (j-1)/(N-1)$, $j = 1, \ldots, (N+3)/4$; $z_j = z_{j-1} + 0.24/(N-1)$, $j = (N+7)/4, \ldots, (3N+1)/4$; $z_j = z_{j-1} + 0.06/(N-1)$, $j = (3N+5)/4, \ldots, N$. The choice $M = 5$, $N = 21$ and $M = 5$, $N = 41$ yields system I and system II, respectively. Because of the physical nature of the

Table IV. Approximations to $u(t, 0, z)$ of Equations (5.6) at Several Times and Points

	System I					System II				
t \ z	0.003	0.006	0.009	0.015	0.075	0.003	0.006	0.009	0.015	0.075
					TOL = 10^{-3}					
0.1	700.9	749.0	757.3	758.6	758.5	706.2	752.0	757.9	758.4	758.4
0.2	952.2	1124.1	1172.6	1185.9	1187.1	956.2	1131.0	1176.1	1185.5	1185.6
0.4	1746.1	2316.2	2514.3	2593.5	2596.4	1738.2	2324.9	2523.5	2594.6	2597.0
0.6	2554.0	3156.2	3287.6	3326.6	3328.1	2511.2	3159.4	3291.1	3326.1	3327.6
0.8	2816.3	3304.3	3382.0	3397.2	3397.9	2757.3	3309.4	3385.1	3397.5	3397.8
0.9	2847.7	3317.5	3388.2	3400.7	3400.9	2786.4	3322.2	3390.9	3400.7	3400.9
1.0	2858.8	3321.9	3390.1	3401.5	3401.6	2797.2	3326.7	3392.7	3401.5	3401.6
					TOL = 10^{-4}					
0.1	701.0	749.5	758.0	759.3	759.3	706.3	752.6	758.7	759.3	759.3
0.2	952.8	1125.3	1174.2	1187.7	1187.9	957.0	1132.5	1177.9	1187.8	1187.9
0.4	1748.4	2319.4	2517.5	2597.1	2600.0	1739.5	2326.9	2525.9	2597.4	2599.9
0.6	2551.8	3153.2	3284.6	3323.7	3325.2	2509.6	3157.3	3289.0	3324.1	3325.6
0.8	2816.7	3304.2	3381.9	3397.2	3397.2	2757.0	3308.9	3384.8	3397.3	3397.6
0.9	2847.7	3317.5	3388.2	3400.7	3400.8	2786.5	3322.2	3391.0	3400.7	3400.9
1.0	2858.7	3321.9	3390.1	3401.5	3401.6	2797.5	3326.8	3392.8	3401.5	3401.6
					TOL = 10^{-5}					
0.1	701.0	749.5	758.0	759.3	759.3	706.3	752.6	758.7	759.3	759.3
0.2	952.8	1125.3	1174.2	1187.7	1187.9	957.0	1132.5	1178.0	1187.8	1187.9
0.4	1748.9	2319.8	2517.7	2597.1	2600.0	1740.1	2327.4	2526.2	2597.6	2600.0
0.6	2552.1	3153.3	3284.7	3323.7	3325.2	2509.4	3156.9	3288.6	3323.7	3325.2
0.8	2815.0	3303.4	3381.4	3396.8	3397.1	2755.3	3307.9	3384.2	3396.8	3397.2
0.9	2846.5	3317.0	3388.0	3400.5	3400.7	2785.5	3321.8	3390.7	3400.6	3400.7
1.0	2858.2	3321.7	3390.0	3401.4	3401.6	2797.0	3326.6	3392.7	3401.5	3401.6

problem it is not necessary to choose $M > 5$. A finer grid at the endpoints of the filament is necessary in order to deal with boundary layers. It should be observed that, because of two reasons, the resulting systems are difficult problems for our explicit integrator. First, the systems are strongly nonlinear. Second, in spite of the relatively small number of gridpoints, we have to deal with a large spectral radius because of the very small size of the r interval.

