BIT 26 (1986), 100–113

LOCAL ERROR CONTROL IN *SDIRK*-METHODS

SYVERT P. NØRSETT and PER G. THOMSEN

Department of Numerical Mathematics,
Technical University of Norway,
N-7034 Trondheim, Norway.

Department for Numerical Analysis,
Technical University of Denmark,
DK-2800 Lyngby, Denmark.

Abstract.

This paper describes some problems that are encountered in the implementation of a class of Singly Diagonally Implicit Runge-Kutta (SDIRK) methods. The contribution to the local error from the local truncation error and the residual error from the algebraic systems involved are analysed. A section describes a special interpolation formula. This is used as a prediction stage in the iterative solution of the algebraic equations. A strategy for computing a starting stepsize is presented. The techniques are applied to numerical examples.

AMS MOS: 65L05, 65L20.

1. Introduction.

The explicit methods most often used for solving systems of ODE's

$$(1) \qquad y' = f(y), \qquad x \geqq a, \qquad y(a) = y_0 \in \mathbb{R}^s$$

have a finite region of absolute stability. When the system (1) is stiff the stability requirements will restrict the stepsize, and methods with unbounded stability domains are preferable. Only methods with some kind of implicitness will have this property, as stated by Lambert [7], "Although no precise result concerning all possible classes of methods exists (naturally!) it is certainly true that for all commonly used methods, explicitness is incompatible with infinite R".

When a Backward Difference Formula (BDF-method) or a SDIRK-method is used, it requires the solution of one or more systems of algebraic equations at each timestep. The algebraic equations are of the form

$$(2) \qquad v = \psi + \gamma h f(v); \qquad v \in \mathbb{R}^s$$

where ψ is computed from past information, h is the current stepsize and γ is a real positive parameter depending on the method being used. For some implicit RK-methods a more natural way of writing the algebraic system would be in

Received December 1984. Revised December 1985.

the traditional derivative-form

$$Z = hf(\psi+\gamma Z), \qquad Z = hf(v). \tag{3}$$

We advance the solution over a step using values of Z found as solutions to equations like (3). For that reason this form would often be preferred.

In Houbak, Nørsett and Thomsen [5] we addressed the problem of what criterion to use for stopping the modified Newton iterations

$$N(J)(v^{(i+1)}-v^{(i)}) = -R(v^i), \qquad i \geqq 0 \tag{4}$$

that are used to solve (2) or (3). We introduced the residual defined by

$$R(v) = v-\psi-\gamma hf(v) \tag{5}$$

(6) and the Newton matrix $N(J) = I-\gamma hJ$

where J is an approximate Jacobian matrix $\partial f/\partial y$ evaluated at some point of the numerical solution. One way to stop the iteration is by using a displacement test, i.e. when

$$\|\Delta_i\| = \|v^{(i+1)}-v^{(i)}\| < \tau \tag{7}$$

for some $i \geqq 0$, where r is a positive iteration tolerance. Another criterion for stopping the iterations (4) is the residual test, and requires that

$$\|R(v^{(i)})\| < \tau, \qquad i \geqq 0. \tag{8}$$

One of the conclusions in our previous work was that the displacement test was the appropriate one to use.

However, the iteration error is not the only error committed when we use a method to solve (1) numerically. The other major contribution is the local truncation error l_n (from the step x_n to x_{n+1}). In most existing codes the stepsize is chosen so that the estimated local error $\hat{l}_n$ satisfies

$$\|\hat{l}_n\| < \varepsilon \tag{9}$$

where ε is the *local error tolerance.*

In this paper we discuss the choice of τ and ε for Runge-Kutta methods. With $\tau = \kappa\varepsilon$ a value for κ is proposed for each given Runge-Kutta method, depending only on the coefficients of the method. In most present codes a very small value of κ is used. This is equivalent to forcing the iterations to satisfy a very strict condition. For the two methods considered here, however, the value of

κ is in the vicinity of 1 and this results in large savings in the number of iterations and thereby in the number of evaluations of the differential equation.

