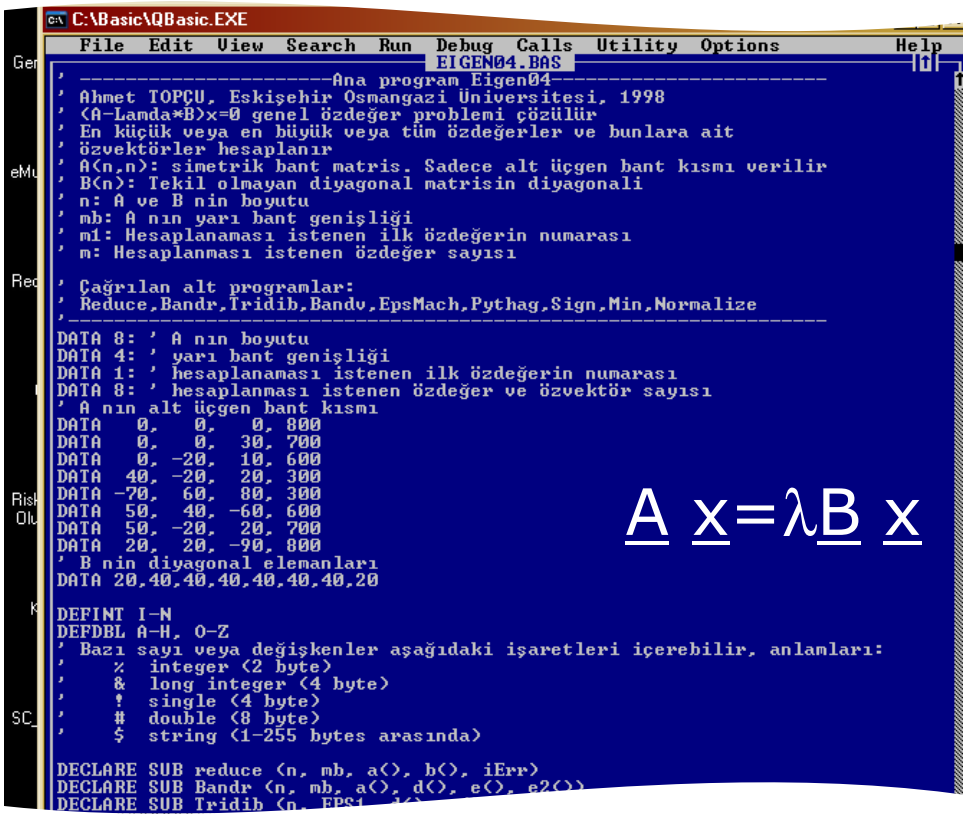
ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ
Mühendislik Mimarlık Fakültesi
İnşaat Mühendisliği Bölümü
E-Posta: ogu.ahmet.topcu@gmail.com
Web: http://mmf2.ogu.edu.tr/atopcu

Bilgisayar Destekli
Nümerik Analiz

Ders notları 2014

Ahmet TOPÇU



34

PROGRAMLAR: Genel özdeğer ve özvektör hesabı

- Birkaç özdeğer ve özvektör - Eigen04

34. PROGRAMLAR: Genel özdeğer ve özvektör hesabı[1] - Eigen04

$$\underline{A}\underline{x} = \lambda \underline{B}\underline{x}$$

Genel özdeğer probleminin en küçük m tane veya en büyük m tane veya ortadaki m tane veya tüm özdeğerlerini hesaplar. A simetrik bant matris, B diyagonal matris, det B≠0 olmalıdır. A nın alt yarı bantı ve B nin sadece diyagonalı programa verilir.

Eigen04 ün çağırdığı alt programlar(Sign, Min, Normalize hariç) Eispack yazılımından alınmıştır.

Çağrılan alt programlar: Reduce, Bandr, Tridib, Bandv, EpsMach, Pytahag, Sign, Min, Normalize.

Örnek:

$$\underline{A}\underline{x} = \lambda \underline{B}\underline{x} \rightarrow A = \begin{bmatrix} 800 & & & & & & & \\ 30 & 700 & & & & & & \\ -20 & 10 & 600 & & & & sim & \\ 40 & -20 & 20 & 300 & & & & \\ & -70 & 60 & 80 & 300 & & & \\ & & 50 & 40 & -60 & 600 & & \\ & & & 50 & -20 & -20 & 700 & \\ & & & & 20 & 20 & -90 & 800 \end{bmatrix}, \quad \underline{B} = \begin{bmatrix} 20 & & & & & & & \\ & 40 & & & & & & \\ & & 40 & & & & & \\ & & & 40 & & & & \\ & & & & 40 & & & \\ & & & & & 40 & & \\ & & & & & & 40 & \\ & & & & & & & 20 \end{bmatrix}$$

Yarı bant=4

Tüm özdeğerler ve özvektörler?

A ve B nin depolanma şekli:

bant genişliğininin aynı olması için eklenen sıfırlar

$$\underline{A} \rightarrow \begin{bmatrix} 0 & 0 & 0 & 800 \\ 0 & 0 & 30 & 700 \\ 0 & -20 & 10 & 600 \\ 40 & -20 & 20 & 300 \\ -70 & 60 & 80 & 300 \\ 50 & 40 & -60 & 600 \\ 50 & -20 & -20 & 700 \\ 20 & 20 & -90 & 800 \end{bmatrix}, \quad \underline{B} \rightarrow \begin{bmatrix} 20 & 40 & 40 & 40 & 40 & 40 & 40 & 20 \end{bmatrix}$$

