Technical Report no. 7
Theory of Condensed Matter Group

A COMPUTER PROGRAM FOR NUMERICAL
SOLUTION OF THE ELLIASHBERG EQUATIONS TO FIND $T_c$

$TCM/4/1974$

by

Philip B. Allen*

Contents

I Formulation p. 2
II Description of the Programs p. 4
III References p. 7
IV Sample Calling Program, Data and Output p. 8
V Listing of Subroutines
   SOLVE p. 10
   F(X) p. 12
   AINVIT p. 13
   CHOLDE p. 15
   CHOLSU p. 16
VI Description of Matrix Manipulation Programs p. 17

* Alfred P. Sloan Fellow. Permanent address: Dept. of Physics, SUNY, Stony Brook, New York 11790. Work supported in part by U.S. National Science Foundation grant no. GH037925
I FORMULATION

The superconducting transition temperature \( T_c \) can be theoretically obtained from the Eliashberg equations\(^1\) if the interaction parameters \( \alpha^2 F(\omega) \) and \( \mu^* \) (characterizing electron-phonon and Coulomb interactions respectively) are known. A simplified version of these equations has been presented by Allen and Dynes\(^2\) based on the work of Bergmann and Rainer\(^3\). The equations are

\[
\rho(T) \psi_m = \sum_{n=0}^{\infty} K_{mn}^{(T)} \psi_n
\]

\[
K_{mn} = \lambda^*(m-n) + \lambda^*(m+n+1) - 2\mu^* - \delta_{mn} \left[ 2m+1 + \lambda^*(0) \right] + 2 \sum_{k=1}^{m} \lambda^*(k) \right]
\]

\[
\lambda^*(k) = 2 \int d\omega \frac{\omega^2 F(\omega)}{\omega^2 + (2\pi kT)^2}
\]

where the eigenvectors \( \psi \) are related to the gap parameter \( \Delta(\omega_m) \equiv \Lambda_m \). The value of \( T_c \) is defined as that temperature where the maximum eigenvalue \( \rho(T) \) becomes zero. In searching for the maximum eigenvalue of a matrix \( K_\psi \) of infinite dimension, it is useful to note that the sequence of maximum eigenvalues \( \rho_N \) of the principal minors \( K_{NN} \) (of order \( N \) by \( N \)) form a series of monotonically increasing bounds to \( \rho \). The procedure used for solving eq (1) for \( T_c \) is to solve successively the problems truncated at \( N = 1, 2, 4, 8, 16, 32, \) and 64. We find that the solution of order 64 x 64 has accurately converged for materials with \( T_c / \sqrt{\omega^2} > 0.01 \) or \( \lambda > 0.5 \), where \( \lambda \) is the electron-phonon coupling strength

\[
\lambda = 2 \int d\omega \omega^2 F(\omega) / \omega = \lambda^*(0)
\]
In practice it is not easy to calculate $T_c$ directly from $\alpha^2 F$ and $\mu^*$. It is much more convenient to assume a value for $T_c$ and vary the amplitude of $\alpha^2 F$ (i.e. vary $\lambda$ keeping the shape of $\alpha^2 F$ fixed.) This way a curve of $T_c$ as a function of $\lambda$ (for fixed $\mu^*$) can be generated, and the value of $T_c$ can be graphically interpolated from the known value of $\lambda$. The phonon coupling enters the kernel (eq.2) only through the function $\lambda^*$ (eq.3). We define a function $f(x)$ which is a normalized version of $\lambda^*$

$$\lambda^*(\lambda) = \lambda f\left(\frac{2\pi kT}{\sqrt{\langle \omega^2 \rangle}}\right)$$  \hspace{1cm} (5)

$$f(x) = \frac{2}{\lambda} \int_{\omega=0}^{\infty} 2\omega^2 F(\omega) \sqrt{\omega^2 + \langle \omega^2 \rangle}$$  \hspace{1cm} (6)

