sine integral. Let us define R_j, the round-off error, by

$$\hat{S}_{p_j} = (\hat{S}_{p_j})^* + R_j, \tag{68}$$

and let us recall, eqs. (4) and (39), that

$$S = \hat{S}_{p_j} + E_{s,j}, \tag{69}$$

where a second subscript has been added to the truncation error E_s to denote its dependence on j. Then the inequality (67) can be written

$$|E_{s,j} + R_j - E_{s,j-1} - R_{j-1}| < \epsilon(1 + |(\hat{S}_{p_j})^*|). \tag{70}$$

We see before us the heart of the difficulty. There is no intrinsic reason why the expression on the left could not be very small, even zero, through cancellation; hence satisfaction of the inequality for a particular ϵ can hardly guarantee a useful upper bound on the total error, $E_{s,j} + R_j$. We choose to bypass this difficulty in the following crude way: on the basis of eq. (36) we assume

$$E_{s,j} - E_{s,j-1} \cong -E_{s,j-1} \cong -8E_{s,j}, \tag{71}$$

and on the basis of eq. (65) we assume

$$R_j - R_{j-1} \cong R_j. \tag{72}$$

Then eq. (70) becomes

$$|-8E_{s,j} + R_j| < \epsilon(1 + |(\hat{S}_{p_j})^*|); \tag{73}$$

hence

$$8\,|E_{s,j} + R_j| - 9\,|R_j| < \epsilon(1 + |(\hat{S}_{p_j})^*|), \tag{74}$$

and

$$|E_{s,j} + R_j| < (\epsilon/8)(1 + |(\hat{S}_{p_j})^*|) + (9/8)\,|R_j|, \tag{75}$$

and, finally, with the help of eq. (65),

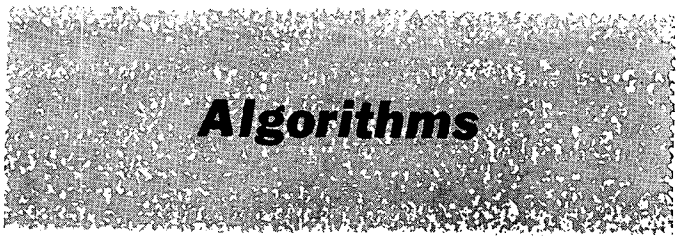
$$|E_{s,j} + R_j| < (\epsilon/8)(1 + |(\hat{S}_{p_j})^*|) + 9\cdot 2^{j-1}\rho M_0. \tag{76}$$

Now $(\hat{S}_{p_j})^*$, ρ, M_0 are all quantities which one can roughly estimate or get rough upper bounds for, and j can be replaced by MAX $-$ 4 if necessary; hence we have a relation between ϵ and the total error, $E_{s,j} + R_j$. As already noted, the result for the cosine integral is exactly the same, replacing S by C in eq. (76). This inequality can be used to determine ϵ so as to achieve a bound on the total error of the result returned by this routine. However, the crudeness of the argument leading to this result should warn the user to treat this estimate with caution.

RECEIVED OCTOBER, 1967; REVISED APRIL, 1968

REFERENCES

1. TEIJELO, LINDA. Algorithm 255. *Comm. ACM 8*, 5 (May 1965), 279.
2. FILON, L. N. G. On a quadrature formula for trigonometric integrals. *Proc. Roy. Soc. Edinburgh 49* (1928–29), 38.
3. HAMMING, R. W. *Numerical Methods for Scientists and Engineers*. McGraw-Hill, New York, 1962, p. 319.
4. FOSDICK, LLOYD D. *Math. Comput. 22* (1968), 77.
5. DAVIS, PHILIP J. *Interpolation and Approximation*. Blaisdell, Waltham, Mass., 1965.
6. LUKE, YUDELL L. On the computation of oscillatory integrals. *Proc. Cambridge Philos. Soc. 50* (1954), 269.
7. CHASE, STEPHEN M. AND FOSDICK, LLOYD D. Algorithm 353—Filon Quadrature. *Comm. ACM 12*, 8 (Aug. 1969), 457–458.