The implicit Runge-Kutta methods have no natural way of generating starting values for the iterations (4).

However, using an interpolation formula based on information from the most recent step, a starting value for the iteration of (4) is found. The derivation of this interpolation formula is discussed in section 3.

In the last section some more details regarding the actual implementation are taken up, details such as the balance between the adjustments to the stepsize based on (9) and the restrictions introduced to ensure convergence in the modified Newton Iterations (4), also how the starting stepsize may be chosen in an efficient way.

2. Solution of the algebraic equations.

Our study will be concentrated upon m-stage Runge-Kutta methods given by

$$(10) \qquad a) \quad Y_i = y_n + h \sum_{j=1}^{m} a_{ij} f(Y_j); \qquad i = 1,\dots,m;\ A = \{a_{ij}\}_{ij=1}^{m}$$

$$b) \quad y_{n+1} = y_n + h \sum_{i=1}^{m} b_i f(Y_i)$$

where y_n is an approximation to $y(x_n)$. An alternative way of writing the method is

$$(11) \qquad a) \quad k_i = hf(y_n + \sum_{j=1}^{m} a_{ij} k_j); \qquad i = 1,\dots,m$$

$$b) \quad y_{n+1} = y_n + \sum_{j=1}^{m} b_i k_i.$$

Although the following discussion is general, we will illustrate the discussion using the following two embedded SDIRK-methods from Nørsett and Thomsen [7].

$$(12) \qquad \begin{array}{c|ccc} 5/6 & 5/6 & & \\ 29/108 & -61/108 & 5/6 & \\ 1/6 & -23/183 & -33/61 & 5/6 \\ \hline b & 25/61 & 36/61 & 0 \\ \hline d & 26/61 & 324/671 & 1/11 \end{array}$$

Method NT I of order 3 with order 2 imbedded method for local error estimation. The order 3 method is B-stable.

$$
(13)\qquad
\begin{array}{c|cccc}
5/6 & 5/6 & & & \\
10/39 & -15/26 & 5/6 & & \\
0 & 215/54 & -130/27 & 5/6 & \\
1/6 & 4007/6075 & -31031/24300 & -133/2700 & 5/6 \\
\hline
b & 32/75 & 169/300 & 1/100 & 0 \\
\hline
d & 61/150 & 2197/2100 & 19/100 & -9/14
\end{array}
$$

Method NT II of order 3 with order 4 imbedded method for local error estimation. The order 3 method is A-stable.

The local error in each method is estimated by

$$
\begin{aligned}
(14)\qquad l_n &= h(d^T - b^T) \otimes [f(Y_1)^T, \ldots, f(Y_m)^T]^T \\
&= (d^T - b^T) \otimes [k_1^T, \ldots, k_m^T]^T.
\end{aligned}
$$

The solution of (10a) or (11a) is usually found by a modified Newton-method. Let $\hat{Y}_i$, $i = 1, \ldots, m$, be the computed solution to (10a) and $\hat{k}_i$, $i = 1, \ldots, m$, be the computed solution to (11a). When (11) is used we solve directly for the quantities that are used to calculate y_{n+1} from (11b). When (10) is used, we solve for $\hat{Y}_i$, $i = 1, \ldots, m$, but we need $f(\hat{Y}_i)$ to insert in (10b) for calculating y_{n+1}. Evaluating $f(\hat{Y}_i)$ costs one extra function evaluation per stage and furthermore, as pointed out by Shampine [11] it also amplifies in the solution when the system is stiff. The build-up errors can be avoided if we put

$$
(15)\qquad [\hat{k}_1^T, \ldots, \hat{k}_m^T]^T = h[f(\hat{Y}_1)^T, \ldots, f(\hat{Y}_m)^T]^T := A^{-1} \otimes \{[\hat{Y}_1^T, \ldots, \hat{Y}_m^T]^T - e \otimes y_n\}
$$

where $e = [1, \ldots, 1]^T \in R^m$.