$$\langle \omega^2 \rangle = \frac{2}{\lambda} \int_{\omega=0}^{\infty} \omega^2 F(\omega)$$  \hspace{1cm} (7)

where $\sqrt{\langle \omega^2 \rangle}$ is a convenient rms phonon frequency. The function $f(x)$ takes the value 1 at $x = 0$ and goes asymptotically to $1/x^2$ at large $x$. For intermediate values of $x$, $f$ is a monotonically decreasing function whose precise shape contains the information about the shape of $\alpha^2 F$ which is relevant to $T_c$. We next split the kernel $K$ into two parts, $A + \lambda B$, where $A$ and $B$ are both independent of $\lambda$.

$$A_{mn} = -2\mu^* - (2m+1)\delta_{mn}$$  \hspace{1cm} (8)

$$B_{mn} = f(2\pi(m-n)T_c/\sqrt{\langle \omega^2 \rangle})$$

$$+ f(2\pi(m+n+1)T_c/\sqrt{\langle \omega^2 \rangle})$$

$$- \delta_{mn} \left[ 1 + 2 \sum_{k=1}^{r} f\left(\frac{2\pi k T_c}{\sqrt{\langle \omega^2 \rangle}}\right) \right]$$  \hspace{1cm} (9)
The procedure now becomes to assume a value for $T_c/\sqrt{\omega^2}$, construct the principle minors $A_N$ and $B_N$, and finally to solve for the coupling strength $\lambda_N$ in $N^{th}$ order using

$$(A_N + \lambda_N B_N) \lambda_N = 0$$

(10)

**II DESCRIPTION OF THE PROGRAMS**

The actual setting up and solving of eq.(10) is done by a subroutine **SOLVE** (TC, EL, CC, NALF), where TC is the assumed value of $T_c/\sqrt{\omega^2}$ and EL is an array of dimension 7 which returns seven successive approximations to $\lambda$ obtained by solving $\frac{E_N}{\lambda_N}$ with dimension $N = 2^0, 2^1, ..., 2^6 = 64$. The parameter CC is the given value of $\mu^*$, which we define by

$$\mu^* = \mu / \left( 1 + \mu \ln(\omega_p / \sqrt{\omega^2}) \right)$$

(11)

There is some confusion about $\mu^*$ in the literature. The purpose of using $\mu^*$ instead of $\mu$ is to make the effective Coulomb cutoff equal to the phonon cutoff. The Coulomb cutoff is approximately the electron plasma frequency $\omega_p$, while the phonon cutoff has been taken to mean two distinct things: either a frequency of order the maximum phonon frequency or the actual maximum frequency at which integration is cut off in a computer program. The second meaning is more natural in many ways but suffers from the disadvantage of arbitrariness. Therefore we have opted for the former meaning. Fortunately $\mu^*$ is not sensitive to small variations (such as replacing $\omega_p$ by $\omega_p$ or $\sqrt{\omega^2}$ by $\omega$). However, some authors, notably
McMillan have used instead of $\sqrt{<\omega^2>}$, a cutoff about 10 times larger. This is enough to affect $\mu^*$ by 30%. In the actual computations described here, an effective $\mu^*(N)$ is used which depends on the truncation point $\omega_N = 2\pi NT_c$, namely

$$\mu^*(N) = \mu^*/\left[ 1 + \mu^* \ln(\sqrt{<\omega^2>}/\omega_N) \right]$$  \hspace{1cm} (12)

This procedure was found to be advantageous to obtain rapid convergence and eliminate small fluctuations in $T_c(\lambda)$ which tend to occur if a fixed $\mu^*$ is used and truncated at some value of $\omega_n$.

The parameter NALF should be read in as 1 if the data $(\alpha^2 F)$ is new and as some other integer if previously read data is to be re-used.

The subroutine SOLVE obtains the function $f(x)$ through a function subprogram called F(X). This program has three different exit points. If the argument X is negative, the subprogram F does not actually calculate $f(x)$ but instead performs the preliminary operations. These consist of reading data and calculating various parameters which are immediately printed out. These are summarized below.

