MODELING, IDENTIFICATION AND CONTROL, 1985, VOL. 6, NO. 3, 141–152
doi:10.4173/mic.1985.3.3

The numerical solution of differential and differential/algebraic systems†

SYVERT P. NØRSETT‡

Keywords: *Initial value systems, algebraic constraints, stiff systems, discontinuities, index.*

Systems of ordinary differential equations (ODE) or ordinary differential/algebraic equations (DAE) are well-known mathematical models. The numerical solution of such systems are discussed. For (ODE) we mention some available codes and stress the need of type insensitive versions. Further the term stiffness is redefined, and ideas on handling discontinuities are presented. The paper ends with a discussion of index for DAE.

1. Introduction

Modeling physical phenomena often leads to differential or algebraic systems. In differential systems we distinguish between ordinary and partial differential equations. As an example of the first set we have population modeling and in the second class the model of a vibrating membrane, with or without an initial transient.

Due to the difficulty involved in solving these differential systems analytically, numerical techniques must be used. Time-dependent partial differential equations are often semi-discretized into ordinary differential systems (ODE). Further stationary partial differential equations are discretized to systems of non-linear or linear algebraic equations.

In this paper we concentrate on systems of ordinary differential systems and their numerical solution. The newest developments are discussed, including both non-stiff and stiff systems. The important question of how to design a code which can handle both cases in an efficient way is discussed together with the problem of how to handle discontinuities in the system efficiently. Finally problems connected to the handling of combined ordinary differential equations and algebraic systems, (DAE) are mentioned. This is a new field of research. Some insights have been gained, but open problems are still at hand.

2. Examples

2.1 One-dimensional heat-conduction

The modeling of heat-conduction in an isotropic one-dimensional rod of dimensionless length l is illustrated. The partial differential equation is the simple parabolic equation

$$\frac{\partial u}{\partial t} = \sigma \cdot \frac{\partial^2 u}{\partial x^2}; \qquad 0 < x < 1, \quad t > 0. \tag{2.1}$$

Received 17 ıne 1985.
† Paper pres ted at the International Seminar on Modern Methods in Dynamic Simulation of Industri Processes, Trondheim, Norway, May 1985.
‡ Dept. of N merical Math. NTH, Trondheim Norway.

σ is a positive constant and $u = u(x, t)$ the temperature in the rod. Let the initial and boundary conditions be given as

$$u(x, 0) = f(x) \tag{2.2a}$$

$$u(0, t) = \varphi_0(t), \qquad u(1, t) = \varphi_1(t). \tag{2.2b}$$

With second order space discretization eqn. (2.1) is approximated by

$$U'(t) = AU(t) + BV(t) \tag{2.3}$$

where

$$\left.\begin{aligned} U(t) &= [U_1(t), \ldots, U_N(t)]^T \\ V(t) &= [U_0(t), U_{N+1}(t)]^T \end{aligned}\right\} \begin{aligned} &U_i(t) \approx u(i\,\Delta x, t), \\ &i = 0, \ldots, N+1. \end{aligned}$$

Eqn. (2.2) is written as

$$U(0) = [f(i\,\Delta x);\ i = 1, \ldots, N] \tag{2.4}$$

$$CV(t) = \Phi(t) = \begin{bmatrix} \varphi_0(t) \\ \varphi_1(t) \end{bmatrix} \in \mathbb{R}^2 \tag{2.5}$$

The matrices A, B and C are given by

$$A = \frac{\sigma}{(\Delta x)^2} \begin{bmatrix} -2 & 1 & & 0 \\ 1 & & & \\ & & & 1 \\ 0 & & 1 & -2 \end{bmatrix} \in \mathbb{R}^{N \times N}$$

$$B = \frac{\sigma}{(\Delta x)^2} \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & & 0 & 1 \end{bmatrix}^T \in \mathbb{R}^{N \times 2}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2 \times 2}$$

Eqn. (2.3) together with eqn. (2.5) is a DAE of the form

$$\left.\begin{aligned} y' &= f(t, y, z); \quad y \in \mathbb{R}^{S_f} \\ 0 &= g(t, y, z); \quad z \in \mathbb{R}^{S_g} \end{aligned}\right\} \tag{2.6}$$

which is normally written as

$$y' = F(t, y) \tag{2.7}$$

since $\partial g/\partial z = I$ is regular and z can be solved from the last system and inserted into the first part.

