Volume 4 (2009) 13 - 27

Southern Africa Journal of Pure and Applied Mathematics

SAJPAM
ISSN:1993-8594
www.samsajournal.org

The Milne error estimator for stiff problems

Ronald Tshelametse
Department of Mathematics
University of Botswana
Private Bag 0022
Gaborone, Botswana.
E-mail tshelame@mopipi.ub.bw

May 29, 2009

ABSTRACT: The paper is concerned with the numerical solution of IVP's for systems of stiff ODE's. Our emphasis is on implicit linear multistep methods (LMM), particularly the backward differentiation formulae (BDF). The local truncation error is usually estimated using the Milne error estimator. In this paper we derive a modified Milne error estimator for stiff problems. We further conduct numerical experiments to show that using the modified Milne error estimates in solving stiff odes leads to improved computational statistics.

1 Introduction

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

$$\frac{du}{dt} = f(t,u), \quad 0 < t \leq T, \quad u(0) = u_0, \quad f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m. \tag{1}$$

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 to solve by standard explicit methods. The time integration of stiff systems is usually achieved using implicit methods, and for many codes by linear multistep methods. 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 a number of consecutive approximations $u_{n-i}, \quad i = 0, 1, \ldots, k$, from which it will be able to compute sequentially the sequence $\{u_n\}, \quad n = 1, 2, \ldots, N$. The integer k is called the step number of the method and for a linear multistep method $k > 1$. When $k = 1$ the method is called a 1-step method. Linear

multistep methods are also called linear k-step methods [2], [4], [6], [7], [8]. In standard constant stepsize form a linear multistep or k-step method is defined thus:

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

where α_i and β_i are constants and $\alpha_0 = 1$. f_{n-i} denotes $f(t_{n-i}, u_{n-i})$, $t_{n-i} = t_n - ih$, $i = 0, 1, \ldots, k$ and h is the stepsize. The condition that $\alpha_0 = 1$ removes the arbitrariness that arises from the fact that both sides of the IVP could be multiplied by the same constant without altering the method. The linear multistep method (2) is said to be explicit if $\beta_0 = 0$ and implicit if $\beta_0 \neq 0$.

Now let $\beta_i = 0,\ i = 1, 2, \ldots, k$ in (2) then the result is a class of methods known as the backward differentiation formulae, BDFs [11]. We concentrate on BDFs which take the form

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

where h_n is the stepsize, k is the order and the coefficients α_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], [3], [9], [12], [16]. At each integration step t_n we must solve the nonlinear equation

$$F(u_n) \equiv u_n + \phi_n - h_n \beta_0 f(u_n) = 0, \tag{4}$$

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

To solve for u_n most codes use the Newton iterative method and its variants in the following form, but could also use a predictor-corrector approach

$$\begin{aligned} W_n^{(l)} \varepsilon_n^{(l)} &= -F(u_n^{(l)}), \\ u_n^{(l+1)} &= u_n^{(l)} + \varepsilon_n^{(l)} \quad l = 0, 1, 2, \ldots, \end{aligned} \tag{5}$$

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

$$W_n^{(l)} = F'(u_n^{(l)}) = I - h_n \beta_0 f'(u_n^{(l)}). \tag{6}$$

The use of the Newton method is due to the stiffness phenomenon. For large problems 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 strategies used in practice to try and minimise the cost of computing the Jacobian and the Newton matrix [3], [5], [10], [13]. These measures are mainly centered around administering 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 linear solves described by (5) and (6).

Despite being useful in the error analysis of linear multistep methods the true local truncation error is rarely used in practical codes. Instead it is estimated using some kind of error estimators, the most common being the *Milne error estimator*. A derivation of the standard Milne error estimator can be found in section 2, where we further derive a modification of the Milne error estimator for stiff problems.

We conduct our numerical experiments using a code from the MATLAB ode suite [15] known as ode15s [14]. The code is a variable step variable order code and integrates stiff initial value ordinary differential equations. In this code there is an option to use either some modified BDFs or use the standard BDFs as 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 difference operator. This is the backward difference form of the interpolating polynomial which matches the back values, u_{n-1}, u_{n-2}, ..., u_{n-k-1} and then is evaluated at t_n. The code implements some form of the Milne error estimator. Most of the test problems used can be found in the MATLAB ode suite [15] .

We conduct our experiments using the default weighted infinity norm. We further evaluate global errors to determine whether the modified code (the one using the our modified Milne error estimator) yields the same solution as the original code.

