stichting
mathematisch
centrum

MC

AFDELING NUMERIEKE WISKUNDE NW 12/74 SEPTEMBER

J.G. VERWER

GENERALIZED LINEAR MULTISTEP METHODS II
NUMERICAL APPLICATIONS

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), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

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

GENERALIZED LINEAR MULTISTEP METHODS II
NUMERICAL APPLICATIONS

by

J.G. VERWER

ABSTRACT

With this report the author proposes a third order formula for the numerical integration of stiff systems of ordinary differential equations. This formula belongs to a special class of generalized linear multistep formulas, i.e. linear multistep formulas of which the scalar coefficients are replaced by operator coefficients. An ALGOL-60 implementation of our method is presented. This procedure supplies the additional starting values and performs stepsize control. Numerical results of the procedure applied on stiff equations are reported.

KEY WORDS AND PHRASES: *Numerical analysis, Ordinary differential equations, Initial value problems, Stiff equations.*

CONTENTS

1. INTRODUCTION

In VAN DER HOUWEN & VERWER [6] we have investigated the *generalized linear multistep method* which may be used to solve numerically initial value problems for systems of ordinary differential equations of the type

$$\frac{dy}{dx} = f(y).$$

This integration method originates from the classical linear multistep method by replacing the coefficients of the integration formula by *rational functions of the Jacobian matrix* J(y). In the report mentioned above special attention has been paid to a class of formulas of which the principal root (the stability function) can be adapted to the problem under consideration, while the parasitic roots are zero. Formulas of this class require an accurate evaluation of the Jacobian matrix to obtain an *adaptive principal root* and *zero-parasitic roots*. However, one may derive consistency conditions for this class of formulas, which do not require an accurate Jacobian. In this situation the stability function is no longer adaptive and the parasitic roots do no longer equal zero. This leads us to the class of formulas with *quasi-zero parasitic roots* and *quasi-adaptive stability function*. From a practical point of view formulas allowing a crude Jacobian may be preferred to formulas requiring a correct Jacobian. In this report we discuss formulas with quasi-zero parasitic roots. We have concentrated on the construction of a *three-step third order formula* which may be used for *efficient integration of stiff equations*. Other special classes of generalized linear multistep methods have been proposed by NØRSETT [12], VAN DER HOUWEN [5] and LAMBERT & SIGURDSSON [8]. In section 7 we present an ALGOL-60 implementation of our third order scheme. This procedure supplies the additional starting values and performs stepsize control. Computational results of this procedure when applied on stiff differential equations are presented in the last section of this report.

2. ALGORITHMS WITH QUASI-ZERO PARASITIC ROOTS

The generalized linear k-step method with *zero-parasitic roots* and *adaptive principal root* is defined by the formula (cf VAN DER HOUWEN & VERWER [6])

$$(2.1) \qquad y_{n+1} = Ry_n + h_n \sum_{\ell=1}^{k} B_\ell [f(y_{n+1-\ell}) - J_n y_{n+1-\ell}],$$

where R and B_ℓ are rational functions of $h_n J_n$, $J_n = J(y_n)$. The principal root is identified with the prescribed stability function R. For formula (2.1) we have derived two types of consistency conditions. The first type requires an exact evaluation of the Jacobian matrix J_n, while the second one allows an inaccurate evaluation of J_n. From a practical point of view, formulas allowing inaccurate Jacobian matrices may be preferred to formulas which strongly depend on a correct evaluation of J_n. Therefore we shall concentrate on the second type. In this case formula (2.1) is transformed into

$$(2.2) \qquad y_{n+1} = Ry_n + h_n \sum_{\ell=1}^{k} B_\ell [f(y_{n+1-\ell}) - J^* y_{n+1-\ell}],$$

where R and B_ℓ are *rational functions of* $h_n J^*$ and where J^* is some approximation to the Jacobian matrix $J(y_n)$. However, scheme (2.2) no longer has zero-parasitic roots and an adaptive stability function. For, when applied to the test equation

$$(2.3) \qquad y' = Jy,$$

J being the Jacobian matrix of the differential equation under consideration, scheme (2.2) is reduced to

$$(2.4) \qquad y_{n+1} = Ry_n + h_n \sum_{\ell=1}^{k} B_\ell (J - J^*) y_{n+1-\ell}.$$

This k-step scheme is reduced to the one-step scheme

$$(2.5) \qquad y_{n+1} = Ry_n,$$

provided that

(2.6) $J^* = J.$

This means that the stability function of scheme (2.2) is a function of the form

(2.7) $R^*(h_nJ,h_nJ^*),$

with the property

(2.8) $R^*(h_nJ,h_nJ) \equiv R(h_nJ).$

In the sequel we shall call formulas of class (2.2) formulas with *quasi-zero parasitic roots* and *quasi-adaptive stability function*. Theoretical aspects concerning the stability behaviour of these schemes will be discussed in a subsequent paper. In this report we shall state a first result as well. To perform a stable integration it is necessary to keep the spectral radius of

(2.9) $B_\ell(hJ^*),\ \ell=1,\ldots,k,$

as small as possible. Therefore, we shall state the following stabilizing conditions for the functions B_ℓ:

(2.10) $B_\ell(z) \to 0$ as $Re(z) \to -\infty.$

At the end of this section we shall state a desirable property of the operators B_ℓ with respect to the adaptivity. When integrating with scheme (2.2) a differential equation which almost behaves like a linear equation

(2.11) $y' = Jy + K,$

we should exploit the adaptivity of our scheme for linear equations. The

local analytical solution of (2.11) is given by

$$y(x_{n+1}) = e^{h_n J} y(x_n) + J^{-1}(e^{h_n J} - I)K, \tag{2.12}$$

and the numerical solution by

$$y_{n+1} = R(h_n J)y_n + h_n \sum_{\ell=1}^{k} B_\ell(h_n J)K, \tag{2.13}$$

provided $J^* = J$.

Therefore, we shall demand the following *adaptivity condition* for the operators B_ℓ:

$$\sum_{\ell=1}^{k} B_\ell(h_n J) = \frac{R(h_n J) - I}{h_n J}. \tag{2.14}$$

If relation (2.14) is satisfied, the nonhomogeneous part of solution (2.12) will be satisfactorily approximated provided that $J^* = J$ and that $R(h_n J)$ approximates $e^{h_n J}$ satisfactorily.

By putting J^* equal to zero formula (2.2) is reduced to

$$y_{n+1} = R(0)y_n + h_n \sum_{\ell=1}^{k} B_\ell(0)f(y_{n+1-\ell}), \tag{2.15}$$

i.e., a formula of the *classical Adams-Bashforth* type. This means that the generalized linear multistep method with quasi-zero parasitic roots may be considered as a stabilized modification of the Adams-Bashforth method. A modification of the Adams-Bashforth methods suggested by NØRSETT [12] turns out to be closely related to the generalized linear multistep methods with quasi-zero parasitic roots, provided that the exponential terms present in the NØRSETT methods are replaced by R. Formula (2.2) also bears a close resemblance to a class of formulas proposed by LAMBERT & SIGURDSSON [8].

3. CONSISTENCY CONDITIONS AND LOCAL TRUNCATION ERROR

We represent scheme (2.2) by the operator equation

$$(3.1)\qquad L[y_n;h_n,J^*] = y_{n+1} - Ry_n - h_n \sum_{\ell=1}^{k} B_\ell[f(y_{n+1-\ell})-J^*y_{n+1-\ell}]$$

and define the numbers

$$(3.2)\qquad q_\ell = \frac{x_{n-\ell}-x_n}{h_n}\ ,\ \ell=0,1,\ldots,k-1.$$

Furthermore, we introduce the functions (cf VAN DER HOUWEN & VERWER [6]).

$$C_0(z) = R(z) - \sum_{\ell=1}^{k} zB_\ell(z),$$

$$(3.3)\qquad C_j(z) = \frac{1}{j!}\sum_{\ell=1}^{k} (j-zq_{\ell-1})q_{\ell-1}^{j-1}B_\ell(z),\ j=1,2,\ldots,$$

and the abbreviations

$$(3.4)\qquad c_j^{(i)} = \frac{d^i}{dz^i} C_j(z)\Big|_{z=0}.$$

By substituting a solution y of the differential equation into (3.1) and by expanding the right member of (3.1) in powers of h_n, we may formally derive *the consistency conditions for p-th order accuracy*, i.e.