2.2. *Flow model*

Next is the mathematical model for a compressible, inviscid, isentropic medium (Courant and Friedrichs 1948)

$$\begin{aligned} &\frac{\partial \rho}{\partial t} + (u \,.\, \nabla)\rho + \rho \nabla \,.\, u = 0; && \text{mass-conservation} \\ &\frac{\partial u}{\partial t} + (u \,.\, \nabla)u + \frac{1}{\rho} \nabla p = 0; && \text{momentum equation} \\ &p - f(\rho) = 0; && \text{equation of state} \end{aligned} \tag{2.8}$$

where ρ is the density, p the pressure and u the velocity. In addition there are initial and boundary conditions. If we let D be the semi-discretized internal density vector, U the similar velocity vector and P the pressure vector, eqn. (2.8) is approximated by the DAE-system

$$D' + C_1(U)D + C_2(D)U = G_1(U, D)$$
$$U' + C_3(U)U + C_4(D)P = G_2(U, D)$$
$$P - F(D) = 0$$

which is of the form of eqn. (2.6). When $P = F(D)$ is replaced in the second equation a system like eqn. (2.7) is again found.

2.3. Electrical network model

This is a model for an electrical network consisting of branches and nodes (Sincoree *et al.* 1979). For a network containing voltage sources and linear capacitors and resistors the equations are

$$\begin{bmatrix} C\dot{V}_C + & 0 & 0 & 0 & 0 & -I & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 & -R & 0 \\ 0 & 0 & -I & 0 & 0 & 0 & 0 & A_E \\ 0 & -I & 0 & 0 & 0 & 0 & 0 & A_C \\ 0 & 0 & 0 & -I & 0 & 0 & 0 & A_R \\ 0 & 0 & 0 & 0 & A_E^T & A_C^T & A_R^T & 0 \end{bmatrix} \begin{bmatrix} V_C \\ V_E \\ V_R \\ I_E \\ I_C \\ I_R \\ V_N \end{bmatrix} = \begin{bmatrix} 0 \\ E \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.9}$$

V_i and I_i ($i = E, C, R$) represent the voltages and currents in branches with voltage sources, capacitors or resistors. The node voltage vector is V_N.

Again a DAE-system in the form of eqn. (2.6) is obtained, but it is now much more complicated. Can it be transformed into a system of the form of eqn. (2.7)? A solution is found in the different sub-matrices of eqn. (2.9).

3. Recent developments for ODE-solvers

As Sections 2.1 and 2.2 indicate, some DAE-systems can be directly rewritten as a system of ODE's. The general form of such systems are

$$y' = F(t, y), \quad t \in (a, b); \quad y(t) \in \mathbb{R}^S. \tag{3.1}$$

In order to understand the problems connected to the solution of eqn. (2.6) we need to know how to solve eqn. (3.1).

The first real method of solving eqn. (3.1) was by Euler and was published in 1743 (Euler (1913) p. 422). However, the real development came around 1890 with the construction of the Adams methods and the Runge–Kutta methods. The modern history is dated to *ca.* 1960 with the work of Dahlquist (1956) and Butcher (1963). Later Dahlquist (1963) introduced the concept of A-stability and thereby started the research for suitable methods for solving stiff ODE's.

