Algorithms

H. J. WEGSTEIN, Editor

ALGORITHM 88
EVALUATION OF ASYMPTOTIC EXPRESSION FOR THE FRESNEL SINE AND COSINE INTEGRALS

John L. Cundiff
Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Ga.

real procedure FRESNEL (u) Result: (frcos, frsin); **value** (u);
comment This procedure evaluates the Fresnel sine and cosine integrals for large u by expanding the asymptotic series given by

$$S(u) = \frac{1}{2} - \frac{\cos(x)}{\sqrt{2\pi x}}\left[1 - \frac{1\cdot 3}{(2x)^2} + \frac{1\cdot 3\cdot 5\cdot 7}{(2x)^4} - \cdots\right] - \frac{\sin(x)}{\sqrt{2\pi x}}\left[\frac{1}{2x} - \frac{1\cdot 3\cdot 5}{(2x)^3} + \frac{1\cdot 3\cdot 5\cdot 7\cdot 9}{(2x)^5} - \cdots\right]$$

and

$$C(u) = \frac{1}{2} - \frac{\sin(x)}{\sqrt{2\pi x}}\left[1 - \frac{1\cdot 3}{(2x)^2} + \frac{1\cdot 3\cdot 5\cdot 7}{(2x)^4} - \cdots\right] - \frac{\cos(x)}{\sqrt{2\pi x}}\left[\frac{1}{2x} - \frac{1\cdot 3\cdot 5}{(2x)^3} + \frac{1\cdot 3\cdot 5\cdot 7\cdot 9}{(2x)^5} - \cdots\right]$$

in which $x = \pi u^2/2$. Reference: Pearcey, T. *Table of the Fresnel Integral to Six Decimal Places.* The Syndics of the Cambridge University Press, Melbourne, Australia (1956).;

```
begin   pi := 3.14159265; arg := pi × (u↑2)/2; temp := 1;
        argsq := 1/(4 × (arg↑2)); term := −3 × argsq;
        series := 1 + term; N := 3;
first:  if temp = series then go to second; temp := series;
        termi := term;
        term := −termi × (4 × N − 7) × (4×N − 5) × (argsq);
        if abs(term) > abs(termi) then go to second;
        series := temp + term; N := N + 1; go to first;
second: series2 := ½ × arg; temp := 0; term := series2;
        N := 2;
loop:   if series2 = temp then go to exit; termi := term;
        term := −termi × argsq × (4×N−5) × (4×N−3);
        if abs(term) > abs(termi) then go to exit;
        temp := series2; series2 := temp + term;
        N := N + 1; go to loop;
exit:   if u < 0 then half := −½ else half := ½;
        frcos := half + (sin(arg) × series − cos(arg) + series2)/
        (pi × u);
        frsin := half − (cos(arg) × series2 + sin(arg) × series)/
        (pi × u)
end FRESNEL;
```

ALGORITHM 89
EVALUATION OF THE FRESNEL SINE INTEGRAL

John L. Cundiff
Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Ga.

real procedure FRESNELSIN (u) Result: (frsin); **value** u;
comment This algorithm computes the Fresnel sine integral defined by,

$$S(u) = \int_0^u \sin \pi t^2/2 \, dt,$$

by evaluating the series expansion

$$S(x) = \sqrt{\frac{2x}{\pi}}\left[\frac{x}{3} - \frac{x^3}{7\cdot 3!} + \frac{x^5}{11\cdot 5!} - \frac{x^7}{15\cdot 7!} + \cdots\right]$$

where $x = \pi u^2/2$. Reference: Pearcey, T. *Table of the Fresnel Integral to Six Decimal Places.* The Syndics of the Cambridge University Press, Melbourne, Australia (1956).;

```
begin   Pi2 := 1.5707963; x := Pi2 x (u↑2); frsin := x/3;
        frsqr := x↑2; N := 3; term := (−x × frsqr)/6;
        frsini := frsin + term/7;
Loop:   if frsin = frsini then go to exit; frsin := frsini;
        term := −term × frsqr/((2×N−1) × (2×N−2));
        frsini := frsin + term/(4×N−1); N := N + 1;
        go to Loop;
exit:   frsin := frsini × u
end FRESNELSIN;
```

Contributions to this department must be in the form stated in the Algorithms Department policy statement (*Communications*, February, 1960) except that ALGOL 60 notation should be used (see *Communications*, May 1960). Contributions should be sent in duplicate to J. H. Wegstein, Computation Laboratory, National Bureau of Standards, Washington 25, D. C. Algorithms should be in the Reference form of ALGOL 60 and written in a style patterned after the most recent algorithms appearing in this department. For the convenience of the printer, please underline words that are delimiters to appear in boldface type.

