TERMINATING SIMPLIFIED NEWTON ITERATIONS: A MODIFIED STRATEGY

Ronald Tshelametse*

ABSTRACT:

The paper deals with the numerical solution of IVPs for systems of stiff ODEs with particular emphasis on implicit linear multistep methods (LMM) particularly the backward differentiation formulae (BDF). In this paper we investigate the current strategies that are used to terminate the Newton iterations in the Matlab Code ode15s we analyse the algorithms for terminating the Newton iterations as implemented in the code ode15s. We thenmodify the existing termination strategy. Our numerical experiments reveal an improvement in terms of computational costs.

* Department of Mathematics, University of Botswana, Private Bag 0022, Gaborone, Botswana.

1 Introduction

The paper is concerned with the numerical solution of initial value problems (IVP) for systems of ordinary differential equations (ODEs). These are usuallywritten in the form

$$\frac{du}{dt} = f(t,u),\ \ 0 < t \le T, \qquad u(0) = u_0,\ \ f: R \times R^m \to R^m \qquad (1)$$

In the literature some initial value problems (1) are referred to as stiff. A prominent feature for these problems is that they are extremely difficult tosolve by standard explicit methods. The time integration of stiff systems isusually achieved using implicit methods, and for many codes by linear multistepmethods. A linear multistep method aims at producing a sequence of values$\{u_n\}$which approximates the true solution of the IVP on the discrete points$\{t_n\}$. Thus the linear multistep formula is a difference equation involving anumber of consecutive approximations $u_{n-i},\ i = 0,1, \dots, k$, from which it willbe able to compute sequentially the sequence $\{u_n\},\ n = 1,2, \dots, N$.Theinteger $k$is called the step number of the method and for a linear multistepmethod $k > 1$. When $k = 1$, the method is called a 1-step method. Linearmultistep methods are also called linear k-step methods [3], [5], [7], [8], [9]. Instandard constant stepsize form a linear multistep or k-step method is defined thus:

$$\sum_{i=0}^{k} \alpha_i u_{n-i} = h \sum_{i=0}^{k} \beta_i f_{n-i}\,, \qquad (2)$$

where $\alpha_i$and β_i are constants and α_0 = 1. f_{n-i}denotes $f(t_{n-i}, u_{n-i}),\ t_{n-i} = t_n - ih,\ i = 0,1, \dots, k$ and $h$is the stepsize. The condition that α_0 = 1removes the arbitrariness that arises from the fact that both sides of the IVPcould be multiplied by the same constant without altering the method. Thelinear multistep method (2) is said to be explicit if$\beta_0 = 0$and implicit if$\beta_0 \neq 0$.

Now let $\beta_i = 0, i = 1,2, \dots, k$in (2) then the result is a class of methods known as the backward differentiation formulae, BDFs [15]. We concentrate onBDFs which take the form

$$u_n + \sum_{i=1}^{k} \alpha_i u_{n-i} = h_n \beta_0 f(u_n) = 0, \qquad (3)$$

Whereh_n is the stepsize, $k$is the order and the coefficients $\alpha_i$depend on k only. In practice codes for integrating stiff IVPs vary the stepsize $h_n$and/or order $k$resulting in variable step variable order BDF implementations [1], [4], [13], [17],[23]. At each integration step $t_n$we must solve the nonlinear equation

$$F(u_n) \equiv u_n + \varphi_n - h_n \beta_0 f(u_n) = 0, \qquad (4)$$

where $\varphi_n = \sum_{i=1}^{k} \alpha_i\, u_{n-i}$is a known value.

To solve for $u_n$most codes use the Newton iterative method and its variantsin the following form

$$W_n^{(l)}\varepsilon_n^{(l)} = -F\left(u_n^{(l)}\right), \qquad u_n^{(l+1)} = u_n^{(l)} + \varepsilon_n^{(l)} \qquad l = 0,12,\ldots \qquad (5)$$

with the starting value $u_n^{(0)}$known and "fairly" accurate. For the full Newtonmethod

$$W_n^{(l)} = F'\left(u_n^{(l)}\right) = I - h_h\beta_0 f'\left(u_n^{(l)}\right) \qquad (6)$$

The use of the Newton method is due to the stiffness phenomenon. For largeproblems evaluating the Jacobian, $f'(u_n^{(l)})$ (and hence the Newton iteration matrix $W_n^{(l)}$) and solving the linear algebraic system are by far the most computationally expensive operations in the integration. There are various strategiesused in practice to try and minimise the cost of computing the Jacobian and theNewton matrix [4], [6], [14], [18]. These measures are mainly centredonadministering the iteration matrix in (6). Other cost saving measures in practical codes include options of using analytical or finite difference Jacobians and at times taking advantage of special structures (banded or sparse) for the linearsolves described by (5) and (6).

Despite being useful in the error analysis of linear multistep methods the truelocal truncation error is rarely used in practical codes. Instead it is estimatedusing some kind of error estimators, the most common being the Milne errorestimator.

We conduct our numerical experiments using a code from the Matlab odesuite [20] known as ode15s [19]. The code is a variable step variable order code and integrates stiff initial value ordinary differential equations. In this codethere is an option to use either some modified BDFs or use the standard BDFsas correctors. The iteration is started with a predicted value

$$u_n^{(0)} = \sum_{m=0}^{k} \nabla^m u_{n-1}$$

where ∇ denotes the backward di¤erence operator. This is the backward difference form of the interpolating polynomial which matches the back values,$u_{n-1}, u_{n-2}, \ldots, u_{n-k-1}$and then is evaluated at t_n. The code implementssome form of the Milne error estimator. Most of the test problems used can befound in the Matlab ode suite [20] .

We conduct our experiments using the default weighted infinity norm. Wefurther evaluate global errors to determine whether the modified code yields the same solution as theoriginal code.

2 Current termination strategies

2.1The underlying theory

In solving the IVP

$$\frac{du}{dt} = f(t,u), \quad 0 < t \le T, \qquad u(0) = u_0, \quad f: R \times R^m \to R^m \qquad (7)$$

these codes compute an approximate solution, u_n, of the implicit equation

$$F(u_n) \equiv u_n + \varphi_n - h_n\beta_0 f(t_n, u_n) = 0, \qquad (8)$$

to satisfy, in principle

$$\left||u_n^* - u_n|\right| \leq k_1 \times tol, \qquad (9)$$

where $tol$is a user specified tolerance, k_1 is a constant usually less than unityand u_n^*denotes the true solution of (8). Alternatively, a test will be to accept$u_n$if the residual satisfies

$$\left||F(u_n)|\right| \leq k_1 \times tol. \qquad (10)$$

In practice most codes that use iterative methods to solve the implicit equations(8) accept the approximation when

$$\left\|u_n^{(l)} - u_n^{(l-1)}\right\| \leq k_1 \times tol, \qquad (11)$$

where $u_n^{(l)}$and $u_n^{(l-1)}$, $l = 0,1,2,$ ...are the successive iterates, or

$$\left\|F\left(u_n^{(l)}\right)\right\| \leq k_1 \times tol. \qquad (12)$$

where $u_n^{(l)}$is the current iterate. The tests are called the displacement test andthe residual test respectively. Houbak et al [11] conduct a comparative studyand reveal that it often takes more computational work to satisfy (12) than (11)with little or no gain in the accuracy of the numerical solution of the associatedinitial value problem.