The word stiff was first used by two engineers, Curtiss and Hirschfelder (1952). A formal definition was given by Lambert (1973). He related the stiffness to the eigenvalue of the Jacobian

$$J = J(t, y) = \frac{\partial F}{\partial y}(t, y).$$

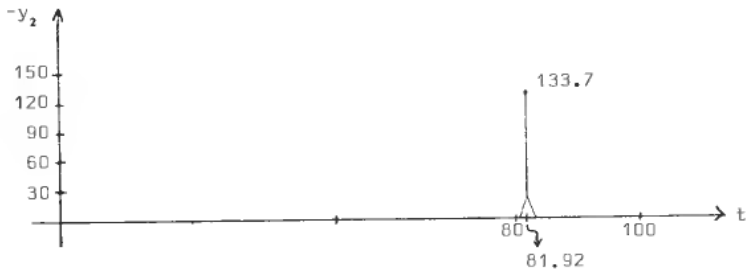

Figure 1. The solution component y_2 of eqn. (3.2)

According to that definition a problem (eqn. (3.1)) with exact solution $y(t)$ is called stiff at t if

$$\mathrm{Re}\,(\lambda_i(J)) < 0, \qquad i = 1, \ldots, s \tag{i}$$

$$\kappa = \max_i \mathrm{Re}(-\lambda_i(J))/\min_i \mathrm{Re}\,(-\lambda_i(J)) \gg 1 \tag{ii}$$

where $\lambda_i(J)$, $i = 1, \ldots, s$ are the eigenvalues of $J(t, y(t))$ for a given t. Obviously this is too strict a definition. It is difficult to give a precise mathematical definition of the word stiffness. Let me give a rather informal definition

> A problem (eqn. (3.1)) is stiff for method M and tolerance ε when the step size selection is restricted by stability rather than accuracy.

Example 1. Let us consider the well-known Van der Poel equation

$$\begin{aligned} y_1' &= y_2 & y_1(0) &= 2 \\ y_2' &= -y_1 + 100(1 - y_1^2)y_2, & y_2(0) &= 0 \end{aligned}; \quad t \in [0, 100] \tag{3.2}$$

Figure 1 gives the value of $-y_2(t)$. For nearly all values of t it is zero. At $t = 81{\cdot}3$, $y_2(t)$ is nearly $-0{\cdot}1$. When t is 81·92, $y_2(t)$ is $-133{\cdot}7$ and at $t = 81{\cdot}95$ it is again back to nearly $-0{\cdot}2$. The first component y_1 has a rather smooth dependency on t.

The eigenvalues for (3·2) behave interestingly. They start as real $-300{\cdot}0$, $-0{\cdot}0033$ and according to the definition of Lambert the system is stiff initially. Later the largest eigenvalue becomes positive while the other remains negative. Just before the peak value in $y_2(t)$ they turn complex before they both stay real and negative.

Using a realistic definition of stiffness, (eqn. (3.21)) is a stiff system during most of the range of definition.

Is this a stiff system for Eulers method? In order to answer that question we apply Eulers method with local error tolerance of 10^{-3} i.e. norm of local error bounded by 10^{-3}. The stepsize selection is shown in Fig. 2.

For nearly the whole interval [0, 100] the step size is approximatively $0{\cdot}67 \cdot 10^{-2}$. The stepsize drops to nearly $0{\cdot}2 \cdot 10^{-4}$ close to the peakvalue at $t \approx 82$. The number of steps is close to 16000, in the peak just under 1000. If the accuracy controls the value of h, increasing the tolerance from 10^{-3} to 10^{-2} should increase h. However, the step variation is nearly unchanged. Only in the peak region is there an observable change. The step size is controlled by stability and not by accuracy.

Obviously this is as it should be. If we now use an A-stable Runge–Kutta method of order 3 (see Nørsett and Thomsen, 1984) and tolerance 10^{-2} we use 78