J. G. HERRIOT, Editor

The following algorithm by Chase and Fosdick relates to the paper by the same authors in the Numerical Analysis department of this issue, on pages 453–457.

This concurrent publication in Communications follows a policy announced by the Editors of the two departments, J. G. Herriot and J. F. Traub, in the March 1967 issue.

ALGORITHM 353
FILON QUADRATURE [D1]

STEPHEN M. CHASE AND LLOYD D. FOSDICK (Recd. 7 July 1967 and 6 Jan. 1969)
Department of Computer Science, University of Illinois, Urbana, IL 61820

KEY WORDS AND PHRASES: quadrature, Filon quadrature, integration, Filon integration, Fourier coefficients, Fourier series
CR CATEGORIES: 5.16

comment FSER1 evaluates the integrals

$$C = \int_0^1 F(X)\cos(M\pi X)\,dX, \qquad S = \int_0^1 F(X)\sin(M\pi X)\,dX$$

using the Filon quadrature algorithm. The user may request an evaluation of C only, S only, or both C and S. FSER1 contains an automatic error-control feature which selects an integration step size on the basis of an error parameter supplied by the user. The Filon quadrature formulas, truncation error, rounding error, and automatic error control are described in a companion paper [1] by the authors.

The calling parameters for this subroutine are defined as follows. F is the name of a FUNCTION subprogram F(X), supplied by the user, which evaluates $F(X)$ appearing in the integrand. EPS is the name for ϵ appearing in inequalities (45) and (46) of [1]. It is used in the error control portion of the algorithm. The error in the computed values of C and S is related to ϵ by the inequality (76) given in [1]. The user must assign a value to EPS before calling FSER1. MAX specifies the maximum number of halvings of the step size that are allowed. The minimum step size, h in equation (16) of [1], is 2^{-MAX}. The user must assign a value to MAX before calling FSER1. M is the parameter appearing in the argument $M\pi X$ of the cosine and sine functions. The user must assign a value to M before calling FSER1. C is the value of the cosine integral determined by FSER1. S is the value of the sine integral determined by FSER1. LC is used on entry as a signal that the user does want C evaluated (LC = 1) or does not want C evaluated (LC = 0). It is used on exit to report the value of h used by the subroutine to evaluate C, this value being 2^{-LC}. The user must assign a value of 1 or 0 to LC before calling FSER1, and if LC = 1 on entry, then the subroutine will assign a new value to LC related to the step size by 2^{-LC}. LS is used on entry as a signal that the user does want S evaluated (LS = 1) or does not want S evaluated (LS = 0). It is used on exit to report the value of

h used by the subroutine to evaluate S, this value being 2^{-LS}. The user must assign a value of 1 or 0 to LS before calling FSER1, and if LS = 1 on entry, then the subroutine will assign a new value to LS related to the step size by 2^{-LS}.

FSER1 calls a subroutine ENDT1 which is also listed below. The purpose of ENDT1 is to perform the end test described by inequalities (45) and (46) of [1].

REFERENCES:

1. FOSDICK, LLOYD D., AND CHASE, STEPHEN M. An algorithm for Filon quadrature. *Comm. ACM 12* (Aug. 1969), 453–457.


```
      SUBROUTINE FSER1(F,EPS,MAX,M, C, S, LC, LS)
      PI = 3.1415926535898
      XM = M
C F1 = COS(M*PI) TEMPORARY.
      F1 = 1 - 2 * ( M- (M/2 ) * 2 )
      F0 = F(0.0)
      F1 = F(1.0) * F1
C 'CIR' WILL BE USED THROUGHOUT THESE COMMENTS TO STAND FOR 'SIN' OR
C 'COS' WHEREVER THOSE TWO SYMBOLS MAY OCCUR.
C NOW DEFINE SUMCIR OF THE ENDPOINTS.
      SUMCOS = (F1 + F0 ) * .5
      SUMSIN = 0.0
      B1 = 2. / 3.
C TMAX IS THE SWITCH-OVER POINT IN THE ANGLE T.
C OUR ANALYSIS INDICATES THAT TMAX = 1/6 IS THE BEST FOR THE ILLIAC II
C WHICH HAS A 44 BIT FLOATING POINT MANTISSA.
      TMAX = 0.166
C N IS THE NUMBER OF THE ITERATION. NOTE THAT WE START AT THE
C FOURTH ITERATION STEP.
C ACTUALLY, THE FIRST EVALUATION OF AN INTEGRAL IS AT N = 5, AND
C THEREFORE, THE FIRST COMPARISON OF VALUES IS AT N= 6.
      N = 4
C BOTH TMAX AND N MAY BE CHANGED IF THE MACHINE FOR WHICH THIS
C ROUTINE IS INTENDED HAS GREATER OR LESS ACCURACY THAN ILLIAC II.
C IF N IS CHANGED , THEN THE CORRESPONDING CHANGES MUST BE MADE
C IN THE ASSIGNMENTS OF H AND NSTOP.
      H = 1. / 16.
C H = 2 ** -N.
      NSTOP = 15
C NSTOP = 2**N - 1
      T = H * XM
      TP = T * PI
      NST = 1
      ASSIGN 67 TO MSWTCH
C LLC AND LLS ARE USED BY THE ROUTINE IN COMPUTED-GO-TO STATEMENTS.
C AS SOON AS LLS AND LLC HAVE BEEN DEFINED, WE CAN USE LS AND LC
C AS RETURN PARAMETERS (SEE ABOVE).
      IF ( LS ) 1, 1, 2
 1    LLS = 2
      GO TO 3
 2    LLS = 1
      LS = MAX
 3    IF ( LC ) 4, 4, 5
 4    LLC = 2
      GO TO 7
 5    LLC = 1
      LC = MAX
 7    LN = 1
C ALL OF THE ABOVE IS EXECUTED ONLY ONCE PER CALL.
C NOW THE ITERATION BEGINS.
 10   ODCOS = 0.
      ODSIN = 0.
C BEGIN SUMMATION FOR ODCOS AND ODSIN.
      DO 65 I = 1, NSTOP, NST
        XI = I
        THA = XI * T
C THA*PI IS THE ANGLE USED IN THIS ITH TERM.
```


(Continued-see next column)

