Algorithms Supplement

Previously published algorithms

The following algorithms have recently appeared in the Algorithms Sections of the specified journals.

(*a*) **Communications of the ACM** (October–December 1968)

338 ALGOL PROCEDURES FOR THE FAST FOURIER TRANSFORM

Two procedures, based on the Cooley–Tukey algorithm, are given. The first computes either the complex Fourier transform or its inverse. The second either computes the Fourier coefficients of a sequence of real data points or evaluates a Fourier series with given cosine and sine coefficients.

339 AN ALGOL PROCEDURE FOR THE FAST FOURIER TRANSFORM WITH ARBITRARY FACTORS

Computes the finite Fourier transform for one variate of dimension nv within a multivariate transform of n complex data values.

340 ROOTS OF POLYNOMIALS BY A ROOT-SQUARING AND RESULTANT ROUTINE

Finds simultaneously zeros of a polynomial of degree n with real coefficients by a root-squaring and resultant routine. It supersedes Algorithm 59.

341 SOLUTION OF LINEAR PROGRAMS IN 0–1 VARIABLES BY IMPLICIT ENUMERATION

Solves the integer linear program

$$\text{minimise } A[0,1] \times x[1] + \ldots + A[0,n] \times x[n]$$
$$\text{subject to } A[i,1] \times x[1] + \ldots + A[i,n] \times x[n] + A[i,0] \geqslant 0 (i = 1, \ldots, m)$$
$$\text{and} \quad x[j] = 0 \text{ or } 1 \ (j = 1, 2, \ldots, n).$$

342 GENERATOR OF RANDOM NUMBERS SATISFYING THE POISSON DISTRIBUTION

Generates a pseudo-random number in the interval 0, 1 *and finds the number px such that*

random $\leqslant$ (*probability that the number is px or less*)
and
random $>$ (*probability that the number is px* $-$ 1 *or less*).

343 EIGENVALUES AND EIGENVECTORS OF A REAL GENERAL MATRIX

Finds all the eigenvalues and eigenvectors of a real general matrix. The eigenvalues are computed by the QR double-step method and the eigenvectors by inverse iteration.

(*b*) **Applied Statistics** (March 1969)

AS8 MAIN EFFECTS FROM A MULTI-WAY TABLE

Calculates the general mean and main effects from a multi-way table stored in standard order and stores them end to end in another array. Algorithm AS9 *performs the inverse operation.*

AS9 CONSTRUCTION OF ADDITIVE TABLE

Produces a multi-way table from a general mean and set of additive main effects supplied in a 1*-way array. Algorithm* AS8 *performs the inverse operation.*

AS10 THE USE OF ORTHOGONAL POLYNOMIALS

Computes the values of successive polynomials $P_0, P_1, \ldots, P_{n-1}$ *orthogonal to a given set of n x-values. The subroutine produces the polynomial values one set at a time, successive calls yielding successive polynomials.*

AS11 NORMALISING A SYMMETRIC MATRIX

Given a symmetric matrix X, stored in lower triangular form, the subroutine computes the diagonal matrix Y whose values are

$$0 \quad \text{if} \quad X(i,i) = 0$$
$$[X(i,i)]^{-1/2} \quad \text{otherwise.}$$

It then overwrites X by the normalised form YXY.

AS12 SUMS OF SQUARES AND PRODUCTS MATRIX

Accumulates the weighted or unweighted means and corrected sums of squares and products matrix for variate values presented unit by unit. An auxiliary subroutine must be supplied by the user to get the data values for successive units.

(*c*) **Numerische Mathematik** (September–November 1968)

RATIONAL CHEBYSHEV APPROXIMATION USING LINEAR EQUATIONS

Generates best, in the Chebyshev sense, weighted rational approximations to continuous functions.

The following papers, containing useful algorithms, have recently appeared in the specified journals.

(*a*) **Numerische Mathematik** (November 1968)

TRIDIAGONALISATION OF A SYMMETRIC BAND MATRIX (Band 12, Heft 4, pp. 231–241)

(*b*) **BIT** (July 1968)

ON THE PRIME ZETA FUNCTION (Bind 8, Hefte Nr. 3, pp. 187–202)

New algorithms

Algorithm 39

AREAS UNDER THE NORMAL CURVE

A. G. Adams
Glaxo Research Ltd.
Greenford

Author's Note:

The procedure *normaltails* evaluates

$$t = \frac{1}{\sqrt{2\pi}} \int_z^\infty exp\left(-\frac{u^2}{2}\right) du$$

using two different approximations.