**Data:**

- TITLE, NAF, DOM (FORMAT 3AH, I5, F5.2)
- AF(I), I = 1, NAF (FORMAT 10E8.3)

**TITLE** = name of material such as LEAD

**NAF** = number of data points for $\alpha^2 F$

**DOM** = increment $\Delta \omega$ of data points in meV.

**AF** = $\alpha^2 F$
Print-out:  TITLE

ELA = \lambda

OM1 = \langle \omega \rangle \text{ in meV}

OM2 = \langle \omega^2 \rangle \text{ in meV}^2

OM4 = \langle \omega^4 \rangle \text{ in meV}^4

When the argument X is positive, the subprogram F calculates 

\[ f(x) \] 
either by Simpson's rule integration (when \( x \leq 5 \)) or by the 
first two terms of a large \( x \) expansion (when \( x > 5 \)), namely

\[ f(x) \approx \left( 1 - \langle \omega^4 \rangle / \langle \omega^2 \rangle^2 \right) / x^2 \]  \hspace{1cm} (13)

The final step is to solve eq.\((10)\). For this purpose a general 
program was used called AINVIT, which was written by C.M.M. Nex. This 
program solves for \( \lambda \) very efficiently using inverse iteration. At 
each stage (except the first one, \( N = 1 \)) the previous value \( \lambda_{N-1} \) 
and eigenvector \( \psi_{N-1} \) are used as the initial trial values for the 
\( N \)th stage. The actual solution is performed by a subroutine named 
CHOLSU, after Cholski decomposition has been performed by a subroutine 
named CHOLDE. Both of these programs were also written by C.M.M. 
Nex. A description and listing of the programs AINVIT, CHOLDE, 
and CHOLSU is attached.

The package described here is completely self-contained. The 
user needs to provide only a brief calling program and data. A 
sample calling program, data, and output are given in the next 
section. The data shown are \( \xi^2 F(\omega) \) for lead as measured by Rowell 
and McMillan \(^5\). Several choices of \( \mu^* \) and \( T_c / \sqrt{\langle \omega^2 \rangle} \) have been made 
which illustrate the convergence, which is very rapid for \( \mu^* = 0 \) 
and \( T_c / \sqrt{\langle \omega^2 \rangle} = 0.1 \). The convergence is only slightly less rapid when 
\( \mu^* = 0.1 \). Convergence is slower for small \( T_c \). When
\( \mu^* = 0.1 \) at \( T_c/\sqrt{\langle \omega^2 \rangle} = 0.01 \), the approximate limit of this program has still not been reached, but convergence will cease being adequate for somewhat smaller values of \( T_c \). To handle smaller \( T_c \), the matrix would have to be enlarged beyond 64 x 64, and perhaps a coarser mesh than the exact Matsubara points \( \omega_n \) could be used. There is no limit on the maximum permissible value of \( T_c/\sqrt{\langle \omega^2 \rangle} \) which this program can handle. The output lists \( T_c/\langle \omega \rangle \) as well as \( T_c/\sqrt{\langle \omega^2 \rangle} \), and gives not only the actual \( \mu^* \) (eq.11) but also the running values of \( \mu^*(N) \) (eq.12). The computing time on an IBM 370 machine for finding seven successive approximations to \( \lambda \) for a given \( T_c \) and \( \mu^* \) is less than 1 sec.

**ACKNOWLEDGEMENTS:** The help of C.M.M. Nex has been invaluable in constructing these programs. Permission to use his programs AINVIT, CHOLDE, and CHOLSU is gratefully acknowledged.