Yarı bant=4

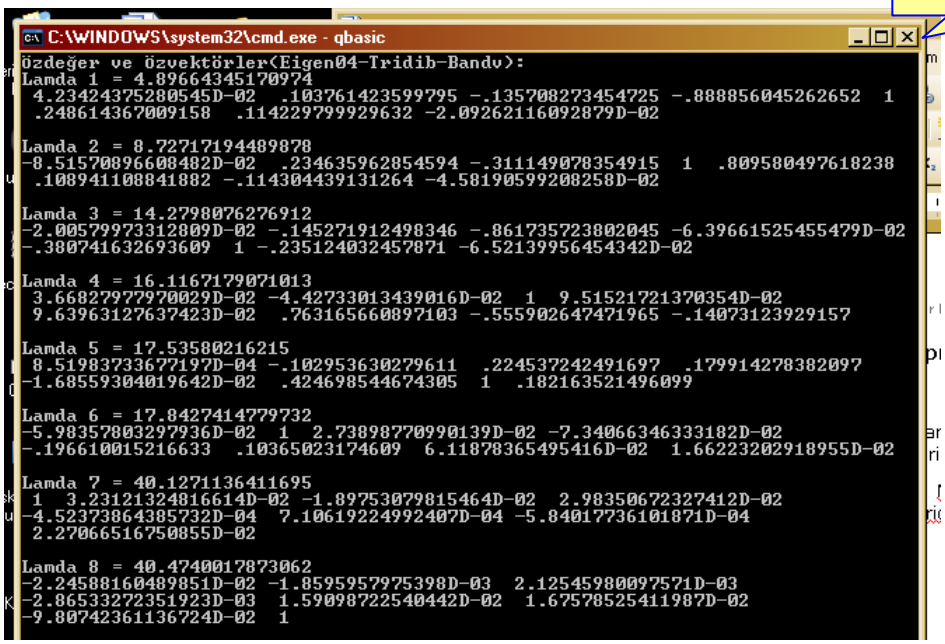
[1] Teori ve örnekler için bak: bölüm 26

Tüm özdeğer ve özvektörleri hesaplamak için ilk dört DATA satırında:

DATA 8 — A nın boyutu
DATA 4 — A nın yarı bant genişliği
DATA 1 — Hesaplanması istenen ilk özdeğerin numarası
DATA 8 — Hesaplanması istenen özdeğer ve özvektör sayısı

verilir:

Eigen04 programının sonucu: Tüm özdeğerler ve özvektörler

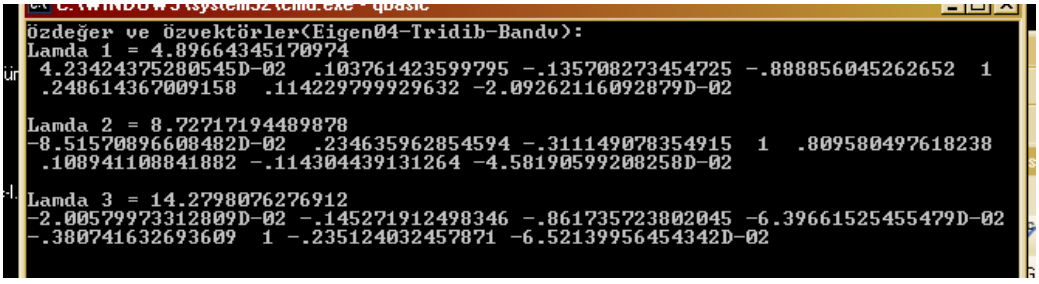


İlk üç özdeğeri ve vektörünü hesaplamak için ilk dört DATA satırında:

DATA 8 — A nın boyutu
DATA 4 — A nın yarı bant genişliği
DATA 1 — Hesaplanması istenen ilk özdeğerin numarası
DATA 3 — Hesaplanması istenen özdeğer ve özvektör sayısı

verilir:

Eigen04 programının sonucu: ilk üç özdeğer ve özvektör

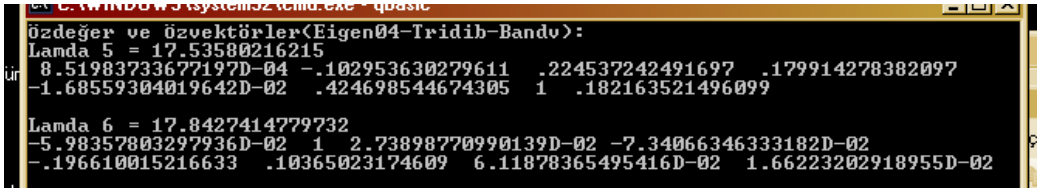


5. ve 6. özdeğeri ve vektörlerini hesaplamak için ilk dört DATA satırında:

DATA 8 — A nın boyutu
DATA 4 — A nın yarı bant genişliği
DATA 5 — Hesaplanması istenen ilk özdeğerin numarası
DATA 2 — Hesaplanması istenen özdeğer ve özvektör sayısı

verilir:

Eigen04 programının sonucu: 5. ve 6. özdeğer ve özvektörler

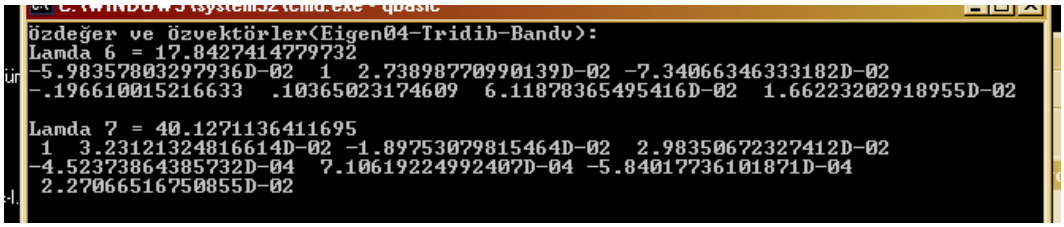


Son iki özdeğeri ve vektörlerini hesaplamak için ilk dört DATA satırında:

DATA 8 — A nın boyutu
DATA 4 — A nın yarı bant genişliği
DATA 6 — Hesaplanması istenen ilk özdeğerin numarası
DATA 2 — Hesaplanması istenen özdeğer ve özvektör sayısı

verilir:

Eigen04 programının sonucu: son iki özdeğer ve özvektör



Eigen 04 ana programı

```
' ----------------------Ana Program Eigen04-----------------------
' Ahmet TOPÇU, Eskişehir Osmangazi Üniversitesi, 1998
' (A-Lamda*B)x=0 genel özdeğer problemi çözülür
' En küçük veya en büyük veya tüm özdeğerler ve bunlara ait
' özvektörler hesaplanır
' A(n,n): simetrik bant matris. Sadece alt bant kısmı verilir
' B(n): Tekil olmayan diyagonal matrisin diyagonali
' n: A ve B nin boyutu
' mb: A nın yarı bant genişliği
' m1: Hesaplanaması istenen ilk özdeğerin numarası
' m: Hesaplanması istenen özdeğer sayısı

' Çağrılan alt programlar:
' Reduce,Bandr,Tridib,Bandv,EpsMach,Pythag,Sign,Min,Normalize
'-----------------------------------------------------------------

DATA 8: ' A nın boyutu
DATA 4: ' yarı bant genişliği
DATA 1: ' hesaplanaması istenen ilk özdeğerin numarası
DATA 8: ' hesaplanması istenen özdeğer ve özvektör sayısı
' A nın alt bant kısmı
DATA   0,   0,   0, 800
DATA   0,   0,  30, 700
DATA   0, -20,  10, 600
DATA  40, -20,  20, 300
DATA -70,  60,  80, 300
DATA  50,  40, -60, 600
DATA  50, -20,  20, 700
DATA  20,  20, -90, 800
' B nin diyagonal elemanları
DATA 20,40,40,40,40,40,40,20

DEFINT I-N
DEFDBL A-H, O-Z
' Bazı sayı veya değişkenler aşağıdaki işaretleri içerebilir, anlamları:
'    %  integer (2 byte)
'    &  long integer (4 byte)
'    !  single (4 byte)
'    #  double (8 byte)
'    $  string (1-255 bytes arasında)

DECLARE SUB reduce (n, mb, a(), b(), iErr)
DECLARE SUB Bandr (n, mb, a(), d(), e(), e2())
DECLARE SUB Tridib (n, EPS1, d(), e(), e2(), BL, BU, m11, m, W(), iND(), iErr)
DECLARE SUB bandv (n, mb, a(), m, W(), z(), iErr)
DECLARE FUNCTION EpsMach# ()
DECLARE FUNCTION Pythag (a, b)
DECLARE FUNCTION Sign (a, b)
DECLARE FUNCTION Min (ia, ib)
DECLARE SUB normalize (n, m, z(), normal)

    READ n

    READ mb

    READ m1

    READ m

    DIM a(n, mb), b(n)
    DIM d(n), e(n), e2(n), iND(n)
    DIM W(n), z(n, m)
```

A ve B nin boyutu

Hesaplanması istenen ilk özdeğerin numarası

Hesaplanması istenen özdeğer ve özvektör sayısı

A nın alt bandı içindeki elemanları

B nin diyagonal elemanları

i, j, k, l, m, n ile başlayan değişkenler tam sayı değişkeni olacak (2 byte)

Bu harf aralığındaki harf ile başlayan değişkenler çift hassasiyetli ondalık sayı değişkeni olacak (8 byte)

Çağrılacak alt programlar ana programa tanıtılıyor

A ve B nin boyutu okunuyor

A nın yarı bant genişlşiği okunuyor

Hesaplanması istenen ilk özdeğer no okunuyor

Hesaplanması istenen özdeğer sayısı okunuyor

Matrisler için yer ayrılıyor

Devamı sonraki sayfada

```
    CLS : ' Ekrası sil
' A nın alt bant kısmını oku
    FOR i = 1 TO n
     FOR j = 1 TO mb
      READ a(i, j)
     NEXT j
    NEXT i
' B nin diyagonal elemanlarını oku
    FOR i = 1 TO n
     READ b(i)
    NEXT i

' Standart özdeğer problemine dönüştür
    CALL reduce(n, mb, a(), b(), iErr)
    IF iErr <> 0 THEN PRINT "HATA(Eigen04-Reduce)="; iErr: END

' A matrisini üçlü bant matrise dönüştür
    CALL Bandr(n, mb, a(), d(), e(), e2())

' m1., m1+1., m1+2., ..., m1+m-1. özdeğeri(m tane) hesapla
    EPS1 = 1E-14: 'Hesap hassasiyeti
    CALL Tridib(n, EPS1, d(), e(), e2(), BL, BU, m1, m, W(), iND(), iErr)
    IF iErr <> 0 THEN PRINT "HATA(Eigen04-Tridib)="; iErr: END

' Verileri tekrar oku
    RESTORE
    READ n
    READ mb
    READ m1
    READ m
' A nın alt bant kısmını tekrar oku
    FOR i = 1 TO n
     FOR j = 1 TO mb
      READ a(i, j)
     NEXT j
    NEXT i
' B nin diyagonal elemanlarını tekrar oku
    FOR i = 1 TO n
     READ b(i)
    NEXT i

' A yı dönüştür
    CALL reduce(n, mb, a(), b(), iErr)

' Özvektörleri hesapla
    CALL bandv(n, mb, a(), m, W(), z(), iErr)
    IF iErr <> 0 THEN PRINT "HATA(Eigen04-Bandv)="; iErr: ' END

' Özvektörleri geri dönüştür
' Hesapla: B*z
    FOR i = 1 TO n
     FOR j = 1 TO m
      z(i, j) = b(i) * z(i, j)
     NEXT j
    NEXT i

' Özvektörleri normalleştir
    CALL normalize(n, m, z(), 1)

    PRINT "Özdeğer ve Özvektörler(Eigen04-Tridib-Bandv):"
    FOR i = 1 TO m
     PRINT "Lamda"; i + m1 - 1; "="; W(i)
     FOR j = 1 TO n
      PRINT z(j, i);
     NEXT j
     PRINT
     PRINT
    NEXT i

END ' Eigen04 ana
```

Eigen 04 ana programı (devam)