We are mainly interested in how to terminate the iterations

$$W_n\left(u_n^{(l+1)} - u_n^{(l)}\right) = -F\left(u_n^{(l)}\right), \qquad l = 0,1,2, \ldots \qquad (13)$$

where $W_n$is an approximation to the Newton iteration matrix, in order to obtain a good approximation, $u_n^{(l+1)}$ to the solution u_n^*. It is common practice toterminate the iterations based on the norm of the difference,

$$d_l = \left\|u_n^{l+1} - u_n^{(l)}\right\|$$

alone. The iterations are terminated as soon as d_l is small enough, but Shampine [21] argued that a small difference $d_l$says nothing about how close $u_n^{(l+1)}$is tou_n^*, nor even that the iteration process is converging. But if the convergencerate factor of the iterative process$\eta < 1$ then a small difference $d_l$implies that$u_n^{(l+1)}$is an acceptable approximation tou_n^*. This is discussed in [2, p612] whereit is shown that under appropriate the assumptions and for $u_n^{(0)}$ a sufficientlygood approximations to the solution u_n^*, then the simplified Newton

methodconverges linearly, that is, $u_n^{(l)} \to u_n^*$with factor $\eta \in (0,1)$. In the stiff ODE applications $u_n^{(0)}$is generally a good approximation.

It can be shown that

$$\left\|u_n^{(l+1)} - u_n^{(l)}\right\| \le \|\tilde{G}\|\left\|u_n^{(l)} - u_n^{(l-1)}\right\|$$

where $G(u) = u - W^{-1}F(u)$, $u^{(l+1)} = G(u^{(l)})$ and $W$is the simplifiedNewton Iteration matrix. Now assume throughout the region of interest that$\|\tilde{G}\|$is bounded above by someη. It is hoped that $\eta < 1$. In fact for the infinitynorm if $\eta < 1$then by a theorem in [12, p111] the iterates u_n^lconverge to thetrue solution u_n^*if $u_n^{(0)}$is sufficiently close to u_n^*. A similar theorem is discussedin [16, p119] for the general norm. There follows

$$\left\|u_n^{(l+1)} - u_n^{(l)}\right\| \le \eta\left\|u_n^{(l)} - u_n^{(l-1)}\right\| \qquad (14)$$

and so at the time $u_n^{(l+1)}$is computed,ηcan be estimated as

$$\eta^{(l)} = \left\|u_n^{(l+1)} - u_n^{(l)}\right\| / \left\|u_n^{(l)} - u_n^{(l-1)}\right\| \qquad (15)$$

Now applying the triangle inequality to

$$u_n^{(l+1)} - u_n^* = (u_n^{(l+1)} - u_n^{(l+2)}) + (u_n^{(l+2)} - u_n^{(l+3)}) + \cdots$$

We get

$$\begin{aligned}\left\|u_n^{(l+1)} - u_n^*\right\| &\le \left\|(u_n^{(l+1)} - u_n^{(l+2)})\right\| + \left\|u_n^{(l+2)} - u_n^{(l+3)}\right\| + \cdots \\ &\le \eta\left\|(u_n^{(l+1)} - u_n^{(l)})\right\| + \eta^2\left\|(u_n^{(l+1)} - u_n^{(l)})\right\| + \cdots \\ &\le \frac{\eta}{1-\eta}\left\|(u_n^{(l+1)} - u_n^{(l)})\right\| \qquad (16)\end{aligned}$$

It is clear that the iteration error should not be larger than the required tolerance. Therefore the iteration can be stopped when

$$\frac{\eta^{(l)}}{1-\eta^{(l)}}\left\|(u_n^{(l+1)} - u_n^{(l)})\right\| \le k.tol, \qquad (17)$$

Where $k > 0$ is a suitable constant and $(u_n^{(l+1)})$ accepted as u_n, an approximationu_n^*,where$\eta^{(l)}$is given by (15). It is clear from (15) that at least two iterationsare required to estimate the rate of convergence and hence apply (17). The rateof convergenceη at the previousintegration step can be approximated by takingthere the largest observed$\eta^{(l)}$. This can then be used to judge if in the currentstep, the iterate after 1 (one) Newton iteration, $u_n^{(1)}$is acceptable. Note that therate at the previous integration step, is only applicable to the current step if thesolution and the factor $h_n\beta_0$remain much unchanged. This is further discussedby Shampine in [21] and Hairer and Wanner in [10, pp119-121].

The estimate of the convergence rate,$\eta^{(l)}$,at the current iterate is also used todecide when to terminate the Newton iterations. If $\eta^{(l)} > \delta$for some,$\delta < 1$,thenthe iteration is regarded as being too slowly convergent and is then terminated and restarted with a different $u_n^{(0)}$obtained using a different stepsize/order andpossibly an updated Jacobian matrix. In practice we set the maximum numberof iterations, $maxit$If the number is reached before the iteration convergesthen the iterations are terminated and the process restarted.

2.2The pseudo code

In ode15s the Newton iterations are terminated as follows. We give each optiona case number for ease of discussion.

- IF $\|(u_n^{(l+1)} - u_n^{(l)})\| \leq 100 * eps$where $eps$is the machine epsilon, the current $u_n^{(l+1)}$is accepted. This is a typical (relative) displacement test. All norms areweightednorms (CASE1).
- ELSE IF the current iteration is the first (CASE2)
  - IF the convergence rate, η_{n-1} from the previous step is available. That is, if the current time step is not the first time step, then the first iterate $u_n^{(1)}$ is accepted If

$$\frac{\eta_{n-1}}{1-\eta_{n-1}}\|(u_n^{(1)} - u_n^{(0)})\| \leq 0.05 \times rtol, \qquad (18)$$

    where $rtol$is now a scalar (CASE2(A)).
  - ELSE the convergence rate is set to zero $(\eta_0 = 0)$ (CASE2(B)).
  - ENDIF
- ELSE IF the convergence rate at the current iterate

$$\eta^{(l)} = \frac{\|u_n^{(l+1)} - u_n^{(l)}\|}{\|u_n^{(l)} - u_n^{(l-1)}\|} > 0.9 \qquad (19)$$

  then the iteration is regarded as too slow and is terminated and restarted with a different$u_n^{(0)}$obtained using a different stepsize/order and possibly an updated Jacobian matrix(CASE3).
- ELSE the convergence rate at the current time step is set to

$$\eta_n = max[0.9 * \eta_n, \eta_n^{(l)}]$$

  (CASE4) and
  - IF

$$\frac{\eta_n}{1-\eta_n}\|(u_n^{(l+1)} - u_n^{(l)})\| \leq 0.5 \times rtol \qquad (20)$$

    then the iterate $u_n^{(L+1)}$is accepted. Note the test (18) is more stringent than (20)because we are using the old rate, η_{n-1} (CASE4(A)).