**III REFERENCES**

1. G.M. Eliashberg, JETP 11, 696 (1960); 12, 1000 (1960).
   Also see for example, J.R. Schrieffer, *Superconductivity* (W.A. Benjamin, New York, 1964) and D.J. Scalapino, in *Superconductivity*, edited by R.D. Parks (M. Dekker, New York, 1969).
2. P.B. Allen and R.C. Dynes, to be published.
4. W.L. McMillan, Phys. Rev. 167, 331 (1968);
IV SAMPLE CALLING PROGRAM, DATA, AND OUTPUT

Calling Program

0001 COMMON OM1, OM2, OM4
0002 DIMENSION EL(7)
0003 TC=-.08
0004 DO 20 I=1,2
0005 TC=TC+.09
0006 CC=-.10
0007 DO 20 J=1,2
0008 CC=CC+.1
0009 NALF=1+J-I
0010 CALL SOLVE(TC, EL, CC, NALF)
0011 20 CONTINUE
0012 STOP
0013 END

Data

LEAD 111 .10
.101E-1 .112E-1 .137E-1 .148E-1 .177E-1 .196E-1 .243E-1 .297E-1 .389E-1 .481E-1
.599E-1 .694E-1 .799E-1 .884E-1 .984E-1 .107E-0 .126E-0 .134E-0 .154E-0 .177E-0
.216E-0 .274E-0 .355E-0 .443E-0 .538E-0 .629E-0 .722E-0 .804E-0 .852E-0 .874E-0
.899E-0 .918E-0 .960E-0 .101E 1 .103E 1 .997E-0 .939E-0 .907E-0 .864E-0 .819E-0
.760E-0 .698E-0 .638E-0 .590E-0 .552E-0 .515E-0 .473E-0 .440E-0 .415E-0 .398E-0
.381E-0 .365E-0 .348E-0 .343E-0 .345E-0 .350E-0 .356E-0 .360E-0 .361E-0 .363E-0
.367E-0 .377E-0 .386E-0 .401E-0 .418E-0 .450E-0 .500E-0 .566E-0 .650E-0 .742E-0
.851E-0 .980E-0 .113E 1 .124E 1 .126E 1 .114E 1 .928E-0 .698E-0 .490E-0 .324E-0
.201E-0 .123E-0 .883E-1 .664E-1 .612E-1 .590E-1 .568E-1 .507E-1 .436E-1 .377E-1
.345E-1 .321E-1 .290E-1 .252E-1 .214E-1 .180E-1 .150E-1 .116E-1 .843E-1 .711E-2
.560E-2
Output

LEAD
ELA = 1.549 DM1 = 5.2 DM2 = 0.310E+02 DM4 = 0.146E+04

GIVEN TC = 0.1000E-01 IN UNITS OF RMS OMEGA
TC = 0.1067E-01 IN UNITS OF OMEGA
MUSTAR = 0.0
MATRIX DIM = 1 LAMBDA = 0.1000E+01 MSTAR = 0.0
MATRIX DIM = 2 LAMBDA = 0.6105E+00 MSTAR = 0.0
MATRIX DIM = 4 LAMBDA = 0.4439E+00 MSTAR = 0.0
MATRIX DIM = 8 LAMBDA = 0.3628E+00 MSTAR = 0.0
MATRIX DIM = 16 LAMBDA = 0.3281E+00 MSTAR = 0.0
MATRIX DIM = 32 LAMBDA = 0.3179E+00 MSTAR = 0.0
MATRIX DIM = 64 LAMBDA = 0.3163E+00 MSTAR = 0.0

GIVEN TC = 0.1000E-01 IN UNITS OF RMS OMEGA
TC = 0.1067E-01 IN UNITS OF OMEGA
MUSTAR = 0.1000
MATRIX DIM = 1 LAMBDA = 0.1166E+01 MSTAR = 0.0763
MATRIX DIM = 2 LAMBDA = 0.7453E+00 MSTAR = 0.0828
MATRIX DIM = 4 LAMBDA = 0.5738E+00 MSTAR = 0.0879
MATRIX DIM = 8 LAMBDA = 0.4996E+00 MSTAR = 0.0936
MATRIX DIM = 16 LAMBDA = 0.4804E+00 MSTAR = 0.1001
MATRIX DIM = 32 LAMBDA = 0.4888E+00 MSTAR = 0.1075
MATRIX DIM = 64 LAMBDA = 0.4985E+00 MSTAR = 0.1162

