SIAM J. NUMER. ANAL.
Vol. 12, No. 6, December 1975

THE USE OF BACK CORRECTIONS IN MULTISTEP METHODS OF NUMERICAL INTEGRATION*

P. BEAUDET† AND T. FEAGIN‡

Abstract. Generalized multistep methods for the numerical solution of nonlinear systems of ordinary differential equations are introduced which allow the correction of previously computed values of the solution at internal points of the grid. These methods are shown to possess enhanced numerical stability. Preliminary numerical results indicate that for some satellite orbit problems these methods also possess greater overall efficiency. A uniformly converging theory of error propagation is presented which is valid for nonasymptotic values of the step size. Experimental results are seen to conform with theory.

1. Introduction. Among the most commonly used methods for solving nonlinear ordinary differential equations are the linear multistep methods [4] in which the numerical solution is generated stepwise at the points of an equally spaced grid. These methods are sometimes called predictor-corrector methods because, as the integration advances, the solution at a point of the grid is first predicted and then usually corrected. The multistep methods considered here are distinguished from the classical methods in that they allow the solution to be corrected at back points as the integration proceeds forward. It is shown that these methods possess enhanced stability and, as a consequence, greater efficiency is realized. The only processes addressed in this paper are constant step size processes; it is assumed that any desired change in time steps is accomplished by a time regularizing transformation. Processes are considered here for the solution of systems of first order ordinary differential equations (Class I),

$$\frac{d\mathbf{y}}{dt}=\mathbf{f}(\mathbf{y}, t) \tag{1}$$

subject to the initial condition $\mathbf{y}(t_0)=\mathbf{y}_0$ and for the solution of systems of second order ordinary differential equations (Class II),

$$\frac{d^2\mathbf{y}}{dt^2}=\mathbf{f}(\mathbf{y}, t) \tag{2}$$

subject to the initial conditions, $\mathbf{y}(t_0)=\mathbf{y}_0$, $d\mathbf{y}/dt(t_0)=\dot{\mathbf{y}}_0$. It should be noted that the solution of the second order equation when $\mathbf{f}=\mathbf{f}(\mathbf{y}, d\mathbf{y}/dt, t)$ may be accomplished by means of combining a process for solving a Class I problem with a process for solving a Class II problem. [1]

In the next section, algorithms for numerical integration which use back corrections are introduced. The procedure for determining the coefficients is

* Received by the editors December 11, 1973, and in final revised form December 29, 1974. This work was supported in part by the National Research Council and the National Aeronautics and Space Administration under Contract No. NAS 5-11790 with Computer Sciences Corporation.

† Computer Sciences Corporation, Silver Spring, Maryland. Now at General Electric Space Division, Beltsville, Maryland, 20705.

‡ National Research Council Research Associate in residence at NASA Goddard Space Flight Center, Greenbelt, Maryland. Now at University of Tennessee Space Institute, Tullahoma, Tennessee, 37388.

described in § 3, and the results are presented. Section 4 displays a unified approach to the theory of error propagation for these algorithms. A comparison is made with errors obtained from numerical experiments. Section 5 contains an analysis of the stability of the algorithms, and the predicted maximum stable step size is compared with that obtained by numerical experiments. Section 6 includes the numerical results which are used to substantiate the conclusion that the higher order back-correction algorithms can lead to more efficient numerical integration because of an enhanced stability region.

2. Description of the algorithms. In this section, back-correcting algorithms for obtaining approximate solutions of ordinary differential equations are described. The predictor-corrector formulas for the Class I back-correcting algorithms are:

(a) Predictor

$$\mathbf{y}_{N+1}=\mathbf{y}_{N-m}+h\sum_{j=1}^{k+1}\beta_j(0)\mathbf{f}_{N-j+1}, \tag{3a}$$

(b) Correctors

$$\mathbf{y}_{N+2-l}=\mathbf{y}_{N-m}+h\sum_{j=1}^{k+1}\beta_j(l)\mathbf{f}_{N-j+2}\quad\text{for } l=1,2,\cdots,m+1. \tag{3b}$$

The formulas are used to propagate the state $\mathbf{y}$ from the Nth grid point to the $(N+1)$st grid point. The correctors are applied to $m+1$ grid points, the $(N+1)$st back through to the $(N+1-m)$th point corresponding to $l=1,2,\cdots,m+1$. The step size is denoted h so that the Nth grid point is associated with time $t=Nh$.

The subscript N on $\mathbf{y}$ or $\mathbf{f}$ designates evaluation at time $t=Nh$; $\beta_j(l)$ is the ordinate form of the coefficients for the process. Quantity $k+1$ is the order of the method in the step size, h.

For Class II problems, the formulas are:

(a) Predictor

$$\mathbf{y}_{N+1}=(m+2)\mathbf{y}_{N-m}-(m+1)\mathbf{y}_{N-m-1}+h^2\sum_{j=1}^{k+1}\gamma_j(0)\mathbf{f}_{N-j+1}, \tag{4a}$$

(b) Correctors

$$\mathbf{y}_{N+2-l}=(m+3-l)\mathbf{y}_{N-m}-(m+2-l)\mathbf{y}_{N-m-1}+h^2\sum_{j=1}^{k+1}\gamma_j(l)\mathbf{f}_{N-j+2}\qquad \text{for } l=1,2,\cdots,m+1, \tag{4b}$$

where $\gamma_j(l)$ denotes the jth coefficient (ordinate form) of the process.

Many algorithms are possible using these formulas. The conventional notations $\mathrm{PE(CE)}^n$ or $\mathrm{P(EC)}^n$ for classical methods can be augmented by using brackets $[\cdot]$ to designate the application to back points. Thus the algorithm $\mathrm{PECE[(CE)^2]}^m$ would designate the sequence of operations:

1. Predict the state at $t=(N+1)h$ and evaluate the derivative.
2. Correct the state at $t=(N+1)h$ and evaluate the derivative.

3. Correct the state at $t = Nh$ and evaluate the derivative.
4. Correct the state at $t = Nh$ and evaluate the derivative a second time.
5. Repeat steps 3 and 4 at the points $t = (N-1)h$, $t = (N-2)h, \cdots, t = (N+1-m)h$.

In this paper, each operation will be designated as a stage of the algorithm. Two types of back-correction algorithms have been investigated. They are PECE[CE]^m and PE[CE]^m. Note that for $m = 0$, the classical PECE and PE methods are obtained.

It often occurs in complicated systems [3] that the derivative $\mathbf{f}$ consists of two parts; $\mathbf{f} = \mathbf{f}_0 + \mathbf{f}_1$, where $\mathbf{f}_0$ is a large component which is not too difficult to compute and $\mathbf{f}_1$ is a small perturbation requiring extensive calculation. For such systems the concept of a pseudoevaluate, hereafter denoted by E^* [5], where only $\mathbf{f}_0$ is computed/recomputed, is particularly fascinating. Numerical stability of such systems is usually governed by the properties of $\mathbf{f}_0$, and stabilization at significantly reduced cost can be achieved by doing only a pseudo-evaluation after each back correction. The satellite ephemeris predication problem is an example of a system for which the concept of pseudoevaluation [7] has been used to improve the efficiency of computation.

As a guide to the user, it was ascertained that the PECE[CE] algorithm for Class I and the PE[CE] algorithm for Class II gave about the most efficient computation for any given order, $p = k+1$. These experiences are based upon the simple harmonic oscillator and satellite trajectory problems.

3. The coefficients. The coefficients for both Class I and Class II methods can be determined in rational form by making use of the operator identity [6]

$$(\nabla_K)^L = h^L\left[\left\{\frac{I - E^{-K}}{-\log(I-\nabla)}\right\}^L E^{-J}\right]E^J D^L, \tag{5}$$

where ∇_K is the backward difference operator ($\nabla \equiv \nabla_1$), L, K, and J are arbitrary integers and h is an arbitrary step size.

$$\nabla_K \mathbf{f}_N = \mathbf{f}_N - \mathbf{f}_{N-K},$$

$$\nabla \mathbf{f}_N = \mathbf{f}_N - \mathbf{f}_{N-1}.$$

E is the shifting operator

$$E^J \mathbf{f}_N = \mathbf{f}_{N+J}.$$

D is the differentiation operator and I is the identity. Methods exist [6] for expanding the operator in brackets in a power series of the backward difference operator so that

$$\nabla_K^L = h^L\left(\sum_{i=0}^{\infty} C_i(J, K, L)\nabla^i\right)E^J D^L. \tag{6}$$

A rational arithmetic computer program was written to obtain the $C_i(J, K, L)$ in rational form. The application of this operational formula upon $\mathbf{y}_{N+2-l}$ with $L = 1$, $K = m+2-l$ yields the predictor ($J = -1, l = 1$) and corrector ($J = 0$) difference forms for Class I back-corrector algorithms.

$$\mathbf{y}_{N+2-l} = \mathbf{y}_{N-m} + h \sum_{i=0}^{k} C_i(J, m+2-l, 1)\nabla^i \mathbf{f}_{N+1+J}. \tag{7}$$

It is noteworthy that the coefficients $C_i(J, K, L)$ are given in rational form so that they find application on all computational facilities.