$$c_j^{(0)} = \frac{1}{j!},\ j=0,1,\ldots,p,$$

$$(3.5)\qquad c_i^{(j-i)} = 0,\ j=1,\ldots,p;\ i=0,\ldots,j-1,$$

or

$$(3.6)\qquad C_j(z) = \frac{1}{j!} + O(z^{p+1-j}),\ z\to 0,\ j=0,\ldots,p,$$

where $z = h_nJ^*$. Recall that the crudeness of J^* does not influence the order of accuracy. We have proved that the maximal attainable order of scheme (2.2) is $p = k$, provided that R is consistent of order k, i.e.

$$(3.7)\qquad \frac{d^i}{dz^i} R(z)\Big|_{z=0} = 1,\ i=0,\ldots,k.$$

In the sequel we shall assume that the order of accuracy p = k. The consistency conditions (3.6) may now be written in the form

$$(3.8)\qquad \sum_{\ell=1}^{k} q_{\ell-1}^{j-1} B_\ell(z) = D_j(z),\ j=1,\ldots,k,$$

where the functions D_j are defined by

$$D_1(z) = \frac{R(z)-1 + \varepsilon_{k+1}(z)}{z},$$

$$(3.9)\qquad D_{j+1}(z) = \frac{jD_j(z)-1 - \varepsilon_{k+1-j}(z)}{z},\ j=1,\ldots,k-1,$$

$$\varepsilon_j(z) = O(z^j),\ z \to 0,\ j=2,\ldots,k+1.$$

The functions ε_j are related with the order terms in (3.6). Note that the (k+1)-st equation of (3.6) does not occur in the form (3.8). This equation will always be satisfied when the above conditions are fulfilled. Next we give the local truncation error. Recalling that p = k and substituting a solution y of the differential equation into (3.1) delivers the local truncation error at the point $x = x_n$, i.e.

$$(3.10)\qquad L[y(x_n);h_n,J^*] = \sum_{j=1} \Bigg[\left(\frac{1}{(k+j)!} - c_{k+j}^{(0)}\right) \frac{d^{k+j}}{dx^{k+j}} y(x)\Big|_{x=x_n} - \sum_{i=0}^{k+j-1} \frac{c_i^{(k-i+j)}}{(k-i+j)!} J^{*k-i+j} \frac{d^i}{dx^i} y(x)\Big|_{x=x_n} \Bigg] h_n^{k+j}.$$

Note that by putting J^* equal to zero, (3.10) is reduced to the local truncation error of the classical Adams-Bashforth formula. For future reference we write out the principal local truncation error for k = 3, i.e.

$$(3.11)\qquad \Big[\Big(\frac{1}{4!}-c_4^{(0)}\Big)\frac{d^4}{dx^4}\,y(x)\Big|_{x=x_n} - \frac{c_0^{(4)}}{4!}\,J^{*4}y(x_n) - \frac{c_1^{(3)}}{3!}\,J^{*3}\,\frac{d}{dx}\,y(x)\Big|_{x=x_n}$$

$$- \frac{c_2^{(2)}}{2!}\,J^{*2}\,\frac{d^2}{dx^2}\,y(x)\Big|_{x=x_n} - c_3^{(1)}J^*\,\frac{d^3}{dx^3}\,y(x)\Big|_{x=x_n}\Big]\,h_n^4.$$

As we have noted before, the maximal attainable order of scheme (2.2) is $p = k$. In general an accurate choice of J^*, i.e.

$$(3.12)\qquad J^* = J_n,$$

will not increase the order. An exception to this rule is made by the one-step scheme

$$(3.13)\qquad y_{n+1} = Ry_n + h_nB_1[f(y_n)-J^*y_n],$$

provided that the operator R is of order greater than one and

$$(3.14)\qquad B_1(z) = D_1(z) = \frac{R(z)-1}{z}\,.$$

In this case the one-step scheme takes the form

$$(3.15)\qquad y_{n+1} = y_n + J^{*-1}(R(h_nJ^*)-I)f(y_n).$$

For non-linear equations the order of (3.15) is $p = 2$, provided that

$$J^* = J_n,$$

whereas for linear equations the order p equals the order of the operator R. These properties may be useful to supply one or more starting values for a k-step formula.

4. OPERATORS WHICH MINIMIZE THE LOCAL TRUNCATION ERROR

The functions ε_j occurring in the consistency conditions (3.9) may be chosen freely provided that $\varepsilon_j(z) = 0(z^j)$, $z \to 0$. In VAN DER HOUWEN & VERWER [6] we have used this freedom to construct a set of integration formulas of which the order may be varied without much computational effort. Numerical results of these formulas turned out to be unsatisfactory. This was caused by the rather big truncation error (compare (3.11)) and by the fact that the stabilizing and adaptivity condition, mentioned in section 2, were not fulfilled. Therefore, it seems advisable to minimize (in some sense) the local truncation error and to match the operators B_ℓ. In this section we shall minimize the local truncation error. To this end we put

$$\varepsilon_j(z) \equiv 0, \quad j=2,\ldots,k+1. \tag{4.1}$$

It is easy to see that by (4.1) conditions (3.6) are reduced to (note $p = k$)

$$\begin{aligned} C_j(z) &= \frac{1}{j!}, \quad j=0,\ldots,k-1, \\ C_k(z) &= \frac{1}{k!} + 0(z). \end{aligned} \tag{4.2}$$

Consequently we have

$$\begin{aligned} c_j^{(i)} &= 0, \quad j=0,\ldots,k-1; \quad i=1,2,\ldots, \\ c_k^{(i)} &= 0, \quad i=2,3,\ldots . \end{aligned} \tag{4.3}$$

The local truncation error (3.10) is now reduced to

$$L[y(x_n);h_n,J^*] = \sum_{j=1} \left[\left(\frac{1}{(k+j)!} - c_{k+j}^{(0)}\right) \frac{d^{k+j}}{dx^{k+j}} y(x)\Big|_{x=x_n} - \sum_{i=1}^{j} c_{k+j-i}^{(i)} J^{*i} \frac{d^{k+j-i}}{dx^{k+j-i}} y(x_n)\Big|_{x=x_n} \right] h_n^{k+j}, \tag{4.4}$$

and the principal local truncation error for k = 3 is reduced to

$$(4.5) \qquad \left[\left(\frac{1}{4!} - c_4^{(0)}\right) \frac{d^4}{dx^4} y(x)\Big|_{x=x_n} - c_3^{(1)} J^* \frac{d^3}{dx^3} y(x)\Big|_{x=x_n} \right] h_n^4 .$$

The functions D_j are given by

$$(4.6) \qquad \begin{aligned} D_1(z) &= \frac{R(z)-1}{z} , \\ D_{j+1}(z) &= \frac{jD_j(z)-1}{z} , \quad j=1,\ldots,k-1. \end{aligned}$$

At the end of this section we observe that by the choice

$$\varepsilon_j(z) \equiv 0, \quad j=2,\ldots,k+1,$$

the stabilizing condition (2.10) is automatically satisfied, provided that

$$R(z) \to 0 \text{ as } \mathrm{Re}(z) \to -\infty.$$

The choice $\varepsilon_{k+1}(z) \equiv 0$ is sufficient for the adaptivity condition (2.14). Moreover, by this choice we are able to use the second order starting mechanism mentioned in section 3.

5. AN EFFICIENT FORMULA FOR THE INTEGRATION OF STIFF EQUATIONS

In this section we shall construct a three-step third order scheme with quasi-zero parasitic roots and minimized local truncation error, which will be used for the integration of stiff differential equations. We require the scheme to be A-stable when $J^* = J(y_n)$. Therefore, for R we have to choose an A-stable stability function. We select the stability function of the $F^{(3)}$ formula of LINIGER & WILLOUGHBY [11], i.e.

$$(5.1) \qquad R(z) = \frac{1 + \frac{1}{2}(1-\alpha)z + \frac{1}{12}(1-3\alpha)z^2}{1 - \frac{1}{2}(1+\alpha)z + \frac{1}{12}(1+3\alpha)z^2} .$$

The free parameter α may be used to fit R exponentially at a real number $z_0 \leq 0$. Then, for α we have

$$\alpha = \frac{1}{3z_0} \frac{e^{z_0}(z_0^2-6z_0+12) - (z_0^2+6z_0+12)}{e^{z_0}(2-z_0) - (2+z_0)}, \quad z_0 \leq 0. \tag{5.2}$$