GIVEN TC = 0.1000E+00 IN UNITS OF RMS OMEGA
TC = 0.1067E+00 IN UNITS OF OMEGA
MUSTAR = 0.0
MATRIX DIM = 1 LAMBDA = 0.1557E+01 MSTAR = 0.0
MATRIX DIM = 2 LAMBDA = 0.1111E+01 MSTAR = 0.0
MATRIX DIM = 4 LAMBDA = 0.1007E+01 MSTAR = 0.0
MATRIX DIM = 8 LAMBDA = 0.9912E+00 MSTAR = 0.0
MATRIX DIM = 16 LAMBDA = 0.9898E+00 MSTAR = 0.0
MATRIX DIM = 32 LAMBDA = 0.9897E+00 MSTAR = 0.0
MATRIX DIM = 64 LAMBDA = 0.9897E+00 MSTAR = 0.0

GIVEN TC = 0.1000E+00 IN UNITS OF RMS OMEGA
TC = 0.1067E+00 IN UNITS OF OMEGA
MUSTAR = 0.1000
MATRIX DIM = 1 LAMBDA = 0.1855E+01 MSTAR = 0.0956
MATRIX DIM = 2 LAMBDA = 0.1411E+01 MSTAR = 0.1023
MATRIX DIM = 4 LAMBDA = 0.1356E+01 MSTAR = 0.1102
MATRIX DIM = 8 LAMBDA = 0.1382E+01 MSTAR = 0.1193
MATRIX DIM = 16 LAMBDA = 0.1396E+01 MSTAR = 0.1300
MATRIX DIM = 32 LAMBDA = 0.1399E+01 MSTAR = 0.1429
MATRIX DIM = 64 LAMBDA = 0.1400E+01 MSTAR = 0.1586
V. LIST OF SUBROUTINES