Both systems were integrated over the interval [0, 1] by calling M3RK for $x_e = 0.1$ (first call), and for $x_e = 0.2$, 0.4, 0.6, 0.8, 0.9 and 1 (subsequent calls). They were integrated for three values of TOL, viz., 10^{-3}, 10^{-4}, and 10^{-5}, again using the second option for the spectral radius.

Some results of the integrations are listed in Tables IV and V. Table IV gives the system I and system II approximations to $u_{1j}(t)$ at the specified output times for some values of $z_j \in [0, z_e/2]$. Results for $z \in [z_e/2, z_e]$ are omitted, as the solution is approximately symmetric with respect to $z_e/2$. With respect to comparing and interpreting results of Table IV, we refer to the remarks made in the preceding example.

Table V yields the same type of information as Table III. The rejections clearly indicate that the guidance of the process is better for smaller values of TOL.

Table V. Information About M3RK Concerning Equations (5.6)

	System I					System II				
t	step	restep	fev	sig	sigma	step	restep	fev	sig	sigma
				TOL = 10^{-3}						
0	0	0	11	11	433026.3	0	0	11	11	433627.5
0.1	148	5	1613	92	358302.0	150	3	1627	96	385907.2
0.2	303	12	3321	299	318363.4	286	7	3177	236	350639.9
0.4	494	16	5457	430	249119.8	565	21	6273	558	290769.0
0.6	632	18	7008	510	214281.4	716	21	7896	603	269756.8
0.8	761	20	8426	582	196941.1	807	21	8920	618	269756.8
0.9	794	20	8819	592	196941.1	891	24	9867	682	240483.4
1.0	820	20	9136	597	196941.1	999	29	11045	777	239930.6
				TOL = 10^{-4}						
0	0	0	11	11	433026.3	0	0	11	11	433627.5
0.1	190	0	1913	53	386532.7	191	0	1923	53	388576.4
0.2	343	0	3513	121	328585.6	355	0	3648	136	356848.8
0.4	589	0	5982	237	250047.3	624	0	6277	220	308949.4
0.6	797	0	8118	292	230593.5	860	0	8675	301	262583.2
0.8	927	0	9315	322	230593.5	980	0	9863	326	262583.2
0.9	958	0	9693	327	230593.5	1016	0	10300	331	262583.2
1.0	989	0	10070	332	230593.5	1051	0	10730	341	262583.2
				TOL= 10^{-5}						
0	0	0	11	11	433026.3	0	0	11	11	433627.5
0.1	238	0	2137	63	388966.2	240	0	2156	64	383643.7
0.2	426	0	3909	141	328075.1	442	0	4062	150	353515.8
0.4	740	0	6866	269	249632.2	797	0	7362	253	301792.6
0.6	997	0	9245	334	233809.7	1093	0	10118	343	257848.1
0.8	1246	0	11506	384	233809.7	1362	0	12509	398	257848.1
0.9	1333	0	12201	404	233809.7	1447	0	13225	413	257848.1
1.0	1385	0	12728	414	233809.7	1499	0	13783	423	257848.1

Because of this, and because of the minimizing property of the degree, we again conclude that it may be preferable to choose TOL not too large. This conclusion particularly applies when integrating strongly nonlinear problems. We conclude this section by observing that all integrations were performed without the necessity of making restarts.

6. CONCLUDING REMARKS

The purpose of this paper was to develop a robust and efficient time integrator that is easy to apply to a wide class of linear, and in particular nonlinear, semidiscretized parabolic problems in one or more dimensions. Although it is not clear that, e.g., for a two-dimensional nonlinear problem, our method needs less computer time than an unconditionally stable method (such as an implicit method or an ADI method), the easy applicability of the subroutine, when used in conjunction with semidiscretization, reduces the human effort needed to solve a problem under consideration. For example, the subroutine is easy to use with a software interface performing the semidiscretization (see, e.g., Sincovec and Madsen [11] or Bakker [1]).