- ELSEIF the iteration has reached the maximum allowed iterations, $l+1$ then it is regarded as too slow and restarted η_{n-1} (CASE4(B)).
- ELSE IF

$$0.5 * tol < \left[\frac{\eta_n}{1-\eta_n}\left\|(u_n^{(l+1)} - u_n^{(l)})\right\|\right]\eta_n^{(maxit-(l+1))} \qquad (21)$$

the iteration is also regarded as too slow and restarted (CASE4(C)). This is because the size of $\left\|(u_n^{(maxit)} - u_n^*)\right\|$ after l iterations can be estimated by

$$\frac{\eta_n^{(maxit-l)}}{1-\eta_n}\left\|(u_n^{(l+1)} - u_n^{(l)})\right\|$$

see Hairer and Wanner [10].

- ENDIF

ENDIF

The norms are either the weighted 2-norm or the weighted in infinity normwith the weights

$$\frac{1.}{\max\left(|u_{n-1}|, \left|u_n^{(l)}\right|, \frac{atol.}{rtol}\right)}, \quad l = 0,1,2,\ldots$$

where for a vector $x$the notation $1./x$ denotes a vector of reciprocals of eachelement x_i, $i = 1,2,\ldots,m$,that is a vector whose elements are $1/x_i$and $|x|$denotes a vector whose elements are $|x_i|$. The parameters $atol$and$rtol$arethe user supplied vectors of absolute and relative tolerances respectively and$atol./rtol$is a vector whose elements are$atol_i/rtol_i$. Note that the weights donot depend upon $atol$and$rtol$unless the magnitude of the elements of u_{n-1} and $u_n^{(l)}$are small compared to $ol./rtol$, referred to as the threshold. The defaultvalue of the threshold is $\frac{10^{-6}}{10^{-3}} = 10^{-3}$.

3. Terminating simplified Newton iterations: A new strategy

3.1 The underlying theory

We apply the theory of Dorsselaer and Spijker [24] to justify a strategy forstopping the Newton iteration in ode codes. The ideas are tested on ode15s.The proposed termination strategy is based on the theory of Dorsselaer andSpijker [24] and Spijker[22] where for particular classes of problems the errorcommitted by stopping the Newton iterations in the numerical solution of stiff IVPs is estimated and related to the local discretization error of the underlyinglinear multistep method. The theory is such that if we assume the initial guessu_0 satisfies

$$\left\| u_n^* - u_n^{(0)} \right\| = O(h^q), \qquad (22)$$

then ([22, Theorem 2.4] for mildly nonlinear problems and [24, Theorem 4.3]for highly nonlinear problems, the relation (22), with a moderate O-constant K,imply that

$$\left\| u_n^* - u_n^{(l)} \right\| = O\left(h^{Q(l)}\right); \qquad Q(l) = l + q, \;\; l \geq 1,$$

for the simplified Newton method. Both the above classes of the highly andmildly nonlinear problems are known to be dissipative, that is the logarithmicnorm of the Jacobian matrixsatisfies

$$\theta = \mu[f'(u)] \leq 0,$$

for $u$in the domain of f. Furthermore according to Dorsselaer and Spijker [24, Remark 2.3, p.188] if the differential equation and the inequality occurring inthe dissipativity condition is replaced by $u'(t) = f\left(u(t)\right)$ and

$$\left\| \tilde{V}(t_1) - V(t_1) \right\| \leq e^{w(t_1 - t_0)} \left\| \tilde{V}(t_0) - V(t_0) \right\|, \qquad w \in R.$$

That is

$$\theta = \mu[f'(u)] \leq w,$$

for $u$in the domain of f, then the theory is still valid. The value of $w$isproblem dependent and could be di¢ cult to establish in practice. Hence we usethe strategy without checking how dissipative the test problem is.

3.2 The proposed strategy

In practical codes the intention is to compute an approximate solution, u_n, at each time step,t_n of the implicit equation

$$F(u_n) \equiv u_n + \varphi_n - h_n \beta_0 f(u_n) \; = 0, \qquad (23)$$

to satisfy, in principle

$$\|u_n^* - u_n\| \leq k.tol, \qquad (24)$$

Where,$tol$is a user specified tolerance, $k$is a constant usually less than unityand $u_n^*$is the true solution. Since the true solution $u_n^*$is generally unknownthe codes commonly use the estimate of the local truncation error to decidewhether to acceptu_n as a good approximation to the true solution. The localtruncation error is usually estimated using the Milne error estimator, in whichthe computed approximation $u_n$is accepted if

$$C\left\|u_n - u_n^{(0)}\right\| \leq k_1.tol, \qquad (25)$$

where,$u_n^{(0)}$is the predicted solution and $C$is the Milne error coefficient. This isusually referred to as the local error test.

The common practice is to solve the nonlinear equation (23) as accuratelyas possible and then use the local error test (25) to check if the nearly exactsolution of the nonlinear system, u_n , can be accepted as a good approximation tothe true solution. Our opinion is that a lot of computations are wasted in tryingto solve the nonlinear system as accurately as possible (until the iterative solverconverges) which could result in 'oversolving'. We propose a strategy which issuch that we terminate the iterations as soon as the current iterate $u_n^{(l)}$satisfiesthe local error test

$$C\left\|u_n^{(l)} - u_n^{(0)}\right\| \leq k_2.tol, \qquad (26)$$

without necessarily satisfying the Newton convergence test and regardless of the number of iterations performed. Of course, if the iterative solveris too slow in converging to the solution of the nonlinear system, or the numberof maximum allowed iterations is reached, then the iterative process should bestopped and appropriate action taken, see section 2.2. Note that iterative solvercan converge to a solution which does not satisfy the local error test. Furthermore note that our proposed strategy reduces to the current implementation ifthe current iterate $u_n^{(l)}$in (26) can also pass the Newton convergence test.

We note with concern that in most practical codes the stepsize, $h_{n+1}$to beused at the next time step, $t_{n+1}$is evaluated by using the local error estimatefrom the previous successful time step, say

$$E_n = C\left\|u_n^{(l)} - u_n^{(0)}\right\|.$$

Hence terminating the Newton iterations before full convergence may seriouslycontaminate the error estimate and thereby destroy the overall stepsize selectionstrategy. The safe use of the new termination strategy might require a morerobust stepsize strategy than the one currently in ode15s.