Table 1 lists the coefficients for the difference form of all Class I back-correction algorithms with $m \leqq 3$. Table 2 gives the same coefficients for Class II methods. For any given order, $p = k+1$, the $\beta(\gamma$ for Class II) ordinate form of the coefficients (as in (3) and (4)) may be computed from (see [6])

$$\beta_{j+1} = (-1)^j \sum_{i=j}^{k} \binom{i}{j} C_i. \tag{8}$$

All computational examples were performed using the ordinate form of the difference equations (i.e., (3) and (4)).

4. Error analysis. Presented in this section is a theory of local error estimation for the back-correction algorithms. A computer program was written using these theoretical results in order to estimate the local truncation and roundoff errors for most of the algorithms of interest. The information thus obtained was used along with the results of the stability analysis (§ 5) to select the most promising algorithms for extensive study and to formulate the conclusions presented at the end of this section. A brief discussion of the contents of this section and the motivations leading to the developed theory is first presented.

In the usual presentation of error analysis [2], an asymptotic theory (meaning valid in the limit $h \to 0$) is developed. For such a theory the errors committed at each step are assumed to emanate from the last stage of the algorithm (i.e., the corrector). Using a small but finite step size h, local errors committed at earlier stages of the algorithm are multiplied by h and are tacitly assumed to be negligible. However, this assumption cannot be made for the case of large step sizes and/or for the back-correction algorithms. The local errors committed in the early stages of the back-correction algorithm are much larger than those committed in the latter stages, and multiplication by h cannot guarantee that they are negligible. This is particularly true with the realistic step sizes used for efficient numerical integration. A nonasymptotic error analysis is therefore desirable and consequently is developed here for the back-correction algorithms. Because errors at the different stages of the algorithm combine coherently with errors from previous time steps to form the error at the final stage, it is difficult to isolate independent local errors associated with the algorithm. To ascribe equivalent independent local errors to the algorithm, it is necessary to compare an inhomogeneous differential equation for the error with the difference equations which describe the errors propagated by the algorithm. In this way, relationships are established between the inhomogeneous term of the error differential equation, the local truncation and roundoff errors for each stage of the algorithm, and the intermediate state errors. The elimination of all intermediate state error variables leads to the desired relationship between the inhomogeneous driving term of the error differential equation and the various truncation and roundoff errors of the algorithm. However, this elimination of all intermediate state error

TABLE 1

Class I coefficients for back-correction algorithms ($m = 0, 1, 2, 3$) (1 *of* 3)

$m = 0$		$m = 1$		
Predictor	Corrector	Predictor	Corrector $l = 1$	Corrector $l = 2$
1	1	2	2	1
1/2	−1/2	0	−2	−3/2
5/12	−1/12	1/3	1/3	5/12
3/8	−1/24	1/3	0	1/24
251/720	−19/720	29/90	−1/90	11/720
95/288	−3/160	14/45	−1/90	11/1440
19087/60480	−863/60480	1139/3780	−37/3780	271/60480
5257/17280	−275/24192	41/140	−8/945	13/4480
1070017/3628800	−33953/3628800	32377/113400	−119/16200	7297/3628800
25713/89600	−8183/1036800	3956/14175	−9/1400	425/290304
26842253/95800320	−3250433/479001600	2046263/7484400	−8501/1496880	530113/479001600
4777223/17418240	−4671/788480	80335/299376	−2368/467775	117943/136857600
703604254357/ 2615348736000	−13695779093/ 2615348736000	5389909963/ 20432412000	−92953787/ 20432412000	1797694357/ 2615348736000
106364763817/ 402361344000	−2224234463/ 475517952000	1364651/5255250	−673175/163459296	4012317/7175168000
1166309819657/ 4483454976000	−132282840127/ 31384184832000	31374554287/ 122594472000	−460024241/ 122594472000	1116664187/ 2414168064000
25221445/98402304	−2639651053/ 689762304000	631693279/ 2501928000	−3012/875875	2214312221/ 5706215424000
8092989203533249/ 32011868528640000	−111956703448001/ 32011868528640000	15587954101729/ 62523180720000	−198060940481/ 62523180720000	10549501921729/ 32011868528640000
85455477715379/ 342372925440000	−50188465/15613165568	120348894184/ 488462349375	−3740727473/ 1275983280000	1393025585029/ 4924902850560000
12600467236042756559/ 51090942171709440000	−2334028946344463/ 786014494949376000	12156987650908561/ 49893498214560000	−27185959324387/ 9978699642912000	2503953954305731/ 10218188434341888000
1311546499957236437/ 5377993912811520000	−301124035185049/ 109285437800448000	611197056507/ 2534852320000	−99059365376/ 38979295480125	16002041245/ 74755836739584

TABLE 1—continued

Class I *coefficients for back-correction algorithms* ($m = 0, 1, 2, 3$) (2 *of* 3)

$m = 2$			
Predictor	Corrector $l = 1$	Corrector $l = 2$	Corrector $l = 3$
3	3	2	1
−3/2	−9/2	−4	−5/2
3/4	9/4	7/3	23/12
3/8	−3/8	−1/3	−3/8
27/80	−3/80	−1/90	−19/720
51/160	−3/160	0	−11/1440
137/448	−29/2240	1/756	−191/60480
265/896	−9/896	1/756	−191/120960
12881/44800	−369/44800	127/113400	−3233/3628800
3591/12800	−25/3584	13/14175	−7/12800
108223/394240	−11899/1971200	5609/7484400	−171137/479001600
1061307/3942400	−2989/563200	19/30800	−9385/38320128
2846527447/10762752000	−50840663/10762752000	10480453/20432412000	−456196373/2615348736000
509240677/1956864000	−30469149/7175168000	275216/638512875	−13681829/106748928000
11037589207/43051008000	−7890747/2050048000	44857009/122594472000	−3033240127/31384184832000
1674820979/6623232000	−9166839/2609152000	307525/980755776	−2138013/28700672000
332193623561/1330667520000	−15741501473/4879114240000	16946703679/62523180720000	−1872789638081/32011868528640000
11553711635/46839496704	−2235639653/750632960000	70327/297797500	−271788294371/5820339732480000
5697885287746637/23361198981120000	−345071605399/124926197760000	10340129027311/49893498214560000	−1931477647562191/51090942171709440000
60297248242327/249852395520000	−64098748895/24918612246528	5325836117/29092418784000	−48719221789199/1572028989898752000

TABLE 1—continued

Class I coefficients for back-correction algorithms $(m = 0, 1, 2, 3)$ (3 *of* 3)

$m = 3$				
Predictor	Corrector $l = 1$	Corrector $l = 2$	Corrector $l = 3$	Corrector $l = 4$
4	4	3	2	1
−4	−8	−15/2	−6	−7/2
8/3	20/3	27/4	19/3	53/12
0	−8/3	−21/8	−8/3	−55/24
14/45	14/45	27/80	29/90	251/720
14/45	0	3/160	1/90	3/160
286/945	−8/945	13/2240	1/756	271/60480
278/945	−8/945	13/4480	0	191/120960
4063/14175	−107/14175	81/44800	−23/113400	2497/3628800
7/25	−94/14175	113/89600	−23/113400	2497/7257600
128242/467775	−547/93555	1851/1971200	−251/1496880	90817/479001600
125812/467775	−2/385	115/157696	−62/467775	443/3942400
337524401/1277025750	−5942359/1277025750	6279127/10762752000	−193087/1857492000	184329877/2615348736000
2657285/10216206	−243808/58046625	19061/39936000	−6887/84084000	1935865/41845579776
10789633/42099750	−7312636/1915538625	5703069/14350336000	−7984463/122594472000	989217599/31384184832000
1328507/5255250	−53500/15324309	9635229/28700672000	−7712/147349125	28380527/1280987136000
487653150169/ 1953849397500	−6272467361/ 1953849397500	1400487457/ 4879114240000	−2658015071/ 62523180720000	5625120019/ 351778775040000
9833816167/ 39874477500	−441713/148898750	219971587/887111680000	−3490255/100037089152	16458903/1394032640000
95057248280749/ 389792954801250	−2146443319519/ 779585909602500	5049187071053/ 23361198981120000	−1442575827281/ 49893498214560000	34944615417419/ 393007247476880000
47028615125656/ 194896477400625	−2915504459/ 1136422608750	682432787777/ 3594030612480000	−4309271/178231803750	3723026089981/ 546427189002240000

TABLE 2

Class II *coefficients for back-correction algorithms* ($m = 0, 1, 2, 3$) (1 *of* 3)