This assignment corresponds to a $P(EC)^N$ scheme for linear multistep methods whereas using

$$
(16)\qquad \hat{k}_i := hf(\hat{Y}_i), \qquad i = 1, \ldots, m,
$$

Table 1. *Comparison between $P(EC)^N E$ and $P(EC)^N$ for stiff problems.*

METHOD	Local error ε	$\kappa = \tau/\varepsilon$	Fcn Calls		Steps		Problem
			$P(EC)^N E$	$P(EC)^N$	$P(EC)^N E$	$P(EC)^N$	
NT 1	10^{-4}	5.00	1139	92	129	14	D5
		0.50	815	170	56	15	
		0.10	290	201	23	16	
		0.01	280	189	16	18	
NT II	10^{-2}	0.67	1458	734	194	80	Van der Pool
		0.20	1266	838	117	78	
		0.10	1365	967	103	80	
		0.01	1267	1212	81	81	

would resemble $P(EC)^N E$ schemes. For nonstiff problems the $P(EC)^N E$ schemes are preferred over $P(EC)^N$ schemes but for stiff problems it turns out that (15) represents the correct way of obtaining the final function values. In order to illustrate the difference in behaviour the method (12) has been tested in the form of (10) using both (15) and (16). The results are shown in Table 1. They were obtained from the program SIMPLE; for details see [8].

Using method NT I in displacement mode and the form (10) we see that $P(EC)^N$ is more efficient than $P(EC)^N E$. The difference is very significant for large values of the parameter κ while it becomes less striking as it decreases. The reason for this behaviour is that a small κ will give smaller errors in $\hat{Y}_i$, $i = 1, \ldots, m$ and thus the influence from the last evaluation of the function $f(\tilde{Y}_i)$ in the $P(EC)^N E$ mode is less important, while for larger κ-values there can be a significant change. This is in full agreement with the observations of Shampine [11].

For the method NT II which uses a mixed displacement and residual mode in the formulation (10) the behaviour is similar. Here the difference between the $P(EC)^N E$ and $P(EC)^N$ modes is not as large as for NT I but the trend is in the same direction. The mixed displacement and residual mode is less sensitive to this phenomenon than the pure displacement mode. The reason for this will become apparent later.

As a general observation we remark that the global error for $P(EC)^N E$ was slightly larger than that for $P(EC)^N$-mode.

Since we are aiming at efficiency and reliability, we are interested in relatively large κ-values and we therefore recommend the use of $P(EC)^N$ mode for stiff problems.

We now address the problem of choosing between the forms (10) and (11). Let us define

$$Y = [Y_1^T, \ldots, Y_m^T]^T, \qquad k = [hk_1^T, \ldots, hk_m^T]^T$$

$$f(u) = [f(u_1)^T, \ldots, f(u_m)^T]^T, \qquad u = [u_1^T, \ldots, u_m^T]^T \in R^{ms}$$

Then (10) and (11) can be written as

$$\begin{cases} Y = e \otimes y_n + hA \otimes f(Y) & (17a) \\ y_{n+1} = y_n + hb^T \otimes f(Y) & (17b) \end{cases}$$

and

$$\begin{cases} k = hf(e \otimes y_n + A \otimes k) & (18a) \\ y_{n+1} = y_n + b^T \otimes k & (18b) \end{cases}$$

When the modified Newton method is used for solving (17a) or (18a) we get

$$N(J)(Y^{i+1}-Y^i) = -Y^i + e \otimes y_n + hA \otimes f(Y^i), \qquad i \geqq 0 \tag{19}$$

where $N(J)$ is defined as in (6), and

$$N(J)(K^{i+1}-K^i) = -K^i + hf(e \otimes y_n + A \otimes K^i), \qquad i \geqq 0 \tag{20}$$

where Y^i and K^i are the iterates obtained by the modified Newton process with Y^0 and K^0 as the starting values.

