The University of Wisconsin
Computer Sciences Department
1210 West Dayton Street
Madison, Wisconsin 53706

NECESSARY AND SUFFICIENT CRITERIA FOR A-STABILITY OF LINEAR MULTI-STEP INTEGRATION FORMULAE

by

Colin W. Cryer*

Technical Report #147

March, 1972

Abstract

Let $\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j$ and $\sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j$ be the characteristic polynomials of a linear multi-step method for the numerical solution of an initial value problem for ordinary differential equations. It is shown that the method is A-stable iff all the following conditions hold: (i) $\beta_k \neq 0$; (ii) $\mathrm{Re}[\rho(\zeta)\overline{\sigma(\zeta)}] \geq 0$ for $|\zeta| = 1$; (iii) ρ and σ satisfy the root condition, that is, their zeros lie inside or on the unit circle and any zeros on the unit circle are simple. For the case when ρ and σ have integer coefficients, a program has been developed which uses the SAC-1 system (system for Symbolic and Algebraic Calculations, version 1) to determine whether the multi-step method (ρ,σ) is A-stable.

NECESSARY AND SUFFICIENT CRITERIA FOR A-STABILITY OF LINEAR MULTI-STEP INTEGRATION FORMULAE

by

Colin W. Cryer*

A linear k-step integration formula for ordinary differential equations,

$$\sum_{j=0}^{k} \alpha_j \, y_{n+j} = h \sum_{j=0}^{k} \beta_j \, y'_{n+j} \qquad (1)$$

is A-stable (Dahlquist [4]) if all the solutions of the finite difference equation obtained by applying (1) to the equation $y' = \lambda y$ tend to zero as n tends to infinity for any fixed $h > 0$ and any λ, $\mathrm{Re}(\lambda) < 0$. Liniger [8] has given useful sufficient conditions for A-stability.

Here we extend the arguments of Liniger to obtain necessary and sufficient conditions for A-stability.

We assume that the reader is familiar with the basic theory of linear multi-step integration methods (see Gear [5], Henrici [6], or Lapidus and Seinfeld [7]). Let

*Sponsored by the Office of Naval Research under Contract No.: N00014-67-A-0128-0004.

$$\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j , \qquad \text{and} \qquad \sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j .$$

As is usual we assume that $\alpha_k \neq 0$ and that ρ and σ have no common zeros. We do not assume that (1) is consistent or convergent although, of course, every useful multi-step method possesses these properties.

Theorem 1

The following statements are equivalent:

(a) (1) is A-stable.

(b) If $Re(q) > 0$ and

$$\rho(\zeta) + q\,\sigma(\zeta) = 0 , \tag{2}$$

then $|\zeta| < 1$.

(c) $\rho(\zeta)/\sigma(\zeta)$ is regular and $Re[\rho(\zeta)/\sigma(\zeta)] \geq 0$ if $|\zeta| > 1$.

(d) $Re[\rho(\zeta)\overline{\sigma(\zeta)}] \geq 0$ if $|\zeta| \geq 1$.

Furthermore, ρ and σ may be interchanged in (b), (c), and (d).

Proof: Criteria (b) and (c) are due to Dahlquist [4, p. 30] who proved that (a) and (b) are equivalent and that (b) implies (c). (In his Lemma 2.1 Dahlquist [4] asserts that (b) and (c) are equivalent, but he only proves that (b) implies (c)).

Let us assume that (c) holds. If $|\zeta| > 1$ then $\sigma(\zeta) \neq 0$ and thus $Re[\rho(\zeta)\overline{\sigma(\zeta)}] \geq 0$. Appealing to

continuity, it follows that (c) implies (d).

Now assume that (d) holds. Let $Re(q) > 0$ and let ζ satisfy (2). Then $\rho(\zeta)\overline{\sigma(\zeta)} + q|\sigma(\zeta)|^2 = 0$. If $|\zeta| \geq 1$ it follows from (d) and (2) that $\sigma(\zeta) = \rho(\zeta) = 0$ which is impossible. Hence $|\zeta| < 1$ so that (d) implies (b). The equivalence of (a), (b), (c), and (d), has thus been established.

To show that ρ and σ may be interchanged, it suffices to consider (b). Let $q' = 1/q$. Since $Re(q) > 0$ iff $Re(q') > 0$, (b), is equivalent to:

(b') If $Re(q') > 0$ and

$$q'\rho(\zeta) + \sigma(\zeta) = 0$$

then $|\zeta| < 1$.

The assertion follows and the proof of the theorem is complete.

A polynomial $\gamma(\zeta)$ will be said to <u>satisfy the root condition</u> (Gear [5, p. 125]) if (i) all the zeros of γ lie inside or on the unit circle and (ii) γ has only simple zeros on the unit circle. With this notation we can state our main result:

<u>Theorem 2</u>

The multi-step method (1) is A-stable iff all the following conditions are satisfied:

(a) $\beta_k \neq 0$ (that is, the method is implicit).

(b) $\mathrm{Re}[\rho(\zeta)\overline{\sigma(\zeta)}] \geq 0$ if $|\zeta| = 1$.

(c) ρ and σ satisfy the root condition.

Proof of necessity: We assume that (1) is A-stable.

Dahlquist [4, p. 30] has shown that condition (a) is necessary, while condition (b) follows from Theorem 1, part (d).

The roots of ρ must lie inside or on the unit circle since otherwise, for sufficiently small q , the roots of $\rho + q\sigma$ would lie outside the unit circle in contradiction to Theorem 1, part (b).

Now suppose that ρ has a zero, ζ_1 say, of multiplicity $m \geq 2$ on the unit circle. Since ρ and σ have no common zeros, $\sigma(\zeta_1) \neq 0$. Thus, for ζ close to ζ_1 , $\rho(\zeta) \sim c(\zeta-\zeta_1)^m$ and $\sigma(\zeta) \sim \sigma(\zeta_1)$ where $c \neq 0$. Let C be a circle with center ζ_1 and sufficiently small radius, and let C_1 be the part of C outside the unit circle. Then, as ζ traverses C_1 , $\arg[\rho(\zeta)\overline{\sigma(\zeta)}]$ changes by at least $3\pi/2$, in contradiction to Theorem 1, part (d). Consequently, ρ can have at most simple zeros on the unit circle. Therefore, ρ must satisfy the root condition.