$m = 0$		$m = 1$		
Predictor	Corrector	Predictor	Corrector $l = 1$	Corrector $l = 2$
1	1	3	3	1
0	−1	−2	−5	−2
1/12	1/12	1/4	9/4	13/12
1/12	0	1/12	−1/6	−1/12
19/240	−1/240	17/240	−1/80	−1/240
3/40	−1/240	1/15	−1/240	0
863/12096	−221/60480	1291/20160	−53/20160	31/60480
275/4032	−19/6048	107/1728	−2/945	31/60480
33953/518400	−9829/3628800	72671/1209600	−743/403200	1571/3628800
8183/129600	−407/172800	3029/51840	−5983/3628800	641/1814400
3250433/53222400	−330157/159667200	1818197/31933440	−15889/10644480	45911/159667200
4671/78848	−24377/13305600	2957909/53222400	−108629/79833600	37633/159667200
13695779093/237758976000	−4281164477/2615348736000	47363747023/871782912000	−1086802397/871782912000	510378643/2615348736000
2224234463/39626496000	−70074463/47551795200	3161165537/59439744000	−230765957/201180672000	106767253/653837184000
132282840127/2414168064000	−1197622087/896690995200	545281125187/10461394944000	−147786791/139485265920	866474507/6276836966400
2639651053/49268736000	−97997951/80472268800	123463024537/2414168064000	−1541202829/1569209241600	739514431/6276836966400
111956703448001/2134124568576000	−264713507083/237124952064000	107191860320507/2134124568576000	−649817790067/711374856192000	10308766513/101624979456000
50188465/975822848	−28500396013/27715903488000	52686555984499/1067062284288000	−606249450503/711374856192000	93945535373/1067062284288000
2334028946344463/46236146761728000	−9720886966413677/10218188434341888000	165459541069766323/3406062811447296000	−543189126550897/681212562289459200	786525034075111/10218188434341888000
301124035185049/6071413211136000	−66387102394771/75133738487808000	5749817169000703/120213981580492800	−3822081922119607/5109094217170944000	98891577246403/1459741204905984000

TABLE 2—continued

Class II *coefficients for back-correction algorithms* ($m = 0, 1, 2, 3$) (2 *of* 3)

$m = 2$

Predictor	Corrector $l = 1$	Corrector $l = 2$	Corrector $l = 3$
6	6	3	1
−8	−14	−8	−3
7/2	23/2	29/4	37/12
−1/6	−11/3	−29/12	−7/6
7/120	9/40	37/240	19/240
1/15	1/120	1/120	1/240
661/10080	−11/10080	31/20160	31/60480
1919/30240	−2/945	31/60480	0
5303/86400	−1259/604800	331/1209600	−289/3628800
53969/907200	−137/72576	11/56700	−289/3628800
4614359/79833600	−44971/26611200	24917/159667200	−3499/53222400
5043/89600	−60523/39916800	3011/22809600	−4139/79833600
23937062833/435891456000	−596426147/435891456000	9038753/79252992000	−106049897/2615348736000
2506651619/46702656000	−1624943167/1307674368000	94709/951035904	−27769877/871782912000
274806748861/5230697472000	−5938232467/5230697472000	915820439/10461394944000	−72041419/2853107712000
808056017723/15692092416000	−116887349/112086374400	485594279/6276836966400	−31740019/1569209241600
53920656305413/1067062284288000	−31125845447/32335220736000	13325679749/194011324416000	−34950809767/2134124568576000
26484196295309/533531142144000	−3734367509/4184557977600	10892084891/177843714048000	−28593026027/2134124568576000
83123033051920717/1703031405723648000	−1414521522705611/1703031405723648000	186776736253879/3406062811447296000	−113097412656737/10218188434341888000
245405591243617109/5109094217170944000	−116573762121913/150267476975616000	26509108106539/537799391281152000	−3142799778343/340606281144729600

TABLE 2—continued

Class II *coefficients for back-correction algorithms* ($m = 0, 1, 2, 3$) (3 *of* 3)

$m = 3$

Predictor	Corrector $l = 1$	Corrector $l = 2$	Corrector $l = 3$	Corrector $l = 4$
10	10	6	3	1
−20	−30	−20	−11	−4
95/6	215/6	51/2	61/4	73/12
−29/6	−62/3	−91/6	−29/3	−17/4
3/8	125/24	467/120	617/240	299/240
1/12	−7/24	−13/60	−7/48	−3/40
409/6048	−95/6048	−19/2016	−137/20160	−221/60480
1919/30240	−1/240	−31/30240	−31/30240	−31/60480
22157/362880	−871/362880	1/28800	−289/1209600	−289/3628800
671/11340	−137/72576	11/56700	−289/3628800	0
131239/2280960	−5219/3193344	15787/79833600	−6059/159667200	317/22809600
4476757/79833600	−911/623700	1981/11404800	−1/41580	317/22809600
14319817741/261534873600	−346038191/261534873600	64485013/435891456000	−15654137/871782912000	29543743/2615348736000
7001962607/130767436800	−105297509/87178291200	82167637/653837184000	−491287/33965568000	11370133/1307674368000
164567067073/3138418483200	−696007099/627683696640	187180067/1743565824000	−125978561/10461394944000	207259963/31384184832000
806786416963/15692092416000	−729496291/713276928000	1450468541/15692092416000	−159744961/15692092416000	52551743/10461394944000
53850754685879/1067062284288000	−5183188039/5472114278400	85614662729/1067062284288000	−18519577621/2134124568576000	391219813/101624979456000
400842473777/8083805184000	−187909629463/213412456857600	6240765413/88921857024000	−15877458547/2134124568576000	4128431/1385795174400
249142904330448677/5109094217170944000	−839190515239151/1021818843434188800	105291366107209/1703031405723648000	−21828473578553/3406062811447296000	1831230458503/786014494949376000
384956080466117/8020556070912000	−327156756127679/425757851430912000	280056655971791/5109094217170944000	−14164288684349/2554547108585472000	18813419306447/10218188434341888000

variables is not straightforward because corresponding state errors occur, evaluated at neighboring points on the grid. An approximate method must be established to propagate these corresponding neighboring errors to the same grid point prior to their elimination.

To propagate the state errors from one grid point to another, the homogeneous error differential equation is used in a manner consistent with other approximations made in the theory. This method of propagating errors is called a "self-consistent approximation" because it is consistent with the error differential equation.

It is well known [2] that if a numerical method is stable, the errors in the state will propagate with characteristics governed by the differential equation. The difference between the computed state $\mathbf{y}_N^{(m+1)}$ and the exact state $\mathbf{y}_N^*$ is the error, $\boldsymbol{\varepsilon}_N = \mathbf{y}_N^{(m+1)} - \mathbf{y}_N^*$. For small errors $\boldsymbol{\varepsilon}_N$, linearization of the error equation is permitted because terms of order $|\boldsymbol{\varepsilon}_N|^2$ will be extremely small compared to local truncation and roundoff errors.

Usually [2] an error equation of the form

$$\dot{\boldsymbol{\varepsilon}} = \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \cdot \boldsymbol{\varepsilon} + \boldsymbol{\Phi} \tag{9}$$

is derived from the associated error difference equation in which the error driving term $\boldsymbol{\Phi}$ is equal to the local sum of errors due to truncation and to round off of the final corrector formula. This equation is only asymptotically accurate and does not reflect higher order errors caused by the predictor formula. When practical step sizes (nonasymptotic values of h) are used, these higher order errors of the predictor often outweigh the corrector errors. The approach to error analysis which follows reflects error propagation at nonasymptotic values of the step size.

With the knowledge that accumulated errors propagate via the homogeneous error equation ($\dot{\boldsymbol{\varepsilon}} = \partial \mathbf{f}/\partial \mathbf{y} \cdot \boldsymbol{\varepsilon}$) in the absence of any further local errors, a self-consistent error driving term $\boldsymbol{\Phi}$ is determined which properly reflects higher order errors introduced by the difference algorithm. The self-consistent approximation which is used involves the propagation of errors expressions over one integration step via use of the homogeneous error differential equation.

Beginning with the error propagation differential equation (9), one can obtain a system of difference equations for the final and intermediate stage errors, $\boldsymbol{\varepsilon}_N^{(l)}$, via the same algorithm used for solving the original differential equation. From the PECE[CE]^m back-correction algorithms (Equations (3) and (4)) for Class $\left\{ \begin{matrix} \mathrm{I} \\ \mathrm{II} \end{matrix} \right\}$,

$$\begin{aligned} \boldsymbol{\varepsilon}_{N+2-l-\delta_{l,0}}^{(l)} = {} & \left\{ \begin{matrix} 1 \\ 3+m-l-\delta_{l,0} \end{matrix} \right\} \boldsymbol{\varepsilon}_{N-m} + \left\{ \begin{matrix} 0 \\ l-m-2+\delta_{l,0} \end{matrix} \right\} \boldsymbol{\varepsilon}_{N-m-1} \\ & + h^c \sum_{j=1}^{k+1} \beta_j(l) \left. \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \right|_{N-j+2-\delta_{l,0}} \cdot \boldsymbol{\varepsilon}_{N-j+2-\delta_{l,0}}^{\min[(j+\delta_{l,0}-\Theta_{l-j}),m+1]} \\ & + h^c \sum_{j=1}^{k+1} \beta_j(l)\, \boldsymbol{\Phi}_{N-j+2-\delta_{l,0}} \end{aligned} \tag{10}$$

where c is the class (1 or 2) of the method and $\boldsymbol{\varepsilon}_N$ without a superscript denotes the error at the final stage, $l=m+1$. The index l is 0 for the predictor and $1, 2, \cdots, m+1$ for the correctors. δ_{ij} is the Kronecker delta function. Θ_{l-j} is the step function