These values for α are lying in the interval $0 \leq \alpha \leq \frac{1}{3}$. If $\alpha \neq 0$ the order of consistency of R equals three, otherwise four. We note that exponential fitting of R is analogous to exponential fitting in the sense of Liniger and Willoughby only if $J^* = J_n$. To find the operators B_ℓ we have to solve the linear system (3.8) for k = 3, where the D_j are defined by relations (4.6). By denoting the denominator of R with Q we find the following expressions for the functions D_j:

$$D_j(z) = \frac{d_{1j}+d_{2j}z}{Q(z)}, \quad j=1,2,3, \tag{5.3}$$

where

$$d_{1j} = 1/j, \; d_{21} = -\alpha/2,$$

$$d_{22} = d_{23} = -(1+3\alpha)/12.$$

Here we may observe that the rational functions occurring in our scheme, all have the same denominator. From a practical point of view such functions may be preferred to functions which do not share this feature. Solving the linear system (3.8) leads to the following expressions for the functions B_ℓ:

$$B_\ell = \sum_{j=1}^{3} b_{\ell j} D_j, \quad \ell=1,2,3, \tag{5.4}$$

where

$$b_{11} = 1,\ b_{12} = \frac{-(q_1+q_2)}{q_1q_2},\ b_{13} = \frac{1}{q_1q_2},$$

$$b_{21} = 0,\ b_{22} = \frac{-q_2}{q_1^2-q_1q_2},\ b_{23} = \frac{1}{q_1^2-q_1q_2},$$

$$b_{31} = 0,\ b_{32} = \frac{-q_1}{q_2^2-q_1q_2},\ b_{33} = \frac{1}{q_2^2-q_1q_2}.$$

By use of the expressions (5.1), (5.3) and (5.4) the three-step third order scheme of the class (2.2), with minimized local truncation error, may be written in the form

$$\begin{aligned} Q(h_nJ^*)y_{n+1} = y_n &+ \sum_{\ell=1}^{k}\sum_{j=1}^{k} h_n b_{\ell j} d_{1j} f(y_{n+1-\ell}) + \\ &+ hJ^*\left[\sum_{\ell=1}^{k}\sum_{j=1}^{k}(h_n b_{\ell j} d_{2j} f(y_{n+1-\ell}) - b_{\ell j} d_{1j} y_{n+1-\ell}) + \frac{1-\alpha}{2} y_n\right] + \\ &+ h^2J^{*2}\left[\sum_{\ell=1}^{k}\sum_{j=1}^{k}(-b_{\ell j} d_{2j} y_{n+1-\ell}) + \frac{1-3\alpha}{12} y_n\right], \end{aligned} \tag{5.5}$$

where $k = 3$.

We emphasize once more that for linear equations

$$y' = Jy + K,$$

scheme (5.5) is reduced to (for all relevant k)

$$y_{n+1} = Ry_n + J^{-1}(R-I)K,$$

provided that $J^* = J$. The stability function R refers to (5.1).

In section 7 we shall present the procedure GMS (generalized multistep), an ALGOL-60 implementation of scheme (5.5). This procedure supplies the additional starting values and performs stepsize control. The starting and stepsize mechanism will be discussed in the next section. Numerical results of the procedure GMS are presented in section 8.

6. A STARTING AND STEPSIZE MECHANISM

Scheme (5.5) presents the advantage of finding the necessary additional starting values in a rather simple way. It is easy to see that only the coefficients $b_{\ell j}$ are dependent of the stepnumber k. Let us put k = 1 and suppose that at the start of the integration process $J^* = J(y_0)$. By these assumptions formula (5.5) is reduced to the second order one-step scheme (3.15) which provides the value y_1. Next we put k equal to two. The relevant coefficients $b_{\ell j}$ are

$$(6.1) \qquad b_{11} = 1,\ b_{12} = 1 - \frac{1}{q_1},\ b_{21} = 0,\ b_{22} = \frac{1}{q_1}.$$

This two-step scheme, which is again of second order, provides the value y_2. Consequently, formula (5.5) provides a starting algorithm which is easy to implement and which computes the additional starting values y_1 and y_2 in second order accuracy.

Next we discuss the step control policy used in GMS. As in most papers, the approach of this problem is somewhat heuristic. When integrating with scheme (5.5) a system of differential equations the crudeness of J^* may cause instability. Consequently, when performing stepsize control we should perform stability control as well as accuracy control. The accuracy control may call for a decrease, or permit an increase, in the stepsize h_n as the computation proceeds, whereas the stability control may call for a reevaluation of the Jacobian matrix. Up to now we did not tackle the problem of interpreting the crudeness of J^* with respect to stability properties of scheme (2.2). Therefore, the stepsize mechanism, as implemented in GMS, does not separate the stability and accuracy control. We have implemented a stepsize mechanism based on the discrepance of linearity (VAN DER HOUWEN

[7]). To a certain extent this type of step control may be considered as a mixture of stability control and accuracy control. In each integration we put k = 2 and compute, by means of the coefficients (6.1), the second order reference solution $\tilde{y}_{n+1}$. For linear equations we have

$$\tilde{y}_{n+1} = y_{n+1},$$

provided that $J^* = J$. So, for linear or almost linear equations our step control does not perform accuracy control. For non-linear equations the discrepance

$$\tilde{y}_{n+1} - y_{n+1}$$

may be considered as a measure of the accuracy of the numerical solution, but also as a measure of the non-linearity of the differential equation. From this point of view we may speak of some kind of stability control. The extra computational work per integration step to compute $\tilde{y}_{n+1}$ consists in:

1° adding some vectors,
2° carrying out two matrix-vector multiplications,
3° one forward-backward substitution.

Because of the second order accuracy of $\tilde{y}_{n+1}$ we have

$$\| y_{n+1} - \tilde{y}_{n+1} \| = c_n h_n^3.$$

Supposing $c_n \approx c_{n-1}$ leads to

$$(6.2) \qquad h_n = \sqrt[3]{\frac{tol}{c_n}} \approx \sqrt[3]{\frac{tol}{c_{n-1}}} = h_{n+1} \sqrt[3]{\frac{tol}{\| y_n - \tilde{y}_n \|}} \approx h_{n-1} \left(\frac{\frac{4}{3}\, tol}{tol + \| y_n - \tilde{y}_n \|} + \frac{1}{3} \right),$$

where tol is a predicted local tolerance. For tol we have chosen

$$\text{absolute tolerance} + \text{relative tolerance} * \| y_n \| .$$

With || || is meant the Euclidean norm. Thus, in each integration step we compute a new steplength h_n by means of (6.2). We decide to use this new steplength only if the decrease or increase in the steplength is more than ten percent. Furthermore, to exclude sudden instabilities we decide to re-evaluate the Jacobian matrix if:

1° the decrease in the steplength is more than ten percent, and a re-evaluation did not take place in the last integration step;

2° ten times the decrease in the steplength is less than ten percent.

When we decide to reevaluate because of point 2, the steplength will be adjusted too. As a result of changing the steplength or reevaluating the Jacobian matrix we have to update some arrays and perform a LU-decomposition.

At the end of this section we observe that it is possible to use GMS without stepsize control. Details are given in section 7.