It is of interest to observe that until now stabilized explicit methods, as well as their implementations, have not received very much attention in the literature. As a consequence, it is most likely that the present implementation and its underlying algorithm can be improved significantly.

ACKNOWLEDGMENTS

The author wishes to thank Prof. P.J. van der Houwen and Prof. T.J. Dekker of the University of Amsterdam for their constructive comments and careful reading of the manuscript. The author is also grateful to M. Bakker for his assistance in programming the Galerkin discretizations, and to B.P. Sommeijer for his assistance in testing the program.

REFERENCES

1. Bakker, M. Software for semi-discretization of time dependent partial differential equations in one space variable. Rep. NW 52/77, Mathematisch Centrum, Amsterdam, 1977.
2. Bakker, M. Unpublished manuscript. Mathematisch Centrum, Amsterdam.
3. Beentjes, P.A. NUMAL, A library of ALGOL 60 procedures in numerical mathematics. Mathematisch Centrum, Amsterdam, 1974, sec. 5.2.1.1.1.1., procedure ARK.
4. Douglas, J., and Dupont, T. Galerkin methods for parabolic equations. *SIAM J. Numer. Anal. 7* (1970), 575–626.
5. Gear, C.W. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, N.J., 1971.
6. Lindberg, B. IMPEX—A program package for solution of systems of stiff differential equations. Rep. NA 72.50, Royal Inst. Technology, Stockholm, 1972.
7. Polak, S.J. Private communication. ISA, Gebouw VM, Philips, Eindhoven, The Netherlands.
8. Richtmeyer, R.D., and Morton, K.W. *Difference Methods for Initial Value Problems.* Interscience, New York, 1967.
9. Ryder, B. G. The PFORT verifier. *Softw. Prac. Exper. 4* (1974), 359–378.
10. Shampine, L., and Gordon, M.K. *Computer solution of ordinary differential equations. The Initial Value Problem*, W.H. Freeman, San Francisco, 1975.
11. Sincovec, R.F., and Madsen, N.K. Software for nonlinear partial differential equations. *ACM Trans. Math. Softw. 1*, 3 (Sept. 1975), 232–260.
12. Te Riele, H.J.J., ed. Colloquium numerieke programmatuur (Dutch), MC syllabus 29.2, Mathematisch Centrum, Amsterdam, 1977
13. van der Houwen, P.J. *Construction of Integration Formulas for Initial Value Problems.* North-Holland, Amsterdam, 1976.
14. Verwer, J.G. A class of stabilized three-step Runge–Kutta methods for the numerical integration of parabolic equations. *J. Computat. Appl. Math. 3* (1977), 155–166.
15. Verwer, J.G. Algorithm 553. M3RK, An explicit time integrator for semidiscrete parabolic equations. *ACM Trans. Math. Softw. 6*, 2 (June 1980), 236–239.
16. Verwer, J.G. Multipoint multistep Runge–Kutta methods I: On a class of two-step methods for parabolic equations. Rep. NW30/76, Mathematisch Centrum, Amsterdam, 1976.
17. Verwer, J.G. Multipoint multistep Runge–Kutta methods II: The construction of a class of stabilized three-step methods for parabolic equations. Rep. NW31/76, Mathematisch Centrum, Amsterdam, 1976.

Received June 1977; revised July 1979; accepted December 1979

ALGORITHM 553

M3RK, An Explicit Time Integrator for Semidiscrete Parabolic Equations [D3]

J. G. VERWER
Mathematical Center, Amsterdam, The Netherlands

Key Words and Phrases: parabolic partial differential equations, semidiscretization, explicit time integrator
CR Categories: 5.17
Language: Fortran

DESCRIPTION

This algorithm is a complement to [2] where it is explained and described.

