Algorithms Supplement

Previously published algorithms

The following Algorithms have been published in the *Communications of the Association for Computing Machinery* during the period January–June 1966.

273 SERREV

Produces the coefficients of the power series $y^i = \sum_{i=j}^{N} C_{ij}x^i$ *where y is the solution of* $f(y) = \sum_{i=1}^{N} A_i y^i = g(x) = \sum_{i=1}^{N} B_i x^i$ *and* $A_1 = 1$.

274 GENERATION OF HILBERT DERIVED TEST MATRIX

Produces n × n matrices a with the properties

(1) *The elements a [i,j] are positive integers*
(2) *The inverse has elements* $(-1) \uparrow (i+j) \times a[i,j]$
(3) *The degree of ill-conditioning increases rapidly with n.*

275 EXPONENTIAL CURVE FIT

Fits a curve defined by the equation $y = a \times exp(b \times x) + c$ *to a set* $[x_i, y_i]$ *of n data points, using the Taylor Series modification of the classical least squares method.*

276 CONSTRAINED EXPONENTIAL CURVE FIT

Fits a curve defined by the equation $y = a \times exp(b \times x) + c$ *to a set* $[x_i, y_i]$ *of n data points, using the Taylor Series modification of the classical least squares method, constraining the curve to pass through the point* (x_k, z).

277 COMPUTATION OF CHEBYSHEV SERIES COEFFICIENTS

Approximates the first N+1 coefficients, a_n, *of the infinite Chebyshev series expansion of a function F(x) defined on* [−1,1].

278 GRAPH PLOTTER

Gives an approximate graphical display of a multivalued function y [i,j] of x [i], on a line printer.

279 CHEBYSHEV QUADRATURE

Evaluates the integral of f(x) between a and b by fitting the 2^{n+1} *point Chebyshev polynomial to the integrand.*

280 ABSCISSAS AND WEIGHTS FOR GREGORY QUADRATURE

281 ABSCISSAS AND WEIGHTS FOR ROMBERG QUADRATURE

282 DERIVATIVES OF e^x/x, $cos(x)/x$ AND $sin(x)/x$

283 SIMULTANEOUS DISPLACEMENT OF POLYNOMIAL ROOTS IF REAL AND SIMPLE

Computes the n roots x of a polynomial equation simultaneously with quadratic convergence.

284 INTERCHANGE OF TWO BLOCKS OF DATA

Transfers the contents of a [1], . . . , a [m] into a [n+1], . . . , a [n+m] while simultaneously transferring the contents of a [n+1], . . . , a [n+m] into a [1], . . . , a [n] without using an appreciable amount of auxiliary memory.

285 THE MUTUAL PRIMAL-DUAL METHOD

Solves the linear programming problem by the Mutual Primal-Dual Simplex method.

286 EXAMINATION SCHEDULING

A heuristic examination time-tabling procedure for scheduling m courses in n time periods.