Since ρ and σ can be interchanged (see Theorem 1), σ must also satisfy the root condition, so that condition (c) is necessary.

Proof of sufficiency: We assume that conditions (a), (b), and (c) hold.

It is convenient to use the transformation

$$z = \frac{\zeta-1}{\zeta+1} \quad , \quad \zeta = \frac{1+z}{1-z} \quad ,$$

which maps the unit disk in the ζ-plane onto the left half plane in the z-plane.

Corresponding to $\rho(\zeta)$ and $\sigma(\zeta)$ we consider the functions

$$r(z) = \left(\frac{1-z}{2}\right)^k \rho\left(\frac{1+z}{1-z}\right) \quad , \quad s(z) = \left(\frac{1-z}{2}\right)^k \sigma\left(\frac{1+z}{1-z}\right) .$$

Clearly s is a polynomial of degree k. Since $s(1) = \beta_k \neq 0$ every zero of s corresponds to a zero of σ. If ζ is a zero of σ then $z = (\zeta-1)/(\zeta+1)$ is a zero of s iff $\zeta \neq -1$. Since σ satisfies the root condition, s has either k or $k-1$ zeros in the closed left z-plane and the imaginary zeros of s are simple. (These observations also hold for r.)

Denote the zeros of s by $z_1, \ldots, z_n$ and let $z_1, \ldots, z_m$ be imaginary. Expanding $r(z)/s(z)$ in partial fractions (Ahlfors [1, p. 32]) we find that

$$\frac{r(z)}{s(z)} = a_0 z + \sum_{j=1}^{m} \frac{a_j}{z-z_j} + \frac{R(z)}{S(z)} , \tag{3}$$

where the a_j are constants, $S(z) = \prod_{j=m+1}^{n} (z-z_j)$, $R(z)$ is a polynomial of degree $n-m$, $z_1, \ldots, z_m$ are distinct, and

$$\sum_{j=1}^{0} \frac{a_j}{a-z_j} = 0 .$$

Condition (b) of the theorem implies that:

$$\mathrm{Re}[r(z)/s(z)] \geq 0 \quad \text{if} \quad \mathrm{Re}(z) = 0 \quad \text{and} \quad s(z) \neq 0 . \tag{4}$$

Using (3), remembering that $R(z)/S(z)$ is bounded on the imaginary axis, and considering what happens as z approaches z_j, we see that (4) implies that the coefficients $a_0, \ldots, a_m$ are real and that

$$\mathrm{Re}[R(z)/S(z)] \geq 0 \quad \text{if} \quad \mathrm{Re}(z) = 0 . \tag{5}$$

The zeros of $S(z)$ lie in the open left half plane so that $R(z)/S(z)$ is regular in the closed right half plane. The limit $R(\infty)/S(\infty)$ exists and it follows from (5) that $\mathrm{Re}[R(\infty)/S(\infty)] \geq 0$. Thus, appealing to the maximum principle for harmonic functions, we see that $\mathrm{Re}[R(z)/S(z)] \geq 0$ if $\mathrm{Re}(z) \geq 0$. Furthermore, since $a_0, \ldots, a_m$ are real, we have that

$$\mathrm{Re}\left[a_0 z + \sum_{j=1}^{m} \frac{a_j}{z-z_j}\right] \geq 0 \quad \text{if} \quad \mathrm{Re}(z) > 0 \ .$$

Using (3) we see that if $\mathrm{Re}(z) > 0$ then $r(z)/s(z)$ is regular and $\mathrm{Re}[r(z)/s(z)] \geq 0$. Returning to the ζ-plane, we see that if $|\zeta| > 1$ then $\rho(\zeta)/\sigma(\zeta)$ is regular and $\mathrm{Re}[\rho(\zeta)/\sigma(\zeta)] \geq 0$. This is precisely criterion (c) of Theorem 1, so that we have established that (1) is A-stable and the proof of the theorem is complete.

We conclude with some remarks:

1. Dahlquist [4, p. 30] noted that σ may have simple zeros on the unit circle but did not pursue this further.

2. Liniger [8] has expressed condition (b) of Theorem 2 in a form which is very suitable for computation.

3. It was not assumed that ρ and σ have real coefficients, but it seems unlikely that integration formulae with complex coefficients will be of any use.

4. In any family of A-stable methods, those methods for which ρ and σ have simple zeros on the unit circle are "extreme points" and thus likely solutions to optimization problems. For example, the trapezoidal

rule ($\rho(\zeta) = \zeta-1$; $\sigma(\zeta) = (\zeta+1)/2$) has the smallest error constant of all A-stable methods (Dahlquist [4, p. 31]).

5. If a and b are any two positive constants then the multi-step method (ρ,σ) is A-stable iff the multi-step method $(a\rho,b\sigma)$ is A-stable. If ρ and σ have rational coefficients we can choose a and b so that $a\rho$ and $b\sigma$ have integer coefficients. Thus, when considering the A-stability of multi-step methods with rational coefficients it may be assumed that ρ and σ have integer coefficients.

6. If ρ and σ have integer coefficients then the conditions of Theorem 2 can be checked using only integer arithmetic. Collins [2] and his co-workers have developed a system, SAC-1 (system for Symbolic and Algebraic Calculations, version 1), which uses list-processing techniques to manipulate polynomials with integer coefficients. Polynomials are represented as lists of appropriate size: for example, a polynomial of degree 200 with coefficients each of which is a one hundred-decimal-digit integer, would be represented as a list occupying approximately 4000 storage locations. In SAC-1 all polynomial manipulations are performed exactly unless the limits of storage or time are exceeded in which case an error

message is generated. SAC-1 is based on FORTRAN and is essentially machine independent; it has been implemented on several different computers. Using SAC-1 the author has developed a subroutine which given two polynomials ρ and σ with integer coefficients, checks the criteria of Theorem 2 and determines whether or not the corresponding method (1) is A-stable. A description and listing of the subroutine is given in the Appendix.

APPENDIX

Determining whether a linear multi-step method with integer coefficients is A-stable.