2 The Milne error estimator

2.1 The standard Milne error estimate

In practice an implicit linear multistep method is implemented by predicting the result at t_n by an explicit formula and then correcting the solution with an implicit formula. We cosider the constant stepsize case. Let the predictor be given by the explicit formula,

$$\alpha_0^* u_n^* + \sum_{i=1}^{k} \alpha_i^* u_{n-i}^* = h \sum_{i=1}^{k} \beta_i^* f_{n-i}, \tag{7}$$

(note that $\beta_0 = 0$) and the corrector by

$$\sum_{i=0}^{k} \alpha_i u_{n-i} = h \sum_{i=0}^{k} \beta_i f_{n-i}, \tag{8}$$

Assuming $z(t)$ to be any sufficiently differentiable function

$$\begin{aligned} & \alpha_0^* z(t) + ... + \alpha_k^* z(t-kh) \\ = \; & h\beta_1^* z'(t-h) + ... + \beta_k^* z'(t-kh) \\ + \; & C_{p+1}^* z^{(p+1)}(t) h^{p+1} + O(h^{p+2}), \end{aligned} \tag{9}$$

and that the predictor and the corrector have the same order p,

$$\begin{aligned} & \alpha_0 z(t) + ... + \alpha_k z(t-kh) \\ = \; & h\beta_0 z'(t) + ... + \beta_k z'(t-kh) \\ + \; & C_{p+1} z^{(p+1)}(t) h^{p+1} + O(h^{p+2}). \end{aligned} \tag{10}$$

Now choose $t = t_n$, $z \equiv u$ and assume that the back values coincide with the exact solution $u(t)$, (i.e. enforcing the localizing assumption) then from (7) and (9) we have

$$\begin{aligned} & \alpha_0^*(u(t_n) - u_n^*) \\ = \; & C_{p+1}^* h^{p+1} u^{(p+1)}(t_n) \\ & + O(h^{p+2}). \end{aligned} \tag{11}$$

From (8) and (10)

$$\begin{aligned} & \alpha_0(u(t_n) - u_n) \\ = \; & -h\beta_0[f(t_n, u_n) - f(t_n, u(t_n))] \\ + \; & C_{p+1} h^{p+1} u^{(p+1)}(t_n) + O(h^{p+2}). \end{aligned} \tag{12}$$

The expression in the square brackets can be written as $J(t_n)(u_n - u(t_n))$, where the matrix $J(t_n)$ is the mean value Jacobian at t_n, that is

$$J(t_n) = \left[\frac{\partial f_i}{\partial u_j}(t_n, \xi_i) \right],$$

where ξ_i is a point in the line segment joining u_n and $u(t_n)$, $i = 1, 2, ...m$. Now since $u_n - u(t_n) = O(h^{p+1})$ and $h\beta_0 J(t_n) = O(h)$ as $h \to 0$ we have

$$\begin{aligned} \alpha_0(u(t_n) - u_n) \quad = \quad & C_{p+1} h^{p+1} u^{(p+1)}(t_n) \\ & + O(h^{p+2}) \end{aligned} \tag{13}$$

Subtracting the expression(11) from (13) we get

$$\begin{aligned} u^{(p+1)}(t_n) \quad = \quad & h^{-p-1} \Big(\frac{C_{p+1}^*}{\alpha_0^*} - \frac{C_{p+1}}{\alpha_0} \Big)^{-1} (u_n - u_n^*) \\ & + O(h). \end{aligned} \tag{14}$$

The unknown derivative $u^{(p+1)}(t_n)$ can be expressed in terms of the difference between the the corrected value u_n and the predicted value u_n^*. Thus the *Milne estimate* of the local truncation error becomes

$$T_n = C_{p+1} \Big(\frac{C_{p+1}^*}{\alpha_0^*} - \frac{C_{p+1}}{\alpha_0} \Big)^{-1} (u_n - u_n^*) + O(h^{p+2}). \tag{15}$$

2.2 Modification of the Milne error test for stiff problems

The expression (12) can be written as

$$\begin{aligned} & \alpha_0(u(t_n) - u_n) \\ = \; & -h\beta_0 J(t_n)(u_n - u(t_n)) \\ + \; & C_{p+1}h^{p+1}u^{(p+1)}(t_n) + O(h^{p+2}), \end{aligned} \tag{16}$$

where $J(t_n)$ is the mean value Jacobian. In the derivation of the Milne error (15) it is assumed that $hJ \to 0$ (a null matrix) as $t \to \infty$, but for stiff problems $hJ \to \infty$ (a matrix with elements whose magnitudes are tending to infinity) as $t \to \infty$ and yet the Milne error estimate (15) is still used in codes for integrating stiff odes. In the analysis that follow we derive the Milne error estimate for stiff problems.