In our proposed implementation we do not always use the converged Newton solution $u_n^{(l)} \sim u_n$in the Milne error test, but in the derivation of the Milneerror estimator the solution $u_n$of the nonlinear system is used. Instead we usethe Newton approximation after$l$iterations, $u_n^{(l)}$where, $= 1,2, \ldots$. This couldrender the use of the Milne error estimate as not applicable in our strategy.Now, note that

$$u_n = u_n^{(l)} + \epsilon_l(h)$$

Where,$\epsilon_l(h) = O(h^{(Q(l)})$where, $Q(l) = l + M$is independent of stiffness and$M = p + 1$, [24]. For $l = 1, Q(l) = p + 2$.This implies that the Milne errorestimate is still valid for $l = 1$, and hence for any $l > 1$. In the case of ode15s,the code estimates the leading term of the local truncation error via,

$$\frac{1}{p+1}h_n^{p+1}u^{(p+1)}(t_n) \approx \frac{1}{p+1}\nabla^{p+1}u_n. \qquad (27)$$

In the code this is implemented as a Milne error test with Milne coefficients as$1/(k+1)$where $k$is the order of the BDF.

Suppose now that after l iteration of the simplified Newton process, andbefore the test for convergence, the local error test is carried out. If the errortest is satisfied, there may be other possibilities, for example

(P.1) accept $u_n^{(l)}$without checking convergence,

(P.2) carry out further Newton iterations until convergence and accept the step.

Using possibility (P.2) will only recover all the computational costs our strategy is trying to save. Note that if the local error test fails, the iterative processis continued.

Our strategy is implemented alongside the other termination strategies inthe original ode15s as depicted in Figure 3.2.

The strategy used in ode15s is shown in Figure 1 (excluding the dottedline part), also see the pseudo code in section 2.2. Our modifications are shownby the addition of dotted lines in Figure 3.2, where by convergence rate test we are referring to the test, if

$$\frac{\eta}{1-\eta}\left\|u_n^{(l)} - u_n^{(l-1)}\right\| \leq k * rtol,$$

then accept the Newton iterate as u_n, where for the first iteration the test ismore strict, $k = 0.05$since $\eta$is the convergence rate factor from the previousstep and $k = 0.5$ for any other iterate. By $iter$we denote the iteration counter,$hmin$, denotes the smallest possible stepsize and $maxit$is the maximum numberof allowed iterations. The quantities$ynew$and $yold$are two successive Newtoniterates and $\varepsilon$is the Newton correction.

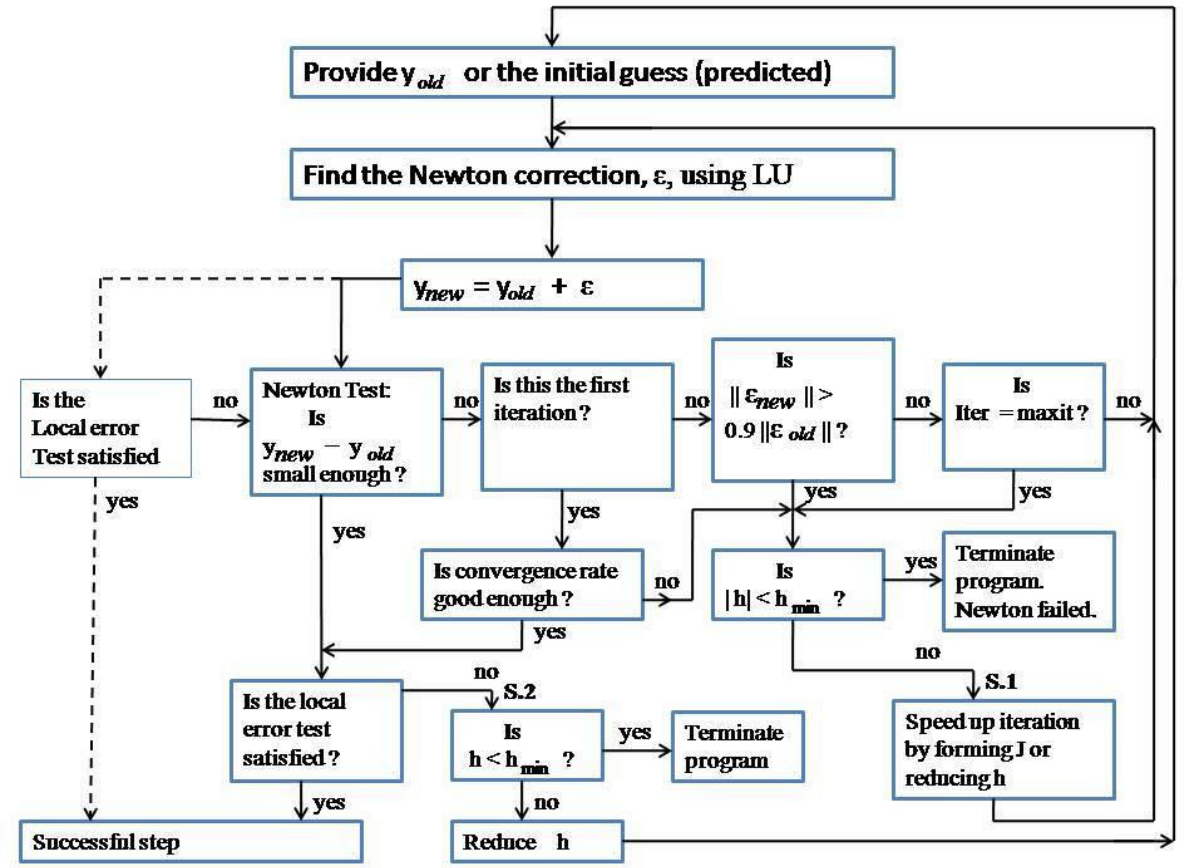

Figure 1: The modified strategy alongside the existing research

4 Numerical Experiments

We incorporate our strategy into ode15s and call the resulting code newode15s.We use various test problems including those in the Matlab ODE suite [19]. Itshould be noted that some of the test problems used in our experiments mightnot satisfy the basic assumptions (a.1) and (a.2) discussed earlier. The statisticsobtained are in Table 1 for the linear test problems, Table 2 for test problemswhich are known to be dissipative and Table 2 (low accuracy), Table 4 (mediumaccuracy), Table 5 (high accuracy) for the nonlinear test problems. To verifythat our new strategy does not yield wrong solutions compared to those obtainedusing ode15s (code A) we compared the infinity norms of the maximum global errors over all the specified output points of the solutions obtained using themodified code, newode15s (code B) to those obtained using ode15s. We also compare the maximum weighted infinity norm of the global errors over all the specified output points.

The theory of Dorsselaer and Spijker [24] is valid when the asymptotic condition, as $h \to 0$ holds. We investigate the significance of this asymptoticcondition, by taking a closer look at ds1ode It has been shown in Dorssalaerand Spijker [24] that the test problem ds1ode satisfy the assumptions (a.1) and(a.2). We force both codes to use smaller stepsizes by demanding more accuracyvia the use of $rtol$and $atol$. In both codes the local error $e_i$in $u_i$isestimatedin each step and made to satisfy

$$|e_i| \leq rtol|u_i| + atol_i, \qquad i = 1,2,\dots,m,$$

and by default $rtol = 10^{-3}$ and $atol_i$ $\quad i = 1,2,\dots,m,$ Test problem ds1ode is ideal forthis investigation as it has only one solution and satisfies the assumptions (a.1)and (a.2). The results obtained are in Table 2.