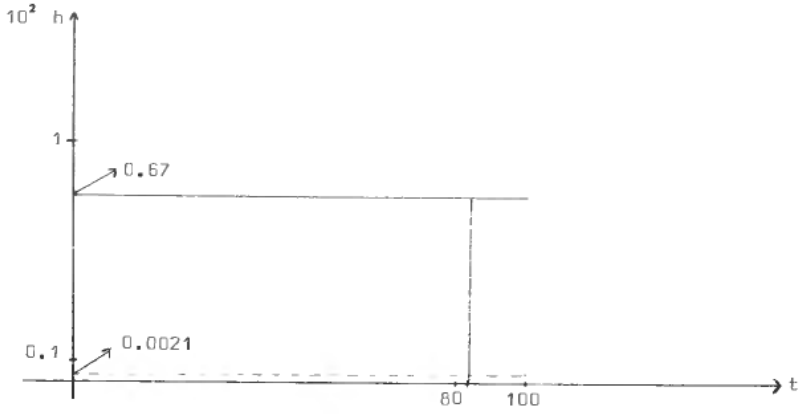

Figure 2. The step size values $100h$.

steps with the largest close to 25. Eighty per cent of the steps are taken in the peak area. When the tolerance is decreased to 10^{-3} the number of steps increases as it should when the accuracy is the controlling factor. In conclusion, the problem is stiff for the Euler method, but not for the given Runge–Kutta method. In short, we can say that stiffness is a problem for the forward Euler, but not for backward Euler (if it is implemented correctly).

The first programs with error control were based on predictor–corrector methods. Later the well-known method of Merson (see Lambert (1973)) was introduced and in fact is still in use today. They were all intended as general purpose solvers, both for non-stiff and stiff systems. Even today Merson's method is used on stiff problems. The users observe an inefficiency, but do not know that the system at hand is stiff.

All production codes are now either intended for non-stiff or for stiff systems. For each class there are three sub-groups of methods,

Linear multi-step methods	(LMM)
Runge–Kutta methods	(RK)
Extrapolation methods	(EXM)

Some good codes

Non-stiff systems:	DVERK (IMSL):	Runge–Kutta of order 5, 6
	RKF45:	Runge–Kutta of order 4, 5
	Merson (NAG):	Runge–Kutta of order 3, 4
	Adams (IMSL, NAG):	Adams-methods
	Diffex:	Extrapolation
Stiff systems:	Gear (NAG, IMSL): LSODE	Backward-differentiation
	SIMPLE:	Singly–diagonally implicit Runge–Kutta of order 3
	STRIDE:	Singly implicit Runge–Kutta
	ROW:	Rosenbrock type methods of order 4
	METAN:	Semi-implicit Runge–Kutta Extrapolation

The main problem is still not solved, how do we know that the problem is stiff?

Mode	eps	# steps	# FIX steps	error at end point	# F-eval	# Jac	# LU	# back solvers	CPU
FIX/MNI	10^{-3}	100	60	1·4(−3)	577	8	36	206	0·64
MNI	10^{-3}	95	0	6·2(−3)	523	8	52	499	0·72
FIX/MNI	10^{-4}	187	150	2·5(−3)	1172	4	36	261	1·15
MNI	10^{-4}	185	0	4·9(−4)	1172	4	71	1160	1·49

Table 1. Results for FIX/MNI: Switching and MNI: Modified Newton only. (# F-eval stands for # calls to righthand side of system).

The obvious remedy is always to use a stiff integrator. But the code will be inefficient on non-stiff systems. In order to tackle that problem we need codes that can change automatically from stiff to non-stiff or *vice versa*. Shampine (1981) calls this type-insensitive codes. His main idea is to use $\|\partial E/\partial y\|_\infty$ in deciding which mode to use. In Nørsett and Thomsen (1985) the ratio between the displacement norm and the redisual norm is used as a measure of stiffness. They use an implicit method and for non-stiff problems a fix-point iteration is used while for stiff systems, a modified Newton iteration.

Example 2. The Van der Poel equation (3.2) is integrated with the Runge–Kutta NTI of Nørsett and Thomsen (1984). The results for local tolerances 10^{-3} and 10^{-4} are shown in Table 1.

We can see that the switching technique is working and saves both the linear algebra involved and therefore the CPU-consumption.

The type insensitivity idea has been tested on both stiff and non-stiff systems. In all cases it works as expected. For example, some problems which are listed as non-stiff were reported stiff for very crude local tolerances. This is in full agreement with the given definition of a stiff system. These ideas will be built into future codes.