A nın alt bant satırları okunuyor

B nin diyagonal elemanları okunuyor

Standart özdeğer problemine dönüştürülüyor

Üçlü diyagonal matrise dönüştürülüyor

m tane özdeğer hesaplanıyor

Veriler tekrar okunacak

Tüm veriler tekrar okunuyor

Tekrar standart özdeğer problemine dönüştürülüyor

m tane özvektör hesaplanıyor

Özvektörler dönüştürülüyor

Özvektörler normalleştiriliyor

Sonuçlar yazdırılıyor

Eispack alt programları:

Eigen01, Eigen02, Eigen03 ve Eigen04 ana programlarının çağırdığı alt programların QBASIC kodları aşağıda verilmiştir. Bu alt programların çoğunluğu EISPACK profesyonel yazılımının parçasıdır ve FORTRAN kodları aşağıdaki adresten alınmıştır:

http://www.netlib.org/eispack

```
SUB Tred2 (n, D(), E(), z())
'------------------------------------------------------------------------
' Real symmetric matrix Z(n,n) will be tridiagonalized using Housholder
' ortogonal similarity transformations

' Input:
' N: Dimension of Z
' Z(n,n): input matrix. Must be real symmetric

' Output:
' D(n): Output matrix. Contains diagonal elements of tridiaginal matrix
' E(n): Output matrix. Contains sub diagonal elements of tridiaginal
' matrix
' Z(n,n): Output matrix. Contains transformation matrix.

' This Subroutine is translated from ALGOL 60 in: Martin, Reinsch and
' Wilkinson. Handbook for auto. comp., Vol. II, 1971
' FORTRAN code: www.netlib.org/eispack
'------------------------------------------------------------------------
' Copy diagonals to d
    FOR i = 1 TO n
     D(i) = z(n, i)
    NEXT i
    IF n = 1 THEN GOTO 510
    FOR i = n TO 2 STEP -1
     L = i - 1
     h = 0
     scale = 0
     IF L >= 2 THEN
      FOR k = 1 TO L
       scale = scale + ABS(D(k))
      NEXT k
     END IF
     IF scale = 0 THEN
      E(i) = D(L)
      FOR j = 1 TO L
       D(j) = z(L, j)
       z(i, j) = 0
       z(j, i) = 0
      NEXT j
      GOTO 290
     END IF
     FOR k = 1 TO L
      D(k) = D(k) / scale
      h = h + D(k) * D(k)
     NEXT k
     f = D(L)
     g = -Sign(SQR(h), f): ' function Call
     E(i) = scale * g
     h = h - f * g
     D(L) = f - g
'Form A*U
     FOR j = 1 TO L
      E(j) = 0
     NEXT j
     FOR j = 1 TO L
      f = D(j)
      z(j, i) = f
      g = E(j) + z(j, j) * f
      jp1 = j + 1
      IF L >= jp1 THEN
       FOR k = jp1 TO L
        g = g + z(k, j) * D(k)
        E(k) = E(k) + z(k, j) * f
       NEXT k
      END IF
      E(j) = g
     NEXT j
' Form P
     f = 0
     FOR j = 1 TO L
      E(j) = E(j) / h
      f = f + E(j) * D(j)
     NEXT j
     hh = f / (h + h)
' Form Q
     FOR j = 1 TO L
      E(j) = E(j) - hh * D(j)
     NEXT j
' Form reduced A
     FOR j = 1 TO L
      f = D(j)
      g = E(j)
      FOR k = j TO L
       z(k, j) = z(k, j) - f * E(k) - g * D(k)
      NEXT k
      D(j) = z(L, j)
      z(i, j) = 0
     NEXT j
290  D(i) = h
    NEXT i
' Accumulation of transformation matrices
    FOR i = 2 TO n
     L = i - 1
     z(n, L) = z(L, L)
     z(L, L) = 1
     h = D(i)
     IF h <> 0 THEN
      FOR k = 1 TO L
       D(k) = z(k, i) / h
      NEXT k
      FOR j = 1 TO L
       g = 0
       FOR k = 1 TO L
        g = g + z(k, i) * z(k, j)
       NEXT k
       FOR k = 1 TO L
        z(k, j) = z(k, j) - g * D(k)
       NEXT k
      NEXT j
     END IF
     FOR k = 1 TO L
      z(k, i) = 0
     NEXT k
    NEXT i

510  FOR i = 1 TO n
     D(i) = z(n, i)
     z(n, i) = 0
    NEXT i
    z(n, n) = 1
    E(1) = 0

END SUB  ' end of Tred2
```

```
FUNCTION Sign (a, b)
  IF b >= 0 THEN Sign = ABS(a) ELSE Sign = -ABS(a)
END FUNCTION ' End of Sign
```

```
FUNCTION Pythag (a, b)
' Finds sqr(a*a+b*b) without overflow or
' destructive underflow
    ABSa = ABS(a)
    ABSb = ABS(b)
    p = ABSa: IF ABSb > p THEN p = ABSb
    IF p = 0 THEN
     Pythag = p
     EXIT FUNCTION
    END IF
    r = ABSa: IF ABSb < r THEN r = ABSb
    r = (r / p) ^ 2
    t = 4 + r
    WHILE t <> 4
     s = r / t
     u = 1 + 2 * s
     p = u * p
     r = (s / u) ^ 2 * r
     t = 4 + r
    WEND
    Pythag = p

END FUNCTION ' End of Pythag
```

```
SUB Tql2 (n, D(), E(), z(), iErr)
'------------------------------------------------------------------------
' Finds all Eigenvalues and eigenvectors of an nxn tridiagonal symmetric
' matrix which is produced by the program Tred2.

' Input:
' N: Dimension of the matrix
' D(n): is input matrix. Contains diagonal elements calulated by Tred2
' E(n): is input matrix. Contains sub diagonal elements calculated by Tred2

' Output:
' D(n): is output matrix. Contains eigenvalues after run Tql2
' Z(n,n): is output matrix. Contains normalized eigenvectors after run Tql2
' iErr: Error flag, ierr=0 no error, iErr<>0 error.
' Used method is QL transformation.

' This Subroutine is translated from ALGOL 60 in: Bowdler, Martin,