7. THE PROCEDURE GMS

In this section we present a first version of the documentation and source text of the procedure GMS which will be inserted in NUMAL, a library of numerical procedures in ALGOL-60 (see HEMKER [4]).

```
AUTHOR: J.G. VERWER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740809.

BRIEF DESCRIPTION:

    GMS SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS AN AUTONOMOUS
    SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS DY/DX = F(Y), BY
    MEANS OF A THIRD ORDER GENERALIZED LINEAR MULTISTEP METHOD;
    IN PARTICULAR THIS PROCEDURE IS SUITABLE FOR THE INTEGRATION
    OF STIFF EQUATIONS.
```

```
KEYWORDS:

    DIFFERENTIAL EQUATIONS,
    INITIAL VALUE PROBLEM,
    AUTONOMOUS SYSTEM,
    STIFF EQUATIONS,
    GENERALIZED LINEAR MULTISTEP METHOD.


CALLING SEQUENCE:

    THE HEADING OF THE PROCEDURE READS:
    "PROCEDURE" GMS(X, XE, R, Y, H, HMIN, HMAX, DELTA, DERIVATIVE,
                    JACOBIAN, AETA, RETA, N, JEV, LU, NSJEV,
                    LINEAR, OUT);
    "VALUE" R;
    "REAL" X, XE, H, HMIN, HMAX, DELTA, AETA, RETA;
    "INTEGER" R, N, JEV, NSJEV, LU;
    "BOOLEAN" LINEAR;
    "ARRAY" Y;
    "PROCEDURE" DERIVATIVE, JACOBIAN, OUT;

    GMS INTEGRATES THE SYSTEM OF DIFFERENTIAL EQUATIONS DY/DX = F(Y)
    FROM X = X0 TO X = XE;

    THE MEANING OF THE FORMAL PARAMETERS IS:
    X:      <VARIABLE>;
            THE INDEPENDENT VARIABLE X;
            ENTRY: THE INITIAL VALUE OF X;
            EXIT : THE END VALUE OF X;
    XE:     <ARITHMETIC EXPRESSION>;
            ENTRY: THE END VALUE OF X;
    R:      <ARITHMETIC EXPRESSION>;
            ENTRY: THE NUMBER OF DIFFERENTIAL EQUATIONS;
    Y:      <ARRAY IDENTIFIER>;
            "ARRAY" Y[1:R];
            THE DEPENDENT VARIABLE;
            ENTRY: THE INITIAL VALUE OF Y;
            EXIT : THE SOLUTION Y AT THE POINT X AFTER EACH
                   INTEGRATION STEP;
    H:      <ARITHMETIC EXPRESSION>;
            ENTRY: THE STEPLENGTH WHEN THE INTEGRATION HAS TO BE
                   PERFORMED WITHOUT THE STEPSIZE MECHANISM, OTHER-
                   WISE THE INITIAL STEPLENGTH (SEE THE PARAMETERS
                   HNIN AND HMAX);
    HMIN, HMAX: <ARITHMETIC EXPRESSION>;
            ENTRY: MINIMAL AND MAXIMAL STEPLENGTH BY WHICH THE INTE-
                   GRATION IS ALLOWED TO BE PERFORMED;
            BY PUTTING HMIN = HMAX THE STEPSIZE MECHANISM IS
            ELIMINATED; IN THIS CASE THE GIVEN VALUES FOR HMIN AND
            HMAX ARE IRRELEVANT, WHILE THE INTEGRATION IS PERFORMED
            WITH THE STEPLENGTH GIVEN BY H;
```

```
DELTA:  <ARITHMETIC EXPRESSION>;
        ENTRY: THE REAL PART OF THE POINT AT WHICH SOME OPERATOR
               MAY BE FITTED EXPONENTIALLY
               (SEE METHOD AND PERFORMANCE);
        ALTERNATIVES:
        DELTA = (AN ESTIMATE OF) THE REAL PART OF THE LARGEST
        EIGENVALUE IN MODULUS OF THE JACOBIAN MATRIX OF THE
        SYSTEM;
        DELTA <= -10**15, IN ORDER TO OBTAIN ASYMPTOTIC STABILITY
        FOR THE OPERATOR MENTIONED ABOVE;
        DELTA = 0, IN ORDER TO OBTAIN A HIGHER ORDER OF ACCURACY
        IN CASE OF LINEAR EQUATIONS;
DERIVATIVE: <PROCEDURE IDENTIFIER>;
        "PROCEDURE" DERIVATIVE(Y); "ARRAY" Y;
        <REPLACEMENT OF THE I-TH COMPONENT OF THE SOLUTION Y BY
        THE I-TH COMPONENT OF THE DERIVATIVE F(Y), I = 1,..., R>;
JACOBIAN: <PROCEDURE IDENTIFIER>;
        "PROCEDURE" JACOBIAN(J, Y); "ARRAY" J, Y;
        WHEN IN GMS JACOBIAN IS CALLED THE ARRAY Y CONTAINS
        THE VALUES OF THE DEPENDENT VARIABLE;
        UPON COMPLETION OF A CALL OF JACOBIAN THE ARRAY J SHOULD
        CONTAIN THE VALUES OF THE JACOBIAN MATRIX OF F(Y);
AETA, RETA: <ARITHMETIC EXPRESSION>;
        ENTRY: MEASURE OF THE ABSOLUTE AND RELATIVE LOCAL
               ACCURACY REQUIRED;
        THESE VALUES ARE IRRELEVANT WHEN THE INTEGRATION IS PER-
        FORMED WITHOUT THE STEPSIZE MECHANISM;
N:      <VARIABLE>;
        EXIT : THE NUMBER OF INTEGRATION STEPS;
JEV:    <VARIABLE>;
        EXIT: THE NUMBER OF JACOBIAN EVALUATIONS;
LU:     <VARIABLE>;
        EXIT: THE NUMBER OF LU-DECOMPOSITIONS;
NSJEV:  <VARIABLE>;
        ENTRY: NUMBER OF INTEGRATION STEPS PER
               JACOBIAN EVALUATION;
        THE VALUE OF NSJEV IS RELEVANT ONLY WHEN THE INTEGRATION
        IS PERFORMED WITHOUT THE STEPSIZE MECHANISM AND THE
        SYSTEM TO BE SOLVED IS NON-LINEAR;
LINEAR: <BOOLEAN EXPRESSION>;
        ENTRY: TRUE WHEN THE SYSTEM TO BE INTEGRATED IS LINEAR,
               OTHERWISE FALSE;
        IF LINEAR IS TRUE THE STEPSIZE MECHANISM IS AUTOMATICALLY
        ELIMINATED;
OUT:    <PROCEDURE IDENTIFIER>;
        "PROCEDURE" OUT;
        <AFTER EACH INTEGRATION STEP ONE MAY OUT ORDER TO PRINT
        THE VALUES OF THE RELEVANT PARAMETERS OCCURRING IN THE
        PARAMETERLIST>.
```

DATA AND RESULTS: SEE REF[2].

PROCEDURES USED:

VECVEC = CP34010,
MATVEC = CP34011,
MATMAT = CP34013,
ELMROW = CP34024,
ELMVEC = CP34020,
DUPVEC = CP31030,
GSSELM = CP34231,
SOLELM = CP34061,
COLCST = CP31131,
MULVEC = CP31020.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: 8 * R + 3 * R * R;

RUNNING TIME:

DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO BE SOLVED.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

THE PROCEDURE GMS DESCRIBES AN IMPLEMENTATION OF A THIRD ORDER THREE-STEP GENERALIZED LINEAR MULTISTEP METHOD WITH QUASI-ZERO PARASITIC ROOTS AND QUASI-ADAPTIVE STABILITY FUNCTION. IN PARTICULAR THE ALGORITHM IS DEVELOPED FOR THE INTEGRATION OF STIFF SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS. THE PROCEDURE SUPPLIES THE ADDITIONAL STARTING VALUES AND PERFORMS A STEPSIZE CONTROL WHICH IS BASED ON THE NON-LINEARITY OF THE DIFFERENTIAL EQUATION. BY THIS CONTROL THE JACOBIAN MATRIX IS INCIDENTALLY EVALUATED. IT IS POSSIBLE TO ELIMINATE THE STEPSIZE CONTROL. THEN, ONE HAS TO GIVE THE NUMBER OF INTEGRATION STEPS PER JACOBIAN EVALUATION. FOR LINEAR EQUATIONS THE STEPSIZE CONTROL IS AUTOMATICALLY ELIMINATED, WHILE THE PROCEDURE PERFORMS ONE EVALUATION OF THE JACOBIAN. MOREOVER, IN THIS CASE THE THREE-STEP SCHEME IS REDUCED TO A ONE-STEP SCHEME. THE PROCEDURE USES ONE FUNCTION EVALUATION PER INTEGRATION STEP AND IT DOES NOT REJECT INTEGRATION STEPS. EACH CHANGE IN THE STEPLENGTH OR EACH REEVALUATION OF THE JACOBIAN COSTS ONE LU-DECOMPOSITION. IT IS POSSIBLE TO FIT SOME OPERATOR (PRESENT IN THE METHOD) EXPONENTIALLY. THIS FITTING IS EQUIVALENT TO FITTING IN THE SENSE OF LINIGER AND WILLOUGHBY, ONLY WHEN THE JACOBIAN MATRIX IS EVALUATED AT EACH INTEGRARTION STEP. WHEN THE SYSTEM TO BE INTEGRATED IS NON-LINEAR AND THE JACOBIAN MATRIX IS NOT EVALUATED AT EACH INTEGRATION STEP, IT IS RECOMMENDED TO FIT AT INFINITY (DELTA <= -10**15). DETAILS ARE GIVEN IN REFERENCE 2.

REFERENCES:

[1] HOUWEN, P. J. VAN DER AND VERWER, J. G.,
GENERALIZED LINEAR MULTISTEP METHODS 1, DEVELOPMENT OF ALGORITHMS WITH ZERO-PARASITIC ROOTS,
REPORT NW 10/74, MATHEMATISCH CENTRUM, AMSTERDAM 1974.
[2] VERWER, J. G.,
GENERALIZED LINEAR MULTISTEP METHODS 2, NUMERICAL APPLICATIONS, REPORT NW 12/74, MATHEMATISCH CENTRUM, AMSTERDAM, 1974.