In this appendix we describe a program, STABLE, which reads in pairs of polynomials $\rho(\zeta)$ and $\sigma(\zeta)$ with real integer coefficients and uses Theorem 2 to determine whether or not the corresponding linear multi-step method (1) is A-stable. The program STABLE uses several subprograms, namely ASTABL, ICWSGN, ROOTCN, ROUTH, PCWSUB, and RCWSUB, which are also described in this appendix. All the programs are written in FORTRAN within the framework of the SAC-1 system (system for Symbolic and Algebraic Computations, version 1) of Collins [2]. Annotated listings of the programs, and sample input and output, are given at the end of this appendix.

A.1 The program STABLE

This program reads in pairs of polynomials $\rho(\zeta)$ and $\sigma(\zeta)$ with integer coefficients. After each pair of polynomials has been read in the polynomials are printed and subroutine ASTABL is called. ASTABL determines whether the corresponding multi-step method is A-stable and prints an appropriate message. Program STABLE stops when the pair $\rho=\sigma=0$ is read in.

The data for each pair of polynomials consists of: (1) a title card which is reproduced; (2) one or more cards containing $\rho(\zeta)$ in external canonical SAC-1 form; (3) one or more cards containing $\sigma(\zeta)$ in external canonical SAC-1 form. The external canonical SAC-1 form for a non-zero polynomial

$$p(\zeta) = \sum_{j=1}^{k} \pm C_j \zeta^{e_j}$$

is $(\pm C_1 ZETA**e_1 \pm C_2 ZETA**e_2 \ldots \pm C_k ZETA**e_k)$ where $C_1, \ldots, C_k$ are strictly positive integers and $e_1 > e_2 > \ldots > e_k$ are non-negative integers.

A.2 The subroutine ASTABL

The input to this subroutine is two SAC-1 polynomials $\rho(\zeta)$ and $\sigma(\zeta)$. The subroutine determines whether the corresponding multi-step method is A-stable by checking the conditions of Theorem 2. An appropriate message is printed.

The following steps are performed:

1. If ρ and σ have a common factor this factor is divided out and the modified polynomials ρ and σ are printed.

2. Condition (a) of Theorem 2 is checked. Since, by definition, a non-zero SAC-1 polynomial has a non-zero leading coefficient, condition (a)

of Theorem 2 is satisfied iff ρ and σ have the same degree.

3. Condition (c) of Theorem 2 is checked with the aid of the function subprogram ROOTCN.

4. Condition (b) of Theorem 2 is checked as follows.

Let $\cos\theta = u$ and $i\sin\theta = v$ so that $e^{i\theta} = u+v$, $e^{-i\theta} = u-v$. Since $v^2 = u^2 - 1$,

$$\rho(e^{i\theta})\sigma(e^{-i\theta}) = \rho(u+v)\sigma(u-v) = R(u) + vI(u)$$

where $R(u)$ and $I(u)$ are polynomials in u. As Liniger [8] has observed, condition (b) of Theorem 2 is satisfied iff $R(u) \geq 0$ for $-1 \leq u \leq +1$.

Let p be the bivariate polynomial $p(u,v) = \rho(u+v)\sigma(u-v)$. In SAC-1, p is represented in the form

$$p(u,v) = \sum_j p_j(u)v^{e_j},$$

where the $p_j(u)$ are polynomials in u and the e_j are non-negative integers. Hence,

$$R(u) = \sum_{\substack{j \\ e_j \text{ even}}} p_j(u)(u^2-1)^{(e_j/2)}.$$

The function subprogram ICWSGN is used to determine whether $R(u) \geq 0$ for $-1 \leq u \leq +1$.

A.3 The function subprogram ICWSGN

The input to this function subprogram is a SAC-1 polynomial in one variable, p, and a closed nontrivial interval $r = [r_\ell, r_r]$ where r_ℓ and r_r are SAC-1 rational numbers. The output is $+1$ if p is strictly positive on r, 0 if p is non-negative on r, and -1 otherwise.

If p is identically zero the output is 0. Otherwise the sign of p on r is determined as follows:

(1) p is strictly positive on r iff p has no zeros in $[r_\ell, r_r]$ and $p(r_t) > 0$ for some $r_t \in [r_\ell, r_r]$.

(2) p is non-negative on r iff p has no zeros of odd multiplicity in (r_ℓ, r_r) and $p(r_t) > 0$ for some $r_t \in [r_\ell, r_r]$.

The location of the zeros of p with respect to r is determined as follows: Let

$$p(z) = a \sum_j (z-z_j)^{e_j} ,$$

and

$$q_m(z) = b_m \prod_{\substack{j \\ e_j=m}} (z-z_j) .$$

Then

$$p(z) = a \prod_m [q_m(z)/b_m]^m$$

so that

{number of zeros of p in $[r_\ell, r_r]$} = $\sum_m m$ {number of zeros of q_m in $[r_\ell, r_r]$} .

Furthermore, p has a zero of odd multiplicity in (r_ℓ, r_r) if q_m has a zero in (r_ℓ, r_r) for some odd m .

The polynomials q_m are constructed using the following recursive procedure. Let

$$p_m(z) = a_m \prod_{\substack{j \\ e_j \geq m-1}} (z-z_j)^{e_j-m+1}$$

so that $p_1(z) = p(z)$. Let v_1 be the greatest square-free divisor of p_m so that

$$v_1(z) = c_1 \prod_{\substack{j \\ e_j \geq m}} (z-z_j) ,$$

and let

$$v_2(z) = p_m(z)/v_1(z) = c_2 \prod_{\substack{j \\ e_j \geq m}} (z-z_j)^{e_j-m} .$$

Let v_3 be the greatest common divisor of v_1 and v_2 so that

$$v_3(z) = c_3 \prod_{\substack{j \\ e_j > m}} (z - z_j) ,$$

and let $v_4 = v_1/v_3$ so that

$$v_4(z) = c_4 \prod_{\substack{j \\ e_j = m}} (z - z_j) .$$

Then $q_m = v_4$ and $p_{m+1} = v_2$.

The above computations are easily carried out using SAC-1. The number of zeros of q_m in $(r_\ell , r_r]$ is computed using the SAC-1 subroutine IRTS. The number of zeros of q_m in $[r_\ell , r_r]$ and (r_ℓ , r_r) is then easily found.