From equation (16) we have

$$\begin{aligned} & \Big(\alpha_0 I - h\beta_0 J(t_n)\Big)(u(t_n) - u_n) \\ = \; & C_{p+1}h^{p+1}u^{(p+1)}(t_n) \\ + \; & O(h^{p+2}). \end{aligned} \tag{17}$$

From (11)

$$\begin{aligned} & \Big(\alpha_0 I - h\beta_0 J(t_n)\Big)(u(t_n) - u_n^*) \\ = \; & \frac{C_{p+1}^*}{\alpha_0^*}\Big(\alpha_0 I - h\beta_0 J(t_n)\Big)u^{(p+1)}(t_n)h^{p+1} \\ + \; & O_s(h^{p+2}), \end{aligned} \tag{18}$$

where O_s can referred to as the stiff order and includes the factor $(\alpha_0 I - h\beta_0 J(t_n))$. Subtracting (17) from (18) we have

$$\begin{aligned} & \Big(\alpha_0 I - h\beta_0 J(t_n)\Big)(u_n - u_n^*) \\ = \; & \Big[-C_{p+1}I + \frac{C_{p+1}^*}{\alpha_0^*}\Big(\alpha_0 I - h\beta_0 J(t_n)\Big)\Big]u^{(p+1)}(t_n)h^{p+1} \\ + \; & O_s(h^{p+2}) + O(h^{p+2}). \end{aligned} \tag{19}$$

Hence

$$\begin{aligned} \Big[\Big(\frac{C_{p+1}^*}{\alpha_0^*}\alpha_0 \; - \; & C_{p+1}\Big)I - \frac{h\beta_0 C_{p+1}^*}{\alpha_0^*}J(t_n)\Big]^{-1} \\ & \times\Big(\alpha_0 I - h\beta_0 J(t_n)\Big)\Big(u_n - u_n^*\Big) \\ = \; & u^{(p+1)}(t_n)h^{p+1} + O_s(h^{p+2}) \\ & + O(h^{p+2}), \end{aligned} \tag{20}$$

where the O_s and O terms are now modified via the factor $[...]^{-1}$. From equation (20) if $hJ \to \infty$, that is the stiff case, we have

$$\begin{aligned} & \Big[- \frac{h\beta_0 C^*_{p+1}}{\alpha^*_0} J(t_n) \Big]^{-1} \Big(- h\beta_0 J(t_n) \Big) \Big(u_n - u^*_n \Big) \\ = u^{(p+1)}(t_n) h^{p+1} \quad & + \quad O(h^{p+2}). \end{aligned} \tag{21}$$

This reduces to

$$\frac{\alpha^*_0}{C^*_{p+1}} (u_n - u^*_n) = u^{(p+1)}(t_n) h^{p+1} + O(h^{p+2}).$$

Thus for stiff problems the Milne error estimate takes the form

$$\Big| \frac{C_{p+1}\alpha^*_0}{C^*_{p+1}} \Big| \|(u_n - u^*_n)\|. \tag{22}$$

For the nonstiff case, that is when $hJ \to 0$ the equation (20) takes the form

$$\begin{aligned} & \Big(\frac{C^*_{p+1}}{\alpha^*_0} \alpha_0 - C_{p+1} \Big)^{-1} \alpha_0 \Big(u_n - u^*_n \Big) \\ = \quad & u^{(p+1)}(t_n) h^{p+1} + O(h^{p+2}). \end{aligned}$$

The error estimate then becomes

$$\begin{aligned} & \left| \frac{\alpha_0 C_{p+1}}{\Big(C^*_{p+1} \frac{\alpha_0}{\alpha^*_0} - C_{p+1} \Big)} \right| \|(u_n - u^*_n)\| \\ = \quad & \left| C_{p+1} \Big(\frac{C^*_{p+1}}{\alpha^*_0} - \frac{C_{p+1}}{\alpha_0} \Big)^{-1} \right| \|(u_n - u^*_n)\|. \end{aligned} \tag{23}$$

This is the standard Milne error estimate.