The first approximation is valid for $z < 1 \cdot 28$ and takes the form

$$t = 0 \cdot 5 - zf(z^2)$$

where f is a rational function.

The second approximation is valid for $z \geqslant 1{\cdot}28$ and takes the form

$$t = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{z^2}{2}\right) f(z)$$

where f is again a rational function. Both approximations minimax the relative error in t.

The procedure was tested on an ICL1903. The results were compared with tables, Abramowitz and Stegun (1964), and with the algorithm by Hill and Joyce (1967). The observed errors were within the limits stated in the comment.

Although no timings have been made, it is expected that the procedure will be fast when $z < 1{\cdot}28$ and at least as fast as an iterative method in other cases.

References

ABRAMOWITZ, M., and STEGUN, I. A. (1964). *Handbook of Mathematical Functions*. National Bureau of Standards.

HILL, I. D., and JOYCE, S. A. (1967). Algorithm 304, Normal Curve Integral, *Communications of the ACM*, Vol. 10, No. 6, p. 374.

```
procedure normaltails (z, ltail, rtail); value z;
real z, ltail, rtail;
comment normaltails calculates the areas under the normal
curve. On exit ltail equals the area from — infinity to z and
rtail equals the area from z to infinity. The smaller of the two
areas has relative error < 1·0₁₀ − 10 and the larger has
absolute error < 1·0₁₀ − 11;
if z = 0·0 then ltail := rtail := 0·5 else
  begin real t, y; Boolean s;
  s := z > 0·0; z := abs(z);
  y := 0·5 × z × z;
  if z < 1·28 then
  t := 0·5 − z × (0·398942280444 − 0·399903438504
  × y/(y + 5·75885480458 − 29·8213557808/
  (y + 2·62433121679 + 48·6959930692 /(y +
  5·92885724438))))
  else
    begin
     y := 0·398942280385 × exp (− y);
     t := if y / z = 0·0 then 0·0 else
     y / (z − 0·000000038052 + 1·00000615302 /
     (z + 0·000398064794 + 1·98615381364 /
     (z − 0·151679116635 + 5·29330324926 /
     (z + 4·83859128080 − 15·1508972451 /
     (z + 0·742380924027 + 30·7899330340 /
     (z + 3·99019417011))))))
    end;
  if s then
    begin
    ltail := 1·0 − t; rtail := t
    end
  else
    begin
    ltail := t; rtail := 1·0 − t
    end
end normaltails
```

Algorithm 40

SPLINE INTERPOLATION OF DEGREE THREE

H. Späth
Institut für Neutronenphysik
und Reaktortechnik
Kernforschungszentrum Karlsruhe
Germany

Author's Note:

If we consider the problem of interpolating between given data points (x_i, y_i), $i = 1, 2, \ldots, n$ with $x_1 < x_2 < \ldots < x_n$ by means of a function s that is smooth in the sense that $s \in C^2[x_1, x_n]$ and $\int_{x_1}^{x_n} [s''(x)]^2 dx$ is a minimum, we know that there is a unique s with these properties, see Greville (1967), and that s consists in each interval $[x_i, x_{i+1}]$, $i = 1, 2, \ldots, n-1$ of a third degree polynomial with $s''(x_1) = s''(x_n) = 0$. It is clear that s is completely determined by the conditions $s(x_i) = y_i$, $i = 1, 2, \ldots, n$ and the values $s''(x_i)$, $i = 2, 3, \ldots, n-1$.

The values $s''(x_i)$, $i = 2, 3, \ldots, n-1$ are obtained by solving the tridiagonal system of linear equations

$$\Delta x_{i-1} s''(x_{i-1}) + 2(\Delta x_i + \Delta x_{i-1}) s''(x_i) + \Delta x_i s''(x_{i+1}) = 6\left(\frac{\Delta y_i}{\Delta x_i} - \frac{\Delta y_{i-1}}{\Delta x_{i-1}}\right) \quad i = 2, \ldots, n-1$$

The matrix of coefficients in this system is symmetric and strictly diagonally dominant. Varga (1962) has shown that such a matrix is positive definite and Martin *et al.* (1963) have shown that in this case the elimination method does not need pivoting and is numerically stable.