If we define V^i by

$$V^i = e \otimes y_n + A \otimes K^i, \qquad i \geqq 0 \tag{21}$$

we easily find $N(J)(V^{i+1}-V^i) = -V^i + e \otimes y_n + hA \otimes f(V^i), \qquad i \geqq 0.$

Hence if $Y^0 = e \otimes y_n + A \otimes K^0$ the processes (19) and (20) are consistent. Runs using NT I have shown that this is indeed the case, and local error tolerance $\varepsilon = 10^{-4}$ for the problem D5 gave the results shown in Table 2.

Table 2. *Comparison between Formulation* (19) *and* (20) *with consistent and inconsistent starting values.*

	Form. (18) with given Y^0	Form. (20) consistent with (19)	Form. (20) not consistent with (19)
# F-Eval.	170	173	189
# Steps	15	15	21
L_2 norm of the error at end point	6.2(−4)	6.5(−4)	8.8(−5)

Consider the relation $Y^{i+1}-Y^i = A \otimes (K^{i+1}-K^i), \quad i \geqq 1.$

According to this and depending on the value of $\|A\|$, small variations in the values produced by (19) or (20) may be present when the residual or displacement test is satisfied. The values of $\|A\|$ and $\|A^{-1}\|$ for the two methods considered are given in table 3.

Table 3. *Norms of the coefficient matrices for NT I and NT II.*

	NT I	NT II
$\|A\|_2$	1.2244	6.5724
$\|A^{-1}\|_2$	2.0997	7.5377

Based on this discussion it has been decided to settle for the $P(EC)^N$ mode using (10) in our implementation of SIMPLE.

3. Starting values for the modified Newton iterations.

For linear multistep methods, like BDF, it is an easy matter to obtain good starting values for the modified Newton process. An interpolation formula based on an appropriate number of previous solution values will provide an explicit predictor.

In RK-methods there are no previous solution values to use for interpolation, only the last accepted y-value. Let this be y_n, at the point x_n, and let the set of $\hat{Y}_i$- or $\hat{K}_i$-values be those used to calculate y_n.

Previous implementation like SIRKUS [9] and SPARKS [4], chose the accepted value directly, i.e., $Y^0 = e \otimes y_n$. This works fine in the cases where the solution does not change rapidly. On the other hand a more accurate prediction can be obtained using an interpolation formula based on the information that is available. Such an interpolation formula will also be useful for generating output values at non-step points. For explicit RK-methods such interpolation formulas have been described by Horn [3]. Addition of extra stages makes one able to find continuous ERK-methods with the same order as the basis method one lower than the basic method over the interval of integration.

For implicit RK-methods related formulae can be derived. For methods equivalent to collocation schemes (see [10]) this is simple. The interpolating polynomial is just the collocation polynomial and the order is $m+1$ for an m-stage method. This type of interpolation is used in the STRIDE package (see [1]).

For the two methods NT I and NT II, both of low order, it is easy to construct interpolation formulas, and the result for NT I is given by:
$y_{n+1}(\theta) \simeq y(x_n+\theta h), \qquad 0 \leqq \theta \leqq 1$:

$$y_{n+1}(\theta) = y_n + h \sum_{i=1}^{3} b_i(\theta) f(Y_i) \tag{22}$$

$$b_1(\theta) = \theta(29-141\theta+216\theta^2)/244$$

$$b_2(\theta) = \theta(-1620+5832\theta-3888\theta^2)/671$$

$$b_3(\theta) = \theta(145-357\theta+216\theta^2)/44.$$

The local error of (22) is $O(h^3)$ for $0 < \theta < 1$.

For NT II we obtain the result:

$$y_{n+1}(\theta) = y_n + h \sum_{i=1}^{3} b_i(\theta) f(Y_i) \tag{23}$$

$$b_1(\theta) = \theta(-100+220\theta+8\theta^2)/300$$

$$b_2(\theta) = \theta(325-130\theta-26\theta^2)/300$$

$$b_3(\theta) = \theta(75-90\theta+18\theta^2)/300$$

with local error $O(h^3)$ for $0 < \theta < 1$. The order $O(h^3)$ is acceptable here because the formulae are intended for the calculation of local output only.