As an example, consider the predictor-corrector pair where the corrector is the second order BDF, $p = 2$,

$$\begin{aligned} & u_n - \frac{4}{3} u_{n-1} + \frac{1}{3} u_{n-2} \\ = \quad & \frac{2}{3} h f(t_n, u_n), \end{aligned} \tag{24}$$

and the predictor is derived to be such that $\alpha_i = \alpha^*_i$ and is to have order $p = 2$. Such a predictor takes the form

$$\begin{aligned} & u_n - \frac{4}{3} u_{n-1} + \frac{1}{3} u_{n-2} \\ = \quad & h \Big[\frac{4}{3} f(u_{n-1}) - \frac{2}{3} f(u_{n-2}) \Big]. \end{aligned} \tag{25}$$

Here $C_{p+1} = -2/9$ and $C^*_{p+1} = 4/9$ and the Milne estimate (22) for the stiff case becomes

$$\frac{1}{2}\|(u_n - u^*_n)\|,$$

and for the nonstiff case, for the predictor-corrector pair (24) and (25) the Milne estimate becomes

$$\frac{1}{3}\|(u_n - u^*_n)\|.$$

The factors in both the stiff and nonstiff case are close. This suggests that the commonly used nonstiff factor of $1/3$ makes the standard Milne error estimate reliable for stiff problems.

The analysis above is for the cases when $hJ \to \infty$ and when $hJ \to 0$. No mention is made for when hJ is of moderate size. To analyse such a situation we consider the predictor corrector pair above. Substituting the values in (20) we have

$$\Big(\frac{2}{3}\Big)^{-1}\Big[I - \frac{4}{9}hJ\Big]^{-1}\Big(I - \frac{2}{3}hJ\Big)(u_n - u^*_n) = u^{(p+1)}(t_n)h^{p+1},$$

where we consider only the principal term on the right hand side. For ease of presentation consider the scalar case where now hJ becomes $z = h\lambda$ with $Re(z) \leq 0$. The function

$$R(z) = \Big(1 - \frac{4}{9}z\Big)^{-1}\Big(1 - \frac{2}{3}z\Big)$$

is analytic for $Re(z) \leq 0$ and so takes its greatest magnitude on the boundary of this region (maximum modulus theorem). Now

$$\lim_{|z|\to\infty} R(z) = \frac{-2/3}{-4/9} = 3/2$$

and by considering the magnitude of $R(z)$ for $Re(z) = 0$

$$|R(iy)| = \Big|\frac{1 - 2iy/3}{1 - 4iy/9}\Big|, \quad y \in \mathbb{R},$$

it is easily shown by considering $|R(iy)|^2$ that

$$\max_{y\in\mathbb{R}} |R(iy)| = 3/2.$$

Therefore

$$\max_{Re(z)\leq 0} |R(z)| = 3/2,$$

suggesting that the asymptotic factors ($hJ \to 0,\ hJ \to \infty$) in the Milne error test are satisfactory.

As another example consider a code from the MATLAB ode suite [15] known as ode15s [14]. The code is a variable step variable order code and integrates stiff initial value ordinary differential equations. In this code there is an option

to use either some modified BDFs or use the standard BDFs as 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 difference operator. This is the backward difference form of the interpolating polynomial which matches the back values, $u_{n-1},\ u_{n-2},\ ...,\ u_{n-k-1}$ and then is evaluated at t_n. To gain insight about this predictor let $k = 1$ then

$$u_n = u_{n-1} + \nabla u_{n-1} = 2u_{n-1} - u_{n-2}.$$

The local truncation error at t_n is given by

$$\begin{aligned} T_n &= u(t_n) - 2u(t_n - h) + u(t_n - 2h) \\ &= u(t_n) - 2[u(t_n) - hu'(t_n) + \frac{h^2}{2}u''(t_n) - ...] \\ &+ [u(t_n) - 2hu'(t_n) + \frac{4h^2}{2}u''(t_n) - ...] \\ &= h^2u''(t_n) + O(h^3). \end{aligned} \tag{26}$$