REFERENCES

1. RYDER, B.G. The PFORT verifier. *Softw. Pract. Exper. 4* (1974), 359–378.
2. VERWER, J.G. An implementation of a class of stabilized explicit methods for the time integration of parabolic equations. *ACM Trans. Math. Softw. 6*, 2 (June 1980), 188–205.

ALGORITHM

[Summary information and part of the listing are printed here. The complete listing is available from the ACM Algorithms Distribution Service (see inside back cover for order form) or may be found in "Collected Algorithms from the ACM."]

NAME(n): indicates a Fortran module with n records
NAME^T(n): indicates "NAME" is included for testing purposes
Contents: TEST1^T(94), DER^T(34), TEST2^T(135), EVAL^T(60), DER^T(42), UPRINT^T(15), M3RK(411), HSTART(26), PARAM(257), POWERM(91), MAXDEG(28), MINDEG(20), STEP(45), ESTIMA(31), NEWH(24), INTER1(30), INTER2(16), SHIFT(17)

Received 22 June 1977; revised 25 July 1979; accepted 11 December 1979.

Author's address: Mathematisch Centrum, 2e Boerhaavestraat 49, 1091 AL Amsterdam, The Netherlands.

```
      SUBROUTINE M3RK(X,XE,N,H,HMIN,SIGMA,TOL,F,Y,                          1Ø
     +                Y1,Y2,YXE,DY,DY1,IFLAG,INFO)                          2Ø
C*********************************************************************    3Ø
C* ABSTRACT                                                          *    4Ø
C*********************************************************************    5Ø
C* M3RK IS DESIGNED TO SOLVE INITIAL VALUE PROBLEMS FOR SYSTEMS OF   *    6Ø
C* ORDINARY DIFFERENTIAL EQUATIONS OF THE FORM                       *    7Ø
C*                  DY/DX=F(Y(1),...,Y(N)),                          *    8Ø
C*                  Y(I) GIVEN AT X,                                 *    9Ø
C* WHICH ORIGINATE FROM SEMI-DISCRETIZATION OF INITIAL-BOUNDARY VALUE *   1ØØ
C* PROBLEMS FOR PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS. M3RK IS     *   11Ø
C* BASED ON STABILIZED,EXPLICIT THREE-STEP RUNGE-KUTTA FORMULAS OF    *   12Ø
C* ORDER ONE AND TWO,OF WHICH THE DEGREE CAN VARY BETWEEN 2 AND 12.   *   13Ø
C*                                                                   *   14Ø
C* M3RK NEEDS 6 ARRAYS OF LENGTH N,WHICH ALL APPEAR IN THE CALL LIST. *   15Ø
C* THE CODE INTEGRATES FROM X TO XE.ON NORMAL RETURN THE PARAMETERS IN *  16Ø
C* THE CALL LIST ARE READY FOR CONTINUING THE INTEGRATION.TO CONTINUE *   17Ø
C* THE INTEGRATION,THE USER NEEDS ONLY TO REDEFINE THE OUTPUT POINT XE *  18Ø
C* AND CALL AGAIN.                                                   *   19Ø
C*                                                                   *   2ØØ
C* M3RK CALLS 11 SUBROUTINES WHICH HAVE BEEN WRITTEN TO STRUCTURE THE *  21Ø
C* PROGRAM.THESE SUBROUTINES ARE:                                    *   22Ø
C* HSTART - HSTART COMPUTES THE INITIAL STEPLENGTH                   *   23Ø
C* PARAM  - PARAM COMPUTES PARAMETERS OF THE VARIOUS IMPLEMENTED     *   24Ø
C*          SCHEMES FROM THE COEFFICIENTS OF STABILITY POLYNOMIALS   *   25Ø
C* POWERM - POWERM ESTIMATES THE SPECTRAL RADIUS OF THE JACOBIAN OF F *  26Ø