SOLVE

subroutine solve(tc,el,cc,nalf)
  common om1,om2,om4
  dimension a(64,64),b(64,64),y(64),el(7),x(64),z(64)
  dimension nt(64),c(32,32),ay(128)
  c solves for lambda given tc in units of om (rms omega)
  c cc is mustar, om is rms freq in millivolts
  c el contains seven successive approximations to lambda
  c nalf is 1 if a new a**2*f is to be read
  1 format(/11h given tc =,e10.4,22h in units of rms omega )
  2 format(13h matrix dim =,i4,9h  lambda=,e10.4,8h  mstar=,f8.4)
  3 format(7x,4htc =,e10.4,18h in units of omega )
  4 format(9h mustar =,f7.4)
  if (nalf .ne. 1) go to 25
    dum=f(-1.)
  25 continue
    td=tc*sqrt(om2)/om1
    print 1,tc
    print 3,td
    print 4,cc
    pi=3.14159
    p=2.*pi*tc
    c use delta fn answer for first approx eigenvalue
    e=(1.+2.*cc)*(1.+p*p)
    c construct first approx eivenvector
    y(1)=1.
    do 30 i=2,64
      y(i)=0.
  30 y(i)=0.
    c construct the missing parts of the matrix
    do 200 n=1,7
      cm=cc
      np=nc+1
      nc=nc*2
      if (n .eq. 1) np=1
      if (n .eq. 1) nc=1
    c calculate missing values of ay(n), the renormalization
      il=nc+1
      iu=2*nc
      if (n .eq. 1) il=1
      do 34 i=il,iu
        s=float(i)*p
        ay(i)=f(s)
  34 end
if (abs(cc) .lt. 1.e-04) go to 35
  c renormalize mustar to the specific cutoff n
  xn=float(nc)
  cm=1./(1./cc-alog(p*xn))
35 continue
  do 100 i=1,nc
  do 100 j=1,i
    a(i,j)=-2.*cm
    if (i .ne. j) go to 40
    a(i,j)=a(i,j)-2.*float(i)+1.
    if (i .lt. np) go to 100
  40 if (i .lt. np) go to 80
    b(i,j)=ay(i+j-1)
    if (i .ne. j) b(i,j)=b(i,j)+ay(i-j)
    if (i .ne. j) go to 80
    if (n .eq. 1) go to 100
    jm=j-1
    do 70 l=1,jm
      70 b(i,j)=b(i,j)-2.*ay(l)
    go to 100
  80 a(j,i)=a(i,j)
  100 continue
  c store a in c because ainvit overwrites a
  if (n .eq. 7) go to 110
  do 105 i=1,nc
  do 105 j=1,nc
  105 c(i,j)=a(i,j)
  110 continue
  ifail=0
  eps=0.002
  call ainvit(a,b,e,64,nc,y,eps,5.0e-06,x,z,30,nt,ifail)
  print 2,nc,e,cm
  c eigenvector x becomes next approx eigenvector
  c restore a out of storage in c
  el(n)=e
  test=0.01*e
  do 120 i=1,nc
    y(i)=x(i)
  120 a(i,j)=c(i,j)
  200 continue
  return
end
function f(x)
c x is 2 pi n tc/om (rms omega)
  common om1,om2,om4
  dimension af(150), title(3)
  1 format(3a4,i4,f10.4)
  2 format(3a4)
  3 format(5f8.3)
  4 format(5h ela=,f6.3,5h om1=,f6.1,5h om2=,e10.3,5h om4=,e10.3)
    if (x .ge. 0.0) go to 100
    c read data for a2f with dom=freq. incr. in milliv.
        open(unit=1, file='a2f.d', status='old')
        read(1,1) title, naf, dom
        write(6,2) title
        read(1,3) (af(i), i=1,naf)
        close(unit=1, status='keep')
    c integrate to get ela, om1, etc.
        e=0.0
        o1=0.0
        o2=0.0
        o4=0.0
        om=0.0
        is=-1
        do 50 i=1,naf
            om=om+dom
            is=-is
            si=2.0
            if (is .gt. 0) si=4.0
            e=e+af(i)*si/om
            o1=o1+af(i)*si
            o2=o2+af(i)*si*om
        50 o4=o4+af(i)*si*om*om*om
            ela=2.0*e*dom/3.0
            om1=2.0*o1*dom/(3.0*ela)
            om2=2.0*o2*dom/(3.0*ela)
            om4=2.0*o4*dom/(3.0*ela)
        write(6,4) ela,om1,om2,om4
        rmsom=sqrt(om2)
        rms=rmsom*11.605
        write(6,'(f12.5)') ' rms frequency=',rmsom,' (meV) ',rms,' (K)'
    f=1.0
    return
100 if (x .gt. 5.0) go to 300
    om=0.
    is=-1
    s=0.
    do 200 i=2,naf
        is=-is
        si=2.0
        if (is .gt. 0) si=4.0
        om=om+dom
200 s=s+si*om*af(i)/(om*om+om2*x*x)
    f=2.0*s*dom/(3.0*ela)
    return
300 f=(1.0-om4/(om2*x)**2)/(x*x)
    return
end
AINVIT