If p has no zeros of odd multiplicity in (r_ℓ , r_r) then p is evaluated at a sequence of points r_t . Initially $r_t = r_r$. If $p(r_t) = 0$ we set $r_t = (r_\ell + r_t)/2$ and re-evaluate $p(r_t)$. Since p has only a finite number of zeros, $p(r_t) \neq 0$ for some r_t , and the sign of p on $[r_\ell , r_r]$ is the same as that of $p(r_t)$.

A.4 The function subprogram ROOTCN

The input to ROOTCN is a SAC-1 polynomial p in one variable. The output is +1 if all the zeros of p lie strictly inside the unit circle, 0 if p satisfies the root condition, and -1 if the root condition is not satisfied.

If p has exact degree k let

$$q(z) = (1-z)^k \, p(\frac{1+z}{1-z}) \ .$$

Then $\zeta \neq -1$ is a zero of $p(\zeta)$ of multiplicity m iff $z = \frac{\zeta-1}{\zeta+1}$ is a zero of $q(z)$ of multiplicity m . Furthermore, $\zeta = -1$ is a zero of $p(\zeta)$ of multiplicity n where n = exact degree of p - exact degree of q .

The mapping $\zeta = \frac{1+z}{1-z}$, $z = \frac{\zeta-1}{\zeta+1}$ maps the unit circle in the ζ-plane onto the left half plane in the z-plane. Hence, a zero of p lies inside, on, or outside the unit circle iff the corresponding zero of q lies to the left of, on, or to the right of the imaginary axis. The location of the zeros of q with respect to the imaginary axis is determined with the aid of the subroutine ROUTH.

A.5 The subroutine ROUTH

The input to ROUTH is a SAC-1 polynomial p in one variable. The output from ROUTH is a list SEQ. If p

has degree less than one then SEQ is the empty list. Otherwise

$$SEQ = (S_1, S_2, \ldots, S_{m+3})$$

where

S_1 = number of zeros of p with strictly negative real part,

S_2 = number of imaginary zeros of p ,

S_3 = number of zeros of p with strictly positive real part,

S_{i+3} = number of imaginary zeros of p with multiplicity i ,

and $m \geq 0$ is the maximum of the multiplicities of the imaginary zeros of p .

Subroutine ROUTH, which is based upon the Routh-Hurwitz algorithm, is described in detail elsewhere (Cryer [3]). The listing of ROUTH given at the end of this appendix is the same as that in Cryer [3] except that all print statements have been deleted and some comments have been changed or deleted.

A.6 The function subprograms PCWSUB and RCWSUB

In the SAC-1 system, if one has polynomials p(x) and q(y) and wishes to form the polynomial p(q(y)) ,