As s'' is a linear function in each interval $[x_i, x_{i+1}]$, $i = 1, 2, \ldots, n-1$ it is easy to calculate $s''(t)$ for any t with $x_1 \leqslant t \leqslant x_n$. Thus in order to integrate s over $[A, B]$ with $x_1 \leqslant A \leqslant B \leqslant x_n$ one makes use of the formula

$$\int_\alpha^\beta s(x)dx = \frac{1}{2}(\beta - \alpha)(s(\alpha) + s(\beta)) - \frac{1}{24}(\beta - \alpha)^3(s''(\alpha) + s''(\beta))$$

which is valid for $x_i \leqslant \alpha < \beta \leqslant x_{i+1}$, $i = 1, 2, \ldots, n$.

References

GREVILLE, T. N. E. (1967). Spline Functions, Interpolation and Numerical Quadrature; *Volume II, Mathematical Methods for Digital Computers*. New York: McGraw-Hill Book Co.

VARGA, R. S. (1962). *Matrix Iterative Analysis*, Englewood Cliffs: Prentice Hall, Inc.

MARTIN, R. S., PETERS, G., and WILKINSON, J. H. (1965). Symmetric Decompositions of a Positive Definite Matrix, *Numerische Mathematik*, Vol. 7, pp. 362–82.

```
procedure splinefit (n, x, y, f2, kenn, der, m, t, f, EXIT);
value n, m, kenn, der; integer n, m, kenn; Boolean der;
array x, y, f2, t, f; label EXIT;
comment For n given data points (x₁, y₁), (x₂, y₂), . . ., (xₙ, yₙ)
this procedure calculates for kenn < 0 the values f2[i] = s''(xᵢ),
i = 1, 2, . . , n that characterise the natural spline function of
degree three through the given points. If kenn = 0 the pro-
cedure also gives at the m abscissae t[j] with x₁ ⩽ t₁ ⩽ . . .
⩽ tₘ ⩽ xₙ the interpolated values f[j] = s(tⱼ) and further if
der = true the values f[m + j] = s'(tⱼ). For kenn > 0 the
array f2 is an input parameter and the interpolation and
optionally the differentiation at the given values tⱼ is performed.
The formal label EXIT is used if either n < 3 or t₁ < x₁
or tₘ > xₙ;
  begin integer i, j, n1, n2, k; real z, h1, h2, h3, h4;
  array h, dy[1 : n], s[1 : n − 1], e[1 : n − 2];
  if n < 3 then goto EXIT;
  n1 := n − 1; n2 := n − 2;
  for i := 1 step 1 until n1 do
    begin
    h[i] : = x[i + 1] − x[i]; dy[i] := (y[i + 1] − y[i]) / h[i]
    end;
  if kenn > 0 then goto CALCULATE;
  f2[1] := f2[n] := 0;
```

```
for i := 2 step 1 until n1 do f2[i] := 6·0 × (dy[i] − dy[i−1]);
z := 0·5 / (h[1] + h[2]); s[1] := − h[2] × z;
e[1] := f2[2] × z; k := 1;
for i := 2 step 1 until n2 do
  begin
  j := i + 1; z := 1·0 / (2·0 × (h[i] + h[j]) + h[i]×s[k]);
  s[i] := − h[j] × z; e[i] := (f2[j] − h[i] × e[k]) × z;
  k := i
  end;
f2[n1] := e[n2];
for i := n2 step −1 until 2 do
  begin
  k := i − 1; f2[i] := s[k] × f2[i + 1] + e[k]
  end;
if kenn < 0 then goto FINALE;
CALCULATE:
if t[1] < x[1] ∨ t[m] > x[n] then goto EXIT;
for i := 1 step 1 until n1 do s[i] := (f2[i + 1] − f2[i]) / h[i];
i := 2; k := 1;
for j := 1 step 1 until m do
  begin
LABEL: if t[j] > x[i] then
    begin
    k := i; i := i + 1;
    goto LABEL
    end
  else
    begin
    h1 := t[j] − x[k]; h2 := t[j] − x[i];
    h3 := h1 × h2; h4 := f2[k] + h1 × s[k];
    z := (f2[i] + f2[k] + h4) / 6·0;
    f[j] := y[k] + h1 × dy[k] + h3 × z;
    if der then f[m + j] := dy[k] + z × (h1 + h2)
      + h3 × s[k] / 6·0
    end
  end;
FINALE:
  end splinefit
```

Contributions for the Algorithms Supplement should be sent to

Mrs. M. O. Mutch
University Mathematical Laboratory
Corn Exchange Street
Cambridge