The interpolation formula can be used as an extrapolation formula as well in order to obtain predicted values Y^0. For that purpose we use

$$Y_j^0 = y_{n+1}(1+(h/h_0)C_j), \qquad j = 1, 2, \ldots, m, \tag{24}$$

where h is the current stepsize, and h^0 the stepsize used in the previous step. This corresponds to $\theta = 1+(h/h_0)C_j$ in (22) or (23). The same type of idea for prediction was used in STRIDE by Burrage, Butcher and Chipman [1]. The interpolation predictor has been compared to the strategy of using the most recent y_n as Y_j^0.

For the problems D5 and the Van der Pool equation using NT I and NT II we obtain the results in Table 4–6.

Table 4. *Results when using interpolation-type predictor (A) and previous solution value (B). Method NT I.*

		# FCN Calls		# Steps		L_2-error at end point	
Problem	REPS = AEPS	A	B	A	B	A	B
D5	10^{-2}	53	36	15	10	8.6(−3)	1.3(−2)
	10^{-4}	91	89	15	14	1.4(−3)	4.3(−3)
	10^{-6}	387	515	50	49	3.5(−5)	7.2(−5)
Van der Pool	10^{-2}	337	322	77	67	1.3(−1)	9.7(−2)
	10^{-3}	559	700	98	99	9.4(−3)	8.6(−3)
	10^{-4}	1147	1528	176	164	1.7(−3)	1.8(−3)

Table 5. *Results when using interpolation-type Predictor (A) and previous solution value (B). Method NT II.*

	Local error tolerance	# Function Calls		# Steps		L_2-error at end point	
Problem	AEPS-REPS	A	B	A	B	A	B
D5	10^{-2}	55	72	12	12	1.8(−2)	2.6(−2)
	10^{-4}	199	180	16	17	3.0(−3)	2.7(−3)
	10^{-6}	582	676	54	53	8.1(−6)	9.3(−5)
VDP	10^{-2}	586	602	77	82	6.6(−3)	3.8(−3)
	10^{-3}	1098	1185	127	118	6.4(−4)	1.4(−3)
	10^{-4}	1701	1729	188	168	1.1(−3)	1.1(−3)

From the tables we conclude that the (A)-type prediction is the overall best way to obtain starting values. The only case when (B) is performing best is in the Van der Pool equation with REPS = AEPS = 10^{-2}. In Table 6 the position of the peak of the second solution component as computed in the same cases is shown. It is seen from these results that the case where the (B)-type prediction was most efficient gave a very bad position for the peak.

Table 6. *Position of peak value for the second component of the Van der Pool Solution as found by SIMPLE for different strategies.*

Method	NT I		NT II	
local error tolerance	A	B	A	B
10^{-2}	74.374	97.576	81.407	81.252
10^{-3}	81.218	81.690	81.218	81.207
10^{-4}	81.270	81.367	81.176	81.185

REMARK. The method NT II was run at first using an interpolation formula different from (23). However, this had bad interpolation properties as may be observed from the results in Table 7.

Table 7. *Results from NT II for problem D5 using alternative interpolation method.*

Local error tolerance	# Function ev.	# Steps	L_2-error at end point
10^{-2}	72	12	2.6(−2)
10^{-4}	180	17	2.7(−3)
10^{-8}	676	53	9.3(−5)

The local truncation error $T_{n+1}(\theta)$ of the interpolation formula (23) can be found as

$$T_{n+1}(\theta) = h^3 E_1(\theta) \cdot F(\curlyvee)(y_n) + O(h^4) \tag{25}$$

where $E_1(\theta) = (-50\theta^3 + 75\theta^2 - 25\theta)/312$ while the formula used to generate the results in Table 6 leads to

$$T_{n+1}(\theta) = h^3 E_2(\theta) \cdot F(\curlyvee)(y_n) + O(h^4). \tag{26}$$

Here $E_2(\theta) = (50\theta^3 - 75\theta^2 + 25\theta)/12$ and hence $E_2(\theta) = -26E_1(\theta)$, which explains why (23) is the better choice. We see that the conditions imposed on the interpolation formula must be selected carefully.