one must first construct the polynomial in two variables $r(y,x) \equiv p(x)$ and then use the function subprogram PSUBST to form $r(y,q(y)) \equiv p(q(y))$. The function subprogram PCWSUB allows the direct substitution of $q(y)$ into $p(x)$. The function subprogram RCWSUB does the same for rational functions.

```
C      PROGRAM STABLE
C*****************************************************************
C      THIS PROGRAM READS IN PAIRS OF POLYNOMIALS RO AND
C      SIGMA, PRECEDED BY IDENTIFYING TITLE CARD,
C      AND CHECKS WHETHER THE LINEAR MULTISTEP METHODS
C      RO(E)Y = H*SIGMA(E)DY ARE A-STABLE
C      AN APPROPRIATE MESSAGE IS PRINTED
C      THE PROGRAM STOPS WHEN RO=SIGMA=0
C*****************************************************************
       COMMON/TR1/AVAIL,STAK,RECORD(72)
       COMMON/TR2/SYMLST
       COMMON/TR3/BETA
       COMMON/TR4/PRIME,PEXP
       IMPLICIT INTEGER (A-Z)
       DIMENSION SPACE(20000),PA(1000)
       DIMENSION TITLE(10)
C*****************************************************************
C      INITIALIZE
C*****************************************************************
       IN=5
       OUT=6
       CALL BEGIN(SPACE,20000)
       BETA=2**33
       CONS=4000000001
       PRIME=GENPR(PA,1000,CONS)
       CONS=PFA(CONS,0)
       CALL ELPOF2(CONS,PEXP,DUM)
       CALL ERLA(CONS)
       SYMLST=0
C*****************************************************************
C      READ IN POLYNOMIALS RO AND SIGMA
C*****************************************************************
       PRINT 10
10     FORMAT('1')
50     CONTINUE
       READ 70,TITLE
70     FORMAT( 10A6)
       PRO=PREAD(IN)
       PSIGMA=PREAD(IN)
       IF( PRO.EQ.0 .AND. PSIGMA.EQ.0 ) GO TO 9500
       PRINT 71,TITLE
71     FORMAT( '0' ,10A6)
       PRINT 100
100    FORMAT( '0 RO=')
       CALL PWRITE(OUT,PRO)
       PRINT 101
101    FORMAT( '0 SIGMA=')
       CALL PWRITE(OUT,PSIGMA)
       CALL ASTABL(PRO,PSIGMA)
       CALL PERASE(PRO)
       CALL PERASE(PSIGMA)
       GO TO 50
C*****************************************************************
C      TIDY UP
C*****************************************************************
9500   CONTINUE
```

```
      CALL ERASE(PRIME)
      CALL ERASE(SYMLST)
      L=LENGTH(AVAIL)
      IF(L.EQ.10000)STOP
      PRINT 9999,L
9999 OFORMAT( '0 NOT ALL STORAGE ERASED.'/
     1' LENGTH OF AVAIL=', I10)
      END
```

```
      SUBROUTINE ASTABL(PR,PS)
C*****************************************************************
C     PR=PRO AND PS=PSIGMA ARE POLYNOMIALS IN ONE VARIABLE
C     THE SUBROUTINE CHECKS WHETHER THE LINEAR MULTISTEP
C     METHOD RO(E)Y = H*SIGMA(E)DY IS A-STABLE
C     AN APPROPRIATE MESSAGE IS PRINTED
C*****************************************************************
      IMPLICIT INTEGER (A-Z)
      IN=5
      OUT=6
      PRO=BORROW(PR)
      PSIGMA=BORROW(PS)
C*****************************************************************
C     CHECK WHETHER RO AND SIGMA HAVE A COMMON FACTOR
C     IF SO, DIVIDE FACTOR OUT
C*****************************************************************
      PRS=PGCD(PRO,PSIGMA)
      IF(PONE(PRS).EQ.1) GO TO 200
      PTEMP=PQ(PRO,PRS)
      CALL PERASE(PRO)
      PRO=PTEMP
      PTEMP=PQ(PSIGMA,PRS)
      CALL PERASE(PSIGMA)
      PSIGMA=PTEMP
      PRINT 150
150   FORMAT( '0 RO AND SIGMA HAVE COMMON FACTOR PRS=')
      CALL PWRITE(OUT,PRS)
      PRINT 155
155   FORMAT( '0 RO/PRS=')
      CALL PWRITE(OUT,PRO)
      PRINT 160
160   FORMAT( '0 SIGMA/PRS=')
      CALL PWRITE(OUT,PSIGMA)
200   CALL PERASE(PRS)
C*****************************************************************
C     CHECK WHETHER RO AND SIGMA HAVE SAME DEGREE
C*****************************************************************
      IF( PDEG(PRO).EQ.PDEG(PSIGMA) )GO TO 1000
      PRINT 250
250   FORMAT( '0 RO AND SIGMA DO NOT HAVE SAME DEGREE')
      GO TO 8500
C*****************************************************************
C     CHECK WHETHER RO AND SIGMA SATISFY THE ROOT CONDITION
C*****************************************************************
1000  CONTINUE
      SRO=ROOTCN(PRO)
      IF( SRO.GE.0 ) GO TO 1100
      PRINT 1050
1050  FORMAT( '0 RO DOES NOT SATISFY ROOT CONDITION')
      GO TO 8500
1100  CONTINUE
      SSIG=ROOTCN(PSIGMA)
      IF(SSIG.GE.0) GO TO 1200
      PRINT 1150
1150  FORMAT( '0 SIGMA DOES NOT SATISFY ROOT CONDITION')
      GO TO 8500
```

```
C*************************************************************
C       SET UP BASIC FUNCTIONS
C       PUPLSV=U+V WITH V THE LEADING VARIABLE
C       PUMINV=U-V WITH V THE LEADING VARIABLE
C       PUUM1=U**2-1
C*************************************************************
1200    CONTINUE
        ONE=PFA(1,0)
        LU=PFA(30,0)
        U=PROSYM(LU)
        CALL ERASE(LU)
        LV=PFA(31,0)
        V=PROSYM(LV)
        CALL ERASE(LV)
        PU1=PFA(1,0)
        BONE=BORROW(ONE)
        PU1=PFL(BONE,PU1)
        PU1=PFL(U,PU1)
        PV1=PFA(1,0)
        BONE=BORROW(ONE)
        PV1=PFL(BONE,PV1)
        PV1=PFL(V,PV1)
        LU=PVLIST(PU1)
        LV=PVLIST(PV1)
        LUV=CONC(LU,LV)
        PU1V=PORDER(PU1,LUV)
        PV1U=PORDER(PV1,LUV)
        PUPLSV=PSUM(PU1V,PV1U)
        PUMINV=PDIF(PU1V,PV1U)
        CALL PERASE(PU1V)
        CALL PERASE(PV1U)
        CALL PERASE(PV1)
        CALL ERASE(LUV)
        PUU=PPROD(PU1,PU1)
        PU0=PQ(PU1,PU1)
        PUUM1=PDIF(PUU,PU0)
        CALL PERASE(PU1)
        CALL PERASE(PU1)
        CALL PERASE(PUU)
        CALL ERASE(ONE)
C*************************************************************
C       COMPUTE PREAL=REAL PART OF P(U,V)
C       WHERE P(U,V)=RO(U+V)*SIGMA(U-V)=SUM ( PJ(U)*V**EJ )
C       THE TERMS PJ(U)*V**EJ ARE EXAMINED ONE BY ONE
C       PREAL=SUM( PJ(U)*(U**2-1)**(EJ/2), EJ EVEN )
C*************************************************************
        PSIGUV=PCWSUB(PUMINV,PSIGMA)
        PROWUV=PCWSUB(PUPLSV,PRO)
        P=PPROD(PSIGUV,PROWUV)
        CALL PERASE(PSIGUV)
        CALL PERASE(PROWUV)
        PREAL=0
2100    IF(P.EQ.0)GO TO 2500
        PJ=PLDCF(P)
        EJ=PDEG(P)
        IF(EJ.NE.2*(EJ/2))GO TO 2200
```

```
      PTEMP1=PPOWER(PUUM1,EJ/2)
      PTEMP2=PPROD(PJ,PTEMP1)
      PTEMP=PSUM(PREAL,PTEMP2)
      CALL PERASE(PREAL)
      PREAL=PTEMP
      CALL PERASE(PTEMP1)
      CALL PERASE(PTEMP2)
2200  CONTINUE
      CALL PERASE(PJ)
      PTEMP=PRED(P)
      CALL PERASE(P)
      P=PTEMP
      GO TO 2100
2500  CONTINUE