Most methods and codes are based on the assumption that the problem is smooth. However, there certainly are problems with discontinuous first derivatives. A simple example is the modeling of a furnace under temperature constraints, the temperature has to be between two limits. The codes will normally solve these equations, but inefficiently. We must hit the point of discontinuity as close as our tolerance admits.

Example 3. In Gear and Østerby (1984) the problem

$$y' = \begin{cases} 0 & x < 40{\cdot}33 \\ 100 & x \geqslant 40{\cdot}33 \end{cases}, \qquad y(0) = 40{\cdot}33$$

is solved by be the code GEAR of Hindmarsch 1974 with a relative error tolerance of 10^{-5}. In Fig. 3 the result is seen as the number of f-evaluations against the integration length. Much extra work is done. This is due to the fact that we do not hit the point of discontinuity in an efficient way.

As Example 3 indicates the existing codes handle problems with discontinuities inefficient and to some extent inaccurate. But a recent work by Enright, Jackson, Nørsett and Thomsen (1985) shows how these problems can be solved. The point is that we locally need a continuous approximation.

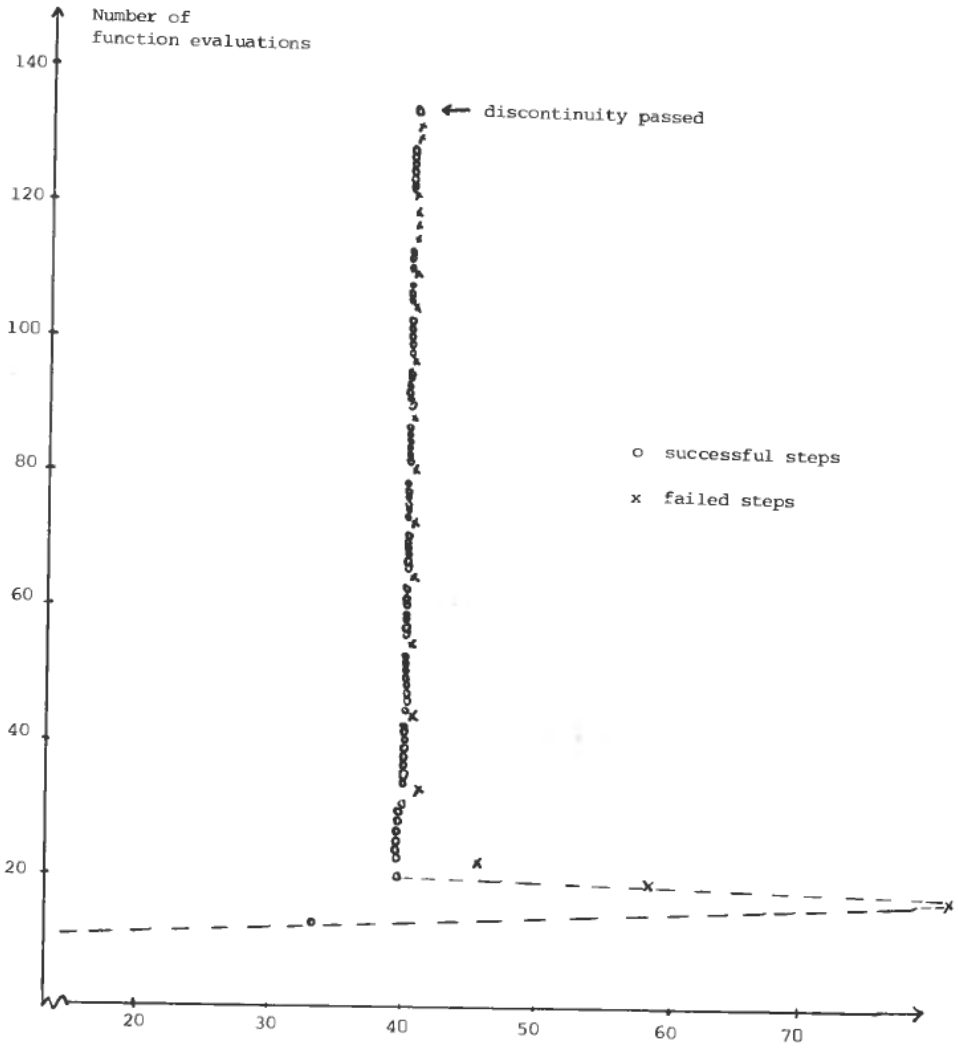

Figure 3. The behaviour of an ODE code at a discontinuity.