EXAMPLE OF USE:

WE CONSIDER THE DIFFERENTIAL EQUATION

DY1/DX = -1000 * Y1 * (Y1 * Y2 - 1.999987),
DY2/DX = -2500 * Y2 * (Y1 + Y2 - 2),

ON THE INTERVAL [0,50], WITH INITIAL VALUE Y1(0) = Y2(0) = 1.
THE REFERENCE SOLUTION AT X = 50 IS GIVEN BY:
Y1(50) = .5976546988,
Y2(50) = 1.4023434075.

```
"BEGIN"
    "PROCEDURE" DER(Y); "ARRAY" Y;
    "BEGIN" "REAL" Y1, Y2;
        Y1:= Y[1]; Y2:= Y[2];
        Y[1]:= -1000 * Y1 * (Y1 + Y2 - 1.999987);
        Y[2]:= -2500 * Y2 * (Y1 + Y2 - 2)
    "END" DER;

    "PROCEDURE" JAC(J, Y); "ARRAY" J, Y;
    "BEGIN" "REAL" Y1, Y2; Y1:= Y[1]; Y2:= Y[2];
        J[1,1]:= 1999.987 - 1000 * (2 * Y1 + Y2);
        J[1,2]:= -1000 * Y1;
        J[2,1]:= -2500 * Y2;
        J[2,2]:= 2500 * (2 - Y1 - 2 * Y2)
    "END" JAC;

    "PROCEDURE" OUTP;
    "IF" X = 50 "THEN"
    "BEGIN"  "REAL"  YE1, YE2;
        YE1:= .5976546988; YE2:= 1.4023434075;
        OUTPUT(61, "("
        "("X = ")", 2D2B,
        "("N = ")", 3ZD2B,
        "("JEV = ")", 3ZD2B,
        "("LU = ")", 3ZD, 2/,
        "("Y1 = ")", +.13D"+2D2B,
        "("REL. ERR. = ")", .2D"+2D, /,
        "("Y2 = ")", +.13D"+2D2B,
        "("REL. ERR. = ")", .2D"+2D")",
        X, N, JEV, LU, Y[1], ABS((Y[1] - YE1) / YE1),
        Y[2], ABS((Y[2] - YE2) / YE2))
    "END" OUTP;
```

```
        "INTEGER" N, JEV, LU; "REAL" X;
        "ARRAY" Y[1:2];
        "PROCEDURE" GMS(X, XE, R, Y, H, HMIN, HMAX, DELTA,
                        DERIVATIVE, JACOBIAN, AETA, RETA, N,
                        JEV, LU, NSJEV, LINEAR, OUT); "CODE" 34191;

        X:= 0; Y[1]:= Y[2]:= 1;
        GMS(X, 50, 2, Y, .01, .001, .5, -"15,
            DER, JAC, "-5, "-5, N, JEV,
            LU, 0, "FALSE", OUTP)
    "END"
```

THIS PROGRAM DELIVERS:

X = 50 N = 109 JEV =   3 LU =   12

Y1 = +.5976547958004"+00 REL. ERR. = .16"-06
Y2 = +.1402343310813"+01 REL. ERR. = .69"-07

SOURCE TEXT:

```
"CODE" 34191;
    "PROCEDURE" GMS(X, XE, R, Y, H, HMIN, HMAX, DELTA, DERIVATIVE,
                    JACOBIAN, AETA, RETA, N, JEV, LU, NSJEV,
                    LINEAR, OUT);
    "VALUE" R;
    "REAL" X, XE, H, HMIN, HMAX, DELTA, AETA, RETA;
    "INTEGER" R, N, JEV, NSJEV, LU;
    "BOOLEAN" LINEAR;
    "ARRAY" Y;
    "PROCEDURE" DERIVATIVE, JACOBIAN, OUT;
    "BEGIN"
        "INTEGER" I, J, K, L, NSJEV1, COUNT, COUNT1, KCHANGE;
        "REAL" A, A1, ALFA, E, S1, S2, Z1, X0, XL0, XL1,
        XL2, ETA, H0, H1, Q, Q1, Q2, Q12, Q22, Q1Q2, DISCR;
        "BOOLEAN" UPDATE, CHANGE, REEVAL, STRATEGY;
        "INTEGER" "ARRAY" RI, CI[1:R];
        "ARRAY" AUX[1:9], BD1, BD2[1:3,1:3], Y1,
        Y0[1:R], HJAC, H2JAC2, RQZ[1:R,1:R], YL, FL[1:3 * R];

        "REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B); "CODE" 34010;
        "REAL" "PROCEDURE" MATVEC(L, U, I, A, B); "CODE" 34011;
        "REAL" "PROCEDURE" MATMAT(L, U, I, J, A, B); "CODE" 34013;
        "PROCEDURE" ELMROW(L, U, I, J, A, B, X); "CODE" 34024;
        "PROCEDURE" ELMVEC(L U, SHIFT, A, B, X); "CODE" 34020;
        "PROCEDURE" DUPVEC(L, U, SHIFT, A, B); "CODE" 31030;
        "PROCEDURE" GSSELM(A, N, AUX, RI, CI); "CODE" 34231;
        "PROCEDURE" SOLELM(A, N, RI, CI, B); "CODE" 34061;
        "PROCEDURE" COLCST(L, U, J, A, X); "CODE" 31131;
        "PROCEDURE" MULVEC(L, U, SHIFT, A, B, X); "CODE" 31020;
```

```
"PROCEDURE" INITIALIZATION;
"BEGIN" LU:= JEV:= N:= NSJEV1:= KCHANGE:= 0; X0:= X; DISCR:= 0
    K:=1; H1:= H0:= H; COUNT:= -2; AUX[2]:= "-14; AUX[4]:= 8;
    DUPVEC(1, R, 0, YL, Y); REEVAL:= CHANGE:= "TRUE";
    STRATEGY:= HMIN ^= HMAX "AND" ^LINEAR; Q1:= -1; Q2:= -2;
    COUNT1:= 0; XL0:= XL1:= XL2:= 0
"END" INITIALIZATION;

"PROCEDURE" COEFFICIENT;
"BEGIN" XL2:= XL1; XL1:= XL0; XL0:= X0;
    "IF" CHANGE "THEN"
    "BEGIN" "IF" N > 2 "THEN"
        "BEGIN" Q1:= (XL1 - XL0) / H1;
            Q2:= (XL2 - XL0) / H1
        "END";
        Q12:= Q1 * Q1; Q22:= Q2 * Q2; Q1Q2:= Q1 * Q2;
        A:= -(3 * ALFA + 1) / 12;
        BD1[1,3]:= 1 + (1 / 3 - (Q1 + Q2) * .5) / Q1Q2;
        BD1[2,3]:= (1 / 3 - Q2 * .5) / (Q12 - Q1Q2);
        BD1[3,3]:= (1 / 3 - Q1 * .5) / (Q22 - Q1Q2);
        BD2[1,3]:= -ALFA * .5 + A * (1 - Q1 - Q2) / Q1Q2;
        BD2[2,3]:= A * (1 - Q2) / (Q12 - Q1Q2);
        BD2[3,3]:= A * (1 - Q1) / (Q22 - Q1Q2);
        "IF" STRATEGY "OR" N <= 2 "THEN"
        "BEGIN" BD1[2,2]:= 1 / (2 * Q1);
            BD1[1,2]:= 1 - BD1[2,2];
            BD2[2,2]:= -(3 * ALFA + 1) / (12 * Q1);
            BD2[1,2]:= -BD2[2,2] - ALFA * .5
        "END"
    "END"
"END" COEFFICIENT;

"PROCEDURE" OPERATOR CONSTRUCTION;
"BEGIN" "IF" REEVAL "THEN"
    "BEGIN" JACOBIAN(HJAC, Y);
        JEV:= JEV + 1; NSJEV1:= 0;
        "IF" DELTA <= -"15 "THEN" ALFA:= 1 / 3 "ELSE"
        "BEGIN" Z1:= H1 * DELTA;
            A:= Z1 * Z1 + 12; A1:= 6 * Z1;
            "IF" ABS(Z1) < .1 "THEN"
            ALFA:= (Z1 * Z1 / 140 - 1) * Z1 / 30 "ELSE"
            "IF" Z1 < -33 "THEN"
            ALFA:= (A + A1) / (3 * Z1 * (2 + Z1)) "ELSE"
            "BEGIN" E:= EXP(Z1); ALFA:= ((A - A1) *
                E - A - A1) / (((2 - Z1) * E - 2 - Z1) *
                Z1 * 3)
            "END"
        "END";
        S1:= -(1 + ALFA) * .5; S2:= (ALFA * 3 + 1) / 12
    "END";
    A:= H1 / H0; A1:= A * A;
    "IF" REEVAL "THEN" A:= H1;
    "IF" A ^= 1 "THEN"
    "FOR" J:= 1 "STEP" 1 "UNTIL" R "DO"
    COLCST(1, R, J, HJAC, A);
```

```
    "FOR" I:= 1 "STEP" 1 "UNTIL" R "DO"
    "BEGIN" "FOR" J:= 1 "STEP" 1 "UNTIL" R "DO"
        "BEGIN" Q:= H2JAC2[I,J]:= "IF" REEVAL "THEN"
            MATMAT(1, R, I, J, HJAC, HJAC)
            "ELSE" H2JAC2[I,J] * A1;
            RQZ[I,J]:= S2 * Q
        "END";
        RQZ[I,I]:= RQZ[I,I] + 1;
        ELMROW(1, R, I, I, RQZ, HJAC, S1)
    "END";
    GSSELM(RQZ, R, AUX, RI, CI); LU:= LU + 1;
    REEVAL:= UPDATE:= "FALSE"
"END" OPERATOR CONSTRUCTION;

"PROCEDURE" DIFFERENCE SCHEME;
"BEGIN" "IF" COUNT =: 1 "THEN"
    "BEGIN" DUPVEC(1, R, 0, FL, YL);
        DERIVATIVE(FL); N:= N + 1; NSJEV1:= NSJEV1 + 1
    "END";
    MULVEC(1, R, 0, Y0, YL, (1 - ALFA) / 2 - BD1[1,K]);
    "FOR" L:= 2 "STEP" 1 "UNTIL" K "DO"
    ELMVEC(1, R, R * (L - 1), Y0, YL, -BD1[L,K]);
    "FOR" L:= 1 "STEP" 1 "UNTIL" K "DO"
    ELMVEC(1, R, R * (L - 1), Y0, FL, H1 * BD2[L,K]);
    "FOR" I:= 1 "STEP" 1 "UNTIL" R "DO"
    Y[I]:= MATVEC(1, R, I, HJAC, Y0);
    MULVEC(1, R, 0, Y0, YL, (1 - 3 * ALFA) / 12 - BD2[1,K]);
    "FOR" L:= 2 "STEP" 1 "UNTIL" K "DO"
    ELMVEC(1, R, R * (L - 1), Y0, YL, -BD2[L,K]);
    "FOR" I:= 1 "STEP" 1 "UNTIL" R "DO"
    Y[I]:= Y[I] + MATVEC(1, R, I, H2JAC2, Y0);
    DUPVEC(1, R, 0, Y0, YL);
    "FOR" L:= 1 "STEP" 1 "UNTIL" K "DO"
    ELMVEC(1, R, R * (L - 1), Y0, FL, H1 * BD1[L,K]);
    ELMVEC(1, R, 0, Y, Y0, 1); SOLELM(RQZ, R, RI, CI, Y)
"END" DIFFERENCE SCHEME;

"PROCEDURE" NEXT INTEGRATION STEP;
"BEGIN" "FOR" L:= 2, 1 "DO"
    "BEGIN" DUPVEC(L * R + 1, (L + 1) * R, -R, YL, YL);
        DUPVEC(L * R + 1, (L + 1) * R, -R, FL, FL)
    "END";
    DUPVEC(1, R, 0, YL, Y)
"END" NEXT INTEGRATION STEP;

"PROCEDURE" TEST ACCURACY;
"BEGIN" K:= 2;
    DUPVEC(1, R, 0, Y1, Y); DIFFERENCE SCHEME; K:= 3;
    ETA:= AETA + RETA * SQRT(VECVEC(1, R, 0, Y1, Y1));
    ELMVEC(1, R, 0, Y, Y1, -1);
    DISCR:= SQRT(VECVEC(1, R, 0, Y, Y));
    DUPVEC(1, R, 0, Y, Y1)
"END" TEST ACCURACY;
```

```
    "PROCEDURE" STEPSIZE;
    "BEGIN" X0:= X; H0:= H1;
        "IF" N <= 2 "AND" ~LINEAR "THEN" K:= K + 1;
        "IF" COUNT = 1 "THEN"
        "BEGIN" A:= ETA / (.75 * (ETA + DISCR)) + .33;
            H1:= "IF" A <= .9 "OR" A >= 1.1 "THEN" A * H0
            "ELSE" H0; COUNT:= 0;
            REEVAL:= A <= .9 "AND" NSJEV1 ~= 1;
            COUNT1:= "IF" A >= 1 "OR" REEVAL "THEN" 0 "ELSE"
            COUNT1 + 1; "IF" COUNT1 = 10 "THEN"
            "BEGIN" COUNT1:= 0; REEVAL:= "TRUE";
                H1:= A * H0
            "END"
        "END" "ELSE"
        "BEGIN" H1:= H; REEVAL:= NSJEV = NSJEV1 "AND"
            ~STRATEGY "AND" ~LINEAR
        "END";
        "IF" STRATEGY "THEN" H1:= "IF" H1 > HMAX
        "THEN" HMAX "ELSE" "IF" H1 < HMIN "THEN" HMIN "ELSE" H1;
        X:= X + H1; "IF" X >= XE "THEN"
        "BEGIN" H1:= XE - X0; X:= XE "END";
        "IF" N <= 2 "AND" ~LINEAR "THEN" REEVAL:= "TRUE";
        "IF" H1 ~= H0 "THEN"
        "BEGIN" UPDATE:= "TRUE"; KCHANGE:= 3 "END";
        "IF" REEVAL "THEN" UPDATE:= "TRUE";
        CHANGE:= KCHANGE > 0 "AND" ~LINEAR;
        KCHANGE:= KCHANGE - 1
    "END" STEPSIZE;


    INITIALIZATION; OUT; X:= X + H1;
    OPERATOR CONSTRUCTION;
    BD1[1,1]:= 1; BD2[1,1]:= -ALFA * .5;
    "IF" ~LINEAR "THEN" COEFFICIENT;
NEXT STEP: DIFFERENCE SCHEME;
    "IF" STRATEGY "THEN" COUNT:= COUNT + 1;
    "IF" COUNT = 1 "THEN" TEST ACCURACY;
    OUT; "IF" X >= XE "THEN" "GOTO" END;
    STEPSIZE; "IF" UPDATE "THEN" OPERATOR CONSTRUCTION;
    "IF" ~LINEAR "THEN" COEFFICIENT;
    NEXT INTEGRATION STEP; "GOTO" NEXT STEP;
END:
"END" GMS;
    "EOP"
```

8. COMPUTATIONAL RESULTS

The procedure GMS was tested on a number of stiff nonlinear systems. As noted before, for linear equations our algorithm is reduced to the $F^{(3)}$ formula of LINIGER & WILLOUGHBY [11]. Numerical results of this formula applied on linear equations are presented in BEENTJES & DEKKER [1]. For each example the following quantities are listed in the tables of results:

sdj: the number of significant digits of the j-th component with respect to a given reference solution, i.e.

$$sdj = -^{10}\log\left|1 - \frac{y_j}{\tilde{y}_j}\right|, \quad |sdj| \leq 14,$$

where $\tilde{y}_j$ denotes the reference solution; from this relation it follows that the absolute error is given by

$$|\tilde{y}_j - y_j| = 10^{-sdj}\tilde{y}_j;$$

hence a negative value of sdj does not necessarily mean an inaccurate result when y_j is very small in magnitude; the procedure was tested on the CYBER 73-26 computer of SARA at Amsterdam; since this computer does not represent more than 14 significant digits we have sdj < 14; when the computer delivered sdj < -14, we have put sdj = = -14; thus, generally, sdj = -14 in the tables means instability;

fev: number of function evaluations necessary to integrate the given integration interval; note that fev also equals the number of integration steps;

jev: number of evaluations of the Jacobian matrix of the system necessary to integrate the given integration interval;

lu: number of LU-decompositions necessary to integrate the given integration interval;

tol: a measure of the required local accuracy; for each example the absolute and relative accuracy parameters aeta and reta are put equal to tol;

hmin: minimal stepsize by which the integration is allowed to be performed;

hmax: maximal stepsize by which the integration is allowed to be performed.

For all examples, the procedure is applied with stepsize control, while the parameter delta is put equal to -10^{15}. This means no exponential fitting has been performed. The quantities listed in the tables of results are obtained by integrating over the interval [initial point, reference point].