      CALL PERASE(PUPLSV)
      CALL PERASE(PUMINV)
      CALL PERASE(PUUM1)
C*****************************************************************
C     TEST WHETHER PREAL IS NON-NEGATIVE ON (-1,+1)
C*****************************************************************
      ONE=PFA(1,0)
      RONE=RPOLY(ONE)
      RMONE=RDIF(0,RONE)
      CALL PERASE(ONE)
      RA=PFL(RONE,0)
      RA=PFL(RMONE,RA)
      SPREAL=ICWSGN(PREAL,RA)
      CALL PERASE(PREAL)
      CALL ERASE(RA)
      IF( SPREAL.GE.0) GO TO 3000
      PRINT 2700
2700  FORMAT( '0 PREAL IS NEGATIVE ON (-1,+1) ')
      GO TO 8500
C*****************************************************************
C     ALL CONDITIONS SATISFIED. METHOD IS A-STABLE
C*****************************************************************
3000  CONTINUE
      PRINT 3100
3100  FORMAT( '0 METHOD IS A-STABLE')
      GO TO 9500
C*****************************************************************
C     ONE CONDITION NOT SATISFIED. METHOD NOT A-STABLE
C*****************************************************************
8500  CONTINUE
      PRINT 8600
8600  FORMAT( '0 METHOD IS NOT A-STABLE')
      GO TO 9500
C*****************************************************************
C     TIDY UP
C*****************************************************************
9500  CONTINUE
      CALL PERASE(PSIGMA)
      CALL PERASE(PRO)
      RETURN
      END
```

```
      FUNCTION ICWSGN(P,R)
C*************************************************************
C     P IS A POLYNOMIAL IN ONE VARIABLE
C     R IS A CLOSED NONTRIVIAL RATIONAL INTERVAL
C     ICWSGN=+1 IF P.GT.0 ON R
C     ICWSGN=0  IF P.GE.0 ON R
C     ICWSGN=-1 IF P TAKES ON NEGATIVE VALUES ON R
C*************************************************************
      IMPLICIT INTEGER (A-Z)
      IN=5
      OUT=6
C*************************************************************
C     TEST WHETHER P=0
C*************************************************************
      ICWSGN=0
      IF(P.EQ.0)RETURN
C*************************************************************
C     SET UP RTWO
C*************************************************************
      ONE=PFA(1,0)
      RONE=RPOLY(ONE)
      RTWO=RSUM(RONE,RONE)
      CALL ERASE(ONE)
      CALL RERASE(RONE)
C*************************************************************
C     DETERMINE LOCATION OF ZEROS
C     P=PRODUCT( QM**M ) WHERE QM CONTAINS THE ZEROS
C     OF P OF ORDER M
C     PM=CURRENT POLYNOMIAL
C     V1=SQUARE FREE DIVISOR OF PM
C     V2=PM/V1, V3=GREATEST COMMON DIVISOR OF V1 AND V2
C     AND V4=V1/V3
C     QM=V4 AND P(M+1)=V2
C     NZ=NUMBER OF ZEROS OF P ON R
C     NM IS NUMBER OF ZEROS OF QM ON R
C     NMH IS NUMBER OF ZEROS OF QM ON HALF-OPEN R
C     NMI IS NUMBER OF ZEROS OF QM IN INTERIOR OF R
C*************************************************************
      RL=FRRST(R)
      RR=FRRST(TAIL(R))
      NZ=0
      M=0
      PM=BORROW(P)
200   IF(PDEG(PM).EQ.0)GO TO 300
C*************************************************************
C     COMPUTE QM
C*************************************************************
      M=M+1
      V1=SQFREE(PM)
      V2=PQ(PM,V1)
      V3=PGCD(V1,V2)
      V4=PQ(V1,V3)
      CALL PERASE(V1)
      CALL PERASE(PM)
      PM=V2
      CALL PERASE(V3)
```

```
      QM=V4
C*************************************************************
C     LOCATE ZEROS OF QM
C*************************************************************
      NMH=RRTS(QM,R)
      NMI=NMH
      NM=NMH
      IF(REVAL(QM,RL).EQ.0)NM=NM+1
      IF(REVAL(QM,RR).EQ.0)NMI=NMI-1
      CALL PERASE(QM)
      NZ=NZ+M*NM
      IF(M.EQ.2*(M/2))GO TO 200
      IF(NMI.EQ.0)GO TO 200
C*************************************************************
C     INTERROR ZERO OF ODD MULTIPLICITY
C*************************************************************
      ICWSGN=-1
      GO TO 500
C*************************************************************
C     P DOES NOT CHANGE SIGN
C     DETERMINE SIGN OF P BY EVALUATING AT A SEQUENCE
C     OF POINTS RT
C*************************************************************
300   RT=BORROW(RR)
350   S=REVAL(P,RT)
      IF(S.NE.0)GO TO 400
      RTEMP1=RSUM(RL,RT)
      RTEMP=RQ(RTEMP1,RTWO)
      CALL RERASE(RT)
      RT=RTEMP
      CALL RERASE(RTEMP1)
      GO TO 350
C*************************************************************
C     P NOT ZERO AT RT
C*************************************************************
400   ICWSGN=S
      IF( (ICWSGN.EQ.1) .AND. (NZ.GT.0)) ICWSGN=0
      CALL RERASE(RT)
      GO TO 500
C*************************************************************
C     TIDY UP
C*************************************************************
500   CONTINUE
      CALL PERASE(PM)
      CALL RERASE(RTWO)
      RETURN
      END
```

```
      FUNCTION ROOTCN(P)
C*************************************************************
C     TESTS WHETHER ROOT CONDITION IS SATISFIED BY P
C     ROOT CONDITION IS THAT
C     (1) ZEROS OF P LIE INSIDE OR ON THE UNIT CIRCLE
C     (2) ZEROS ON UNIT CIRCLE ARE SIMPLE
C     STRICT ROOT CONDITION SATISFIED IF ALL THE ZEROS OF P
C     LIE INSIDE THE UNIT CIRCLE
C     ROOTCN=-1 IF ROOT CONDITION NOT SATISFIED
C     ROOTCN=0 IF ROOT CONDITION SATISFIED
C     ROOTCN=1 IF STRICT ROOT CONDITION SATISFIED ( IN
C     PARTICULAR, IF P HAS DEGREE ZERO)
C*************************************************************
      IMPLICIT INTEGER (A-Z)
      IN=5
      OUT=6
C*************************************************************
C     TEST WHETHER P IS TRIVIAL
C*************************************************************
      ROOTCN=+1
      IF (PDEG(P).EQ.0)RETURN
C*************************************************************
C     USE VARIABLE Z IF NOT VARIABLE OF P. OTHERWISE USE ZZ
C*************************************************************
      LZ=PFA(35,0)
      Z=PROSYM(LZ)
      LP=PVLIST(P)
      IF( Z.NE.FIRST(LP))GO TO 15
      LZ=PFA(35,LZ)
      CALL ERASE(Z)
      Z=PPOSYM(LZ)
15    CONTINUE
      CALL ERASE(LZ)
      CALL ERASE(LP)
C*************************************************************
C     SET UP RTRANS=(Z+1)/(-Z+1)
C     AND RF=(-Z+1)**(DEG(P))
C*************************************************************
      ONE=PFA(1,0)
      PZ1=PFA(1,0)
      PZ1=PFL(ONE,PZ1)
      PZ1=PFL(Z,PZ1)
      PZ0=PQ(PZ1,PZ1)
      PPZP1=PSUM(PZ0,PZ1)
      PMZP1=PDIF(PZ0,PZ1)
      PF=PPOWER(PMZP1,PDEG(P))
      RTRANS=RPOLY2(PPZP1,PMZP1)
      RF=RPOLY(PF)