4. Iteration error tolerance in relation to local error tolerance.

Locally there are two types of errors committed, the local truncation error and the iteration error from the algebraic system. Each error is controlled locally. Usually the truncation error will be bounded by a user-defined local error tolerance ε while the iteration error is made small compared to ε, satisfying (8) for $\tau \ll \varepsilon$. In the programs SIRKUS, [9] and in SPARKS, [4], $\tau = \varepsilon/100$ was used.

In some cases iteration to convergence has been applied. This corresponds to $\tau \sim u$, where u is the unit round-off-error of the computer used. In their program STRIDE, Burrage, Butcher and Chipman [1] use a different approach. They estimate the number of iterations necessary to obtain a displacement error that satisfies (7) with $\tau = \varepsilon$.

What strategy is best and what value should be used for τ is, to quote Shampine [11] "a research question which needs attention. It is clear that τ must be smaller than $\varepsilon \cdots$. However, the smaller τ is made, the more it costs to compute y^*. Experiments say that τ a great deal smaller than ε does not improve the solutions of the differential equation".

We agree with most of this. But that we should need τ smaller than ε is not obvious and may not be correct. In fact this will depend on the method used.

In [5] the following relation between the exact local truncation error, the computed local truncation error and the iteration error is found

$$(27) \qquad \begin{aligned} \hat{l}_1 &\simeq l_1 + (b-a)^T A^{-1} \otimes N^{-1}(\hat{J})(R(\hat{Y}) \\ &\simeq l_1 + (b-a)^T A^{-1} \otimes N^{-1}(\hat{J})[hA \otimes (J-\hat{J})\Delta_i]. \end{aligned}$$

For most stiff problems $\|N^{-1}(\hat{J})\|$ is bounded by 1. Further $\|N^{-1}(\hat{J})[hA \otimes (J-\hat{J})]\|$ is an estimate for the rate-of-convergence of the modified Newton iteration and it must be smaller than 1 for convergence. The contribution to the local error from the algebraic system is then bounded by $\|(b-a)^T A^{-1}\|\tau$. It is seen that this contribution will depend on the coefficients of the method used. Since we are interested in controlling the total local contribution to the error we have chosen to use the following

$$(28) \qquad \|l_1\| < \varepsilon/2$$

$$(29) \qquad \|(b-a)^T A^{-1}\|\tau < \varepsilon/2.$$

Hence τ is defined by $\tau = \kappa\varepsilon$ where

$$(30) \qquad \kappa = (2\|(b-a)^T A^{-1}\|)^{-1}.$$

Values of κ for different methods are given in Table 8.

Table 8. *Values for κ. The methods* (7.6), (6.8) *and* (4.15) *refer to* [7]. *NMI is from* [9].

Method	κ, L_2-norm	κ, L_∞-norm
NT I	4.04	4.58
NT II	0.56	0.64
(7.6)	8.97	9.61
(6.8)	3.97	4.5
SIRKUS	0.17	0.21
NM I	0.11	0.15
(4.15)	0.065	0.077

For the method (4.15) the last stage is explicit. In Butcher notation it can be defined in the form

$$\begin{array}{c|cc} c & A & \\ \bar{c} & d^T & 0 \\ \hline & b^T & 0 \\ \hline & a^T & \bar{a} \end{array} \qquad \bar{c}, \bar{a} \in \mathbb{R}$$

In this case (27) becomes

$$\hat{l}_1 \simeq l_1 + (b-a)^T A^{-1} \otimes N^{-1}(\hat{J})R(\hat{Y}) + \bar{a} d^T A^{-1} \otimes (hA \otimes \hat{J}) N^{-1}(\hat{J}) R(\hat{Y}). \tag{31}$$

Hence residual test is recommended for this case and the value of κ is given by

$$\kappa^{-1} = 2(\|(b-a)^T\| + |\bar{a}| \cdot \|d^T A^{-1}\|). \tag{32}$$

A number of experiments have been carried out to give evidence to the above considerations. The two problems, D5 and the Van der Pool equation ($\varepsilon = 100$) have been used with different values for κ and error tolerances 10^{-4} and 10^{-3} respectively. The starting values in the iterations have been obtained by interpolation type prediction.