Example 4. The problem

$$y' = \begin{cases} y & \text{if status} = 0 \\ -y/2 & \text{if status} = 1 \end{cases}, \qquad x \in [0, 10]$$

$$y(0) = 1$$

with

$$g(x, y) = \begin{cases} y - 2 & \text{if status} = 0 \\ y - 1 & \text{if status} = 1 \end{cases}$$

and

status = 0 initially
status is reset to 0 if $g(x, y) \leqslant 0$ and status = 1
status is reset to 1 if $g(x, y) \geqslant 0$ and status = 0

$\log_{10}$ to 1	Standard approach			Interpolant approach with u_0^1			Interpolant approach with u_1		
	Error	Steps	f-Evals	Error	Steps	f-Evals	Error	Steps	f-Evals
−1	1·32	20	148	0·215	14	79	0·137	14	88
−2	7·85	37	317	0·321	20	109	0·362	20	118
−3	20·83	69	573	0·414	24	129	0·523	24	138
−4	33·75	133	1105	0·732	39	204	0·622	39	213
−5	54·96	175	1355	0·555	68	349	0·629	68	358
−6	26·86	240	1764	0·481	112	569	0·637	112	578
−7	32·75	374	2630	0·740	199	1004	0·647	199	1013
−8	38·99	569	3793	0·589	345	1734	0·655	345	1743

Table 2. Numerical results for the discontinuous problem.

Hence y is held between 1 and 2. This is a simple model for a furnace. There are 7 points of discontinuity in [0, 10]. The code DNORK based on a Runge–Kutta pair of orders 3 and 4 were used and the results are shown in Table 2.

The code performs well both with respect to efficiency and accuracy. Similar ideas are presented in Enright *et al.* (1985) for DVERK, RKF45 and other RK-methods in general.

4. Differential algebraic systems

Section 1 showed that differential/algebraic problems occur in a natural way. The question is rather, how do we solve such equations? For the present discussion we assume the system to be

$$\left.\begin{aligned} y' &= f(t, y, z), \quad y \in \mathbb{R}^r \\ 0 &= g(t, y, z), \quad z \in \mathbb{R}^s \end{aligned}\right\} \tag{3.1}$$

This form covers most practical problems, but in some cases the DAE will be given in fully implicit form

$$F(t, w', w) = 0, \qquad w = [y^T, z^T]^T.$$

Obviously this is a more complicated system, but the different difficulties can be seen by considering eqn. (3.1). As in Petzold (1982) let us consider the system

$$\begin{aligned} v_1 &= \varphi(t) \\ v_1' &= v_2 \\ v_2' &= v_3 . \end{aligned}$$

In matrix notation this takes the form

$$\begin{aligned} y' &= Ay + Bz, \qquad y = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\ 0 &= Cy - \varphi(t), \quad z = v_3 \end{aligned}$$

$$\text{with} \quad A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad C[1 \quad 0]$$

The simplest method for eqn. (3.1) could be the backward Euler Method,

$$y_{n+1} - y_n = hf(t_{n+1}, y_{n+1}, z_{n+1})$$
$$0 = g(t_{n+1}, y_{n+1}, z_{n+1})$$

In our example this turns out as

$$v_{1,n+1} = \varphi_{n+1}$$
$$v_{2,n+1} = \frac{1}{h}(v_{1,n+1} - v_{1,n})$$
$$v_{3,n+1} = \frac{1}{h}(v_{2,n+1} - v_{2,n})$$