EXAMPLE 8.1

A non-linear system from chemistry (GEAR [3]):

$$\frac{dy_1}{dx} = -1000y_1(y_1+y_2-1.999987),$$

$$\frac{dy_2}{dx} = -2500y_2(y_1+y_2-2),$$

$$y_1(0) = y_2(0) = 1,\ 0 \le x \le 50.$$

The eigenvalues δ_1 and δ_2 of the Jacobian are

$$\delta_1 \approx -3500,\ \delta_2 \approx 0,\ \delta_2 < 0 \text{ at } x = 0 \text{ and}$$

$$\delta_1 \approx -4100,\ \delta_2 \approx 0,\ \delta_2 < 0 \text{ at } x = 50.$$

Reference solution:

$$\tilde{y}_1(\tfrac{1}{64}) = .999853854436,\ \tilde{y}_1(50) = .5976546988,$$

$$\tilde{y}_2(\tfrac{1}{64}) = 1.00014243203,\ \tilde{y}_2(50) = 1.40234334075.$$

Relevant parameters:

$$h = hmin = .001,\ hmax = .5.$$

TABLE 8.1.1. Results obtained at $x = \frac{1}{64}$.

tol	sd1	sd2	fev	jev	lu
10^{-4}	12.2	12.1	7	3	7
10^{-5}	12.2	12.0	7	3	7
10^{-6}	12.3	12.1	7	3	7
10^{-7}	12.2	12.2	7	3	7
10^{-8}	12.3	12.1	7	3	7
10^{-9}	12.3	12.1	7	3	7

TABLE 8.1.2. Results obtained at x = 50.

tol	sd1	sd2	fev	jev	lu
10^{-4}	7.2	7.6	113	3	17
10^{-5}	7.7	8.0	113	3	17
10^{-6}	7.2	7.6	113	3	17
10^{-7}	7.4	7.8	113	3	17
10^{-8}	7.6	7.9	140	3	21
10^{-9}	8.7	9.2	297	8	28

EXAMPLE 8.2

A non-linear system describing the motion of control rod in a nuclear reactor (LINIGER & WILLOUGHBY [11]):

$$\frac{dy_1}{dx} = 10y_2 + .125y_3 - (60 - .125y_3)y_1,$$

$$\frac{dy_2}{dx} = .2(y_1 - y_2),$$

$$\frac{dy_3}{dx} = 1,$$

$$y_1(0) = y_2(0) = y_3(0) = 0, \quad 0 \le x \le 400.$$

The eigenvalues δ_1, δ_2 and δ_3 of the Jacobian are

$$\delta_1 \approx -60,\ \delta_2 \approx -.17,\ \delta_3 = 0 \text{ at } x = 0 \text{ and}$$
$$\delta_1 \approx -10,\ \delta_2 \approx 0,\ \delta_2 < 0,\ \delta_3 = 0 \text{ at } x = 400.$$

Reference solution:

$$\tilde{y}_1(10) = .02344886,\ \tilde{y}_1(400) = 27.110701,$$
$$\tilde{y}_2(10) = .01301528,\ \tilde{y}_2(400) = 22.242211,$$
$$\tilde{y}_3(10) = 10,\qquad \tilde{y}_3(400) = 400.$$

Relevant parameters:

h = hmin = .01, hmax = 1.

TABLE 8.2.1. Results obtained at x = 10.

tol	sd1	sd2	fev	jev	lu
10^{-3}	3.1	2.1	20	3	14
10^{-4}	3.1	2.1	20	3	14
10^{-5}	3.4	2.4	24	3	20
10^{-6}	4.5	3.5	41	3	20
10^{-7}	5.6	4.6	81	3	19

TABLE 8.2.2. Results obtained at x = 400.

tol	sd1	sd2	fev	jev	lu
10^{-3}	2.3	2.4	410	3	14
10^{-4}	2.6	2.5	411	4	15
10^{-5}	3.6	3.5	439	12	29
10^{-6}	4.9	4.8	612	34	58
10^{-7}	6.3	6.1	1384	98	124

EXAMPLE 8.3

A non-linear system from reactor kinetics (LINIGER & WILLOUGHBY [10]):

$$\frac{dy_1}{dx} = .01-[1+(y_1+1000)(y_1+1)][.01+y_1+y_2],$$

$$\frac{dy_2}{dx} = .01-[1+y_2^2][.01+y_1+y_2],$$

$$y_1(0) = y_2(0) = 0,\ 0 \le x \le 100.$$

The eigenvalues δ_1 and δ_2 of the Jacobian are

$$\delta_1 \approx -1012,\ \delta_2 \approx -.01 \text{ at } x = 0 \text{ and}$$

$$\delta_1 \approx -21.7,\ \delta_2 \approx -.089 \text{ at } x = 100.$$

Reference solution:

$$\tilde{y}_1(10) = -.10975436,\ \tilde{y}_1(100) = -.99164207,$$

$$\tilde{y}_2(10) = .09977673,\ \tilde{y}_2(100) = .98333636.$$

Relevant parameters:

$$h = hmin = .01,\ hmax = 1.$$

TABLE 8.3.1. Results obtained at x = 10

tol	sd1	sd2	fev	jev	lu
10^{-3}	5.0	5.0	20	3	14
10^{-4}	5.0	5.0	21	3	14
10^{-5}	5.0	5.0	21	3	14
10^{-6}	5.0	5.0	22	3	14
10^{-7}	5.0	5.0	23	4	15
10^{-8}	5.0	5.0	24	4	16

TABLE 8.3.2. Results obtained at x = 100

tol	sd1	sd2	fev	jev	lu
10^{-3}	2.5	2.6	110	3	14
10^{-4}	2.5	2.6	111	3	14
10^{-5}	3.1	3.1	113	5	17
10^{-6}	4.8	4.8	139	16	32
10^{-7}	8.5	7.8	219	31	42
10^{-8}	6.2	6.2	474	49	61

EXAMPLE 8.4

A large, non-linear system from chemistry (DATTA [2]):

$$\frac{dy_1}{dx} = -K_1 y_1,$$

$$\frac{dy_2}{dx} = K_1 y_1 + K_{11}K_{14}y_4 + K_{19}K_{14}y_5 - K_3 y_2 y_3 - K_{15}y_2 y_{12} - K_2 y_2,$$

$$\frac{dy_3}{dx} = K_2 y_2 - K_5 y_3 - K_3 y_2 y_3 - K_7 y_{10} y_3 + K_{11}K_{14}y_4 + K_{12}K_{14}y_6,$$

$$\frac{dy_4}{dx} = K_3 y_2 y_3 - K_{11}K_{14}y_4 - K_4 y_4,$$

$$\frac{dy_5}{dx} = K_{15}y_2 y_{12} - K_{19}K_{14}y_5 - K_{16}y_5,$$

$$\frac{dy_6}{dx} = K_7 y_{10} y_3 - K_{12}K_{14}y_6 - K_8 y_6,$$

$$\frac{dy_7}{dx} = K_{17}y_{10} y_{12} - K_{20}K_{14}y_7 - K_{18}y_7,$$

$$\frac{dy_8}{dx} = K_9 y_{10} - K_{13}K_{14}y_8 - K_{10}y_8,$$

$$\frac{dy_9}{dx} = K_4 y_4 + K_{16}y_5 + K_8 y_6 + K_{18}y_7,$$

$$\frac{dy_{10}}{dx} = K_5y_3 + K_{12}K_{14}y_6 + K_{20}K_{14}y_7 + K_{13}K_{14}y_8 - K_7y_{10}y_3 +$$

$$- K_{17}y_{10}y_{12} - K_6y_{10} - K_9y_{10},$$

$$\frac{dy_{11}}{dx} = K_{10}y_8,$$

$$\frac{dy_{12}}{dx} = K_6y_{10} + K_{19}K_{14}y_5 + K_{20}K_{14}y_7 - K_{15}y_2y_{12} - K_{17}y_{10}y_{12},$$

with

$$K_2 = K_6 = 10.0,\ K_1 = K_5 = 0.1,\ K_{10} = 5.0,$$

$$K_{14} = 30.0,\ K_4 = K_{16} = K_8 = K_{18} = 2.5,$$

$$K_3 = K_7 = K_9 = K_{11} = K_{19} = K_{12} = K_{20} = K_{13} = 50.0,$$

$$K_{15} = K_{17} = 100.0;$$

$$y_1(0) = 1,\ y_i(0) = 0,\ i=2,\ldots,12;\ 0 \le x \le 50.$$

The eigenvalue with greatest magnitude at x = 0 equals ≈ -1555.16.