Test Problem	Code	Time Steps	Failed Steps	f evals	$\partial f/\partial y$ Evals	LUs	Linear Solves	Flop Count	$\|GE\|_\infty$	$\|GE\|_{w\infty}$
Low accuracy solution: $atol = 10^{-6}$, $rtol = 10^{-3}$										
a2ode	A	119	0	154	1	23	142	121599	9.722e-5	0.575
	B	119	0	131	1	23	119	115022	9.7.7e-5	0.576
a3ode	A	138	0	282	1	25	276	81953	0.0033	0.5448
	B	138	0	144	1	25	138	69623	0.0033	0.5448
b5ode	A	1243	140	2774	1	335	2766	676899	0.0131	482.95
	B	1243	140	1531	1	335	1523	497381	0.0131	482.95
Fem1ode	A	57	9	164	1	23	113	2987466*	0.0225	200.86
	B	79	18	167	1	36	116	4154929	0.0224	168.92
Fem2ode	A	49	12	167	5	21	117	104542*	0.0037	10.782
	B	92	21	201	5	36	151	143741	0.0038	11.202
Hb3ode	A	806	201	1863	1	325	1860	247876	4.770e-30	0.227
	B	761	190	1144	1	314	1141	220740	4.776e-30	0.228
Medium accuracy solutions: $atol = 10^{-6}$, $rtol - 10^{-6}$										
a2ode	A	208	1	277	1	35	265	208309	6.7609e-7	0.0717
	B	208	1	222	1	35	210	192530	6.7600e-7	0.0682
a3ode	A	226	2	462	1	39	456	132381	1.0163e-5	0.0034
	B	226	2	236	1	39	230	112287	1.0163e-5	0.0034
b5ode	A	2827	1	5664	1	235	5656	1289519	3.2047e-4	81.7048
	B	2827	1	2837	1	235	2829	876791	3.2047e-4	81.7048
Fem1ode	A	411	101	770	1	201	719	22653031	0.0013	32.3111
	B	399	98	647	1	195	596	21366984	0.0012	42.6499
Fem2ode	A	74	13	222	6	25	162	141393*	6.6488e-6	0.3020
	B	143	28	276	5	48	226	203546	6.6855e-6	2.0950
Hb3ode	A	781	195	1794	1	324	1791	245255	2.3552e-30	0.1122
	B	782	199	1183	1	326	1180	227851	3.1542e-30	0.1504
High accuracy solution: $atol = 10^{-9}$, $rtol = 10^{-7}$										
a2ode	A	278	3	357	1	47	344	271587	1.332e-7	0.0556
	B	278	3	297	1	47	284	254603	1.313e-7	0.0548
a3ode	A	330	5	676	1	56	670	180291	3.087e-7	1.624e-4
	B	330	5	346	1	56	340	150971	3.087e-7	1.624e-4
b5ode	A	6052	2	12116	1	404	12108	2713944	5.672e-5	15.680
	B	6052	2	6064	1	404	6056	1830380	5.672e-5	15.680
Fem1ode	A	1231	323	2232	1	626	2181	6966142	3.0938e-4	10.098
	B	1219	321	1913	1	621	1862	6723222	3.102e-4	11.243
Fem2ode	A	87	12	256	7	27	186	160991*	7.438e-7	0.039
	B	181	25	322	6	56	262	240599	7.459e-7	0.388
Hb3ode	A	780	201	1814	1	330	1811	245596	8.834e-30	0.421
	B	774	207	1191	1	334	1188	230511	5.127e-30	0.245

Table 1: Statistics obtained for linear test problems for ode15s (code A) andnewode15s (code B). The test problems, except fem2ode, have constant Jacobians. The starred(*) entries are for test problems where the modified code performs more flops.

Low accuracy solution: $atol = 10^{-6},\ rtol = 10^{-3}$										
Test Problem	Code	Time Steps	Failed Steps	f evals	$\partial f/\partial y$ Evals	LUs	Linear Solves	Flop Count	$\|GE\|_\infty$	$\|GE\|_{w\infty}$
ds1ode	A	52	0	75	1	14	73	22634*	4.972e-7	4.967e-4
	B	60	0	62	1	16	60	23420	1.835e-4	1.834e-2
ds2ode	A	144	4	258	2	31	256	44062*	7.808e-4	4.457
	B	180	12	212	2	45	210	52341	9.960e-4	5.116
ds4ode	A	143	96	391	38	133	389	82571	4.119e-4	0.523
	B	107	35	196	17	66	194	48181	2.410e-3	4.350
fem1ode	A	57	9	164	1	23	113	2987466*	0.0225	200.86
	B	79	18	167	1	36	116	4154929	0.0224	168.92
will1ode	A	117	0	194	1	26	141	3652725	1.069e-6	0.0038
	B	117	0	170	1	26	117	3483056	1.021e-6	0.0036
Medium accuracy solutions: $atol = 10^{-6},\quad rtol - 10^{-6}$										
ds1ode	A	110	0	134	1	22	132	41218*	1.0692e-10	1.0682e-07
	B	125	1	129	1	27	127	43583	1.8782e-07	1.8763e-04
ds2ode	A	227	6	404	2	45	402	63887	3.8673e-06	0.3850
	B	250	10	273	2	53	271	63875	4.3663e-06	0.3713
ds4ode	A	156	59	354	29	91	352	73706	3.3804e-06	0.0798
	B	154	26	233	11	59	231	59062	5.8989e-05	0.0994
fem1ode	A	411	101	770	1	201	719	22653031	0.0013	32.3111
	B	399	98	647	1	195	596	21366984	0.0012	42.6499
will1ode	A	225	1	321	1	38	268	5818749	2.0356e-08	7.1613e-05
	B	224	1	279	1	38	226	5522525	1.9905e-08	7.0091e-05
High accuracy solution: $atol = 10^{-9},\quad rtol = 10^{-7}$										
ds1ode	A	138	2	171	1	28	169	50271*	9.0049e-11	8.996e-8
	B	163	4	174	1	36	172	5440	1.921e-08	1.920e-5
ds2ode	A	322	8	612	2	61	610	88793	4.349e-7	0.0546
	B	315	10	341	2	62	339	76754	4.405e-7	0.0526
ds4ode	A	174	44	381	21	76	379	73872	4.387e-7	0.0116
	B	159	14	220	6	42	218	53172	6.971e-6	0.0119
fem1ode	A	1231	323	2232	1	626	2181	6966142	3.0938e-4	10.098
	B	1219	321	1913	1	621	1862	6723222	3.102e-4	11.243
will1ode	A	301	2	429	1	49	376	7674510	1.008e-9	3.547e-6
	B	301	2	358	1	49	305	7170928	1.170e-9	4.168e-6

Table 2: Statistics obtained for the test problems which are known to be dissipative for ode15s (code A) and newode15s (code B). The starred (*) entriesare for test problems where the modified code performs more flops.