In the next two tables we give the data as follows

A:	Number of function evaluations
B:	Number of Steps
C:	Norm of the error at the end point
D:	The position of the peak in y_2 for the Van der Pool equation

Table 9. *NT II with different values for κ in $\tau = \kappa\varepsilon$. The norm used is L_2.*

κ	D5: A/B/C $\varepsilon = 10^{-4}$	Van der Pool: A/B/C/D $\varepsilon = 10^{-3}$
100	76/15/1.4(−2)	640/ 98/2.4(−2)/85.045
10	138/17/2.2(−3)	1025/119/3.2(−3)/81.161
5	159/16/3.4(−3)	1135/121/9.1(−4)/81.209
1	187/16/3.1(−3)	1210/113/5.6(−4)/81.198
0.5	200/16/3.9(−3)	1429/115/8.9(−4)/81.191
0.1	268/16/2.1(−3)	1448/112/1.1(−3)/81.180
0.01	283/16/7.3(−4)	1793/114/5.0(−4)/81.177

Table 10. *NT I with different values for κ in $\tau = \kappa\varepsilon$. The norm used is L_∞.*

κ	D5: A/B/C $\varepsilon = 10^{-4}$	Van der Pool: A/B/C/D $\varepsilon = 10^{-3}$
100	51/16/1.4(−2)	401/115/1.6(−2)/84.013
10	72/16/4.8(−3)	484/101/7.1(−3)/82.698
5	91/15/1.4(−3)	629/103/8.7(−3)/81.060
4	91/15/1.4(−3)	661/109/1.1(−2)/80.941
3	100/17/1.2(−3)	685/103/7.0(−3)/81.520
1	121/18/1.7(−3)	888/104/4.0(−3)/81.237
0.1	163/16/4.5(−5)	1194/102/1.8(−3)/81.098
0.01	209/15/4.6(−4)	1607/106/1.6(−3)/81.063

From the tables we draw the following conclusions:

1) Nothing is gained by making κ very small. On the other hand, when κ is too large the global error is affected. The reason is that when κ is large the iteration error is the dominant local contribution to the global error.

2) The stepsize is unaffected by the choice of κ. As κ decreases the number of function evaluations increases, meaning that each step involves more work.

3) We recommend the κ-values from table 6 but as seen from the results in Table 9 and 10 the exact value is not very critical for the performance.

The choice of starting stepsize is a problem that in most library routines is left for the user to supply. It is then expected that the routine will make adjustments based upon satisfying the local error tolerance in the first step. However, if the initial choice is outside the asymptotic region for the local error, the method will not give an appropriate reduction. The order will be zero rather than p for a pth order method. The result is a rejection of the first step maybe several times as illustrated in Table 11 where NT I has been used to solve problem D5 with $\varepsilon = 10^{-4}$ choosing an initial stepsize $h_0 = 0.1$.

Table 11. *NT I on D5*; $\varepsilon = 10^{-4}$; *ROC = Rate of Convergence*; *EFAC = Factor to modify the stepsize*: *EFAC* $> 0.8 \Rightarrow$ *accept*, *EFAC* $< 0.8 \Rightarrow$ *reject*, *H = the proposed stepsize.*

ROC	EFAC	H
0.02	0.48	0.0381
0.02	0.49	0.0148
0.02	0.51	0.0060
0.01	0.56	0.0027
0.05	0.68	0.0015
0.01	0.85	0.0015

As the table shows a total of 5 attempts had to be made. Each of them leads to almost the same value of EFAC. The order is not $p = 2$ as the control assumes but rather $p = 0$. If the order is assumed to be $p = 0$ one is led directly to the correct stepsize. However, in general this would be a rather bad idea if we happened to be inside the asymptotic region with the first guess for h. A strategy proposed by Hairer, Nørsett and Wanner [2] can be used for this purpose, the basic ideas being as follows:

Let the norm of the local error for the method be

$$E_L \approx C \cdot h^{p+1} \|\psi\| \tag{32}$$

where C is a characteristic error constant and ψ contains elementary differentials of order $p+1$. We can obtain a very rough but indicative estimate for $\|\psi\|$ by

$$\|\psi\| = \|y''\|^{(p+1)/2}. \tag{33}$$

This estimate it at least correct for the case $y' = \lambda y$. We can obtain an estimate of $y''(x_0)$ by

$$y''(x_0) = (d/dx) f(y(x_0)) \approx d^{-1}(f(y_0 + d f(y_0)) - f(y_0)) \tag{34}$$

where d is chosen as a multiple of the error constant for the computer in use. If the local error tolerance is ε we will obtain a value for the initial stepsize h_0 given by

$$h_0 = (\varepsilon/C)^{1/(p+1)} / \sqrt{\|y''\|}. \tag{35}$$

The initial point might be non-typical for the solution over the interval of interest and a step of length h_0 is taken using the forward Euler method. After this step another stepsize h_1 is estimated using the same strategy. The starting stepsize is then chosen as $h = min(h_0, h_1)$. The total cost is 4 function evaluations which is equivalent to the work in a normal step of a method of order 3 using one iteration in each stage.

This method of computing the starting stepsize has been implemented in SIMPLE and run on our favourite examples D5 and Van der Pool, and it was found that the estimated stepsize was accecpted in all cases but one. In the exceptional case a reduction by a factor of 3 was needed but this was inside the asymptotic range and thus acceptable. The other cases led to initial stepsizes that could be increased by factors in the range 1.1 to 3.0 after the first step.

REFERENCES

1. K. Burrage, J. C. Butcher and F. H. Chipman, *An implementation of singly diagonally implicit Runge-Kutta methods*, BIT, 20, (1980).
2. E. Hairer, S. P. Nørsett and G. Wanner: To appear.
3. M. K. Horn, *Fourth- and fifth-order, scaled Runge-Kutta algorithms for treating dense output*, SIAM J. Numer. Anal. 20, (1983).
4. N. Houbak and P. G. Thomsen, *SPARKS-A FORTRAN subroutine for the solution of large systems of stiff ODE's with sparse Jacobians*, NI-79-02, DTH, Lyngby, Denmark.
5. B. Houbak, S. P. Nørsett and P. G. Thomsen, *Displacement or residual test in the application of implicit methods for stiff problems*, NI-83-04 DTH, Lyngby, Denmark. (To appear JIMA, Numer. Anal.).
6. J. D. Lambert, *The initial value problem for ordinary differential equations: A survey*, in *The State of the Art in Numerical Analysis*, ed. D. Jacobs, Acad. Press, New York, (1977).
7. S. P. Nørsett and P. G. Thomsen, *Imbedded SDIRK-methods of basic order three*, Mathematics and Computation 8/82. NTH, Trondheim, Norway.
8. S. P. Nørsett and P. G. Thomsen, *SIMPLE – a stiff system solver* (to appear), (1984).
9. S. P. Nørsett, *Semi-explicit Runge-Kutta methods*, Report no. 6/1974, ISBN 82-7151-009-6, Dept. of Mathematics, University of Trondheim, Norway.
10. S. P. Nørsett and G. Wanner, *Perturbed collocation and Runge-Kutta methods*, Numer. Math. 38, (1981).
11. L. F. Shampine, *Implementation of implicit formulas for the solution of ODE's*, SIAM J. SCI. STAT. COMPUT., 1, (1980).
12. L. F. Shampine, *Evaluation of implicit formulas for the solution of ODE's*, BIT, 19 (1979).