C* MAXDEG - MAXDEG COMPUTES THE MAXIMAL DEGREE OF THE FORMULAS,      *   27Ø
C*          WHICH IS ALLOWED WITH RESPECT TO INTERNAL STABILITY      *   28Ø
C* MINDEG - MINDEG COMPUTES THE MINIMAL DEGREE OF THE FORMULAS,      *   29Ø
C*          WHICH IS ALLOWED WITH RESPECT TO ABSOLUTE STABILITY      *   3ØØ
C* STEP   - STEP CONTAINS THE ACTUAL INTEGRATOR                      *   31Ø
C* ESTIMA - ESTIMA COMPUTES A LOCAL ERROR BOUND AND ESTIMATES A LOCAL *   32Ø
C*          ERROR                                                    *   33Ø
C* NEWH   - NEWH DELIVERS A NEW STEPLENGTH                           *   34Ø
C* INTER1 - INTER1 PERFORMS INTERPOLATION AFTER A CHANGE OF THE      *   35Ø
C*          STEPLENGTH                                               *   36Ø
C* INTER2 - INTER2 INTERPOLATES THE SOLUTION AT THE OUTPUT POINT XE  *   37Ø
C* SHIFT  - SHIFT SHIFTS THE DATA FOR A NEXT STEP                    *   38Ø
C*                                                                   *   39Ø
C* THESE SUBROUTINES ARE COMPLETELY LOCAL,I.E.THE INFORMATION THEY    *  4ØØ
C* NEED IS PASSED THROUGH THE PARAMETER LISTS.THE WHOLE PACKAGE HAS   *  41Ø
C* BEEN TESTED ON A CDC CYBER 73-28 USING AN ARITHMETIC PRECISION OF  *  42Ø
C* 14 DIGITS.THE CODE POWERM USES THE CDC SYSTEM SUBPROGRAMS RANSET   *  43Ø
C* AND RANF CONSTITUTING A RANDOM GENERATOR.RANSET AND RANF MUST BE   *  44Ø
C* REPLACED WHEN USING THE PROGRAM ON ANOTHER COMPUTER.              *   45Ø
C* THE CODES CALLED BY M3RK USE NO MACHINE DEPENDENT CONSTANTS.      *   46Ø
C* M3RK USES ONE MACHINE DEPENDENT CONSTANT,NAMELY THE ARITHMETIC    *   47Ø
C* PRECISION OF THE COMPUTER.IN THE PROGRAM THE INTERNAL VARIABLE APR, *  48Ø
C* WHICH REPRESENTS THE ARITHMETIC PRECISION,EQUALS 1.ØE-14. APR MUST *  49Ø
C* BE CHANGED ACCORDINGLY WHEN USING THE PROGRAM ON ANOTHER COMPUTER. *   5ØØ
C*                                                                   *   51Ø
C* THE WHOLE PROGRAM PACKAGE IS ACCEPTED BY THE PFORT VERIFIER.THE    *  52Ø
C* PFORT VERIFIER IS A PROGRAM WHICH CHECKS A FORTRAN PROGRAM,I.E. A  *   53Ø
C* MAIN PROGRAM AND SUBPROGRAMS,FOR ADHERENCE TO PFORT,A PORTABLE     *   54Ø
C* SUBSET OF AMERICAN NATIONAL STANDARD FORTRAN(SEEI1 ).THE WHOLE     *   55Ø
C* PROGRAM PACKAGE IS COMPLETELY EXPLAINED AND DESCRIBED IN I2 .      *   56Ø
C*********************************************************************    57Ø
C* MEANING OF THE PARAMETERS                                         *   58Ø
C*********************************************************************    59Ø
C* X      - VARIABLE    : INDEPENDENT VARIABLE                       *   6ØØ
C* XE     - EXPRESSION  : OUTPUT POINT AT WHICH SOLUTION IS DESIRED  *   61Ø
C* N      - EXPRESSION  : NUMBER OF EQUATIONS                        *   62Ø
```

```
C* H      - VARIABLE    : STEPLENGTH                                          *   63Ø
C* HMIN   - VARIABLE    : MINIMAL STEPLENGTH                                  *   64Ø
C* SIGMA  - ARRAY       : AN ARRAY OF LENGTH 2 CONTAINING ESTIMATES           *   65Ø