Reference solution:

$$\tilde{y}_3(\tfrac{1}{64}) = .115825313_{10}-3,\ \tilde{y}_3(50) = .334500768_{10}-1,$$

$$\tilde{y}_5(\tfrac{1}{64}) = .179687_{10}-12,\quad \tilde{y}_5(50) = .407994_{10}-5.$$

$$\tilde{y}_9(\tfrac{1}{64}) = .476247_{10}-10,\quad \tilde{y}_9(50) = .149109210_{10}-1,$$

$$\tilde{y}_{12}(\tfrac{1}{64}) = .2269204_{10}-8,\quad \tilde{y}_{12}(50) = .914169996.$$

Relevant parameters:

$$h = hmin = .0005,\ hmax = .5.$$

TABLE 8.4.1. Results obtained at $x = \frac{1}{64}$

tol	sd3	sd5	sd9	sd12	fev	jev	lu
10^{-3}	5.4	.3	1.1	2.3	8	3	8
10^{-4}	5.4	.3	1.1	2.3	8	3	8
10^{-5}	5.4	.3	1.1	2.3	8	3	8
10^{-6}	5.4	.3	1.1	2.3	8	3	8
10^{-7}	5.4	.3	1.1	2.3	8	3	8
10^{-8}	5.4	.3	1.1	2.2	8	3	8

TABLE 8.4.2. Results obtained at x = 50

tol	sd3	sd5	sd9	sd12	fev	jev	lu
10^{-3}	5.8	5.0	4.4	5.9	115	3	18
10^{-4}	5.8	5.2	4.4	5.9	115	3	18
10^{-5}	5.8	4.5	4.7	6.3	115	3	19
10^{-6}	5.8	4.5	4.8	6.2	124	3	26
10^{-7}	6.1	3.8	6.2	7.7	211	4	36
10^{-8}	7.5	5.3	7.1	9.2	584	6	41

EXAMPLE 8.5

A highly stiff non-linear system representing a set of chemical reaction rate equations (ROBERTSON [13]):

$$\frac{dy_1}{dx} = -.04y_1 + 10^4 y_2 y_3,$$

$$\frac{dy_2}{dx} = .04y_1 - 10^4 y_2 y_3 - 3_{10}7y_2^2,$$

$$\frac{dy_3}{dx} = 3_{10}7y_2^2,$$

$$y_1(0) = 1,\ y_2(0) = y_3(0) = 0,\ 0 \le x \le 10.$$

These equations are dependent; by use of the initial values we have

$$y_1(x) + y_2(x) + y_3(x) = 1.$$

By means of this relation we remove the first component and the system is reduced to

$$\frac{dy_1}{dx} = .04 - .04(y_1+y_2) - 10^4 y_1 y_2 - 3_{10}7y_1^2,$$

$$\frac{dy_2}{dx} = 3_{10}7y_1^2,$$

$$y_1(0) = y_2(0) = 0, \quad 0 \le x \le 10.$$

The eigenvalues δ_1 and δ_2 of the Jacobian are

$$\delta_1 \approx -.04, \quad \delta_2 \approx 0 \text{ at } x = 0 \text{ and}$$

$$\delta_1 \approx -10000, \quad \delta_2 \approx 0 \text{ at } x = 10.$$

Reference solution:

$$\tilde{y}_1(10) = .1623391063_{10}-4,$$

$$\tilde{y}_2(10) = .1586138424.$$

Relevant parameters:

$$h = hmin = .0005, \quad hmax = .5.$$

TABLE 8.5.1. Results obtained at x = 10

tol	sd1	sd2	fev	jev	lu
10^{-3}	2.2	3.0	39	3	24
10^{-4}	2.5	3.1	54	3	29
10^{-5}	4.2	4.1	46	5	30
10^{-6}	4.6	4.6	65	5	41
10^{-7}	5.5	5.5	113	5	48
10^{-8}	6.2	6.1	218	9	56
10^{-9}	6.9	7.2	457	9	62

REMARKS:

Example 8.5 is a very stiff non-linear system. The eigenvalues change from 0, -.04 to 0, -10^4 over the range x = 0 to x = 10. In fact, most of this change occurs in the first few steps, e.g. $|\delta_1|$ changes from .04 to 2405 within x = 0 to x = 0.02. Thus, this example represents one of the severest tests. We may illustrate this by presenting table 8.5.2. The results are obtained with the initial steplength h = .001 (hmin = .001, hmax = .5).

TABLE 8.5.2.

tol	sd1	sd2	fev	jev	lu
10^{-3}	-14	-14			
10^{-4}	-14	-14			
10^{-5}	4.2	4.0	53	6	35
10^{-6}	4.6	4.6	64	6	41
10^{-7}	5.3	5.5	111	7	45
10^{-8}	6.2	6.1	218	10	56
10^{-9}	6.8	7.0	456	11	63

The instabilities occurring for tol = 10^{-3}, 10^{-4} are more or less due to the second order starting formula which provides the first solution vector. We may explain this assertion as follows: at x = 0 the first and second derivatives of y_2 are equal to zero, whereas the third derivative of y_2 is not equal to zero. However, our starting formula is of second order and represents only the first- and second derivatives of y for h small enough. Consequently, for each initial steplength h our starting formula provides $y_2 = 0$. The reference solution at x = .001 is

$$\tilde{y}_1 = .239983_{10}{}^{-4},$$

$$\tilde{y}_2 = .160009_{10}{}^{-4}.$$

For the lower tolerances and initial steplength h = .001 the value $y_2 = 0$ at x = .001 appears to be fatal. When we start the integration process at x = .001 with the initial vector given above and initial steplength h = .001, no instabilities will arise. Results of this integration are given in table 8.5.3. At x = .001 the eigenvalues δ_1 and δ_2 are $\delta_1 \approx -.04$, $\delta_2 \approx 0$.

TABLE 8.5.3.

tol	sd1	sd2	fev	jev	lu
10^{-3}	3.9	3.8	33	3	17
10^{-4}	4.0	3.9	34	3	18
10^{-5}	5.2	5.0	38	3	24
10^{-6}	5.0	4.6	59	4	38
10^{-7}	5.1	5.4	109	5	46
10^{-8}	6.4	6.3	214	9	55
10^{-9}	6.9	7.2	454	9	61

REFERENCES

[1] BEENTJES, P.A. & DEKKER, K., *Colloquium stijve differentiaalvergelijkingen*, MC Syllabus 15.3, Mathematisch Centrum, Amsterdam (Dutch), 1974.

[2] DATTA, A.K., *An evaluation of the approximate inverse algorithm for numerical integration of stiff differential equations*, Technical report MS/67/84, Imperial Chemical Industries Ltd., Cheshire, 1967.

[3] GEAR, C.W., *The automatic integration of stiff ordinary differential equations*, Information Processing 68, 187-194, ed. A.J.H. Morrell, North-Holland Publishing Co., Amsterdam, 1968.

[4] HEMKER, P.W. (ed.), *NUMAL, A library of numerical procedures in ALGOL-60, index and KWIC index*, Report NW 8/73, Mathematisch Centrum, Amsterdam, 1973.

[5] HOUWEN, P.J. VAN DER, *A note on two-step integration formulas of third order accuracy with prescribed stability function*, Report NR 26/72, Mathematisch Centrum, Amsterdam, 1972.

[6] HOUWEN, P.J. VAN DER & VERWER, J.G., *Generalized linear multistep methods I, Development of algorithms with zero-parasitic roots*, Report NW 10/74, Mathematisch Centrum, Amsterdam, 1974.

[7] HOUWEN, P.J. VAN DER, *Construction of integration formulas for initial value problems*, North-Holland Publishing Company, Amsterdam, (to be published).

[8] LAMBERT, J.D. & SIGURDSSON, S.T., *Multistep methods with variable matrix coefficients*, SIAM J. Numer. Anal. 9, 715-733, 1972.

[9] LAMBERT, J.D., *Computational methods in ordinary differential equations*, John Wiley and Sons, London, 1973.

[10] LINIGER, W. & WILLOUGHBY, R.A., *Efficient numerical integration of stiff systems of ordinary differential equations*, Technical Report RC-1970, IBM Thomas J. Watson Research Centre, Yorktown Heights, N.Y., 1967.

[11] LINIGER, W. & WILLOUGHBY, R.A., *Efficient numerical integration of stiff systems of ordinary differential equations*, SIAM J. Numer. Anal. 7, 47-66, 1970.

[12] NØRSETT, S.P., *An A-stable modification of the Adams-Bashforth methods*, Lecture Notes in Mathematics 109, ed. A. Dold and B. Echmann, Springer-Verlag, Berlin, 1969.

[13] ROBERTSON, H.H., *The solution of a set of reaction rate equations in numerical analysis*, ed. J. Walsh, Thompson Book Co., Washington, 1967.