(Concluded)


```
C CIR(I*T*PI) IS CALCULATED HERE USING THE IDENTITY
C CIR ( INTEGER MULTIPLE OF PI + FRACTIONAL MULT OF PI )
C = COS(INTEGER*PI) * CIR(FRAC*PI)
C = (+ OR -) * CIR(FRAC*PI).
        FRAC = THA
        IN = THA
        THA = IN
        FRAC = (FRAC - THA) * PI
C THA IS A FLOATING POINT INTEGER, FRAC IS THE FRACTIONAL PART *PI.
        COSIP = 1 - 2*(IN - 2*(IN/2))
        TEMP1 = COSIP * F(XI*H)
C TEMP1 = COS(INTEGER PART) * F(I*H).
        GO TO ( 50 , 55 ) , LLS
 50     ODSIN = TEMP1 * SIN(FRAC) + ODSIN
 55     GO TO ( 60 , 65 ) , LLC
 60     ODCOS = TEMP1 * COS(FRAC) + ODCOS
 65     CONTINUE
      GO TO MSWTCH,(67,70)
 67   NST = 2
C NOW HAVE MADE UP FOR THE FIRST 4 ITERATION STEPS, SO RESET THESE
C TWO NUMBERS TO LOOK LIKE THE GENERAL CASE.
      NSTOP = 16
C NSTOP = 2**N (IN CASE YOU CHANGE STARTING VALUE OF N).
      ASSIGN 70 TO MSWTCH
      GO TO 92
 70   TSQ = TP*TP
      IF (T -TMAX) 74, 74, 75
C 74 IS THE POWER SERIES FOR SMALL T, 75 IS THE CLOSED FORM USED WITH
C LARGER VALUES OF T.
C THE POWER SERIES ARE (WITH 'TN' = TP**N)
C A = (2./45.)*T3 - (2./315.)*T5 + (2./4725.)*T7
C B = (2./3.) + (2./15.)*T2 - (4./105.)*T4 + (2./567.)*T6
C     - (4./22275.)*T8
C G = (4./3.) - (2./15.)*T2 + (1./210.)*T4 - (1./11340.)*T6
C THE NEXT TERM IN G IS TOO SMALL. IT IS (1./997920.)*T8
 74   A = TP * TSQ * (1. - TSQ * (1. - TSQ / 15.) / 7.) / 22.5
      B2 = B1 * TSQ* .2
      B3 = B2 * TSQ * 2./7.
      B4 = B3 * TSQ / 10.8
      B5 = B4 * TSQ * 14./275.
      B = B1+ B2 - B3 + B4 - B5
      G = 2.*B1 - B2+ B3/ 8. - B4/40.
C G = 2.*B1 - B2 + B3/8. - B4/40. + 5.*B5/896. IF YOU WANT THE T8
C TERM INCLUDED IN G.
      GO TO 80
C CLOSED FORM OF THE COEFFICIENTS, WHERE AGAIN 'TN' MEANS TP**N.
C A = 1./TP + COS(TP)*SIN(TP)/T2 - 2.*(SIN(TP))**2/T3
C B = 2.*((1 + (COS(TP))**2)/T2 - 2.*SIN(TP)*COS(TP)/T3)
C G = 4.*(SIN(TP)/T3 - COS(TP)/T2)
 75   IN = T
      TEMP1 = 1 - 2 * ( IN - 2 * ( IN / 2 ) )
      TEMP2 = IN
C TEMP1 IS COS ( INTEGER PART OF TP), TEMP2 IS FRACTIONAL PART OF TP.
      TEMP2 =(T - TEMP2 ) * PI
      S1 = TEMP1 * SIN (TEMP2)
C S1 = SIN(TP)
      C1 = TEMP1 * COS (TEMP2)
C C1 = COS(TP)
      P = S1 * C1
      S1SQ = S1 * S1
      A = (((-2.*S1SQ/TP) + P)/TP +1.)/ TP
      B = 2. * ((-2.* P/ TP)+ 2. -S1SQ) / TSQ
      G = 4. * (S1 / TP - C1)/ TSQ
 80   GO TO (81, 85), LLS
C HAVE CALCULATED THE COEFFICIENTS, NOW READY FOR THE INTEGRATION
C FORMULAS.
 81   T2 = H* (A * (F0 - F1) + B * SUMSIN + G * ODSIN)
C ENDT1 IS A SUBROUTINE WHICH CHECKS FOR THE CONVERGENCE OF THE
C ITERATIONS. ENDT1 REQUIRES THE PRESENT VALUE TO AGREE WITH THE
C PREVIOUS VALUE TO WITHIN EPS2, WHERE
C EPS2 = (1.0 + ABSF(PRESENT VALUE))*EPS
C EPS IS SUPPLIED BY THE USER.
      CALL ENDT1   (PVT2, T2, EPS, S, LLS, LN)
      GO TO ( 85, 84 ), LLS
 84   LS = N
 85   GO TO (86,90),LLC
C THIS IS THE COSINE INTEGRAL.
 86   T1 = H * ( B * SUMCOS + G * ODCOS)
      CALL ENDT1   (PVT1, T1, EPS, C, LLC, LN)
      GO TO ( 90, 89 ), LLC
 89   LC = N
 90   LN = 2
C NOW TEST TO SEE IF DONE.
      IF (LLC + LLS - 3) 92, 92, 100
 92   N = N + 1
C THIS IS THE BEGINING OF THE ITERATION.
      IF (N-MAX) 95, 95,100
 95   H = .5 * H
      T = .5 * T
      TP = .5 * TP
      NSTOP = 2 * NSTOP
      SUMSIN = SUMSIN + ODSIN
      SUMCOS = SUMCOS + ODCOS
      GO TO 10
 100  S = T2
      C = T1
      RETURN
      END
      SUBROUTINE ENDT1   (PREVQT, QUANT,EPS, VALUE, L1, L2)
      GO TO ( 29, 20), L2
 20   REPS = EPS * (1.0 + ABS(QUANT))
 23   IF (ABS(PREVQT - QUANT) - REPS) 25, 25, 29
 25   VALUE = QUANT
      L1 = 2
      GO TO 30
 29   PREVQT = QUANT
 30   RETURN
      END
```