Test Problem	Code	Time Steps	Failed Steps	f evals	$\partial f/\partial y$ Evals	LUs	Linear Solves	Flop Count	$\|GE\|_\infty$	$\|GE\|_{w\infty}$
buiode	A	66	6	129	2	18	122	36002	4.071e-4	1.108
	B	68	6	110	1	19	106	35881	4.125e-4	1.039
brussode	A	93	8	373	2	24	170	24498222	0.0104	4.3730
	B	112	6	241	1	27	139	23686772	0.0280	11.6329
chm6ode	A	170	2	227	2	35	216	71681*	0.676	0.657
	B	182	5	208	2	39	197	72967	1.272	1.522
chm7ode	A	55	0	71	1	12	67	20811	1.657e-4	0.596
	B	55	0	59	1	12	55	20132	1.703e-4	0.596
chm9ode	A	866	254	2572	81	366	2247	463398*	1.154e5	4.007e7
	B	1447	458	2731	56	711	2506	685062	1.157e5	5.445e7
d1ode	A	69	8	133	4	21	120	75165*	0.0181	0.762
	B	108	16	154	3	36	144	86974	0.0691	17.193
ds1ode	A	52	0	75	1	14	73	22634*	4.972e-7	4.967e-4
	B	60	0	62	1	16	60	23420	1.835e-4	1.834e-2
ds2ode	A	144	4	258	2	31	256	44062*	7.808e-4	4.457
	B	180	12	212	2	45	210	52341	9.960e-4	5.116
ds4ode	A	144	96	391	38	133	389	82571	4.119e-4	0.523
	B	180	35	196	17	66	194	48181	2.410e-3	4.350
gearode	A	19	1	33	2	6	26	23695	1.658e-4	0.252
	B	19	0	23	1	6	19	23260	1.659e-4	0.252
hb1ode	A	212	20	442	11	63	396	83788*	5.709er-5	2.528
	B	320	33	451	6	109	455	1103232	2.317e-4	9.020
hb2ode	A	578	49	793	2	116	786	158812	1.680e8	0.330
	B	573	49	676	1	115	672	151932	1.868e8	0.645
vdpode	A	823	854	2250	39	331	2132	338972*	3.157	2.103e3
	B	973	303	1749	35	444	1643	372079	3.100	2.103e3
willode	A	117	0	194	1	26	141	3652725	1.069e-6	0.0038
	B	117	0	170	1	26	117	3483056	1.021e-6	0.0036

Table 3: Statistics obtained for ode15s (code A) and newode15s (code B) for thenonlinear test problems at low accuracy. The starred (*) entriesare for test problems where the modified code performs more flops.

Test Problem	Code	Time Steps	Failed Steps	f evals	$\partial f/\partial y$ Evals	LUs	Linear Solves	Flop Count	$\|GE\|_\infty$	$\|GE\|_{w\infty}$
buiode	A	86	1	122	2	16	115	45609	7.0345e-7	0.0791
	B	85	0	89	1	15	85	43201	2.8035e-6	2.7956
brussode	A	232	8	532	1	38	430	38145947	5.1709e-5	0.0238
	B	237	10	361	1	40	259	35677605	6.0365e-5	0.0290
chm6ode	A	283	2	398	2	47	387	113025*	9.7389e-4	0.3407
	B	375	12	433	2	76	422	137989	98.003e-4	1.4270
chm7ode	A	105	1	132	1	20	128	31817	2.0935e-6	0.0063
	B	105	1	111	1	20	107	30815	2.1044e-6	0.0063
chm9ode	A	2384	298	5559	67	520	5290	968658*	1.1208e3	55.4688
	B	5112	927	7947	60	1608	7706	1909656	0.8677e3	12.3361
d1ode	A	175	16	390	3	43	380	115190*	9.2574e-5	0.3084
	B	583	93	858	3	176	848	226236	1.1.61e-4	0.3084
ds1ode	A	110	0	134	1	22	132	41218*	1.0692e-10	1.0682e-7
	B	125	1	129	1	27	127	43583	1.8782e-7	1.8763e-4
ds2ode	A	227	6	404	2	45	402	63887	3.8673e-6	0.3850
	B	250	10	273	2	53	271	63875	4.3663e-6	0.3713
ds4ode	A	156	59	354	29	91	352	73706	3.3804e-6	0.0798
	B	154	26	233	11	59	231	59062	5.8989e-5	0.0994
gearode	A	35	0	49	1	10	45	35345	1.5144e-7	1.5663e-4
	B	35	0	39	1	10	135	34976	2.1741e-7	3.6314e-4
hb1ode	A	335	26	642	16	81	576	123055*	2.3624e-6	0.8697
	B	668	96	1071	8	231	1037	242138	2.0614e-5	9.3996
hb2ode	A	1365	41	2395	2	134	2388	331713	7.1225e5	0.0025
	B	1361	39	1443	1	131	1439	279043	2.0069e6	0.0025
vdpode	A	1651	283	3854	24	393	3781	534215*	0.0017	46.8094
	B	1928	312	2745	26	487	2666	539846	0.0015	41.4994
willode	A	225	1	321	1	38	268	5818749	2.0356e-8	7.1613e-5
	B	224	1	279	1	38	226	5522525	1.9905e-8	7.0091e-5

Table 4: Statistics obtained for the nonlinear test problems for the originalode15s (A) and the newode15s (B) at medium accuracy solutions.The starred (*) entries are for test problems where the modified code performsmore flops.