$$\Theta_u = \begin{cases} 1, & u \leqq 0, \\ 0, & u > 0. \end{cases}$$

Minor modifications to equation (10) are required for the PE[CE]^m type algorithms. In (10), $\boldsymbol{\varepsilon}_N^{(l)}$ is interpreted as the lth stage estimate of the "ultimate" error $\mathbf{e}_N$ in $\mathbf{y}_N$ at the final stage of the numerical integration process. $\boldsymbol{\varepsilon}_N^{(l)}$ is *not* the error $\mathbf{e}_N^{(l)} = \mathbf{y}_N^{(l)} - \mathbf{y}_N^*$ in $\mathbf{y}$ at the lth stage; the $\boldsymbol{\varepsilon}$'s always refer to some estimate of $\mathbf{e}_N$, the error at the final stage of the algorithm. $\mathbf{e}_N^{(l)}$ satisfies a different system of difference equations which are derived from the difference equations (3) and (4) for the state. The procedure which will be followed is to compare (10) with the difference equations satisfied by $\mathbf{e}_N^{(l)}$. A correspondence will then be made between the $\boldsymbol{\Phi}_N$ of (10) and the local truncation (and roundoff) errors $\mathbf{T}_N^{(l)}$ of the predictor and correctors. The difference equations satisfied by $\mathbf{e}_N^{(l)}$ are obtained by subtracting the exact state $\mathbf{y}_{N+2-l}^*$ from both sides of (3) and (4), expressing all terms on the right-hand side as errors $\mathbf{e}_N^{(l)}$, setting the final stage error to $\mathbf{e}_N$ and identifying the remaining terms as the truncation error. The resulting expression is:

$$\begin{aligned} \mathbf{e}_{N+2-l-\delta_{l,0}}^{(l)} = &\begin{Bmatrix} 1 \\ (3+m-l-\delta_{l,0}) \end{Bmatrix} \boldsymbol{\varepsilon}_{N-m} + \begin{Bmatrix} 0 \\ (l-m-2+\delta_{l,0}) \end{Bmatrix} \boldsymbol{\varepsilon}_{N-m-1} \\ &+ h^c \sum_{j=1}^{k+1} \beta_j(l) \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\bigg|_{N-j+2-\delta_{l,0}} \cdot \mathbf{e}_{N-j+2-\delta_{l,0}}^{\min[(j+\delta_{l,0}-\Theta_{l-j}),m+1]} \\ &+ \mathbf{T}_{N+2-l-\delta_{l,0}}^{(l)}. \end{aligned} \tag{11}$$

In this expression terms of order $|\mathbf{e}|^2$ have been assumed insignificant. $\mathbf{T}_{N+2-l-\delta_{l,0}}$ is the truncation and roundoff error of the predictor and corrector difference equations.

In order to proceed any further with (10) and (11), it would be customary [4] to assume a constant Jacobian matrix $\partial\mathbf{f}/\partial\mathbf{y}$, independent of time. The procedure which will be followed is tantamount to this. Global error propagation (error after many steps of integration) depends on the dynamic characteristics of the Jacobian matrix $\partial\mathbf{f}/\partial\mathbf{y}$. Thus, to project errors accurately over long times, it is necessary to expand the Jacobian in powers of the step size and to consider the effects of higher variations of $\mathbf{f}$ with respect to the state variables. The corrections (errors) due to the time dependence of the Jacobian matrix are often negligible compared to the higher order direct errors resulting from the truncations of various stages of the algorithm. At each stage of the algorithm, the

$$\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\bigg|_{N-j+2-\delta_{l,1}}$$

term of (10) and (11) and the $\mathbf{\Phi}_{N-j+2-\delta_{l,1}}$ term is expanded about the solution at t_N, and terms of order

$$h\mathbf{f}\frac{\partial^2\mathbf{f}}{\partial\mathbf{y}^2}\bigg|_N$$

are neglected compared to $\partial\mathbf{f}/\partial\mathbf{y}|_N$.

The resulting Jacobian matrix $\partial\mathbf{f}/\partial\mathbf{y}|_N$ is then factored from the summations in (10) and (11). These equations are then put into normal form by the application of a rotation matrix $\mathbf{R}$, which (if appropriate) diagonalizes $\partial\mathbf{f}/\partial\mathbf{y}|_N$ and transforms the error components into normal coordinates. The eigenvalues of $h^c(\partial\mathbf{f}/\partial\mathbf{y})|_N$ are designated $\bar{h}_i^c$, $i=1,2,\cdots$, and the diagonal matrix is designated $\bar{\mathbf{h}}^c$; so that

$$\bar{\mathbf{h}}^c=\begin{pmatrix} \bar{h}_1^c & 0 & 0 & \cdots \\ 0 & \bar{h}_2^c & 0 & \cdots \\ 0 & 0 & \cdot & \\ \vdots & \vdots & & \ddots \end{pmatrix}=h^c\mathbf{R}\cdot\frac{\partial\mathbf{f}}{\partial\mathbf{y}}\bigg|_N\cdot\mathbf{R}^{-1}.$$

Also, the N error components in normal coordinates are designated $\boldsymbol{\varepsilon}'$ and $\mathbf{e}'$ so that $\boldsymbol{\varepsilon}'=\mathbf{R}\cdot\boldsymbol{\varepsilon}$ and $\mathbf{e}'=\mathbf{R}\cdot\mathbf{e}$ (i.e., primed quantities designate normal coordinates). In a parallel fashion, it is also assumed that the error driving term

$$\mathbf{\Phi}'_{N-j+2-\delta_{\ell,0}}=\mathbf{R}\cdot\mathbf{\Phi}_{N-j+2-\delta_{\ell,0}}$$

in (10) is a slowly varying function of time, as is the Jacobian matrix; it is also expended in a time series about t_N.

In what follows, the leading term in a more uniformly valid expansion for the error driving term $\mathbf{\Phi}$ of (9) is obtained. In this expansion, the leading term depends only on the local Jacobian matrix, but includes the most important contribution from each stage of the algorithm. By subtracting the normal forms of (10) and (11), one obtains the equations

$$\boldsymbol{\delta}_{N+2-l-\delta_{l,0}}^{(l)}=\bar{h}^c\sum_{j=1}^{m+1-\delta_{l,0}}\beta_j(l)\boldsymbol{\delta}_{N+2-j-\delta_{l,0}}^{(j+\delta_{l,0}-\Theta_{l-j})}+\mathbf{T}'^{(l)}_{N+2-l-\delta_{l,0}}-h^c\mathbf{\Phi}'_N\sum_{j=1}^{k+1}\beta_j(l), \tag{12}$$

where $\boldsymbol{\delta}_N^{(l)}=\mathbf{e}_N'^{(l)}-\boldsymbol{\varepsilon}_N'^{(l)}$ is the difference between the lth stage error and the lth stage estimate of the "ultimate" error. $\mathbf{T}'$ in (12) has also been rotated into normal coordinates. Note that $\boldsymbol{\delta}_N^{(m+1)}$ vanishes because $\mathbf{e}_N^{(m+1)}=\boldsymbol{\varepsilon}_N^{(m+1)}$. For $l=0,1,2,\cdots,m+1$, (12) forms a system of $m+2$ difference equations in the $m+2$ unknowns $\boldsymbol{\delta}_{n+2-l'}^{(l)}$, $l=0,\cdots,m$ and $l'=l$ or $l+1$, $\mathbf{\Phi}'$ corresponding to the PECE[CE]^m algorithm. (For the PE[CE]^m algorithm, the $\boldsymbol{\delta}^{(1)}$ equation is omitted and $m+1$ equations in $m+1$ unknowns are obtained.) Note that in (12), $\boldsymbol{\delta}_{()}^{(l)}$ always appears with a subscript of the form $(N+2)-l'$ with l' always equal to l or to $l+1$.

To solve the system of equations (12), it is necessary to propagate forward the $\boldsymbol{\delta}_{()}^{(l)}$'s which are "delayed" by one step. A self-consistent approximation is made by recognizing that the errors propagate as $e^{\bar{h}N}$ (solution of the homogeneous error equation). We write

$$\boldsymbol{\delta}^{(l)}_{N+2-l+1} \simeq \boldsymbol{\delta}^{(l)}_{N+2-l}\, e^{\bar{h}}. \tag{13}$$

Eliminating the $\boldsymbol{\delta}^{(l)}_{n+2-l}$ variables from the system of $m+1$ equations (12) permits the determination of $\boldsymbol{\Phi}'_N$, a function of $\bar{h}$. It is convenient for numerical calculations to expand $\boldsymbol{\Phi}'_N$,

$$\boldsymbol{\Phi}'_N = \sum_i \boldsymbol{\phi}^{(i)}_N \bar{h}^i. \tag{14}$$

Each $\boldsymbol{\phi}^{(i)}_N$ is a linear combination of the local truncation and roundoff errors in each stage of the algorithm. A computer program was written to compute the coefficients $\boldsymbol{\phi}^{(i)}_N$ for many algorithms. The local roundoff errors were assumed constant in the computer simulations and consistent with the word size of the computer. The local truncation errors are of the form

$$\mathbf{T}^{(l)}_{N+2-l-\delta_{r,0}} = \frac{h^{k+c} C^{(l)}_{k+c} \mathbf{y}^{(k+c)}_N}{(k+c)!},$$

where $C^{(l)}_{k+c}$ is the error coefficient for the lth stage of the algorithm.