      CALL PERASE(PZ0)
      CALL PERASE(PZ1)
      CALL PERASE(PPZP1)
      CALL PERASE(PMZP1)
      CALL PERASE(PF)
C*************************************************************
C     COMPUTE Q(Z)=P((Z+1)/(-Z+1))*(-Z+1)**(DEG(P))
C*************************************************************
```

```
      RTEMP1=RPOLY(P)
      RTEMP2=RCWSUB(RTRANS,RTEMP1)
      RTEMP3=RPROD(RTEMP2,RF)
      Q=BORROW(FIRST(RTEMP3))
      CALL RERASE(RTEMP1)
      CALL RERASE(RTEMP2)
      CALL RERASE(RTEMP3)
      CALL RERASE(RTRANS)
      CALL RERASE(RF)
C*****************************************************************
C     CALL ROUTH
C*****************************************************************
      CALL ROUTH(Q,SEQ)
C*****************************************************************
C     DETERMINE IF ROOT CONDITION SATISFIED
C     NN=NUMBER OF ZEROS OF Q WITH NEGATIVE REAL PART
C     NI=NUMBER OF IMAGINARY ZEROS OF Q
C     NP=NUMBER OF ZEROS OF Q WITH POSITIVE REAL PART
C     NIS=NUMBER OF SIMPLE IMAGINARY ZEROS OF Q
C     NC=NUMBER OF ZEROS OF P ON UNIT CIPCLE
C     NCS=NUMBER OF SIMPLE ZEROS OF P ON UNIT CIRCLE
C     NCM=NUMBER OF MULTIPLE ZEROS OF P ON UNIT CIRCLE
C     NCM1=NUMBER OF ZEROS OF P AT -1
C*****************************************************************
      NN=0
      NI=0
      NP=0
      NIS=0
      IF(SEQ.EQ.0)GO TO 300
      TEMP1=SEQ
      CALL ADV(NN,TEMP1)
      CALL ADV(NI,TEMP1)
      CALL ADV(NP,TEMP1)
      IF(TEMP1.EQ.0)GO TO 300
      CALL ADV(NIS,TEMP1)
300   CONTINUE
      NCM1=PDEG(P)-PDEG(Q)
      NC=NI+NCM1
      NCS=NIS
      IF(NCM1.LE.1)NCS=NCS+NCM1
      NCM=PDEG(P)-(NN+NCS+NP)
      ROOTCN=0
      IF((NP.GT.0).OR.(NCM.GT.0))ROOTCN=-1
      IF(ROOTCN.EQ.0  .AND.  NC.EQ.0 )ROOTCN=1
      CALL ERASE(SEQ)
      CALL PERASE(Q)
      RETURN
      END
```

```
      SUBROUTINE ROUTH(U,SEQ)
C*************************************************************
C     INPUT IS THE POLYNOMIAL U
C     OUTPUT IS THE LIST SEQ
C     SEQ=0 IF U HAS DEGREE ZERO
C     SEQ(1)=NUMBER OF ZEROS OF U WITH STRICTLY
C     NEGATIVE REAL PART
C     SEQ(2)=NUMBER OF IMAGINARY ZEROS OF U
C     SEQ(3)=NUMBER OF ZEROS OF U WITH STRICTLY
C     POSITIVE REAL PART
C     SEQ(I+3)=NUMBER OF IMAGINARY ZEROS OF U WITH
C     MULTIPLICITY I
C*************************************************************
      IMPLICIT INTEGER (A-Z)
      DIMENSION W(2)
      IN=5
      OUT=6
C*************************************************************
C     TEST WHETHER U HAS DEGREE ZERO
C*************************************************************
      IF(PDEG(U).GT.0) GO TO 20
      SEQ=0
      RETURN
20    CONTINUE
C*************************************************************
C     T=U IF U HAS POSITIVE LEADING COEFFICIENT
C     T=-U OTHERWISE
C*************************************************************
      T=PABS(U)
C*************************************************************
C     SET UP W(1) AND W(2)
C*************************************************************
      W(1)=0
      W(2)=0
      NT=PDEG(T)
100   T1=PRED(T)
      T2=PDIF(T,T1)
      NT2=PDEG(T2)
      GAMMA=NT-NT2
      EPS=GAMMA/2
      TAU=GAMMA-2*EPS
      S=(-1)**EPS
      J=TAU+1
      IF(S.EQ.+1)WT=PSUM(W(J),T2)
      IF(S.EQ.-1)WT=PDIF(W(J),T2)
      CALL PERASE(W(J))
      W(J)=WT
      CALL PERASE(T)
      CALL PERASE(T2)
      T=T1
      IF(T.NE.0)GO TO 100
C*************************************************************
C     COMPUTE V,P1,P2
C*************************************************************
      V=PGCD(W(1),W(2))
      P1=PQ(W(1),V)
```

```
      P2=PQ(W(2),V)
      CALL PERASE(W(1))
      CALL PERASE(W(2))
C*************************************************************
C     LOCATE ZEROS OF V
C*************************************************************
      NV=PDEG(V)
      NI=0
      SEQ=0
      NOLD=-1
      Z=FIRST(V)
      H=V
200   NH=PDEG(H)
      IF(NH.EQ.0)GO TO 300
      HD=PDERIV(H,Z)
      HT=PGCD(H,HD)
      G=PQ(H,HT)
      N=NROOTS(G)
      ND=NOLD-N
      IF(NOLD.GE.0)SEQ=PFA(ND,SEQ)
      NOLD=N
      NI=NI+N
      CALL PERASE(H)
      H=HT
      CALL PERASE(HD)
      CALL PERASE(G)
      GO TO 200
300   CALL PERASE(H)
      IF(NOLD.GE.0)SEQ=PFA(NOLD,SEQ)
      SEQ=INV(SEQ)
      NP=(NV-NI)/2
      NN=NP
C*************************************************************
C     LOCATE ZROS OF P
C*************************************************************
      NP1=PDEG(P1)
      IF(NP1.EQ.0)GO TO 500
      DELV=0
      B1=PSIGN(P1)
      M1=PDEG(P1)
401   B2=PSIGN(P2)
      M2=PDEG(P2)
      N=M1-M2
      C=B1/B2
      E=C*(-1)**N
      IF(C.LT.0) DELV=DELV+1
      IF(E.LT.0)DELV=DELV-1
      B1=B2
      M1=M2
      DUM=PSREM(P1,P2)
      IF(DUM.EQ.0)GO TO 404
      CALL PERASE(P1)
      P1=P2
      DUM1=PCONT(DUM)
      P2=PSQ(DUM,DUM1)
      CALL PERASE(DUM)
```

```
      CALL ERLA(DUM1)
      IF( (B2.EQ.-1) .AND. (B2**N.EQ.1)) GO TO 401
      DUM=P2
      P2=PNEG(P2)
      CALL PERASE(DUM)
      GO TO 401
404   CONTINUE
      NP=NP+(NP1+DELV)/2
      NN=NN+(NP1-DELV)/2
500   CONTINUE
      CALL PERASE(P1)
      CALL PERASE(P2)
C*****************************************************************
C     STORE ANSWERS
C*****************************************************************
      SEQ=PFA(NP,SEQ)
      SEQ=PFA(NI,SEQ)
      SEQ=PFA(NN,SEQ)
      RETURN
      END
```

```
      FUNCTION RCWSUB(R,S)
C*****************************************************************
C     SUBSTITUTES THE RATIONAL FUNCTION R
C     INTO THE RATIONAL FUNCTION S
C     R AND S ARE RATIONAL FUNCTIONS IN ONE VARIABLE
C*****************************************************************
      IMPLICIT INTEGER (A-Z)
      IN=5
      OUT=6
      LR=RVLIST(R)
      LS=RVLIST(S)
      LRS=CONC(LR,LS)
      RTEMP1=RORDER(S,LRS)
      RCWSUB=RSUBST(R,RTEMP1)
      CALL RERASE(RTEMP1)
      CALL ERASE(LRS)
      RETURN
      END



      FUNCTION PCWSUB(P,Q)
C*****************************************************************
C     SUBSTITUTES THE  POLYNOMIAL P INTO THE POLYNOMIAL Q
C     P AND Q ARE POLYNOMIALS IN ONE VARIABLE
C*****************************************************************
      IMPLICIT INTEGER (A-Z)
      IN=5
      OUT=6
      LP=PVLIST(P)
      LQ=PVLIST(Q)
      LPQ=CONC(LP,LQ)
      PTEMP1=PORDER(Q,LPQ)
      PCWSUB=PSUBST(P,PTEMP1)
      CALL PERASE(PTEMP1)
      CALL ERASE(LPQ)
      RETURN
      END
```