' Reinsch and Wilkinson. Handbook for auto. comp., Vol. II, 1971
' FORTRAN code: www.netlib.org/eispack
'------------------------------------------------------------------------
    iErr = 0
    IF n = 1 THEN EXIT SUB
    FOR i = 2 TO n
     E(i - 1) = E(i)
    NEXT i
    f = 0
    tst1 = 0
    E(n) = 0
    FOR L = 1 TO n
     j = 0
     h = ABS(D(L)) + ABS(E(L))
     IF tst1 < h THEN tst1 = h
' Look for small sub-diogonal element
      FOR m = L TO n
       tst2 = tst1 + ABS(E(m))
       IF tst2 = tst1 THEN EXIT FOR
      NEXT m
      IF m = L THEN GOTO 220
130   IF j = 30 THEN
' Set error-no convergence to an eigenvalue after 30 iteration
       iErr = L
       EXIT SUB
      END IF
     j = j + 1
' Form shift
      L1 = L + 1
      L2 = L1 + 1
      g = D(L)
      p = (D(L1) - g) / (2 * E(L))
      r = Pythag(p, 1): ' Function call
      D(L) = E(L) / (p + Sign(r, p)): ' Function Call
      D(L1) = E(L) * (p + Sign(r, p)): 'Function Call
      dL1 = D(L1)
      h = g - D(L)
      IF L2 <= n THEN
       FOR i = L2 TO n
        D(i) = D(i) - h
       NEXT i
      END IF
      f = f + h
' QL Transformation
      p = D(m)
      c = 1
      c2 = c
      eL1 = E(L1)
      s = 0
      mmL = m - L
      FOR i = m - 1 TO L STEP -1
       c3 = c2
       c2 = c
       s2 = s
       g = c * E(i)
       h = c * p
       r = Pythag(p, E(i))
       E(i + 1) = s * r
       s = E(i) / r
       c = p / r
       p = c * D(i) - s * g
       D(i + 1) = h + s * (c * g + s * D(i))

' Form vector
       FOR k = 1 TO n
        h = z(k, i + 1)
        z(k, i + 1) = s * z(k, i) + c * h
        z(k, i) = c * z(k, i) - s * h
       NEXT k
      NEXT i
      p = -s * s2 * c3 * eL1 * E(L) / dL1
      E(L) = s * p
      D(L) = c * p
      tst2 = tst1 + ABS(E(L))
      IF tst2 > tst1 THEN GOTO 130
220   D(L) = D(L) + f
     NEXT L
' Order eigenvalues
     FOR L = 2 TO n
      i = L - 1
      k = i
      p = D(i)
      FOR j = L TO n
       IF D(j) < p THEN
        k = j
        p = D(j)
       END IF
      NEXT j
' Order iegenvectors
      IF k <> i THEN
       D(k) = D(i)
       D(i) = p
       FOR j = 1 TO n
        p = z(j, i)
        z(j, i) = z(j, k)
        z(j, k) = p
       NEXT j
      END IF
     NEXT L

END SUB ' End of Tql2
```

```
SUB Normalize (n, m, z(), normal)
' Vekctors of z(n,m) will be normalized
' Normal<>0:  normalize so that max element of vector is 1.
' Normal=0: do nothing
    IF normal = 0 THEN EXIT SUB
    FOR i = 1 TO m
     p = 0
     FOR j = 1 TO n
      IF ABS(z(j, i)) > ABS(p) THEN p = z(j, i)
     NEXT j
     IF p = 0 THEN EXIT SUB
     p = 1 / p
     FOR j = 1 TO n
      z(j, i) = z(j, i) * p
     NEXT j
    NEXT i
END SUB  ' End of Normalize
```

```
FUNCTION EpsMach#
' determines Machine epsilon
' IMPORTANT: ALL CONSTANTS AND VARIALES
' MUST BE IN DOUBLE
   a = 4# / 3#
110 b = a - 1#
   c = b + b + b ' DO NOT CHANGE INTO 3*B
   Eps = ABS(c - 1#)
   IF Eps = 0# GOTO 110
   EpsMach = Eps

END FUNCTION ' end of EpsMach
```

```
SUB Tred1 (n, a(), d(), e(), e2())
' This subroutine reduces a real symmetric matrix to a symmetric
' tridiagonal matrix using orthogonal similarity transformations.

' on input:
'  n is the order of the matrix.
'  a contains the real symmetric input matrix.  only the
'    lower triangle of the matrix need be supplied.

' on output:
'  a contains information about the orthogonal trans-
'    formations used in the reduction in its strict lower
'    triangle. The full upper triangle of a is unaltered.
'  d contains the diagonal elements of the tridiagonal matrix.
'  e contains the subdiagonal elements of the tridiagonal
'    matrix in its last n-1 positions.  e(1) is set to zero.
'  e2 contains the squares of the corresponding elements of e.
'    e2 may coincide with e if the squares are not needed.
' This subroutine is a translation of the algol procedure tred1,
' num. math. 11, 181-195(1968) by Martin, Reinsch, and Wilkinson.
' handbook for auto. comp., vol.ii-linear algebra, 212-226(1971).
' This version dated August 1983.
' FORTRAN code: www.netlib.org/eispack
'------------------------------------------------------------------
    FOR i = 1 TO n
     d(i) = a(n, i)
     a(n, i) = a(i, i)
    NEXT i
'
    FOR i = n TO 1 STEP -1
     L = i - 1
     h = 0
     scale = 0
' scale row
     FOR k = 1 TO L
      scale = scale + ABS(d(k))
     NEXT k
     IF scale = 0 THEN
      FOR j = 1 TO L
       d(j) = a(L, j)
       a(L, j) = a(i, j)
      a(i, j) = 0
      NEXT j
      e(i) = 0
      e2(i) = 0
      GOTO 300
     END IF

     FOR k = 1 TO L
      d(k) = d(k) / scale
      h = h + d(k) * d(k)
     NEXT k
     e2(i) = scale * scale * h
     f = d(L)