Algorithms

Algorithm 12. *SCALECHOLESKI*

Miss C. M. Devine,
Medical Research Council,
Computer Services Centre.

```
procedure scalecholeski(B,n,scale,l);value n,scale;integer n,
scale,l;array B;

comment scalecholeski inverts a symmetric positive definite
matrix of order n stored as an upper triangle by rows in loca-
tions B[0] to B[n×(n+1)/2−1]. If scale = 1 each row and
column is scaled so that the diagonal elements are between
1 and 100, and the matrix is rescaled after inversion. On
exit l normally contains 0 but if the matrix is singular, or very
nearly so, l will contain the number of the row of the matrix on
which the inversion process breaks down. The expression
"very nearly singular" means that rounding errors in the
computing have been sufficiently large to make one of the
leading minors of the matrix either zero or negative.
    The procedure contains three sub-procedures which find the
scale factors, scale the matrix, and invert it. The inversion
process is the Choleski method which is described in Davies,
"Statistical Methods in Research and Production", 3rd Ed.,
Oliver and Boyd, 1957, Chapter 8, Appendix 8B. The comput-
ing method is based on London University Library Routine 900
written by I. M. Kabhaza in EMA and CHLF3;

begin integerarray P[1:n];
  procedure findscalefactors(B,P,n);value n;array B;
    integerarray P;integer n;
  begin integer p,i,j,k;real x;
    k:=0;
    for i:=1 step 1 until n do
```

```
    begin j:=n-i+1;x:=B[k];p:=0;
      if x≤0 then goto L13;
      if x>100·0 then goto L12;
      if x>1 then goto L13;
      L11:x:=100·0×x;p:=p+1;if x<1 then goto L11;
        goto L13;
      L12:x:=0·01×x;p:=p-1;if x>100 then goto L12;
      L13:P[i]:=p;k:=k+j
    end
  end;

  procedure scalematrix(B,P,n);value n;array B;
    integer array P;integer n;
  begin integer i,j,k,p;
    k:=0;
    for i:=1 step 1 until n do for j:=i step 1 until n do
    begin p:=P[i]+P[j];B[k]:=B[k]×10↑p;k:=k+1
    end
  end;

  procedure choleski(B,n,l);value n;array B;integer n,l;
  begin real d,x;integer m,j,p,q,r,s,t;
    comment first stage of inversion is to replace the matrix B
      by U where B=U'U and U is an upper triangular matrix;
    r:=l:=0;
    for p:=1 step 1 until n do for q:=p step 1 until n do
    begin x:=B[r]; if q=1 then goto L4;if p=1 then goto L2;
      s:=p-1;t:=q-1;
      for j:=2 step 1 until p do
      begin x:=x-B[s]×B[t];
        s:=s+n-j+1;t:=t+n-j+1
      end;if p≠q then goto L2;
      L4:if x≤0 then begin l:=p;goto L15 end;
      d:=1/sqrt(x);
      L2:B[r]:=x×d;r:=r+1
    end;

    comment second stage is to replace U by its inverse;
    t:=0;
    for q:=1 step 1 until n do
    begin B[t]:=1/B[t];s:=0;
      for p:=1 step 1 until q-1 do
      begin d:=0;r:=s+q-p-1;m:=r+1;
        for j:=s step 1 until r do
        begin d:=B[j]×B[m]+d;m:=m+n-p+s-j
        end;B[r+1]:=-B[t]×d;s:=s+n-p+1
      end;t:=t+n-q+1
    end;

    comment third stage forms inverse of B as product of
      inverses of U and U';
    r:=0;for p:=1 step 1 until n do
    begin s:=r;for q:=p step 1 until n do
      begin m:=r+n-q;d:=0;
        for j:=r step 1 until n do
        begin d:=B[j]×B[s]+d;s:=s+1
        end;B[r]:=d;r:=r+1
      end
    end;
  L15:end;

  if scale=1 then
  begin findscalefactors(B,P,n);scalematrix(B,P,n)
  end;
  choleski(B,n,l);
  if scale=1 then scalematrix(B,P,n)
end
```

Algorithm 13. *NORMALAREA* A. Bergson, Computing Laboratory, Sunderland Technical College.

Author's note:

NORMALAREA will find the area under a normal curve for a given ordinate t, i.e. $\phi(x)\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{t}e^{-\frac{1}{2}x^2}dx$. *NORMALAREA* actually computes $\phi_1(x)=\frac{1}{\sqrt{2\pi}}\int_{0}^{|t|}e^{-\frac{1}{2}x^2}dx$, and if $t>0$ then $\phi(x)=0\cdot5+\phi_1(x)$, and for $t\leqslant 0$ $\phi(x)=0\cdot5-\phi_1(x)$.

If $t<4$ then $\phi_1(x)$ is calculated from the power series of the integral, i.e.

$$\phi_1(x)=\frac{1}{\sqrt{2\pi}}\left[t-\frac{t^3}{2\times3\times1!}+\frac{t^5}{2^2\times5\times2!}-\frac{t^7}{2^3\times7\times3!}+\frac{t^9}{2^4\times9\times4!}-\cdots\right]$$

using a recurrence relation and stopping when the modulus of the difference of two successive terms is $\leqslant 10^{-9}$.