Since the error driving term $\boldsymbol{\Phi}_N$ is linear in the truncation and roundoff errors, it is possible to write it in the form

$$\boldsymbol{\Phi}'_N = C^*_{k+c}(\bar{h}) \frac{h^{k+c} \mathbf{R} \cdot \mathbf{y}^{(k+c)}_N}{(k+c)!} + \text{roundoff error},$$

where $C^*_{k+c}(\bar{h})$ are complicated but computable functions of the Jacobian eigenvalue $\bar{h}^c$. $C^*_{k+c}(\bar{h})$ is designated as the error coefficient of the entire algorithm. Error coefficients as a function of $\bar{h}$ were calculated for algorithms PECE[CE]^m and PE[CE]^m for all orders up to 16. Estimates of the local error in the state variable for the simple harmonic oscillator were made. Reasonable agreement was obtained between the theoretically predicted and actually observed local errors. Figure 1 depicts a typical result of such a comparison. No attempt was made to extract global error estimates by integrating the homogeneous error differential equation (9).

Figure 1 shows a theoretical curve of accuracy versus effective stepsize (i.e., number of function evaluations for eight cycles of simple harmonic motion). After one step of a PECE[CE]^2 integration process, theory gives slightly lower accuracy predictions than what is actually observed. The spike in the data corresponds to a change in sign of the error as a function of step size. Also shown is the global accuracy after eight cycles of the motion. Even after eight cycles, the phenomenon of numerical instability occurs just beyond the theoretical step size at which numerical instability should ensue, according to the stability analysis presented in the next section. These results tend to support the nonasymptotic theory of error analysis.

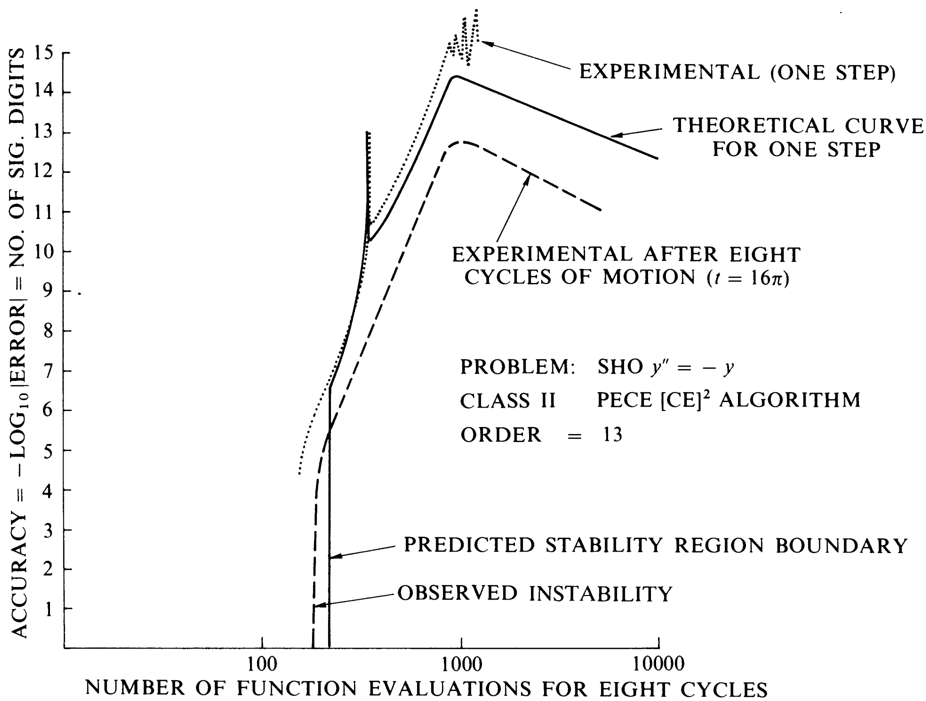

FIG. 1. *Comparison of theory with experimental accuracy for* 13*th order* PECE$[CE]^2$ *algorithm*

The errors calculated from the nonasymptotic theory were used to draw the following conclusions with respect to the efficiencies of the various back-correction algorithms. Here the efficiency of an algorithm is interpreted as the reciprocal of the number of function evaluations required to achieve a given number of significant digits of accuracy.

Without regard to stability, the classical PE algorithm is the most efficient algorithm with few exceptions. When stability is taken into account, often the PECE algorithm is more efficient.

For a given order, some Class I and Class II back-correction algorithms, (e.g., $k = 4$, $m = 1$) surpass the efficiency of classical algorithms.

Relative to the classical algorithms, there is little if any loss in efficiency caused by the additional function evaluations of a back-correction algorithm.

When the additional function evaluations required by the back-correcting algorithms are pseudo-evaluations E^*, significant increases in efficiency over classical algorithms normally occur.

These advantages become more meaningful when the improved stability characteristics of these methods are considered in the next section.

5. Stability analysis. Error analysis alone gives insufficient information to judge the applicability of a given algorithm to the solution of a particular problem. The stability of the algorithm must also be considered. It is well known [4] that when all orders are considered, the most efficient multistep algorithms border upon instability. The stiffer the differential equations, the more stable the algorithm must be. The classical stability analysis [2] is applicable to the back-correction algorithms. Two algorithms are considered here: PE[CE]^m and PECE[CE]^m.

The system of scalar error equations (normal form of equation (11)) without truncation or roundoff errors may be applied in analyzing the stability of these algorithms [4]. The following error equations pertain to both Class I and Class II algorithms.

(a) Predictor error equation.

$$e_{N+1}^{(0)} = \sum_{j=0}^{k^*} a_j(0)\, e_{N-m-j}^{(m+1)} + \bar{h}^c \sum_{j=0}^{k} \beta_j(0)\, e_{N-j}^{\min[(j+1),m+1]}. \tag{15a}$$

(b) Corrector error equations.

$$\begin{aligned} e_{N+2-l}^{(l)} = \sum_{j=0}^{k^*} a_j(l)\, e_{N-m-j}^{(m+1)} + \bar{h}^c \Bigg\{ \sum_{j=0}^{l-2} \beta_j(l)\, e_{N+1-j}^{(j+1)} \\ + \sum_{j=l-1}^{m} \beta_j(l)\, e_{N+1-j}^{(j)} + \sum_{j=m+1}^{k} \beta_j(l)\, e_{N+1-j}^{(m+1)} \Bigg\}. \end{aligned} \tag{15b}$$

In these equations, $\bar{h}^c$ is used to represent an eigenvalue of the Jacobian matrix times the step size (squared for Class II). For the algorithms presented in § 2, k^* is 0 (or 1) for Class I (or Class II); the analysis, however, holds for larger values of k^*. If there is a growing extraneous solution to these equations, then the method is unstable.

In the algorithms that were tested (equations (3) and (4)), $k^* = 0$ for all Class I algorithms and $a_0(l) = 1$. For the Class II algorithms, $k^* = 1$, $a_0(l) = 3 + m - l$, and $a_1(l) = l - m - 2$.

Assuming solutions to difference equation (15) of the standard form $e_N^{(l)} = \zeta_l Z^N$ leads to a system of linear equations in ζ_l. The existence of a nontrivial solution requires the determinant of the coefficients of ζ_l to vanish. For the PECE[CE]^m algorithm these equations are

$$\begin{aligned} \zeta_0 Z^{q+1} - \sum_{j=0}^{k^*} a_j(0) \zeta_{m+1} Z^{q-m-j} \\ - \bar{h}^c \left[\sum_{j=0}^{m} \beta_j(0) \zeta_{j+1} Z^{q-j} + \sum_{j=m+1}^{k} \beta_j(0) \zeta_{m+1} Z^{q-j} \right] = 0, \end{aligned} \tag{16a}$$

where $q = \max[k, m + k^*]$

$$\begin{aligned} \zeta_l Z^{q-l+1} - \sum_{j=0}^{k^*} a_j(l) \zeta_{m+1} Z^{q-m-j-1} \\ - \bar{h}^c \left[\sum_{j=0}^{l-2} \beta_j(l) \zeta_{j+1} Z^{q-j} + \sum_{j=l-1}^{m} \beta_j(l) \zeta_j Z^{q-j} + \sum_{j=m+1}^{k} \beta_j(l) \zeta_{m+1} Z^{q-j} \right] = 0, \end{aligned} \tag{16b}$$

where $q = \max[k, m + k^* + 1]$.