$C_0^* = C_1^* = 0$ and $C_2^* = 1$. Thus we have $C_{k+1}^* = 1$ and $\alpha_0^* = 1$. Similarly $C_{k+1}^* = 1$ and $\alpha_0^* = 1$ for $k = 2, 3, 4, ...$ So that in this case the Milne error estimate for the stiff case (22) reduces to

$$|C_{k+1}|\|(u_n - u_n^*)\|, \tag{27}$$

and for the nonstiff case (23) we have

$$\left|C_{k+1}\Big(1 - \frac{C_{k+1}}{\alpha_0}\Big)\right|\|(u_n - u_n^*)\|. \tag{28}$$

The leading coefficients of the BDFs [11] are in Table 1. Thus for the predictor

order	α_0	C_{k+1}
1	-1	-1/2
2	1/3	-2/9
3	-2/11	-3/22
4	3/25	-12/125
5	-12/137	-10/137

Table 1: Coefficients of the BDFs

corrector implementation as discussed above the factors in the Milne error estimates for stiff and nonstiff applications are as in Table 2. The largest variation

order	stiff case	nonstiff case
1	1/2	1
2	2/9	2/15
3	3/22	6/11
4	12/125	4/75
5	10/137	60/137

Table 2: Coefficients of the Milne error estimate

for the stiff and nonstiff coefficients is for the BDF of order 5 where the two coefficients vary by a factor of 6.

In ode15s the leading term of the BDF of order $p = k$ the local truncation error is approximated as

$$\frac{1}{k+1}h^{k+1}u^{(k+1)} \approx \frac{1}{k+1}h^{k+1}\nabla^{k+1}u_n.$$

In the code the local error test is implemented as, accept the Newton iterate $u_n^{(l)}$ (after the Newton method has converged) if

$$\frac{1}{k+1}\|u_n^{(l)} - u_n^{(0)}\| < rtol. \tag{29}$$

This is typically based on the Milne error estimate (23). In Table 3 the coefficients for the Milne error estimate (29) in ode15s (as verified in the actual code) are compared to the coefficients we derived for the stiff case in Table 2. It is evident that the difference is not that large (factor 2.28 at its worst!) so

order	stiff case (X)	ode15s (Y)	difference factor (Y/X)
1	1/2	1/2	1.00
2	2/9	1/3	1.50
3	3/22	1/4	1.83
4	12/125	1/5	2.08
5	10/137	1/6	2.28

Table 3: Comparing coefficients of the Milne error estimate

that we do not expect a huge variation in numerical statistics obtained when using either set of coefficients, particularly when implementing lower order BDFs. Furthermore the default coefficients in ode15s are larger than the coefficients we derived, hence the code using our coefficients could lead to a local error test which is easier to satisfy than the local error test in the original ode15s. This could lead to reduced computational costs.

3 Verification of the theory of the Milne error estimate

3.1 Numerical experiments

The aim of this experiment is to investigate whether there are any significant differences in the statistics when using the default coefficients of the error test in `ode15s` and when using the coefficients we derived for the stiff case in Table 3. We use the default weighted infinity norm and most of the test problems used are obtained from the MATLAB ode suite [15] except the following. The test problems are nonlinear and the results are in Table 4.

3.2 A dissipative stiff problem (ds1ode)

This is a simple example of a nonlinear stiff problem from [17]. The example reads,

$$u'(t) = -\sigma(u(t)^3 - 1), \quad u(0) = u_0. \tag{30}$$

$\sigma > 0$ is a parameter determining the stiffness of the problem, and $0.5 < u_0 < 1.5$. The problem is solved for $t \in [0, 10]$ with $\sigma = 1000000$ and $u_0(0) = 1.2$. The Jacobian matrix is computed once during the entire integration. The eigenvalue of the Jacobian is as given below.

```
t = 0; eigenvalue: -4320000
```

3.3 A dissipative stiff problem (ds2ode)

ds2ode is a stiff dissipative nonlinear problem obtained from Dorssalaer and Spijker [17]. The problem parameters are $k_1 = 1e6$ and $k_2 = 1.3$. The problem takes the form

$$\begin{aligned} u_1' &= -k_1 u_1(u_1 - 1) \\ u_2' &= k_1 u_1(u_1 - 1) - k_2 u_2 \end{aligned} \tag{31}$$

and is solved for $t \in [0, 100]$ with initial conditions $u_1(0) = 2$ and $u_2(0) = 0$. The Jacobian matrix is computed twice during the entire integration. The eigenvalues of the Jacobian are as given below.

```
t = 0; eigenvalues: 1.0e+06 * (-0.00000130000000, -3.00000000000000)

t = 0.61072938898683; eigenvalues: 1.0e+05 *
(-0.00001300000000, -9.99773554068031)
```