If $t\geqslant 4$ an asymptotic series, as derived by Scarborough (1), is used; viz.

$$\phi_1(x)=\frac{1}{2}-\frac{e^{-\frac{1}{2}t^2}}{\sqrt{2\pi}t}\left[1-\frac{1}{t^2}+\frac{1\times3}{t^4}-\frac{1\times3\times5}{t^6}+\frac{1\times3\times5\times7}{t^8}-\cdots\right]$$

Terms in parenthesis are taken until the modulus of a term is $\geqslant$ the modulus of the preceding term. Again a recurrence type relation is used.

NORMALAREA was coded in ALGOL for a National-Elliott 803, and gave satisfactory results correct to eight significant figures as checked with *Biometrika Tables* (2). A similar procedure by MacLaren (3), procedure *PHI*, was found to use more than three times the storage taken by *NORMALAREA*, but was considerably faster. However, using the parameters given in *PHI*, *NORMALAREA* was more accurate.

References

(1) SCARBOROUGH, J. B. (1950). *Numerical Mathematical Analysis*, pp. 391–4.

(2) PEARSON, E. S., and HARTLEY, H. O. (1958). *Biometrika Tables for Statisticians*, pp. 104–10.

(3) MACLAREN, M. D. (1965). "Procedure for the Normal Distribution Functions. Algorithm 272", *Communications of the Association for Computing Machinery*, Vol. 8, No. 12, pp. 789–90.


```
real procedure NORMALAREA(t);
value t; real t;
begin
  comment the constants 2·506628275 (root 2 pi) and 10^-9
    should be quoted according to the accuracy of the machine on
    which the procedure is implemented;
  real u1,u2,u3,a,a1,tsq,tsq2; integer i;
  a1:= abs(t); tsq:= t × t; tsq2:= ·5 × tsq;
```

```
  a:= 0;
  if a1 < 4 then
  begin
    u1:= a1;
    for i:= 1, i + 1 while abs(u3 − u2) > 10^−9 do
    begin
      u2:= tsq2 × (1 − i − i) × u1/(i × (1 + i + i));
      a:= a + u1;  u3:= u1;  u1:= u2
    end of i;
    a:= a/2·506628275
    end for t < 4
  else
  begin
    u1:= 1;
    for i:= 1, i + 1 while abs(u2) < abs(u3) do
    begin
      u2:= − u1 × i/tsq;
      a:= a + u1;  u3:= u1;  u1:= u2
    end of i;
    a:= ·5 − a × exp(− tsq2)/(a1 × 2·506628275)
  end for t ⩾ 4;
  if t > 0 then NORMALAREA:= ·5 + a else
    NORMALAREA:= ·5 − a
end of procedure NORMALAREA
```

Algorithm 14. *SCANNET* B. J. Benzimra, Ministry of Aviation, London.

Author's note:

This algorithm is offered, in response to the plea of J. Boothroyd (1), as an example of a non-trivial recursive procedure which arose naturally.

Reference

(1) BOOTHROYD, J. (1965). "PERM. Algorithm 6", *The Computer Bulletin*, Vol. 9, No. 3.

```
procedure scannet(i,j); value i,j; integer i,j;
  begin comment this processes a valid event-activity network
        consisting of nE events interconnected by nA activities.
        At entry to the procedure the network is defined by the
        global integer arrays Aterm, Adur, Achain [1:nA],
        Echain, Encom, Edat [1:nE] as follows:—
        Aterm[j] contains the terminating event number for
          activity j
        Adur [j] contains the estimated duration of activity j
        Echain and Achain list, in chained form, the activity
              numbers m,n,p, . . . , z leading out of event
              k so that Echain[k]=m, Achain[m]=n,
              Achain[n]=p . . . with the end of the chain
              designated by zero, Achain[z]=0.
        Encom[k] contains the total number of activities
              leading into event k, unless k is a source
              event for which Encom[k]=−1. The pro-
              cedure reduces to zero the entries for non-
              source events.
        Edat[k] contains the starting date, expressed as an
              integer in suitable time units relative to some
              datum, for all source events k. For non-
              source events Edat[k]=0.
        At exit from the procedure Edat[k] contains, for all k,
        the earliest occurrence date for each event while
        Acomp[1:nA] contains the earliest completion date of
        each activity. For a network with a single source
        event s the single call scannet(Echain[s],s) is sufficient.
        For a network with multiple sources the procedure
        must be called once for each source event. This may
        be conveniently accomplished by
          for i:=1,i+1 while i⩽nE do if Encom[i]=−1 then
        scannet(Echain[i],i). Only one variable is needed as
        workspace irrespective of the depth of recursion and
        integer k is thus global to the procedure;
    if i ≠ 0 then
      begin k:=Aterm[i];
        Acomp[i]:=Edat[j]+Adur[i];
        Edat[k]:=if Edat[k]>Acomp[i]then Edat[k] else
          Acomp[i];
        Encom[k]:=Encom[k]−1;
        if Encom[k]=0 then scannet(Echain[k],k);
        scannet(Achain[i],j)
      end
  end of procedure scannet
```