Test Problem	Code	Time Steps	Failed Steps	f evals	$\partial f/\partial y$ Evals	LUs	Linear Solves	Flop Count	$\|GE\|_\infty$	$\|GE\|_{w\infty}$
Buiode	A	115	2	170	2	21	163	57691	1.135e-7	0.0791
	B	118	0	122	1	20	118	55510	3.582e-6	2.7956
Brussode	A	344	10	712	1	51	610	51295624	6.065e-6	0.0238
	B	344	10	466	1	51	364	45780199	8.645e-6	0.0290
chm6ode	A	686	20	1470	2	115	1459	317963*	6.320e-6	0.3407
	B	1378	67	1722	3	284	1706	491702	1.374e-4	1.4270
chm7ode	A	142	2	178	1	27	174	40312	3.220e-7	0.0063
	B	142	2	150	1	27	146	38978	3.220e-7	0.0063
chm9ode	A	5989	494	11960	66	1000	11695	2111245*	25.663	55.4688
	B	27341	3158	38198	67	6496	37929	8424935	36.164	12.3361
d1ode	A	403	10	1019	2	93	1012	200903*	3.036e-7	0.3084
	B	973	144	1384	3	290	1374	337027	1.520e-6	0.3084
ds1ode	A	138	2	171	1	28	169	50271*	9.005e-11	1.0682e-7
	B	163	4	174	1	36	172	54440	1.921e-8	1.8763e-4
ds2ode	A	322	8	612	2	61	610	88793	4.349e-7	0.3850
	B	315	10	341	2	62	339	76754	4.405e-7	0.3713
ds4ode	A	174	44	381	21	76	379	73872	4.387e-7	0.0798
	B	159	14	220	6	42	218	53172	6.971e-6	0.0994
Gearode	A	46	0	65	1	11	61	43976	1.514e-7	1.5663e-4
	B	45	0	50	1	11	46	43200	2.174e-7	3.6314e-4
hb1ode	A	451	27	840	15	96	778	159399*	2.177e-7	0.8697
	B	851	124	1335	10	282	1293	308995	1.149e-6	9.3996
hb2ode	A	3523	49	6098	2	250	6091	799092	5.917e3	0.0025
	B	3545	56	3666	2	263	3659	676348	3.114e3	0.0025
Vdpode	A	3342	326	6093	31	515	5999	846205*	1.008e-9	46.8094
	B	4235	510	5508	30	852	5417	1026181	1.170e-9	41.4994
Willode	A	301	2	429	1	49	376	7674510	1.008e-9	7.1613e-5
	B	301	2	358	1	49	305	7170928	1.170e-9	7.0091e-5

Table 5: Statistics obtained for the nonlinear test problems for the originalode15s (A) and the newode15s (B) at high accuracy.The starred (*) entries are for test problems where the modified code performs more flops.

5Conclusions

The error norms in our tables for the various test problems confirm that the solutions obtained using newode15s have similar accuracy to the solutions obtainedusing ode15s. There are a very small number of exceptions to this statement. Of particular interest in the results are the total number of linear algebraicequations solves (linear solves), that is the number of forward and backwardsubstitutions and the total number of function evaluations. For test problemssatisfying the assumptions of the theory of Dorsselaer and Spijker [24] and wherethe Jacobian is analytically computed, we expect the Newton iterations to beterminated after 1 (one) iterations. For the general test problems we anticipatethat the new termination strategy will reduce the number of Newton iterations,i.e. forward and backward substitutions and to reduce the number of functionevaluations.

The theory is trivially illustrated by the statistics obtained for the linear testproblems, a2ode, a3ode and b5ode (Table 1) These problems have analytical(constant) Jacobians. In most cases only 1 Newton iteration per time step isrequired to satisfy our termination strategy and there is a significant reductionin linear solves and function evaluations. This is clear from Table 6 which showspercentage reductions in linear solves, function evaluations and flop counts forthe test problems a2ode, a3ode and b5ode for all levels of accuracy.

Test problem: a2ode

					Percentage savings		
Level of Accuracy	Code	Func evals	Linear solves	Flop Count	Func evals	Linear solves	Flop count
Low	A B	154 131	142 119	121599 115022	14.94	16.20	5.41
Medium	A B	277 222	265 210	208309 192530	19.86	20.75	7.57
High	A B	357 297	344 284	271587 254603	16.81	17.44	6.25
Test problem: a3ode							
					Percentage savings		
Level of Accuracy	Code	Func evals	Linear solves	Flop Count	Func evals	Linear solves	Flop count
Low	A B	282 144	276 138	81953 69623	48.94	50.00	15.05
Medium	A B	462 236	456 230	132381 112287	48.92	49.56	15.18
High	A B	676 346	670 340	180291 150971	48.82	49.25	16.26
Test problem: b5ode							
					Percentage savings		
Level of Accuracy	Code	Func evals	Linear solves	Flop Count	Func evals	Linear solves	Flop count
Low	A B	2774 1531	2706 1523	676899 497381	44.81	43.72	26.62
Medium	A B	5664 2837	5656 2829	1289519 876791	49.91	49.98	32.01
High	A B	12116 6064	12108 6056	2713944 1830380	49.95	49.98	32.56

Table 6: Percentage savings of function evaluations, linear solves and flop counts due to the modified code for linear test problems a2ode, a3ode and b5ode.

We notice that for linear test problems Table 1, the problem a2ode, a3odeand b5ode, have constant Jacobians, so the simplified Newton iteration willconverge in one Newton iteration. However in ode15s, the convergence testsare based on, initially, the predicted value $u_n^{(0)}$. Hence sometimes two iterations(at most) will be required to satisfy the convergence test.

For the large, linear, constant Jacobian test problem fem1ode, we note thatfor low accuracy solutions the modified code performs more flops. This could beattributed to a possible collapse in the step size strategy for the modified code.But as the accuracy is increased (medium and high accuracy) the modified codeperforms less steps. This shows the significance of the asymptotic condition,as,$h \to 0$. For the linear, time-dependent Jacobian test problem fem2ode the higher flop count of the modified code for all levels of accuracy is mainly due to the collapse in the stepsize. Notethat there is no guarantee that a collapsed stepsize strategy will be stabilisedby the asymptotic condition (i.e. by demanding higher accuracy).

The theory is also verified (despite the Jacobian matrix being computednumerically) by the large, dissipative, nonlinear test problem will1ode For thistest problem only one(1) Newton iteration is required at each integration stepas predicted by the theory. Also see Table 2 for dissipative test problems. Thesavings in function evaluations and linear solves are significant as indicated inTable 7 that shows percentage savings for this test problem.

Test Problem: will1ode

					Percentage savings		
Level of Accuracy	Code	Func evals	Linear solves	Flop Calculation	Func evals	Linear solves	Flop count
Low	A B	194 170	141 117	3652725 3483056	12.37	17.02	4.64
Medium	A B	321 279	268 226	5818749 5522525	13.08	15.67	5.09
High	A B	429 358	376 305	7674510 7170928	16.55	18.88	6.56

Table 7: Percentage savings of function evaluations, linear solves and flop counts due to the modified code for the nonlinear test problem will1ode.

We also note a general agreement with the theory for some of the nonlineartest problems despite the fact that most Jacobians are computed numericallyand their dissipativity is not known. This is shown by the test problems chm7odeand gearode Table 8 shows the percentage savings.

It is also interesting to note that for some test problems the 'crash' in thestep size strategy leads to reduced time steps and hence an overall reduction in flop counts. Test problems showing this behaviour include ds4ode, hb2ode andhb3ode. In fact for hb2ode and hb3ode the step size behaviour for the modifiedcode almost follow the same curve as the original code.

We therefore conclude that the new termination strategy can lead to significant computational savings when used in solving stiff systems of ODEs withanalytical (constant) Jacobians, particularly if the system is large and the function f is computationally expensive to evaluate.