For the PE[CE]^m algorithm, the equation for ζ_1 is eliminated; and where it appears on the right-hand side of (16), ζ_1 is replaced by ζ_0. The resulting determinant of the coefficients, as a function of Z and $\bar{h}$, is called the characteristic polynomial $P(Z, \bar{h}^c)$. One (or two for Class II) of the roots Z of the polynomial $P(Z, \bar{h}^c)=0$ corresponds to the characteristic solution of the error differential equation and follows the error characteristics of the original differential equation. This root is called a principal root; the other roots are called extraneous. All principal roots will have values $\sim e^{\bar{h}}+o(h^{n+1})$, where n is the order of the method. If any extraneous root has magnitude greater than 1, then the errors will become unbounded after many integration steps and the method is said to be absolutely $\bar{h}$-*unstable*. This definition is distinct from the concept of absolute stability which requires the principal roots to have magnitude less than 1 as well. The roots of $P(Z, \bar{h}^c)=0$ are dependent on the value of $\bar{h}^c$. The region in the $\bar{h}^c$ complex plane which characterizes the boundary between $\bar{h}$-stable and $\bar{h}$-unstable algorithms is determined by the solution of the equation

$$P(e^{i\theta}, \bar{h}^c)=0$$

for real θ, $0<\theta\leqq 2\pi$. A computer program was constructed to generate the region of stability for these algorithms. Figure 2 shows the stability diagram comparison of the classical, 15th order, Class II PECE (Störmer–Cowell) algorithm compared to that of the 15th order Class II PE[CE] back-correction algorithm. Each algorithm requires two function evaluations per step.

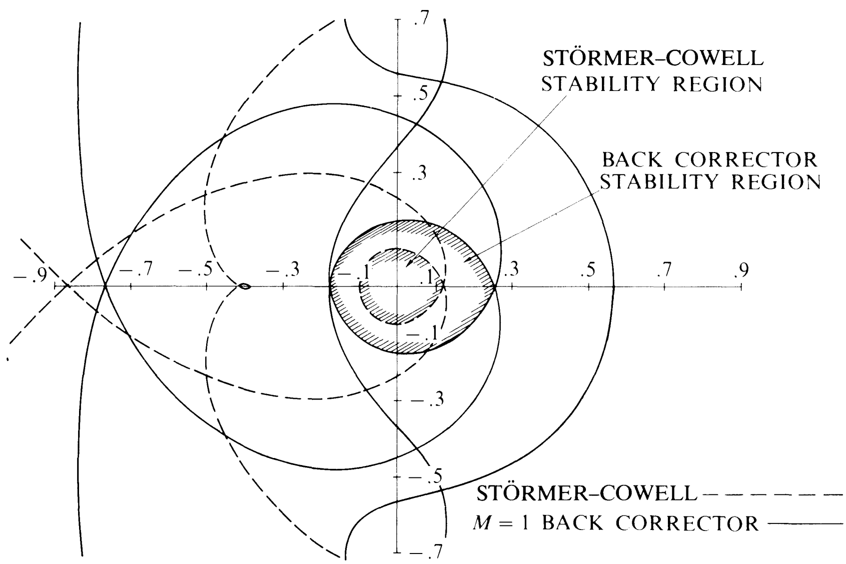

FIG. 2. *Region of stability in the complex $\bar{h}^2$ plane for 15th order Class II processes*

For oscillatory type problems, where the Class II Jacobian has negative eigenvalues, the intersection of the stability diagram curve with the negative real axis determines the maximum step size that can be taken for stable integration. Table 3 shows values of this negative h as a function of order for the PE, PECE classical, and PE[CE], PECE[CE], PE[CE]^2, PECE[CE]^2, PE[CE]^3, PECE[CE]^3 back-correction algorithms. Note that for the classical Cowell algorithms, the region of stability decreases geometrically with increasing order, while for the $m=2$ back-correction algorithm, the degradation with increasing order appears

TABLE 3

Stability limits, $(-\bar{h}^2)_{max}$, for oscillatory type Class II problems as a function of order for various algorithms

Order $=K+1$ \ Algorithm	Classical PE	Classical PECE	PE[CE]	PE[CE]^2	PE[CE]^3	PECE[CE]	PECE[CE]^2	PECE[CE]^3
2	.500*	A-Stable	.5	Unstable	Unstable	.385	Unstable	Unstable
3	.604*	12.0	.638	.101	Unstable	2.91	.002	Unstable
4	.840*	12.0	.817	.394	$8.\times10^{-5}$	1.27	1.38	$3.\times10^{-5}$
5	2.06*	1.53	.805	.618	.0806	1.19	.750	.806
6	2.64	2.33	.793	.635	.465	1.16	.767	2.28
7	.853	2.30	.812	.631	.506	1.13	.774	2.41
8	.386	1.96	.969	.621	.519	1.09	.780	2.34
9	.185	1.56	.752	.607	.530	1.06	.786	2.33
10	.091	1.24	.589	.588	.539	1.02	.792	2.32
11	.045	.968	.466	.560	.549	.962	.798	2.31
12	.023	.757	.369	.524	.559	.894	.803	2.30
13	.011	.616	.290	.482	.568	.806	.808	2.24
14	.0058	.502	.225	.436	.531	.698	.813	2.22
15	.0029	.100	.172	.389	.485	.567	.818	2.17
16	.00145	.050	.132	.340	.268	.417	.822	2.05
17	.00073	.026	.100	.293	.158	.265	.826	1.83
18	.00037	.014	.075	.248	.0959	.154	.829	1.57
19	.000185	.007	.057	.205	.0574	.087	.193	1.35
20	.00009	.004	.043	.164	.0336	.049	.095	1.12

*The stability region for these cases depends on the detailed distinction between principal and extraneous roots.

to be much slower. The small arrows in Table 3 indicate that a topologically different portion of the stability diagram controls stability for negative $\bar{h}$, and so at these places the trends in $\bar{h}$ with increasing order may be expected to change. The most remarkable feature of the back-correction algorithms, as can be seen in Table 3, is the improved region of stability relative to classical on-grid algorithms.

6. Numerical results. In order to illustrate the accuracy, stability, and overall efficiency of the methods described above, a number of them have been used to obtain the numerical solution of the differential equations describing the fully perturbed motion of the ATS-F geosynchronous satellite, and the numerical solution of the equation describing the motion of a simple harmonic oscillator (SHO). The equation to be solved for the simple harmonic oscillator is

$$\ddot{y} + y = 0$$

subject to the initial conditions

$$y(0) = 0, \qquad \dot{y}(0) = 1.$$

The solution is well known: $y(x) = \sin x$.

The accuracy of a given method is ascertained by comparing the numerical solution with the known solution at $x = 16\pi$. The number of significant digits of accuracy (negative logarithm of the absolute error) at $x = 16\pi$ is plotted against the total number of function evaluations on a log scale. Such a graph is called an efficiency curve. If it is assumed (for more complicated problems than SHO) that the evaluation of the second derivative of y consumes the dominant portion of computer time, then the curve so plotted reflects the accuracy achieved for a given amount of work. In the truncation error limited region, the efficiency curve of a process has a slope proportional to the order of the process. Numerical instability manifests itself when the accuracy drops off more rapidly than that indicated by the order of the process. This instability occurs at larger step sizes (fewer function evaluations) on the left portion of the efficiency curves. Roundoff error becomes evident when the curve peaks and the slope becomes negative at smaller step sizes. It should be noted that high order processes give evidence of encountering instability at smaller step sizes than lower order processes.

In Figs. 3, 4, 5 and 6 the efficiency curves representing methods of order 4, 8, 12, 16 and 20 as applied to the harmonic oscillator are depicted. The cases where $m = 0$ denote the use of classical Adams or Störmer–Cowell methods. The cases where $m \geqq 1$ denote the use of m back-corrections. Notice how the step size where instability ensues (for a given high order method) is affected by the introduction of back corrections.

Figure 7 shows the efficiency graph after 27 of the fully perturbed ATS-F satellite orbits integrated on the Goddard Trajectory Determination Subsystem (GTDS) using the most efficient (12th order) Class II Störmer–Cowell method (dashed line). Also shown (solid line) is the most efficient (17th order) back-correction algorithm ($m = 2$) that could be found for this orbit. The 17th order PECE*[CE*]^2 method provides an approximate solution which is from two to four significant figures more accurate than the one obtained using the 12th order PECE* method. It is concluded from these and other, as yet unpublished,