Algorithm 15. *GRAM* R. J. Ord-Smith, Computing Laboratory, Bradford Institute of Technology.

```
procedure Gram (c,d,m,n,e); value m,n; array c,d;
integer m,n; label e;
comment The first four parameters are input parameters and
c,d are also output parameters. If at the call of the procedure
these arrays contain the coefficients of the (n−1)th and (n−2)th
degree Gram polynomials respectively (produced by a previous
call of the procedure) in order of increasing powers of x, and
followed by the normalizing factors of the polynomials, the
procedure replaces these coefficients and normalizing factors by
those of the nth and (n−1)th Gram polynomials and their
corresponding normalizing factors. If called with n=1 the
first and zeroth polynomials and their factors are initialized.
The user must, at call time, specify a label as his last parameter
to which the procedure will exit if it encounters the situation
n=0 or n=m.
  The Gram polynomials p(m,n,x) with n=0 (1) m−1 are a set
of polynomials possessing the property of discrete orthogonality
with respect to the equidistant key points x=−l,−l+1, . . . ,
0, . . . ,l−1,l where m=2×l+1.
  Although the construction is effectively using the well known
recurrence relation relating three successive Gram polynomials,
this algorithm describes the simpler explicit generation of the
coefficients;
if n=0 v n=m then goto e else
if n=1 then
begin d[0]:= 1; d[1]:= m; comment this is the normalizing
    factor for the zeroth polynomial;
    c[0]:= 0; c[1]:= 2/(m−1);
    c[2]:= m×(m+1)/(3×(m−1)); comment this is the n.f.
    for the first polynomial;
end     else
begin real a,b,t; integer i,k;
    k:= n×(m−n);
    a:= (4×n−2)/k;
    b:= −(n−1)×(m+n−1)/k;
    t:= c[0]; c[0]:= b×d[0];
    d[0]:= t; t:= c[1];
    for i:= 1 step 1 until n−2 do
    begin c[i]:= a×d[i−1] + b×d[i];
        d[i]:= t; t:= c[i+1]
    end;
```

```
    c[n − 1]: = b × d[n − 2];
    d[n − 1]: = t; d[n]: = c[n];
    c[n]: = a × t;
    c[n + 1]: = (m + n) × (2 × n − 1) × d[n]/((m − n) ×
        (2 × n + 1));
    comment this is the n.f. for the nth polynomial;
end
```

Editor's note

Material for this Supplement should be sent to the Algorithms Editor

P. Hammersley,
The City University,
St. John Street,
London, E.C.1.

Correspondence (*continued from p. 320*)

This is always possible because U is similar to a symmetric matrix. The solution of (2) when $l = 0$ is

$$w(t) = \left[\exp\left(-\frac{1}{(\Delta x)^2}\,Ut\right)\right]c$$

$$= \sum_{i=0}^{N} \alpha_i z_i \exp\left(\frac{-\lambda_i t}{(\Delta x)^2}\right) \tag{7}$$

or $$w(n\Delta t) = \sum_{i=0}^{N} \alpha_i z_i [\exp(-r\lambda_i)]^n. \tag{8}$$

On the other hand the solution of (4) with $k = 0$ is

$$v^n = \{[I + r\theta U]^{-1}[I - r(1-\theta)U]\}^n c \tag{9}$$

$$= \sum_{i=0}^{N} \alpha_i z_i \left(\frac{1 - r(1-\theta)\lambda_i}{1 + r\theta\lambda_i}\right)^n. \tag{10}$$

The replacement of (2) by (4) therefore replaces each factor $\exp(-r\lambda_i)$ in one time step of (8) by a factor

$$\frac{1 - r(1-\theta)\lambda_i}{1 + r\theta\lambda_i} \tag{11}$$

whence the result follows.