Let us first remark that the exact solution is

$$v_1(t) = \varphi(t)$$
$$v_2(t) = \varphi'(t)$$
$$v_3(t) = \varphi''(t)$$

The first steps with backward Euler then give

(0)
$$v_{1,0} = \alpha$$
$$v_{2,0} = \beta$$
$$v_{3,0} = \gamma$$

(1)
$$v_{1,1} = \varphi(h)$$
$$v_{2,1} = \frac{1}{h}(\varphi(h) - \alpha)$$
$$v_{3,1} = \frac{1}{h}\left(\frac{1}{h}\varphi(h) - \frac{1}{h}\alpha - \beta\right) = \frac{1}{h^2}(\varphi(h) - \alpha - h\beta)$$

(2)
$$v_{1,2} = \varphi(2h)$$
$$v_{2,2} = \frac{1}{h}(\varphi(2h) - \varphi(h))$$
$$v_{3,2} = \frac{1}{h^2}(\varphi(2h) - 2\varphi(h) + \alpha)$$

(3)
$$v_{1,3} = \varphi(3h)$$
$$v_{2,3} = \frac{1}{h}(\varphi(3h) - \varphi(2h))$$
$$v_{3,3} = \frac{1}{h^2}(\varphi(3h) - 2\varphi(2h) + \varphi(h))$$

In fact for $n \geqslant 3$

$$v_{1,n} = \varphi(nh)$$

$$v_{2,n} = \frac{1}{h}(\varphi(nh) - \varphi((n-1)h))$$

$$v_{3,n} = \frac{1}{h^2}(\varphi(nh) - 2\varphi((n-1)h) + \varphi((n-2)h))$$

For the error we thus have: (except rounding errors)

$$\left.\begin{aligned} e_{1,n} &= v_{1,n} - v_1(t_n) = 0 \\ e_{2,n} &= v_{2,n} - v_2(t_n) = 0(h) \\ e_{3,n} &= v_{3,n} - v_3(t_n) = 0(h) \end{aligned}\right\} \quad n \geqslant 3$$

For

$$\left.\begin{aligned} n = 2: \quad & v_{3,n} && \text{diverges as } h \to 0 \\ n = 1: \quad & v_{3,n}, v_{2,n} && h \to 0 \\ n = 0: \quad & && \text{They all diverge } h \to 0 \end{aligned}\right\} \text{unless the correct initial values are given}$$

The question is then, how could we foresee this behaviour, so different from the normal ODE-case?

Gear and Petzold (1982) have discussed this problem and by considering the *index* of a problem, insight can be gained. For eqn. (3.1) we have

Index 1:

$$\frac{\partial g}{\partial z} \text{ is non-singular.}$$

When we differentiate the algebraic part of eq. (3.1) we get

$$0 = \frac{\partial g}{\partial t} + \frac{\partial g}{\partial y} \cdot f + \frac{\partial g}{\partial z} z'.$$

Hence a differential system results.

The BDF-methods for eqn. (3.1) are formulated as

$$\left.\begin{aligned} \sum_{i=0}^{k} \alpha_i y_{n+i} &= hf(t_{n+k}, y_{n+k}, z_{n+k}) \\ 0 &= g(t_{n+k}, y_{n+k}, z_{n+k}) \end{aligned}\right\} \tag{3.2}$$

From Gear and Petzold (1982)

> The error order of eqn. (3.2) for index 1 is $0(h^k)$, i.e. the order is as for normal ODE.

For the popular Runge–Kutta methods nothing is stated in the literature, but we obviously should have the same result. (This will be handled in a forthcoming thesis

at NTH.) The general class of m-stage Runge–Kutta methods can be defined by

$$\left.\begin{aligned} Y_i &= y_n + h\sum_{j=1}^{m} a_{ij} f(t_n + c_j h,\ Y_j,\ Z_j) \\ 0 &= g(t_n + c_i h,\ Y_i,\ Z_i) \\ y_{n+1} &= y_n + h\sum_{i=1}^{n} b_i f(t_n + c_i h,\ Y_i,\ Z_i) \\ 0 &= g(t_{n+1},\ y_{n+1},\ z_{n+1}) \end{aligned}\right\} \quad i = 1, \ldots, m \tag{3.3}$$

(Other definitions are possible and will be considered.)