     g = -Sign(SQR(h), f)
     e(i) = scale * g
     h = h - f * g
     d(L) = f - g

     IF L <> 1 THEN
' form a*u
      FOR j = 1 TO L
       e(j) = 0
      NEXT j
      FOR j = 1 TO L
       f = d(j)
       g = e(j) + a(j, j) * f
       jp1 = j + 1
       IF L >= jp1 THEN
        FOR k = jp1 TO L
         g = g + a(k, j) * d(k)
         e(k) = e(k) + a(k, j) * f
        NEXT k
       END IF
       e(j) = g
      NEXT j
' form P
      f = 0
      FOR j = 1 TO L
       e(j) = e(j) / h
       f = f + e(j) * d(j)
      NEXT j
      h = f / (h + h)
' form Q
      FOR j = 1 TO L
       e(j) = e(j) - h * d(j)
      NEXT j
' form reduced a
      FOR j = 1 TO L
       f = d(j)
       g = e(j)
       FOR k = j TO L
        a(k, j) = a(k, j) - f * e(k) - g * d(k)
       NEXT k
      NEXT j
     END IF

     FOR j = 1 TO L
      f = d(j)
      d(j) = a(L, j)
      a(L, j) = a(i, j)
      a(i, j) = f * scale
     NEXT j
300  NEXT i

END SUB ' end of Tred1
```

```
SUB Trbak1 (n, a(), e(), m, z())
' Trbak1-Eispack
IF n = 1 THEN EXIT SUB
FOR i = 2 TO n
 L = i - 1
 IF e(i) <> 0 THEN
  FOR j = 1 TO m
   s = 0
   FOR k = 1 TO L
    s = s + a(i, k) * z(k, j)
   NEXT k
   s = (s / a(i, L)) / e(i)
   FOR k = 1 TO L
    z(k, j) = z(k, j) + s * a(i, k)
   NEXT k
  NEXT j
 END IF
NEXT i

END SUB ' end of Trbak1
```

```
SUB Tinvit (n, d(), e(), e2(), m, W(), iND(), z(), iErr)
'------------------------------------------------------------------
' This subroutine finds those eigenvectors of a TRIDIAGONAL
' SYMMETRIC matrix corresponding to specified eigenvalues,
' using inverse iteration.
' This subroutine is a translation of the inverse iteration technique
' in the ALGOL procedure TRISTURM by Peters and Wilkinson.
' HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 418-439(1971).

' On Input
'  N is the order of the matrix.
'  D contains the diagonal elements of the input matrix.
'  E contains the subdiagonal elements of the input matrix
'    in its last N-1 positions.  E(1) is arbitrary.
'  E2 contains the squares of the corresponding elements of E,
'     with zeros corresponding to negligible elements of E.
'     E(I) is considered negligible if it is not larger than
'     the product of the relative machine precision and the sum
'     of the magnitudes of D(I) and D(I-1).  E2(1) must contain
'     0.0e0 if the eigenvalues are in ascending order, or 2.0e0
'     if the eigenvalues are in descending order.  If  BISECT,
'     TRIDIB, or  IMTQLV  has been used to find the eigenvalues,
'     their output E2 array is exactly what is expected here.
'  M is the number of specified eigenvalues.
'  W CONTAINS the M eigenvalues in ascending or descending order.
'  iND contains in its first M positions the submatrix indices
'      associated with the corresponding eigenvalues in W
'      1 for eigenvalues belonging to the first submatrix from
'      the top, 2 for those belonging to the second submatrix, etc.

' On Output
'  All input arrays are unaltered.
'  Z contains the associated set of orthonormal eigenvectors.
'    any vector which fails to converge is set to zero.
'  IERR is set to
'       Zero  for normal return,
'       -iR   if the eigenvector corresponding to the iR-th
'             eigenvalue fails to converge in 5 iterations.

'  RV1, RV2, RV3, RV4, and RV6 are temporary storage arrays.
' REFERENCES  B. T. SMITH, J. M. BOYLE, J. J. DONGARRA, B. S. GARBOW, Y. IKEBE,
' V. KLEMA, B. MOLER, MATRIX EIGEN SYSTEM ROUTINES - EISPACK GUIDE,
' SPRINGER-VERLAG, 1976.
' ROUTINES CALLED  (NONE)
' FORTRAN code: www.netlib.org/eispack
' ------------------------------------------------------------------
    DIM RV1(n), RV2(n), RV3(n), RV4(n), RV6(n)

     iErr = 0
     IF m <= 0 THEN EXIT SUB
     u = 0
     x0 = 0
     iTag = 0
     Order = 1 - e2(1)
     iQ = 0
' ESTABLISH AND PROCESS NEXT SUBMATRIX
100  iPi = iQ + 1
     FOR iQ = iPi TO n
      IF iQ = n THEN EXIT FOR
      IF e2(iQ + 1) = 0 THEN EXIT FOR
     NEXT iQ
' FIND VECTORS BY INVERSE ITERATION
     iTag = iTag + 1
     iSi = 0
     FOR iR = 1 TO m
      IF iND(iR) <> iTag THEN GOTO 920
      iTS = 1
      x1 = W(iR)
      IF iSi <> 0 THEN GOTO 510
' CHECK FOR ISOLATED ROOT
      xu = 1
      IF iPi = iQ THEN RV6(iPi) = 1: GOTO 870
      aNorm = ABS(d(iPi))
      iP = iPi + 1
      FOR i = iP TO iQ
       t = ABS(d(i)) + ABS(e(i))
       IF t > aNorm THEN aNorm = t
      NEXT i
' EPS2 is THE CRITERION FOR GROUPING,
' EPS3 REPLACES ZERO PIVOTS AND EQUAL ROOTS ARE MODIFIED BY EPS3,
' EPS4 is TAKEN VERY SMALL TO AVOID OVERFLOW
' EpsMach is hardware dependent parameter specifying the relativ
' Precision of floating point arithmetic.
      Eps2 = .001 * aNorm
      eps3 = EpsMach * aNorm
```