subroutine ainvit(a,b,e,nr,nc,y,eps,emach,x,z,nmx,nt,ifail)
dimension b(nr,nc),a(nr,nc),y(nc),x(nc),z(nc),nt(nc)
c symmetric version - overwrites the lower triangle of a only
n=nc
dun=e*e
if (ifail) 30,33,30
30 dx=0.
dy=0.
do 32 i=1,n
d1=-a(i,i)*y(i)
d2=-b(i,i)*y(i)
do 31 j=1,i
d1=d1+a(i,j)*y(j)
d2=d2+b(i,j)*y(j)
do 34 j=1,n
d1=d1+a(j,i)*y(j)
d2=d2+b(j,i)*y(j)
dx=dx+y(i)*d1
dy=dy+y(i)*d2
e=dx/dy
33 ifail=0
do 2 i=1,n
do 1 j=1,i
1 a(i,j)=a(i,j)+e*b(i,j)
2 z(i)=0.0
call cholde(a,nt,nr,nc,emach)
dy=0.0
iac=3
ncpt=0
noit=0
3 do 4 i=1,n
x(i)=b(i,i)*y(i)
do 35 j=1,n
35 x(i)=x(i)-b(j,i)*y(j)
do 4 j=1,i
4 x(i)=x(i)-b(i,j)*y(j)
call cholsu(a,nt,x,nr,nc)
xnorm=0.
icpt=ncpt
do 6 i=1,n
d=abs(x(i))
if (d-xnorm) 6,6,5
5 xnorm=d
ncpt=1
6 continue
dx=1.0/x(ncpt)
do 7 i=1,n
7 x(i)=x(i)*dx
AINVIT (continued)

noit=noit+1
iac=iac+1
if (abs(dx-dy)-eps) 20,20,8
8  if (noit-nmx) 9,9,19
9  if (icpt-ncpt) 10,11,10
10  iac=1
go to 17
11  if (iac-3) 17,12,12
12  do 15 i=1,n
   yr=(z(i)-y(i))**2
   dum=z(i)-2.0*y(i)+x(i)
   if (abs(dum)-emach) 13,13,14
13  yr=0.0
  go to 15
14  yr=yr/dum
15  x(i)=z(i)-yr
  do 16 i=1,n
16  y(i)=x(i)
   yr=(dz-dy)**2
   yr=yr/(dz-2.0*dy+dx)
   dx=dz-yr
   iac=1
17  do 18 i=1,n
   z(i)=y(i)
18  y(i)=x(i)
dz=dy
dy=dx
go to 3
19  ifail=-nmx-1
20  eps=abs(dx-dy)
e=e+dx
   ifail=ifail+noit
return
end
subroutine cholde(a,nt,nr,nc,emach)
dimension a(nr,nc),nt(nc)
if (nc .eq. 1) return
n=nc
do 10 ii=2,n
i=ii-1
yr=abs(a(i,i))
in=i
do 2 j=ii,n
if (yr-abs(a(j,j))) 1,2,2
1 yr=abs(a(j,j))
in=j
2 continue
nt(i)=in
if (in-1) 6,6,3
3 do 12 j=1,i
  dum=a(i,j)
  a(i,j)=a(in,j)
12 a(in,j)=dum
4 do 14 j=i,n
  dum=a(j,i)
  a(j,i)=a(in,j)
5 a(in,i)=dum
  dum=a(i,i)
  a(i,i)=a(in,in)
  a(in,in)=dum
6 a(i,i)=sign(sqrt(yr),a(i,i))
  if(abs(a(i,i))-emach) 7,7,8
7 a(i,i)=emach*1.0e-05
8 do 9 j=ii,n
  a(j,i)=a(j,i)/a(i,i)
  dum=a(j,i)*sign(1.0,a(i,i))
  do 9 k=ii,j
9 a(j,k)=a(j,k)-dum*a(k,i)
10 continue
if (abs(a(n,n))-emach) 13,13,14
13 a(n,n)=emach*1.0e-05
return
14 a(n,n)=sign(sqrt(abs(a(n,n))),a(n,n))
return
end
subroutine cholsu(a,nt,x,nr,nc)
dimension a(nr,nc),nt(nc),x(nc)
if (nc .eq. 1) go to 10
n=nc
do 2 ii=2,n
i=ii-1
in=nt(i)
if (in-i) 1,2,1
1 dum=x(in)
x(in)=x(i)
x(i)=dum
2 continue
x(1)=x(1)/abs(a(1,1))
do 4 ii=2,n
i=ii-1
do 3 j=1,i
3 x(ii)=x(ii)-a(ii,j)*x(j)
4 x(ii)=x(ii)/abs(a(ii,ii))
x(n)=x(n)/a(n,n)
do 6 ij=2,n
ii=n-ij+2
i=ii-1
x(i)=sign(1.0,a(i,i))*x(i)
do 5 j=ii,n
5 x(i)=x(i)-a(j,i)*x(j)
6 x(i)=x(i)/abs(a(i,i))
do 8 ii=2,n
i=n-ii+1
in=nt(i)
if (in-i) 7,8,7
7 dum=x(in)
x(in)=x(i)
x(i)=dum
8 continue
return
10 x(1)=x(1)/a(1,1)
return
end
VI DESCRIPTION OF MATRIX MANIPULATION PROGRAMS