C*                        OF THE SPECTRAL RADIUS OF THE JACOBIAN OF F         *   66Ø
C* TOL    - EXPRESSION  : LOCAL ERROR TOLERANCE                               *   67Ø
C* F      - SUBROUTINE  : DERIVATIVE                                          *   68Ø
C* Y      - ARRAY       : SOLUTION VECTOR AT X,INPUT AND WORK ARRAY           *   69Ø
C* Y1     - ARRAY       : SOLUTION VECTOR AT X-H,WORK ARRAY                   *   7ØØ
C* Y2     - ARRAY       : SOLUTION VECTOR AT X-2H,WORK ARRAY                  *   71Ø
C* YXE    - ARRAY       : SOLUTION VECTOR AT XE,OUTPUT AND WORK ARRAY         *   72Ø
C* DY     - ARRAY       : DERIVATIVE VECTOR AT X,WORK ARRAY                   *   73Ø
C* DY1    - ARRAY       : DERIVATIVE VECTOR AT X-H,WORK ARRAY                 *   74Ø
C* IFLAG  - VARIABLE    : ERROR FLAG                                          *   75Ø
C* INFO   - ARRAY       : INTEGER ARRAY OF LENGTH 15.INFO IS USED TO          *   76Ø
C*                        PASS INFORMATION TO INITIALIZE THE CODE,TO          *   77Ø
C*                        PASS INFORMATION BETWEEN THE MAIN PROGRAM AND       *   78Ø
C*                        THE SUBPROGRAMS,TO PASS INFORMATION TO THE          *   79Ø
C*                        USER ABOUT THE STATUS OF THE INTEGRATION,AND        *   8ØØ
C*                        TO RETAIN INFORMATION FOR SUBSEQUENT CALLS.         *   81Ø
C*************************************************************************   82Ø
C* FIRST CALL TO M3RK                                                        *   83Ø
C*************************************************************************   84Ø
C* THE USER MUST PROVIDE STORAGE IN HIS CALLING PROGRAM FOR THE ARRAYS       *   85Ø
C* IN THE CALL LIST.HE HAS TO SUPPLY THE SUBROUTINE F(N,Y) FOR               *   86Ø
C* EVALUATING THE DERIVATIVES DY(I)/DT,I=1,...,N,WHICH MUST BE OVER-         *   87Ø
C* WRITTEN ON Y(I).FOR THE SPECTRAL RADIUS THERE EXIST THREE OPTIONS         *   88Ø
C* WHICH MUST BE SELECTED WITH INFO(2).IN INFO(3) THE USER MUST GIVE         *   89Ø
C* A MAXIMUM FOR THE NUMBER OF EVALUATIONS OF F(Y) TO BE SPENT(ON RE-        *   9ØØ
C* TURN IT MAY OCCUR THAT INFO(3) IS EXCEEDED WITH ABOUT 5Ø)                 *   91Ø
C************************                                                   *   92Ø
C* INPUT PARAMETERS                                                        *   93Ø
C************************                                                   *   94Ø
C* X      - INITIAL VALUE OF THE INDEPENDENT VARIABLE                      *   95Ø
C* XE     - OUTPUT POINT AT WHICH SOLUTION IS DESIRED                      *   96Ø
C* N      - NUMBER OF EQUATIONS                                            *   97Ø
C* SIGMA  - (1)= AN UPPER ESTIMATION OF THE SPECTRAL RADIUS OF THE JA-     *   98Ø
C*               COBIAN OF F IN CASE OF OPTION 1.FOR OPTION 2 AND 3 NO     *   99Ø
C*               INITIALIZATION IS REQUIRED                                *  1ØØØ