3.4 A variant of the Prothero and Robinson test problem (ds4ode)

ds4ode is also a stiff dissipative nonlinear problem obtained from Dorssalaer and Spijker [17]. The problem is such that for $k = 100$,

$$\begin{aligned} u_1' &= 1 + k(u_2 - \sin(1/4 + 2u_1)) \\ u_2' &= 2\cos(1/4 + 2u_1) - k * (u_2 - \sin(1/4 + 2u_1)) \end{aligned} \tag{32}$$

and is solved for $t \in [0; 1]$ with initial conditions $u_1(0) = 0$ and $u_2(0) = \sin(1/4)]$. The Jacobian matrix is computed 37 times during the entire integration. The eigenvalues of the Jacobian at some time steps are as given below.

```
t = 0; eigenvalues: 1.0e+02 * (-1.93782484342129, 0)

t = 0.28807629014181: eigenvalues: 1.0e+02 * (-1.35537084977037, 0)

t = 0.48346055190935: eigenvalues: -69.26662799360155, 0

t = 0.74372441760717: eigenvalues: 33.23377987125123, 0

t = 0.93838296753870: eigenvalues: 1.0e+02 * (1.05611779565109, 0)
```

3.5 A scalar parabolic IVP(will1ode)

The test problem will1ode, obtained from [18], is a discretized nonlinear, one space dimensional, scalar parabolic initial boundary value problem of the type

$$u_t = f\left(t, x, u, \frac{\partial}{\partial x}\left(p(t,x)\frac{\partial u}{\partial x}\right)\right), \quad 0 < t \leq T, \quad x \in \Omega = (0,1), \tag{33}$$

where $u(0,t) = b_0(t),\ u(1,t) = b_1(t),\ 0 < t \leq T,\ u(x,0) = u^0(x),\ 0 \leq x \leq 1$. Functions $p(t,x)$ and $f(t,x,a,b)$ satisfy the conditions

$$p(t,x) \geq p_0 > 0,\ 0 < t \leq T,\ x \in \Omega,$$

and

$$\partial f(t,x,a,b)/\partial b \geq f_0 > 0,\ 0 < t \leq T,\ x \in \Omega, a, b \in \mathbb{R}.$$

Suppose that in addition to constants p_0 and f_0 there exists real numbers f_{-1}, f_1 such that

$$f_{-1} \leq \partial f(t,x,a,b)/\partial a \leq f_1, \quad 0 < t \leq T,\ x \in \Omega, a, b \in \mathbb{R}.$$

The discretization of the problem in space is discussed in detail in [18], where it is shown that the logarithmic infinity norm satisfies

$$\mu_\sim[f'(t,u)] \leq f_1, \quad 0 \leq t \leq T, \quad u \in D.$$

We set our problem to be such that

$$f(t,x,a,b) \equiv \sin a - a + b$$

with

$$p(x) = 1 + \frac{1}{1+x}.$$

So that

$$\frac{\partial u}{\partial t} = \sin u - u + \frac{\partial}{\partial x}\left(\Big(1 + \frac{1}{1+x}\Big)\frac{\partial u}{\partial x}\right).$$

where $u(0,t) = 1,\ u(1,t) = 0,\ u(x,0) = \frac{1}{1+x^2},\ \ 0 \le t \le T,\ 0 \le x \le 1$. Here

$$p(x) \ge 1, \quad 0 \le t \le T,\ 0 \le x \le 1,$$

$$\frac{\partial f}{\partial b} = 1, \quad 0 \le t \le T,\ 0 \le x \le 1,$$

$$\frac{\partial f}{\partial a} = \cos a - 1 \implies -2 \le \frac{\partial f}{\partial a} \le f_1 = 0$$

The logarithmic infinity norm of this problem satisfies

$$-2 \le \mu_\infty[f'(t,u)] = \cos u - 1 \le 0, \quad 0 \le t \le T, \quad u \in D.$$

Hence the test problem willode is dissipative, see Dorssalaer and Spijker [17].