AINVIT

SUBROUTINE AINVIT(A,B,E,NR,NC,Y,EPS,EMACH,X,Z,NMX,NT,IFAIL)
DIMENSION B(NR,NC),A(NR,NC),Y(NC),X(NC),Z(NC),NT(NC)

PURPOSE To find a root of the eigenproblem \((A+\lambda B)x = 0\) using inverse iteration
The root nearest to a given number \(E\) is located or that nearest a given eigenvector by first forming the Rayleigh quotient. For the method see Wilkinson "The Algebraic Eigenvalue Problem"

input

\(A\) matrix as in problem definition
\(B\) matrix as in problem definition
\(E\) approximate root (if appropriate)
\(NR\) First dimension of arrays \(A\) and \(B\) in calling routine
\(NC\) dimension of matrices of problem
\(Y\) initial 'approximate' eigenvector \((1,1,1,\ldots)\) will often suffice
\(EPS\) accuracy required in eigenvalue
\(EMACH\) machine accuracy (\(5.0E-6\) in single precision)
\(NMX\) maximum number of iterations allowed (\(\sim 30\))
\(IFAIL\) = 0 if approximate eigenvalue given 
\# 0 if approximate eigenvector given

\(X\), \(Z\) are used as working space

output

\(E\) computed eigenvalue
\(EPS\) estimated error in computed eigenvalue
\(X\) computed eigenvector
\(IFAIL\) = 0 accuracy not achieved after \(NMX\) iterations
\# 0 the number of iterations used

notes

this routine uses GELM and SUBS if \(A\) and \(B\) are symmetric then calls to these subroutines should be replaced by calls to CHOLB and CHOLSU
SUBROUTINE CHOLDE(A, NT, NR, NC, EMACH)
DIMENSION A(NR, NC), NT(NC)

**Purpose** to perform Cholski decomposition on a real symmetric matrix. A is factorised into $U^T D U$ where $D$ is a diagonal matrix whose non-zero elements are ±1. These signs are stored with the diagonal elements of $U$ thus \( \det(A) = \text{product of diagonal elts. of } U \) ± sign of the product. Only the lower triangle of $A$ need be defined.

**Input**
- $A$ matrix to be factorised; lower triangle only (overwritten)
- NR first dimension of array $A$ in calling
- NC dimension of matrix of problem
- EMACH machine accuracy

**Output**
- $A$ the matrix $U^T$ stored in lower triangle
- NT list of pivotal rows

---

SUBROUTINE CHOLSU(A, NT, X, NR, NC)
DIMENSION A(NR, NC), NT(NC), X(NR)

**Purpose** to solve the set of equations $Ax = b$, $A$ real symmetric, after $A$ has been factorised by CHOLDE; i.e., compute $A^{-1} X$.

**Input**
- $A$ output from CHOLDE
- NT output from CHOLDE
- $X$ right-hand side of matrix equation; (overwritten)
- NR first dimension of array $A$ in calling routine
- NC dimension of matrix $A$

**Output**
- $X$ calculated solution $= A^{-1} X$