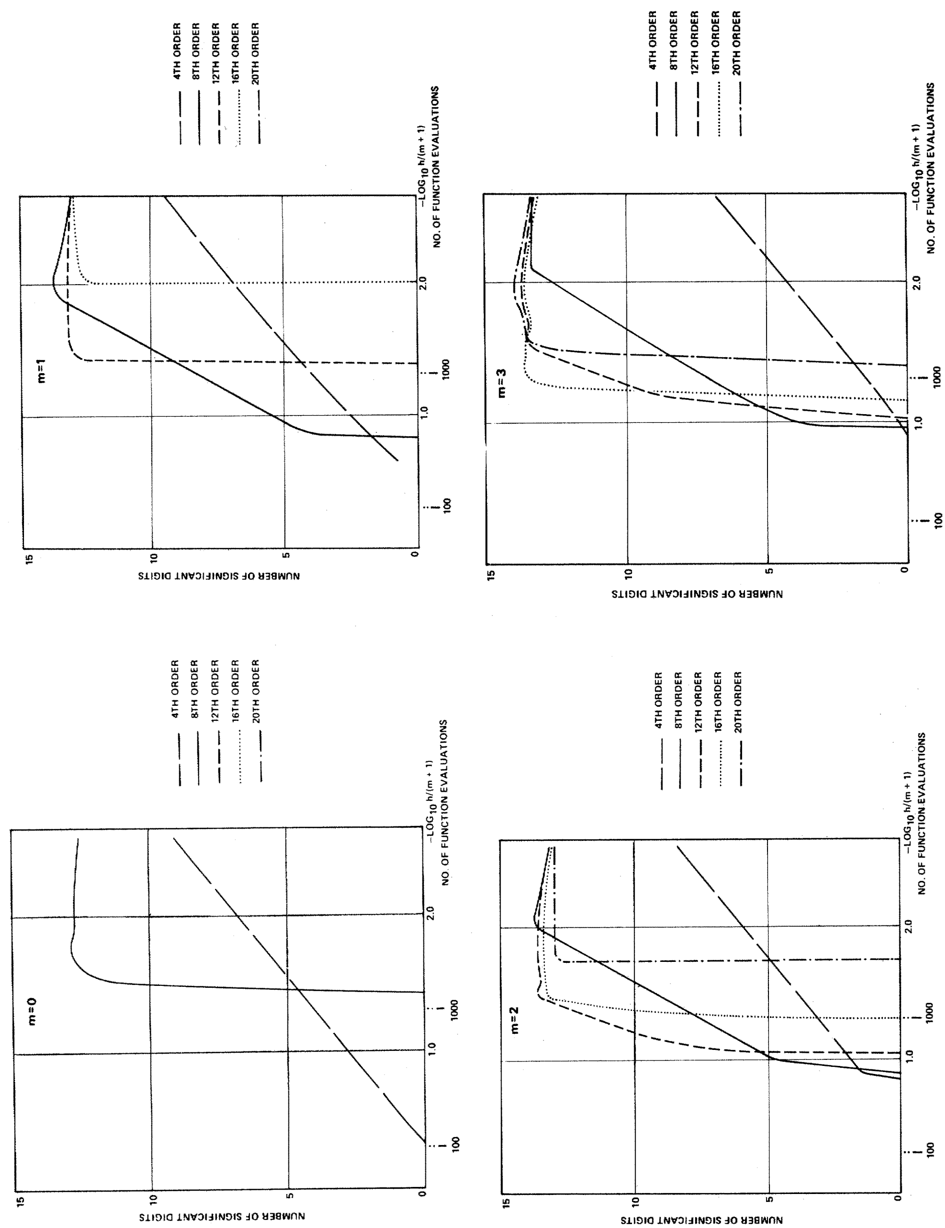

FIG. 3. *Efficiency curves for Class I processes of the type* PE[CE]^m *for simple harmonic oscillator:* $Y_1' = Y_2$, $Y_2' = -Y_1$ *(for eight cycles)*

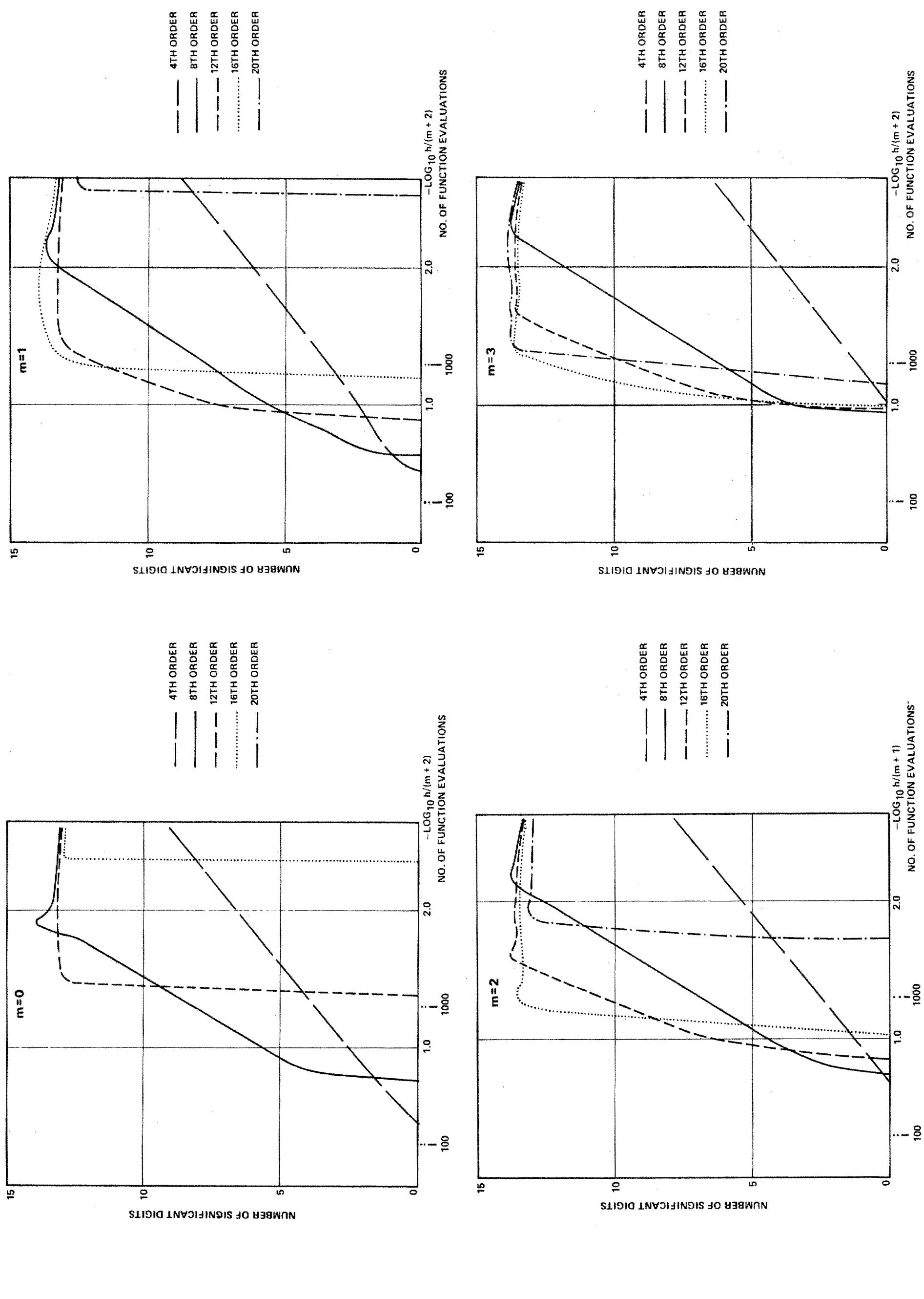

FIG. 4. *Efficiency curves for Class I processes of the type* PECE[CE]^m *for simple harmonic oscillator:* $Y'_1 = Y_2$, $Y'_2 = -Y_1$ *(for eight cycles)*

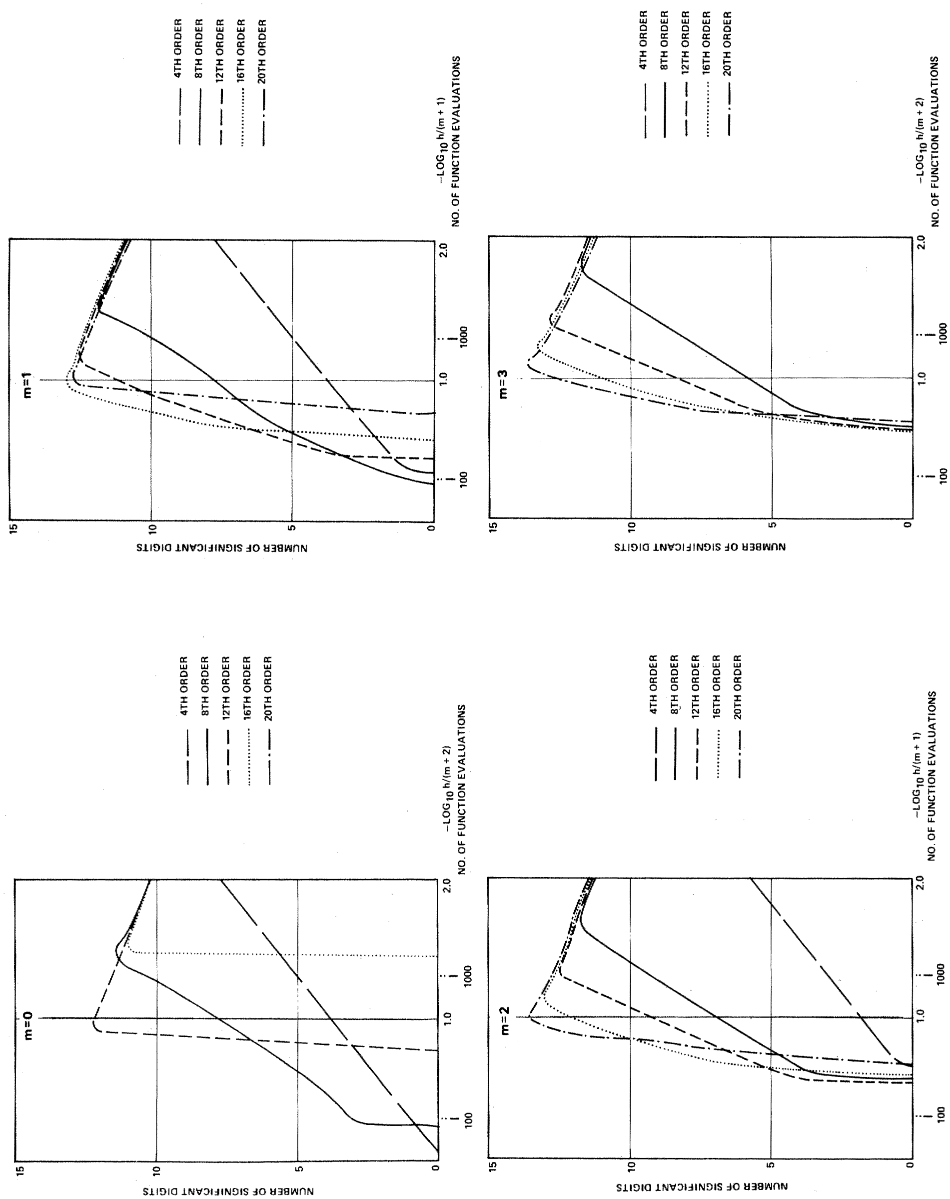

FIG. 5. *Efficiency curves for Class* II *processes of the type* PE[CE]^m *for simple harmonic oscillator:* $Y'' = -Y$ *(for eight cycles)*

