MATHEMATICS OF COMPUTATION, VOLUME 30, NUMBER 134
APRIL 1976, PAGES 216–219

Error Analysis for a Stiff System Procedure

By Arthur David Snider

Abstract. The analysis of the local truncation error in a numerical scheme for integrating stiff differential equations, presented in a recent paper by Guderley and Hsu, holds only in special circumstances, but a more general analysis preserves the main conclusions. Several modifications of the scheme are also considered.

1. In a recent paper, Guderley and Hsu [1] give an excellent detailed analysis of a predictor-corrector scheme invented by Certaine [2] for integrating stiff systems of differential equations. Section 3 of their paper presents a derivation of the local truncation error and argues that it can be simply related to the difference between the predicted and corrected values under some circumstances. The present paper attempts to clarify these circumstances, pointing out an effect overlooked in [1] but essentially recovering the original conclusion, and suggests a slight modification which improves the accuracy.

2. Following [1], we transform a stiff system of ordinary differential equations of the form

$$dy/dx + \Lambda y = f(x, y) \tag{1}$$

(x is the independent variable, y and f are vectors, and Λ is a matrix) into the equivalent integral equation

$$y(x) = e^{-\Lambda(x-\xi)}y(\xi) + \int_\xi^x \exp(\Lambda(\tau - x))f(\tau, y(\tau))d\tau. \tag{2}$$

It is assumed that Λ is a diagonal matrix (the off-diagonal terms are absorbed by the function f). For the numerical scheme using the mesh points $x_n = x_0 + nh$, the predictor for $y(x_{n+1})$ is

$$y^p_{n+1} = \exp(-\Lambda h)y_n + \int_{x_n}^{x_{n+1}} \exp(\Lambda(\tau - x_{n+1}))f_1(\tau)d\tau, \tag{3}$$

where $f_1(\tau)$ is the polynomial interpolating $f(x, y)$ at the $(k + 1)$ points (x_{n-k}, y_{n-k}) to (x_n, y_n). Then the polynomial $f_2(\tau)$ is formed, interpolating $f(x, y)$ at the points (x_{n-k+1}, y_{n-k+1}) to (x_n, y_n), and the point (x_{n+1}, y^p_{n+1}). The equation for the corrector y^c_{n+1} is the same as (3), but with $f_2(\tau)$ replacing $f_1(\tau)$:

$$y^c_{n+1} = \exp(-\Lambda h)y_n + \int_{x_n}^{x_{n+1}} \exp(\Lambda(\tau - x_{n+1}))f_2(\tau)d\tau. \tag{4}$$

Received January 30, 1973; revised July 7, 1975.
AMS (MOS) subject classifications (1970). Primary 65L05.
Key words and phrases. Stiff differential equations.

When the scheme is stable, most of the truncation error is generated locally; hence the hypothesis is made that the values of y_j for $j \leqslant n$ are exactly equal to the true solution $y(x_j)$, and the local truncation error $y(x_{n+1}) - y^c_{n+1}$ is to be computed. Under such conditions, the only errors involved in (3) and (4) come from the approximation of the function $f(x, y(x))$.

For the predictor formula, f_1 interpolates f at its previous $k + 1$ values, assumed exact. Writing $f(x)$ for $f(x, y(x))$, [1] correctly gives the difference as

$$\epsilon_1(x) = f(x) - f_1(x) = \frac{f^{(k+1)}(x')}{(k+1)!} \prod_{j=0}^{k} (x - x_{n-k+j}), \tag{5}$$

for some intermediate value x', which depends on x and which changes from component to component. Thus we can write

$$y(x_{n+1}) - y^p_{n+1} = \int_{x_n}^{x_{n+1}} \exp(\Lambda(\tau - x_{n+1}))\epsilon_1(\tau)\,d\tau. \tag{6}$$

Since the factors multiplying the components of $f^{(k+1)}(x')$ are of fixed sign in the interval of integration, we can use the mean value theorem for integrals to derive

$$y(x_{n+1}) - y^p_{n+1} = h^{k+2} C^p f^{(k+1)}(\tilde{x}), \tag{7}$$

where

$$C^p = C^p(\Lambda h) = \int_0^1 \exp(\Lambda h(\xi - 1)) \prod_{l=0}^{k} (\xi + l)\,d\xi/(k+1)!. \tag{8}$$

Observe however, that in the corrector formula $f_2(x)$ interpolates $f(x)$ at the points $x_{n-k+1}, x_{n-k+2}, \dots, x_{n-1}$, and x_n, but (and here is the effect overlooked in [1]) at x_{n+1} it takes the value $f(x_{n+1}, y^p_{n+1})$, which, unless f does not depend on y at all, is not equal to $f(x_{n+1}, y(x_{n+1}))$. Thus, the error is more complicated than equation (5). Adding and subtracting $f_3(x)$, the polynomial which interpolates $f(x, y(x))$ (exactly) at the points x_{n-k+1} through x_{n+1}, and using the Lagrange interpolation formula, we can write

$$\begin{aligned} f(x) - f_2(x) &= f(x) - f_3(x) + f_3(x) - f_2(x) \\ &= \frac{f^{(k+1)}(x'')}{(k+1)!} \prod_{j=0}^{k+1} (x - x_{n-k+j}) \\ &\quad + [f(x_{n+1}, y(x_{n+1})) - f(x_{n+1}, y^p_{n+1})] \prod_{j=1}^{k} \frac{x - x_{n-k+j}}{x_{n+1} - x_{n-k+j}}, \end{aligned} \tag{9}$$

for some intermediate x''. This yields

$$\begin{aligned} y(x_{n+1}) - y^c_{n+1} &= h^{k+2} C^c f^{(k+1)}(\tilde{x}') \\ &\quad + h\beta[f(x_{n+1}, y(x_{n+1})) - f(x_{n+1}, y^p_{n+1})], \end{aligned} \tag{10}$$

where

$$C^c = C^c(\Lambda h) = \int_0^1 \exp(\Lambda h(\xi - 1)) \prod_{l=-1}^{k-1} (\xi + l)\,d\xi/(k+1)! \tag{11}$$

and

(12) $$\beta = \beta(\Lambda h) = \int_0^1 \exp(\Lambda h(\xi - 1)) \prod_{l=0}^{k-1} (\xi + l)\, d\xi / k!.$$