C* TOL    - LOCAL ERROR TOLERANCE                                          *  1Ø1Ø
C* Y      - VECTOR OF INITIAL VALUES OF THE DEPENDENT VARIABLES            *  1Ø2Ø
C* INFO   - (1)= Ø TO INDICATE FIRST CALL                                  *  1Ø3Ø
C*          (2)= 1 TO INDICATE OPTION 1 FOR THE SPECTRAL RADIUS,I.E.       *  1Ø4Ø
C*               THE USER MUST INITIALIZE SIGMA(1)                         *  1Ø5Ø
C*             = 2 TO INDICATE OPTION 2 FOR THE SPECTRAL RADIUS,I.E.       *  1Ø6Ø
C*               THE CODE INITIALIZES SIGMA(1) AND CONTROLS SIGMA(1)       *  1Ø7Ø
C*             = 3 TO INDICATE OPTION 3 FOR THE SPECTRAL RADIUS,I.E.       *  1Ø8Ø
C*               THE CODE ONLY INITIALIZES SIGMA(1) AT THE FIRST CALL      *  1Ø9Ø
C*          (3)= MAXIMUM NUMBER OF F(Y) EVALUATIONS TO BE SPENT            *  11ØØ
C************************                                                   *  111Ø
C* OUTPUT PARAMETERS                                                       *  112Ø
C************************                                                   *  113Ø
C* X      - LAST POINT REACHED IN INTEGRATION.NORMALLY X IS SLIGHTLY       *  114Ø
C*          BEYOND XE                                                      *  115Ø
C* H      - INITIAL STEPLENGTH FOR SUBSEQUENT CALL                         *  116Ø
C* HMIN   - MINIMAL STEPLENGTH USED BY M3RK                                *  117Ø
C* SIGMA  - (1)= UPPER ESTIMATION OF THE SPECTRAL RADIUS INITIALIZED BY    *  118Ø
C*               THE USER OR BY POWERM                                     *  119Ø
C*        - (2)= INACCURATE ESTIMATION OF THE SPECTRAL RADIUS USED FOR     *  12ØØ
C*               ITS CONTROL                                               *  121Ø
C* Y      - SOLUTION VECTOR AT X                                           *  122Ø
C* Y1     - SOLUTION VECTOR AT X-H                                         *  123Ø
C* Y2     - SOLUTION VECTOR AT X-2H                                        *  124Ø
```

```
C* YXE    - SOLUTION VECTOR AT OUTPUT POINT XE                              *  125Ø
C* DY     - DERIVATIVE VECTOR AT X                                          *  126Ø
C* DY1    - DERIVATIVE VECTOR AT X-H                                        *  127Ø
C* IFLAG - = Ø NORMAL RETURN,I.E.OUTPUT POINT IS REACHED                    *  128Ø
C*          = 1 OUTPUT POINT IS NOT REACHED.THE MAXIMUM NUMBER OF           *  129Ø
C*              F(Y)-EVALUATIONS HAS BEEN SPENT.THE PROCESS CAN BE CON-     *  13ØØ
C*              TINUED BY INCREASING INFO(3) AND CALLING AGAIN              *  131Ø
C*          = 2 MAXIMAL DEGREE FALLS OUTSIDE THE RANGE AS TOL/APR IS        *  132Ø
C*              TOO SMALL.THE PROCESS IS NOT STARTED                        *  133Ø
C*          = 3 POWERM FAILED IN THE ESTIMATION OF THE SPECTRAL RADIUS.     *  134Ø
C*              THE PROCESS IS DISCONTINUED                                 *  135Ø
C* INFO   - (1) = 1 TO INDICATE THAT THE NEXT CALL IS A SUBSEQUENT ONE      *  136Ø
C*          (2) = 1 IN CASE OF OPTION 1 OR 3,ELSE 2                         *  137Ø