This simple relationship between the stability properties of equations (2) and (4) does *not* necessarily persist when Crank and Nicolson's procedure with $\theta = \frac{1}{2}$ is applied to a non-linear or non-autonomous problem (Rosenbrock and Storey, 1965, pp. 173–175). Some formulae giving improved stability and truncation error have been suggested in an earlier note (Rosenbrock, 1963).

References

PARKER, I. B., and CRANK, J. (1964). "Persistent discretization errors in partial differential equations of parabolic type", *The Computer Journal*, Vol. 7, pp. 163–167.

KEAST, P., and MITCHELL, A. R. (1966). "On the instability of the Crank–Nicholson formula under derivative boundary conditions", *The Computer Journal*, Vol. 8, pp. 110–114.

CRANK, J., and NICOLSON, P. (1947). "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type", *Proc. Camb. Phil. Soc.*, Vol. 43, pp. 50–67.

ROSENBROCK, H. H., and STOREY, C. (1965). *Computationa Techniques for Chemical Engineers*, pp. 8–15 (Pergamon Press).

ROSENBROCK, H. H. (1963). "Some general implicit processes for the numerical solution of differential equations" *The Computer Journal*, Vol. 5, pp. 329–330.

Yours faithfully,

H. H. ROSENBROCK

Control Systems Centre,
University of Manchester,
Institute of Science and Technology,
Sackville Street, Manchester 1.
16 June 1966.

To the Editor,
The Computer Journal.

Sir,

I should like to reply to the letter by K. Wright (this *Journal*, May 1966, p. 115) about my paper entitled "Error curves for Lanczos' 'selected points' method" (this *Journal*, January 1966, p. 372). I apologise for stating that Wright's statement about the form for the residual (this *Journal*, January 1964, p. 358) is incorrect. His letter clearly shows the source of my confusion.

However, I do not agree with the simpler derivation in the letter for the form of the residual. Although

$$r(x) = \dot{e}(x) - e(x)\frac{\partial F}{\partial y} \cdots$$

there is no justification in dropping the whole right-hand side except for the first term. The derivation given in my paper based on the Picard iteration does show how errors build up.

Incidentally, there are two typographical errors in my paper. In Table 4, H_{31} should read $-0{\cdot}089142227$ instead of $-0{\cdot}08142227$. In Table 5, G_{41} should read $0{\cdot}37699459$ instead of $0{\cdot}376994519$.

Yours sincerely,

W. KIZNER

Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive,
Pasadena,
California 91103
1 August 1966

Don't be surprised

autumn

I can brush away your punching problems too!

(... Cards or Paper Tape)

With a staff of over 400 highly trained punch operators, it is not surprising that A.D.P. is recognised as the most efficient dataprep organization in the United Kingdom. A nation-wide delivery and collection service and the strict adherence to accuracy and time schedule contributes largely to the increasing success of A.D.P. Whatever your punching problem, large or small, A.D.P. can deal with it swiftly and economically at any of their regional service centres.

SERVICE CENTRES

ENGLAND
LONDON – Annabelle House, 28 Staines Road, Hounslow. Phone HOU 3294. Telex 262063.
BRISTOL – Southey House, Wine Street, Bristol, 1. Phone Bristol 26813. Telex 44657.
MANCHESTER – Royal Buildings, 2 Mosley Street, Manchester. Phone Central 5803.
BRADFORD – Oak Mills, Clayton, Bradford. Phone Queensbury 3496.
DERBY – Laurie House, Colyear Street, Derby. Phone Derby 48109.
NORTHAMPTON – 39-41, Bridge Street, Northampton. Phone Northampton 39345.
SCOTLAND
BATHGATE – Gardners Lane, South Bridge Street, Bathgate, West Lothian. Phone Bathgate 3927

ADP **AUTOMATIC DATA PROCESSING LTD.**

Head Office: Annabelle House, 28 Staines Rd., Hounslow, Middlesex. Phone Hounslow 3294

A.D.P. For the best punching in the business!