Tinvit sonraki sayfada devam ediyor

Tinvit devamı

```
      Uk = iQ - iPi + 1
      Eps4 = Uk * eps3
      Uk = Eps4 / SQR(Uk)
      iSi = iPi
505    iGroup = 0
      GOTO 520
' LOOK FOR CLOSE OR COINCIDENT ROOTS
510    IF ABS(x1 - x0) >= Eps2 THEN GOTO 505
      iGroup = iGroup + 1
      IF Order * (x1 - x0) <= 0 THEN x1 = x0 + Order * eps3
' ELIMINATION WITH INTERCHANGES AND INITIALIZATION OF VECTOR
520    V = 0
      FOR i = iPi TO iQ
       RV6(i) = Uk
        IF i <> iPi THEN
         IF ABS(e(i)) >= ABS(u) THEN
' WARNING: A DIVIDE CHECK MAY OCCUR HERE IF E2 ARRAY HAS
' NOT BEEN SPECIFIED CORRECTLY
          xu = u / e(i)
          RV4(i) = xu
          RV1(i - 1) = e(i)
          RV2(i - 1) = d(i) - x1
          RV3(i - 1) = 0
          IF i <> iQ THEN RV3(i - 1) = e(i + 1)
          u = V - xu * RV2(i - 1)
          V = -xu * RV3(i - 1)
          GOTO 580
         END IF
         xu = e(i) / u
         RV4(i) = xu
         RV1(i - 1) = u
         RV2(i - 1) = V
         RV3(i - 1) = 0
        END IF
        u = d(i) - x1 - xu * V
        IF i <> iQ THEN V = e(i + 1)
580    NEXT i

      IF u = 0 THEN u = eps3
      RV1(iQ) = u
      RV2(iQ) = 0
      RV3(iQ) = 0
' BACK SUBSTITUTION
600    FOR i = iQ TO iPi STEP -1
       RV6(i) = (RV6(i) - u * RV2(i) - V * RV3(i)) / RV1(i)
       V = u
       u = RV6(i)
      NEXT i
' ORTHOGONALIZE WITH RESPECT TO PREVIOUS MEMBERS OF iGroup
      j = iR
      FOR jj = 1 TO iGroup
       DO
        j = j - 1
       LOOP WHILE iND(j) <> iTag
       xu = 0
       FOR i = iPi TO iQ
        xu = xu + RV6(i) * z(i, j)
       NEXT i
       FOR i = iPi TO iQ
        RV6(i) = RV6(i) - xu * z(i, j)
       NEXT i
      NEXT jj

      aNorm = 0
      FOR i = iPi TO iQ
       aNorm = aNorm + ABS(RV6(i))
      NEXT i
      IF aNorm < 1 THEN
' FORWARD SUBSTITUTION
       IF iTS = 5 THEN GOTO 830
       IF aNorm = 0 THEN
        RV6(iSi) = Eps4
        iSi = iSi + 1
        IF iSi > iQ THEN iSi = iPi
        GOTO 780
       END IF
       xu = Eps4 / aNorm
       FOR i = iPi TO iQ
        RV6(i) = RV6(i) * xu
       NEXT i
' ELIMINATION OPERATIONS ON NEXT VECTOR ITERATE
780    FOR i = iP TO iQ
       u = RV6(i)
       IF RV1(i - 1) = e(i) THEN
        u = RV6(i - 1)
        RV6(i - 1) = RV6(i)
       END IF
       RV6(i) = u - RV4(i) * RV6(i - 1)
      NEXT i
      iTS = iTS + 1
      GOTO 600
' SET ERROR if NON-CONVERGED EIGENVECTOR
830    iErr = -iR
      xu = 0
      GOTO 870
     END IF
' NORMALIZE SO THAT SUM OF SQUARES is 1
      u = 0
      FOR i = iPi TO iQ
       u = Pythag(u, RV6(i))
      NEXT i
      xu = 1 / u
870    FOR i = 1 TO n
       z(i, iR) = 0
      NEXT i
      FOR i = iPi TO iQ
       z(i, iR) = RV6(i) * xu
      NEXT i
      x0 = x1
920   NEXT iR
    IF iQ < n THEN GOTO 100

END SUB ' End of Tinvit
```

```
SUB Tridib (n, EPS1, d(), e(), e2(), BL, BU, m11, m, W(), iND(), iErr)
'------------------------------------------------------------------
' This subroutine finds those eigenvalues of a TRIDIAGONAL
' SYMMETRIC matrix between specified boundary indices, using bisection.
'
' On Input:
'  N is the order of the matrix.
'  EPS1 is an absolute error tolerance for the computed eigenvalues.
'   If the input EPS1 is non-positive, it is reset for each submatrix
'   to a default value, namely, minus the product of the relative machine
'   precision and the 1-aNorm of the submatrix.
'  D contains the diagonal elements of the input matrix.
'  E contains the subdiagonal elements of the input matrix
'    in its last N-1 positions.  E(1) is arbitrary.
'  E2 contains the squares of the corresponding elements of E. E2(1) is arbitrary.
'  M11 specifies the lower boundary index for the desired eigenvalues.
'  M specifies the number of eigenvalues desired.  The upper
'   boundary index M22 is then obtained as M22=M11+M-1.

' On Output
'  EPS1 is unaltered unless it has been reset to its (last) default value.
'  D and E are unaltered.
'  ELEMENTS of E2, corresponding to elements of E regarded as negligible,
'   have been replaced by zero causing the matrix to split into a direct
'   sum of submatrices. E2(1) is also set to zero.

'  BL and BU DEFINE an interval containing exactly the desired  eigenvalues.
'  W contains, in its first M positions, the eigenvalues
'   between indices M11 and M22 in ascending order.
'  IND contains in its first M positions the submatrix indices
'   associated with the corresponding eigenvalues in W.
'   1 for eigenvalues belonging to the first submatrix from
'   the top, 2 for those belonging to the second submatrix, etc.
'  IERR is set to
'    Zero       for normal return,
'    3*N+1      if multiple eigenvalues at index M11 make
'               unique selection impossible,
'    3*N+2      if multiple eigenvalues at index M22 make
'               unique selection impossible.
'
'  RV4 and RV5 are temporary storage arrays.
' REFERENCES  B. T. SMITH, J. M. BOYLE, J. J. DONGARRA, B. S. GARBOW,
'             Y. IKEBE, V. C. KLEMA, C. B. MOLER, *MATRIX EIGEN-
'             SYSTEM ROUTINES - EISPACK GUIDE*, SPRINGER-VERLAG, 1976.
' ROUTINES CALLED  (NONE)
' This subroutine is a translation of the ALGOL procedure BISECT,
' NUM. MATH. 9, 386-393(1967) by Barth, Martin, and Wilkinson.
' HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 249-256(1971).
' FORTRAN code : www.netlib.org/eispack/tridib.f
'------------------------------------------------------------------
    DIM RV4(n), RV5(n)

    iErr = 0
    iTag = 0
    xu = d(1)
    x0 = d(1)
    u = 0
' LOOK FOR SMALL SUB-DIAGONAL ENTRIES AND DETERMINE AN
' INTERVAL CONTAINING ALL THE EIGENVALUES
    FOR i = 1 TO n
     x1 = u
     u = 0
     IF i <> n THEN u = ABS(e(i + 1))
     dMin1 = d(i) - (x1 + u): IF dMin1 < xu THEN xu = dMin1
     dMax1 = d(i) + (x1 + u): IF dMax1 > x0 THEN x0 = dMax1
     IF i <> 1 THEN
      Tst1 = ABS(d(i)) + ABS(d(i - 1))
      Tst2 = Tst1 + ABS(e(i))
      IF Tst2 <= Tst1 THEN e2(i) = 0
     END IF
    NEXT i
    dMax1 = ABS(xu): IF ABS(x0) > dMax1 THEN dMax1 = ABS(x0)
    x1 = dMax1 * EpsMach * n
    xu = xu - x1
    t1 = xu
    x0 = x0 + x1
    t2 = x0
' DETERMINE AN INTERVAL CONTAINING EXACTLY THE DESIRED EIGENVALUES
    iPi = 1
    iQ = n
    m1 = m11 - 1
    IF m1 <> 0 THEN
     iSTURM = 1
3050  V = x1
     x1 = xu + (x0 - xu) * .5
```