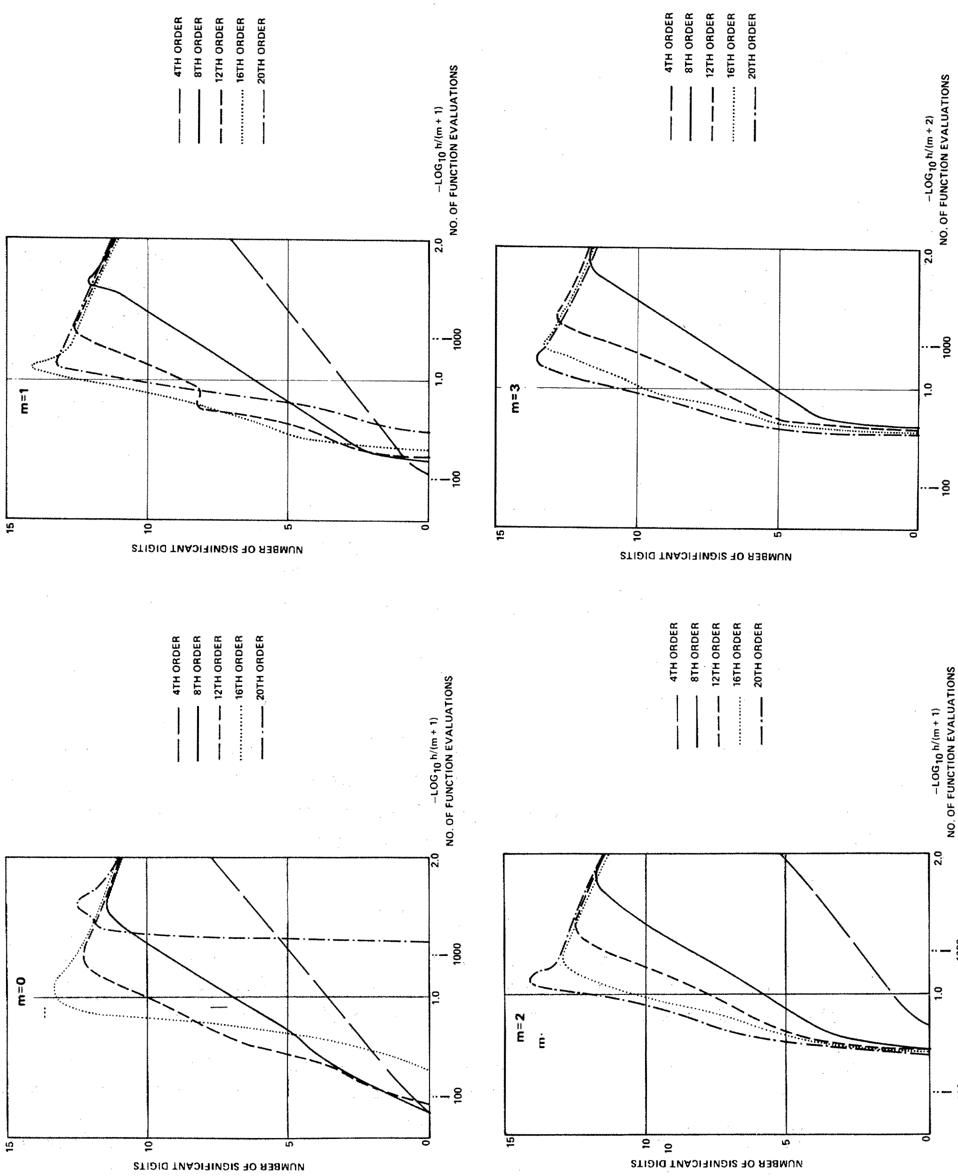

FIG. 6. *Efficiency curves for Class* II *processes of the type* PE[CE]^m *for simple harmonic oscillator:* $Y'' = -Y$ *(for eight cycles)*

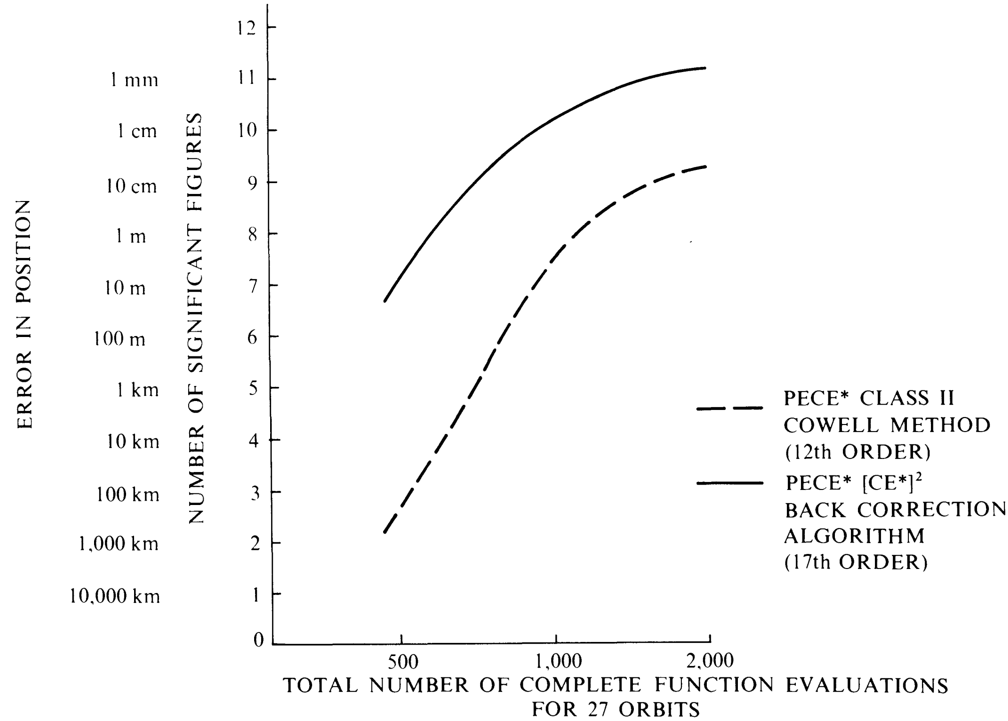

FIG. 7. *Efficiency graph for the integration of* 27 *orbits of the fully perturbed, geosynchronous* ATS-F *satellite*

numerical experiments that the enhanced stability afforded by back-correction algorithms permits the effective use of higher order algorithms.

Acknowledgments. The authors would like to thank Dr. C. E. Velez for his constructive criticisms and for making the Goddard Trajectory Determination System available for applying back-correction algorithms to the artificial satellite problem. The authors also wish to thank Mr. Joseph Vincent La Monte for making several computer runs.

REFERENCES

[1] J. L. MAURY AND G. P. BRODSKY, *Cowell type numerical integration as applied to satellite orbit computation.* Doc. 553-69-46, NASA Goddard Space Flight Center, Greenbelt, Md., 1969.

[2] P. HENRICI, *Discrete Variable Methods in Ordinary Differential Equations*, John Wiley, New York, 1962.

[3] W. M. KAULA, *Theory of Satellite Geodesy*, Blaisdell, Los Angeles, 1966.

[4] L. LAPIDUS AND J. M. SEINFELD, *Numerical Solution of Ordinary Differential Equations*, Academic Press, New York, 1971, pp. 153–241.

[5] T. D. TALBOT, *A predict-partial correct integration scheme for integrating spacecraft trajectories*, Deep Space Network, Space Programs Summary 37–59, Vol. II, Jet Propulsion Laboratory, Pasadena, Calif., 1969, pp. 78–84.

[6] C. E. VELEZ AND J. L. MAURY, *Derivation of Newtonian-type integration coefficients and some applications to orbit calculations*, NASA Tech. Note D-5958, 1970.

[7] W. W. WAGNER AND C. E. VELEZ (eds.), *Goddard trajectory determination subsystem mathematical specifications*, Doc. 552-72-244, NASA Goddard Space Flight Center, Greenbelt, Md., 1972.