When g_z is singular a higher index system results. Let us assume that $g_z = 0$. Then

$$0 = g_t + g_y \cdot f$$

and by differentiation

$$0 = g_{tt} + g_{ty} f + g_{yt} f + g_{yy} ff + g_y f_t + g_y f_y f + g_y f_z z'$$

Index 2: $g_y f_z$ is non-singular

Results on error behavior are in general not known.

In short, the number of times we have to differentiate the algebraic conditions in eqn. (3.1) in order to get a differential equation corresponds to the index of the system.

When we differentiate once the algebraic equation in our example we get

$$0 = \underbrace{CA}_{\underbrace{(0\ \ 1)}_{D}}y + \underbrace{CB}_{0}z - \varphi'$$

One more differentiation yields

$$0 = \underbrace{DA}_{0}y + \underbrace{DB}_{1}z - \varphi'' \Leftrightarrow 0 = z - \varphi''$$

and finally

$$z' = \varphi'''$$

The *index* is *3*.

For index higher than 2 little is known. Let me only refer to the paper of Gear and Petzold (1982) and also the paper by Løstedt and Petzold (1983).

References

Butcher, J. C. (1963). Coefficients for the study of Runge–Kutta integration processes. *J. Australian Math. Soc.* **3**, 185–201.

Courant, R., and Friedrichs, K. O. (1948). *Supersonic Flow and Shock Waves* (Springer-Verlag, New-York-Heidelberg-Berlin).

CURTISS, C. F., and HIRSCHFELDER, J. O. (1952). Integration of stiff equations. *Proc. Nat. Acad. Science, U.S.A.*, **38,** 235–243.

DAHLQUIST, G. (1956). Numerical integration of ordinary differential equations. *Math. Scandinavica*, **4,** 33–50.

DAHLQUIST, G. (1963). A special stability problem for linear multistep methods. *BIT*, **3,** 27–43.

ENRIGHT, W., JACKSON, K., NØRSETT, S. P., and THOMSON, P. G. (1985). Continuous Runge–Kutta methods for dense output and discontinuities. Report No. 180/85, Dept. of Computer Science, University of Toronto, Canada.

EULER (1913) Opera omnia, series prima Vol. II, Leipzig and Berlin.

GEAR, C. W., and PETZOLD, L. R. (1982). ODE Methods for the Solution of Differential/Algebraic Systems. Report No. SAND82-8051, Sandia National Laboratories, Livermore, CA, U.S.A.

GEAR, C. W., and ØSTERBY, O. (1984). Solving ordinary differential equations with discontinuities. *ACM Trans. Math. Software*, **10,** 23–44.

LAMBERT, J. D. (1973). *Computational Methods in Ordinary Differential Equations.* (John Wiley & Sons, New York.)

LØTEDT, P. and PETZOLD, L. R. (1983). Numerical Solution of Nonlinear Differential Equations with Algebraic Constraints. Report No. SAND83-8877, Sandia National Laboratories, Livermore, CA, U.S.A.

NØRSETT, S. P., and THOMSEN, P. G. (1984). Embedded SDIRK-methods of basic order three, *BIT*, **24,** 634–646.

NØRSETT, S. P., and THOMSEN, P. G. (1985). Switching between modified Newton and fixpoint iteration for implicit ODE-solvers. (Submitted BIT).

PETZOLD, L. R. (1982). Differential/algebraic equations are not ODEs. *SIAM J. Sci. Stat. Comput.*, **3,** 367–384.

SHAMPINE, L. F. (1981). Type insensitive ODE codes based on implicit A-stable formulas. *Math. Comp.*, **36,** 499–510.

SINCOREE, R. F., DEMBART, B., EPTON, M. A., MANKE, J. W., and YIP, E. L. (1979). Solvability of Large-Scale Descriptor Systems. Report Boeing Computer Services Company, Seattle, WA, U.S.A.