In [1] it is argued that, in the circumstances when the components of $f(x, y(x))$ are given *exactly* by polynomials of degree $k + 1$ in x (so that $f^{(k+1)}$ is constant), the ratios of corresponding components of $y(x_{n+1}) - y^p_{n+1}$ to $y(x_{n+1}) - y^c_{n+1}$ are constants; and they use this conclusion to derive an easily computed expression for the truncation error. We see that this statement is not precisely true because of the last term in (10); it would be true only if we *also* assume that $f(x, y)$ is independent of y (i.e., f is simply a polynomial in x, in which case (1) can be integrated analytically). However, if $f(x, y)$ is smooth enough, we can write

(13) $$f^{(k+1)}(\tilde{x}) = f^{(k+1)}(x_{n+1}) + O(h),$$

and similarly for $f^{(k+1)}(\tilde{x}')$; and because $y(x_{n+1}) - y^p_{n+1}$ is $O(h^{k+2})$,

(14) $$f(x_{n+1}, y(x_{n+1})) - f(x_{n+1}, y^p_{n+1}) = O(h^{k+2}),$$

so that (7) becomes

(15) $$y(x_{n+1}) - y^p_{n+1} = h^{k+2} C^p f^{(k+1)}(x_{n+1}) + O(h^{k+3}),$$

and (10) becomes

(16) $$y(x_{n+1}) - y^c_{n+1} = h^{k+2} C^c f^{(k+1)}(x_{n+1}) + O(h^{k+3}).$$

Since C^c, C^p, and $C^c - C^p$ all have nonsingular limits as $h \longrightarrow 0$, we can manipulate these equations to express

(17) $$y(x_{n+1}) - y^c_{n+1} = C^c(C^p - C^c)^{-1}(y^c_{n+1} - y^p_{n+1}) + O(h^{k+3}),$$

a useful formula for numerically estimating the principal part of the truncation error if $f^{(k+1)}(x_{n+1}) \neq 0$. This idea reduces to Milne's method [3] in case $\Lambda = 0$.

Guderley and Hsu, as we have mentioned, quote (17) without the error term and propose that it be used for step control. We agree wholeheartedly, but we wish to add that with very little additional work we can get one more order of accuracy by finally setting

(18) $$y_{n+1} = (C^p - C^c)^{-1}(C^p y^c_{n+1} - C^c y^p_{n+1}),$$

since (17) implies (recall that these matrices are diagonal and commute)

(19) $$y(x_{n+1}) - y_{n+1} = O(h^{k+3});$$

this is just a variant of Richardson extrapolation. Of course, the stability computations of [1] can easily be modified accordingly (though they are still somewhat forbidding, and most practitioners will probably want to depend on step control for stability).

3. Finally, we would like to comment on two other plausible modifications for improving the accuracy. First, we consider the possibility of iterating the corrector equation (4) until it is satisfied with the same number, y^c_{n+1}, appearing as the value on the left-hand side and as the interpolated value of $f_2(\tau)$ at x_{n+1}. However, the only

change in the analysis that would result is the appearance of y^c_{n+1} instead of y^p_{n+1} in equation (10), so that the leading term in (10) is still $O(h^{k+2})$. This is an old story; when the predictor has the same order of accuracy as the corrector, the asymptotic behavior of the discretization error is not improved by iterating the latter [3, p. 261].

A more promising suggestion would be to retain the point (x_{n-k}, y_{n-k}) in interpolating $f(x, y)$ in the corrector formula, so that $(k + 2)$ values are used in (4). In this case, the error analysis would yield

$$\begin{aligned}y(x_{n+1}) - y^c_{n+1} &= h^{k+3}C^c f^{(k+2)}(\tilde{x}') \\ &\quad + h\beta[f(x_{n+1}, y(x_{n+1})) - f(x_{n+1}, y^p_{n+1})],\end{aligned} \tag{20}$$

which is $O(h^{k+3})$. However, this would increase the complexity of the code (requiring a $(k + 1)$-point *and* a $(k + 2)$-point interpolation) and destroy the step control test (17): Surely the use of equation (18) is more advantageous.

Department of Mathematics
University of South Florida
Tampa, Florida 33620

1. K. G. GUDERLEY & C. C. HSU, "A predictor-corrector method for a certain class of stiff differential equations," *Math. Comp.*, v. 26, 1972, pp. 51–69. MR **45** #8001.

2. J. CERTAINE, "The solution of ordinary differential equations with large time constants," in *Mathematical Methods for Digital Computers*, A. Ralston and H. S. Wilf (editors), Wiley, New York, 1960, pp. 128–132. MR **22** #8691.

3. P. HENRICI, *Discrete Variable Methods in Ordinary Differential Equations*, Wiley, New York, 1962, pp. 255–262. MR **24** #B1772.