C*          (3) = UNCHANGED                                                 *  138Ø
C*          (4) = TOTAL NUMBER OF INTEGRATION STEPS PERFORMED,I.E.          *  139Ø
C*                ACCEPTED AND REJECTED ONES                                *  14ØØ
C*          (5) = NUMBER OF REJECTED INTEGRATION STEPS                      *  141Ø
C*          (6) = NUMBER OF RESTARTS INITIATED BY THE CODE                  *  142Ø
C*          (7) = TOTAL NUMBER OF DERIVATIVE EVALUATIONS                    *  143Ø
C*          (8) = NUMBER OF DERIVATIVE EVALUATIONS USED FOR THE ESTIMA-     *  144Ø
C*                TION AND CONTROL OF THE SPECTRAL RADIUS                   *  145Ø
C*          (9) = CURRENT DEGREE                                            *  146Ø
C*          (1Ø)= MAXIMAL DEGREE FOR THE FIRST ORDER FORMULAS               *  147Ø
C*          (11)= MAXIMAL DEGREE FOR THE SECOND ORDER FORMULAS              *  148Ø
C*          (12)= CURRENT ORDER                                             *  149Ø
C*          (13)= NUMBER OF STEPS PERFORMED AFTER START OR RESTART          *  15ØØ
C*          (14)= NUMBER OF STEPS PERFORMED AFTER CHANGE OF H OR ORDER      *  151Ø
C*          (15)= NUMBER OF STEPS PERFORMED AFTER ESTIMATION OF SPEC-       *  152Ø
C*                TRAL RADIUS                                               *  153Ø
C*                                                                          *  154Ø
C* IT IS EMPHASIZED THAT THE OUTPUT LIST GIVEN ABOVE IS NOT STRICTLY        *  155Ø
C* VALID WHEN ON RETURN IFLAG IS EQUAL TO 2 OR 3.                           *  156Ø
C* IT IS FURTHER EMPHASIZED THAT ON RETURN THE ARRAY Y ALWAYS CONTAINS      *  157Ø
C* THE SOLUTION VECTOR AT THE POINT X.THUS WHEN IFLAG=3,THE USER HAS        *  158Ø
C* THE POSSIBILITY TO RESTART THE PROCESS AT THE POINT X,PROVIDED HE        *  159Ø
C* IS ABLE TO TAKE ACTION WITH RESPECT TO THE ESTIMATION OF THE SPEC-       *  16ØØ
C* TRAL RADIUS.                                                            *  161Ø
C************************************************************************  162Ø
C* SUBSEQUENT CALLS TO M3RK                                                *  163Ø
C************************************************************************  164Ø
C* ON RETURN OF M3RK ALL PARAMETERS ARE READY FOR CONTINUING THE INTE-      *  165Ø
C* GRATION,PROVIDED IFLAG IS NOT EQUAL TO 2 OR 3.IF XE IS REACHED AND      *  166Ø
C* A NORMAL CONTINUATION IS DESIRED,THE USER NEEDS ONLY TO DEFINE A NEW*    167Ø
C* OUTPUT POINT XE AND CALL AGAIN.IF ON RETURN IFLAG=1 AND THE USER         *  168Ø
C* WANTS TO CONTINUE,HE ONLY NEEDS TO INCREASE INFO(3) AND CALL AGAIN.     *  169Ø
C* THE PROGRAM IS WRITTEN IN SUCH A WAY THAT THE CHOICE OF OUTPUT          *  17ØØ
C* POINTS DOES NOT AFFECT THE INTEGRATION PROCESS ITSELF.BETWEEN SUB-      *  171Ø
C* SEQUENT CALLS THE USER MAY INCREASE TOL.ALL OTHER PARAMETERS MUST       *  172Ø
C* REMAIN UNCHANGED.THE COUNTERS IN INFO ARE USED ACCUMULATIVELY.          *  173Ø
```