We solve this test problem with a constant mesh size, with the number of equations, N=500 over a time range $0 \le t \le 1000$. The sparse option is set to "on" in both codes with the pattern of the Jacobian matrix provided. The Jacobian sparsity pattern is coded thus

```
e = ones(N,1);
out1 = spdiags([e e e], -1:1, N, N);
```

The Jacobian matrix is computed once during the entire integration. The eigenvalue with the largest real part in magnitude (eig1) and the eigenvalue with the smallest real part in magnitude (eig2) are as given below.

```
t = 0; eig1 = 1.0e+04 * -1.99814454811956;
       eig2 = 1.0e+04 * -0.00201621167723
```

Test Problem	Code	Time steps	Failed steps	f evals	$\partial f/\partial y$ evals	LUs	Linear solves	Flop count
buiode	Y	115	2	170	2	21	163	57691
	X	105	1	154	2	19	147	53635
brussode	Y	344	10	712	1	51	610	51295624
	X	304	9	700	1	45	598	46871120
chm6ode	Y	686	20	1470	2	115	1459	317963
	X	543	11	1078	3	83	1062	239826
chm7ode	Y	142	2	178	1	27	174	40312
	X	126	2	158	1	24	154	36604
chm9ode	Y	5989	494	11960	66	1000	11695	2111245
	X	4352	329	9367	77	627	9058	1555550
d1ode	Y	403	40	1019	2	93	1012	200903
	X	201	18	466	3	37	456	126057
ds1ode	Y	138	2	171	1	28	169	50271
	X	145	2	214	1	29	212	49208
ds2ode	Y	322	8	612	2	61	610	88793
	X	268	5	501	2	49	499	73839
ds4ode	Y	174	44	381	21	76	379	73872
	X	162	49	364	24	78	362	72584
gearode	Y	46	0	65	1	11	61	43976
	X	44	0	63	1	11	59	42278
hb1ode	Y	451	27	840	15	96	778	159399
	X	388	23	783	15	83	721	142427
hb2ode	Y	3523	49	6098	2	250	6091	799092
	X	3047	52	5311	4	234	5298	701290
vdpode	Y	3342	326	6093	31	515	5999	846205
	X	2815	322	6165	36	485	6056	793627
will1ode	Y	301	2	429	1	49	376	7674510
	X	273	1	390	1	44	337	6917850

Table 4: Statistics obtained for nonlinear test problems for `ode15s` with the default coefficients (Y) and `ode15s` with the stiff case coefficients we derived (X). The values of *rtol* and *atol* were set to $rtol = 10^{-7}$ and $atol = 10^{-9}$, for all the test problems except for chm6ode and hb2ode where the values of *atol* were set to 10^{-13} and 10^{-20} respectively. We regard solutions obtained as high accuracy solutions.

3.6 Conclusions

From our analysis (section 2.2) it is evident that care should be taken when implementing asymptotic analysis in stiff cases. This is because we get different sets of the Milne error coefficients when we take stiffness into account (not assuming the quantity $hJ \to 0$ or $hJ = O(h)$).

We recommend that if solutions of high accuracy are sought then the coefficients we derived should be used. This is evident from Table 4 where the modified code performs better for all the test problems.

From the results obtained we conclude that there is a significant variation in the numerical statistics obtained when using either set of coefficients. With the stiff coefficients (X), the overall cost is generally reduced. The global errors are of the same order for both the original and the modified code. This confirms the theoretical expectation that the use of our coefficients generally leads to reduced computational costs.

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

[3] G. D. Byrne and A. C. Hindmarsh. A polyalgorithm for the numerical solution of ordinary differential equation. *Comm. ACM*, 1, No. 1:71–96, March 1975.

[4] W. H. Enright, T. E. Hull, and B. Linberg. Comparing numerical methods for stiff systems of ODEs. *BIT*, 15:10–48, 1975.

[5] 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.

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

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

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

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

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

[11] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems.* John Wiley and Sons, 1991.

[12] L. F. Shampine and P. Bogacki. The effect of changing the stepsize in the linear multistep codes. *SIAM J. Sci. Stat. Comput.*, 10:1010–1023, September 1989.

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

[14] Lawrence F. Shampine and Mark W. Reichelt. The MATLAB ODE suite code ode15s. *(from ftp.mathworks.com in the directory pub/mathworks/toolbox/matlab/funfun)*, 1997.

[15] Lawrence F. Shampine and Mark W. Reichelt. The MATLAB ODE suite. *SIAM J. Sci. Stat. Comput.*, 18, No. 1:1–22, January 1997.

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

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

[18] J. G. Vewer and J. M. Sanz-Serna. Convergence of method of lines approximations to partial differential equations. *Computing*, 33:297–313, 1984.