ALGORITHM 90
EVALUATION OF THE FRESNEL COSINE INTEGRAL

JOHN L. CUNDIFF
Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Ga.

real procedure FRESNELCOS (u) result: (frcos); **value** (u);
comment This algorithm computes the Fresnel cosine integral defined by

$$C(u) = \int_0^u \cos \frac{\pi t^2}{2} \, dt,$$

by evaluating the series expansion

$$C(u) = \sqrt{\frac{2x}{\pi}} \left[1 - \frac{x^2}{5 \cdot 2!} + \frac{x^4}{9 \cdot 4!} - \frac{x^6}{13 \cdot 6!} + \cdots \right],$$

where $x = \pi u^2/2$. Reference: PEARCEY, T. *Table of the Fresnel Integral to Six Decimal Places.* The Syndics of the Cambridge University Press, Melbourne, Australia (1956).;

```
begin  pi2 := 1.5707963;  x := pi2 × (u↑2);  frcos := 1;
       xsqr := x↑2;  N := 3;  term := −xsqr/2;
       frcoi := 1 + (term/5);
loop:  if frcoi = frcos then go to exit;  term := −term ×
       xsqr/((2×N−2 × (2×N−3));  frcos := frcoi;  frcoi :=
       frcos + term/(4×N−3);  N := N + 1;  go to loop;
exits: frcos := u × frcos
end FRESNELCOS;
```

ALGORITHM 91
CHEBYSHEV CURVE-FIT

ALBERT NEWHOUSE
University of Houston, Houston, Texas

procedure CHEBFIT(m, n, X, Y); **integer** m, n; **array** X, Y;
comment This procedure fits the tabular function $Y(X)$ (given as m points (X, Y)) by a polynomial $P = \sum_{i=0}^{n} A_i X^i$. This polynomial is the best polynomial approximation of $Y(X)$ in the Chebyshev sense. Reference: STIEFEL, E. *Numerical Methods of Tchebycheff Approximation,* U. of Wisc. Press (1959), 217–232;

```
begin array X[1:m], Y[1:m], T[1:m], A[0:n], AX[1:n+2],
        AY[1:n+2], AH[1:n+2], BY[1:n+2], BH[1:n+2];
    integer array IN [1:n+2];  real TMAX, H;  integer i,
        j, k, imax;
    comment Initialize;
    k := (m−1)/(n+1);
    for 1 := 1 step 1 until n+1 do IN [i] := (i−1)×k + 1;
    IN[n+2] := m;
    START: comment Iteration begins;
    for i := 1 step 1 until n+2 do
        begin AX[i] := X[IN[i]];
              AY[i] := Y[IN[i]];
              AH[i] := (−1)↑(i−1)
        end i;
    DIFFERENCE: comment divided differences;
    for i := 2 step 1 until n+2 do
        begin
        for j := i−1 step 1 until n+2 do
        begin BY[j] := AY[j];
              BH[j] := AH[j]
        end j;
        for j := i step 1 until n+2 do
        begin AY[j] := (BY[j] −BY[j−1])/
                  (AX[j] −AX[j−i+1]);
              AH[j] := (BH[j] −BH[j−1])/
                  (AX[j] −AX[j−i+1])
        end j;
     end i;
     H := −AY[n+2]/AH[n+2];
     POLY: comment polynomial coefficients;
     for i := 0 step 1 until n do
         begin A[i] := AY[i] +AH[i] ×H;
               BY[i] := 0
         end i;
     BY[1] := 1;  TMAX := abs(H);  imax := IN[1];
     for i := 1 step 1 until n do
         begin
         for j := 0 step 1 until i−1 do
             begin
             BY[i+1−j] := BY[i+1−j] −BY[i−j] ×X[IN[i]];
             A[j] := A[j] +A[i] ×BY[i+1−j]
             end j;
         end i;
     ERROR: comment compute deviations;
     for i := 1 step 1 until m do
         begin T[i] := A[n];
         for j := 0 step 1 until n do T[i] := T[i] X[i] +A[n−j];
         T[i] := T[i] −Y[i];
         if abs(T[i]) ≤ TMAX then go to L1;
         TMAX := abs(T[i]);
         imax := i
L1:      end i;
     for i := 1 step 1 until n+2 do
         begin
         if imax < IN[i] then go to L2;
         if imax = IN[i] then go to FIT end
         end i;
L2:  if T[imax] × T[IN[i]] < 0 then go to L3;
     IN[i] := imax;
     go to START;
L3:  if IN[1] < imax then go to L4;
     for i := 1 step 1 until n+1 do IN[n+3−i] := IN[n+2−i];
     IN[i] := imax;
     go to START;
L4:  if IN[n+2] ≤ imax then go to L5;
     IN[i−2] := imax;
     go to START;
L5:  for i := 1 step 1 until n+1 do IN[i] := IN[i+1];
     IN[n+2] := imax;
     go to START;
FIT: end CHEBFIT
```

CERTIFICATION OF ALGORITHM 60
ROMBERG INTEGRATION (F. L. Bauer, *Comm. ACM*, June 1961)

KARL HEINZ BUCHNER
Lurgi Gesellschaft fur Mineraloltechnik m.b.H., Frankfurt, Germany

Since August 1961, the Rombert Integration has been successfully applied in FORTRAN language to various problems on an IBM 1620. Due to its elegant method and the memory saving features, the Romberg Integration has succeeded other methods in our program library, e.g., the Newton-Cotes integration of order 10.

Reference is made to Stiefel, *Numerische Mathermatik* (Teubner Verlag. Stuttgart). Stiefel discusses in his book various methods of numerical integration including the Romberg algorithm.