Sample Input

```
********************      EULER'S METHOD      ********************
(+1ZETA**1-1ZETA**0)                                    RO
(+1ZETA**0)                                             SIGMA
*************      FULLY IMPLICIT METHOD      *************
(+1ZETA**1-1ZETA**0)                                    RO
(+1ZETA**1)                                             SIGMA
****************      TRAPEZOIDAL METHOD      ****************
(+1ZETA**1-1ZETA**0)                                    RO
(+1ZETA**1+1ZETA**0)                                    SIGMA
************      THIRD ORDER ADAMS MOULTON      ************
(+1ZETA**3-1ZETA**2)                                    RO
(+9ZETA**3+19ZETA**2-5ZETA**1+1ZETA**0)                 SIGMA
**************************      END      **************************
+0                                                      RO
+0                                                      SIGMA
```

Sample Output

```
******************        EULER'S METHOD        ******************

 RO=
(+1ZETA**1-1ZETA**0)

 SIGMA=
(+1ZETA**0)

 RO AND SIGMA DO NOT HAVE SAME DEGREE

 METHOD IS NOT A-STABLE

**************        FULLY IMPLICIT METHOD        **************

 RO=
(+1ZETA**1-1ZETA**0)

 SIGMA=
(+1ZETA**1)

 METHOD IS A-STABLE

****************        TRAPEZOIDAL METHOD        ****************

 RO=
(+1ZETA**1-1ZETA**0)

 SIGMA=
(+1ZETA**1+1ZETA**0)

 METHOD IS A-STABLE

***********        THIRD ORDER ADAMS MOULTON        ***********

 RO=
(+1ZETA**3-1ZETA**2)

 SIGMA=
(+9ZETA**3+19ZETA**2-5ZETA**1+1ZETA**0)

 SIGMA DOES NOT SATISFY ROOT CONDITION

 METHOD IS NOT A-STABLE
```

References

[1] Ahlfors, L. V.: Complex Analysis, second edition. New York: McGraw-Hill, 1966.

[2] Collins, G. E.: The SAC-1 system: an introduction and survey. Proc. Second Symposium on Symbolic and Algebraic Manipulation, Los Angeles, March 1971.

[3] Cryer, C. W.: A proof of the instability of backward-difference multistep methods for the numerical integration of ordinary differential equations. Technical Report No. 117, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, May 1971.

[4] Dahlquist, G. D.: A special stability problem for linear multi-step methods. BIT 3, 27-43 (1963).

[5] Gear, C. W.: Numerical Initial Value Problems in Ordinary Differential Equations. Englewood Cliffs: Prentice-Hall, 1971.

[6] Henrici, P.: Discrete Variable Methods for Ordinary Differential Equations. New York: Wiley, 1962.

[7] Lapidus, L. and J. H. Seinfeld: Numerical Solution of Ordinary Differential Equations. New York: Academic Press, 1971.

[8] Liniger, W.: A criterion for A-stability of linear multi-step integration formulae. Computing 3, 280-285 (1968).