Test Problem: chm7ode							
					Percentage savings		
Level of Accuracy	Code	Func evals	Linear solves	Flop Calculation	Func evals	Linear solves	Flop count
Low	A B	71 59	67 55	20811 20132	16.90	17.91	3.26
Medium	A B	132 111	128 107	31817 30815	15.91	16.41	3.15
High	A B	178 150	174 146	40312 38978	15.73	16.09	3.31

Test Problem: gearode							
					Percentage savings		
Level of Accuracy	Code	Func evals	Linear solves	Flop Calculation	Func evals	Linear solves	Flop count
Low	A B	33 23	26 19	23695 23260	30.30	26.92	1.84
Medium	A B	49 39	45 35	35345 34976	20.41	22.22	1.04
High	A B	65 50	61 46	43976 43200	23.08	24.59	1.76

Table 8: Percentage savings of function evaluations, linear solves and flop countsdue to the modified code for the nonlinear test problems chm7ode and gearode.

The results of investigating the significance of the asymptotic condition,as,$h \to 0$, on the test problem ds1ode further reveal that our new terminationstrategy is ideal for high accuracy solutions As we demand more accuracy the flop counts for the modified code are reduced relative to the flop counts of the original code. This is also illustrated by the steady increase in the percentage savings in flop counts as higher accuracy is demanded for test problems a3odeopand b5ode in Table 8 and will1ode in Table 7.

Despite the general agreement with the theory for some of the test problems,we observe that our new strategy leads to an increase in computational costs forsome test problems as indicated by stars in our Tables. For the starred problems we observe that the modified code (B) performs more LU decompositions thanthe original code (A). This is mainly because of the increased number of timesteps possibly due to the contaminated stepsize. Increased number of time stepsleads to an increase in the number of LU factorizations performed and hence anincrease in the overall computational cost. It must be noted that for some testproblems the reduction in linear solves and function evaluations outweighs theincrease in number of time steps (and hence LU decompositions) resulting inlower computational costs for the modified code. This can be seen in the large,nonlinear test problem, brussode. Also note that for brussode, the stepsizestrategy stabilises as accuracy is increased (see Table 5).

The safe use of the new termination strategy requires a more robust step sizestrategy than the one currently in ode15s. The stepsize strategy in the modified code (adopted from the original code) is not robust enough to handle theperturbations in the error estimate due to the new stopping criterion. It shouldalso be noted that some of the test problems used in our experiments might notsatisfy the basic assumptions (a.1) and (a.2) discussed earlier.

6 Suggestions for future research

It has been established that the new stopping criterion can significantly reduces the number of function evaluations and forward and backward substitutions.As demonstrated by some test problems it is clear that a more robust stepsizestrategy is required if the new termination strategy is to be safely used. Thiswould significantly reduce the number of time steps taken and hence the LUfactorizations.

A possible way of minimizing the contamination of the predicted stepsize isto use not only the error at the previous step E_n, but to use in some way, say bythe process of extrapolation or interpolation, the previous error estimates E_{n-i}, $i = 0,1,2,\ldots$.to estimate the error at the next time step and hence predict thestepsize h_n.

References

[1]. Peter N. Brown, George D. Byrne, and Alan C. Hindmarsh. VODE: a variable-coefficient ODE solver. SIAM J. Sci. Stat. Comput., 10, No. 5:1038–1051, September 1989.

[2]. Peter N. Brown and Alan C. Hindmarsh. Matrix-free methods for stiff systems of ODE' SIAM J. Numer. Anal., 23, No. 3:610–s.638, June 1986.

[3]. John C. Butcher. Numerical methods for ordinary differential equations.John Wiley, 2003.

[4]. G. D. Byrne and A. C. Hindmarsh. A polyalgorithm for the numericalsolution of ordinary deferential equation. Comm. ACM, 1, No. 1:71–96,March 1975.

[5]. W. H. Enright, T. E. Hull, and B. Linberg. Comparing numerical methodsfor stiff systems of ODEs. BIT, 15:10– 1975.48.

[6]. G. Gheri and P.Marzulli. Parallel shooting with error estimate for increasing the accuracy. J. Comput. and Appl. Math., 115, Issues 1-2:213–227,March 2000.

[7]. E. Hairer. Backward error analysis for linear multistep methods. Numer.Math., 84:2:199–232, 1999.

[8]. E. Hairer. Conjugate-symplecticity of a linear multistep methods. J. Computational Mathematics, 26:5:657–659, 2008.

[9]. E. Hairer and C. Lubich. Symmetric multistep methods over long times. Numer. Math., 97:4:699–723, 2004.

[10]. E. Hairer and G. Wanner. Solving Ordinary Differential Equations II – Stiff and Algebraic Problems. Springer Verlag, Berlin, Germany, second revisededition, 1996.

[11]. N. Houbak, S. P. Nørsett, and P. G. Thomsen. Displacement or residual testin the application of implicit methods for stiff problems. IMA J. Numer.Anal., 5:297–305, 1985.

[12]. E. Isaacson and H. B. Keller. Analysis of numerical methods. John Wileyand Sons, 1966.

[13]. Kenneth R. Jackson. The numerical solution of stiff IVPs for ODEs. J.Applied Numerical Mathematics, 1995.

[14]. C. T. Kelley. Iterative methods for linear and nonlinear equations. Societyfor Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995.

[15]. J. D. Lambert. Numerical Methods for Ordinary Differential Systems. JohnWiley and Sons, 1991.

[16]. James M. Ortega and W. C. Rheinboldt. Iterative solution of nonlinearequations in several variables. Academic Press, London, England, 1970.

[17]. L. F. Shampine and P. Bogacki. The effect of changing the step size inthe linear multistep codes. SIAM J. Sci. Stat. Comput., 10:1010–1023,September 1989.

[18]. Lawrence F. Shampine. Numerical solution of ordinary differential equations. Chapman and Hall, 1994.

[19]. Lawrence F. Shampine and Mark W. Reichelt.The MATLABODE suite code ode15s.(from ftp.mathworks.com in the directorypub/mathworks/toolbox/matlab/funfun), 1997.

[20]. Lawrence F. Shampine and Mark W. Reichelt. The MATLAB ODE suite. SIAM J. Sci. Stat. Comput., 18, No. 1:1– January 1997.

[21]. L.F. Shampine. Implementation of implicit formulas for the solution ODEs.SIAM J. Sci. Stat. Comput., 1, No.1:103–118, March 1980.

[22]. M. N. Spijker. The effect of the stopping of the Newton iteration in implicitlinear multistep methods. J. Applied Numerical Mathematics, 18:367–386,1995.

[23]. Peter Tischer. A new order selection strategy for ordinary differential equation solvers. SIAM J. Sci. Stat. Comput., 10:1024–1037, September 1989.

[24]. J. L. M. van Dorsselaer and M. N. Spijker. The error committed by stoppingthe Newton iteration in the numerical solution of stiff initial value problems.IMA J. Numer. Anal., 14:183–209, 1994.