[*ALGORITHMS ARE CONTINUED ON PAGE 286*]

Observations and Conclusions

The following observations and inferences are based on the initial experiences with a multiprogramming system:

1. It has been possible to make a significant improvement in the utilization of the main-frame time of the computer with large classes of real problems running at Lewis Research Center.

2. The interrupt feature is the basic tool that permits the automatic time sharing of independently coded and unrelated problems.

3. A clock that interrupts periodically would be a highly desirable tool to be used in the management of more general multiprogramming systems. Such a clock would be useful in preventing a "looping" problem from "hogging" the computer, as well as for running-time accounting.

4. Multiprogramming provides the incentive for more efficient problem and system codes by removing the output barrier. Previously, there was little motivation to prepare efficient codes in output-limited problems, since there was little advantage to be gained. The multiprogramming system has provided a method of exploiting such gains and has produced pressure for more efficient codes.

5. Operator attention to the input-output devices, such as a tape change, no longer need delay the computer. The computer can push ahead any problem that is not using the device undergoing the change.

6. Multiprogramming seems to develop a trend towards splintering of problems and their input-output tasks. This appears to have two advantages for multiprogramming in computers of limited store and input devices. First, it provides a much better opportunity to obtain a feasible mix of problems in the high-speed memory. Splintering of problems or data-reduction tasks will increase the possibility of getting compatible problems for a parallel-operation schedule.

7. Scheduling of parallel problems could be facilitated by an inexpensive procedure for code relocation. It would be desirable to relocate the codes of a current problem in the high-speed memory without more than a nominal delay in computation.

8. On-line debugging of problems is costly of computer time and an inefficient use of computer capabilities. However, from an individual problem standpoint, it is often an effective method of debugging. Parallel operation during on-line debugging, if it could be made safe, may reduce the cost of on-line debugging to a point where it is feasible.

REFERENCE

1. Turner, L. R., and Rawlings, J. H. Realization of randomly timed computer input and output by means of an interrupt feature. *IRE Trans. EC-7*, no. 2 (June 1958), 141–149.

ALGORITHM 92
SIMULTANEOUS SYSTEM OF EQUATIONS AND MATRIX INVERSION ROUTINE

Derek Johann Roek
Applied Physics Laboratory of Johns Hopkins University, Silver Spring, Maryland


```
procedure SIMULTANEOUS (U, W, C, X, B, n, kount, eps,
  absf) ;
array U, W, C, X, B ; integer n, kount ;
  real eps; real procedure absf;
comment This procedure solves the problem Ux := b for the
  vector x. It assumes the problem written in the form x'U' := b',
  where ' denotes transpose. The procedure is completed in n
  cycles and may be iterated kount times (kount ≦ 6). The trans-
  pose of U is in U[,] and the row vector b' is in B. The integer n
  is the dimension of U, and the solution row vector x' is in X.
  The matrix C is a check of accuracy. It should have b' in its
  first row, the first element b1 of b' along its main diagonal,
  and zeros elsewhere. The real number eps checks to see how close
  the actual result is to this theoretical one. Also if we let b' :=
  (1, 0, ··· , 0), then this procedure finds the inverse W[,] of U.
  The function absf finds the absolute value of its argument. The
  procedure chooses the column vectors of U as the row vectors of
  W in the 0th cycle of the first iteration. For all subsequent itera-
  tions, the row vectors of W, computed at the nth cycle of the
  last iteration, are the row vectors of W in the 0th cycle ;
begin integer i, j, k, p ; real bh, b1, Z ;
  for j := 1 step 1 until n do
      for i := 1 step 1 until n do W[j, i] := U[i, j];
  S1: for j := 1 step 1 until n do
        for i := 1 step 1 until n do C[i, j] := 0 ;
  for j := 1 step 1 until n do
      begin for k := 1 step 1 until n do
        begin C[j, j] := C[j, j] + W[j, k] × U[k, j] end;
        if j = 1 then Z := B[j]/C[j, j] else Z := 1/C[j, j];
        for k := 1 step 1 until n do
            begin X[k] := Z × W[j, k];
              W[j, k] := X[k]
            end k;
        for k := 1 step 1 until n do
            begin if k = j then go to S2 else
              for p := 1 step 1 until n do
                  C[k, j] := C[k, j] + U[p, j] × W[k, p];
            if j = 1 then bh := B[j] else bh := 1;
            if k = 1 then b1 := B[j] else b1 := 0;
            for p := 1 step 1 until n do
            begin X[p] := bh × W[k, p] + (b1 − C[k, j]) ×
            W[j, p];
              W[k, p] := X[p]
            end p;
  S2:        if k = j ∧ j = n then go to S3
            end k;
      end j;
  S3: for j := step 1 until n do
          if absf(absf(C[j, j]) − absf(B[1])) > eps then go to S4;
      go to S6;
  S4: if kount > 0 then go to S5 else go to S6;
  S5: kount := kount − 1;
      go to S1;
  S6: for j := step 1 until n do
          X[j] := W[1, j];
  S7: end SIMULTANEOUS
```