Tridib sonraki sayfada devam ediyor

Tridib devamı

```
     IF x1 = V GOTO 3980
     GOTO 3320
3060  IF iSi - m1 < 0 THEN
3065   xu = x1: GOTO 3050
     END IF
     IF iSi - m1 > 0 THEN
3070   x0 = x1: GOTO 3050
     END IF
     xu = x1
     t1 = x1
    END IF
    m22 = m1 + m
    IF m22 <> n THEN
     x0 = t2
     iSTURM = 2
     GOTO 3050
3080  IF (iSi - m22) < 0 THEN GOTO 3065
     IF (iSi - m22) > 0 THEN GOTO 3070
3085  t2 = x1
    END IF
    iQ = 0
    iR = 0
' ESTABLISH AND PROCESS NEXT SUBMATRIX, REFINING
' INTERVAL BY THE GERSCHGORIN BOUNDS
3100 IF iR = m GOTO 3001
    iTag = iTag + 1
    iPi = iQ + 1
    xu = d(iPi)
    x0 = d(iPi)
    u = 0
    FOR iQ = iPi TO n
     x1 = u
     u = 0
     V = 0
     IF iQ <> n THEN
      u = ABS(e(iQ + 1))
      V = e2(iQ + 1)
     END IF
     dMin1 = d(iQ) - (x1 + u)
     IF dMin1 < xu THEN xu = dMin1
     dMax1 = d(iQ) + (x1 + u)
     IF dMax1 > x0 THEN x0 = dMax1
     IF V = 0 THEN EXIT FOR
    NEXT iQ
    x1 = ABS(xu)
    dMax1 = ABS(x0)
    IF dMax1 > x1 THEN x1 = dMax1
    IF EPS1 <= 0 THEN EPS1 = -x1
    IF iPi = iQ THEN
' CHECK FOR ISOLATED ROOT WITHIN INTERVAL
     IF (t1 > d(iPi)) OR (d(iPi) >= t2) THEN GOTO 3940
     m1 = iPi
     m2 = iPi
     RV5(iPi) = d(iPi)
     GOTO 3900
    END IF
    x1 = x1 * (iQ - iPi + 1)
    BL = t1: dMax1 = xu - x1
    IF dMax1 > BL THEN BL = dMax1
    BU = t2: dMin1 = x0 + x1
    IF dMin1 < BU THEN BU = dMin1
    x1 = BL
    iSTURM = 3
    GOTO 3320
3200 m1 = iSi + 1
    x1 = BU
    iSTURM = 4
    GOTO 3320
3220 m2 = iSi
    IF m1 > m2 GOTO 3940
' FIND ROOTS BY BISECTION
    x0 = BU
    iSTURM = 5
    FOR i = m1 TO m2
     RV5(i) = BU
     RV4(i) = BL
    NEXT i
' LOOP FOR K-TH EIGENVALUE
' FOR K=M2 STEP -1 UNTIL M1 DO --
' (-DO- NOT USED TO LEGALIZE -COMPUTED GO TO-)
    k = m2
3250 xu = BL
    FOR i = k TO m1 STEP -1
     IF xu >= RV4(i) THEN GOTO 3260
     xu = RV4(i)
     EXIT FOR
3260 NEXT i
    IF x0 > RV5(k) THEN x0 = RV5(k)
' NEXT BISECTION STEP
3300 x1 = (xu + x0) * .5
    Tst1 = ABS(xu) + ABS(x0) + ABS(EPS1)
    Tst2 = Tst1 + ABS(x0 - xu) / 2!
    IF Tst2 = Tst1 GOTO 3420
'IN-LINE PROCEDURE FOR STURM SEQUENCE
3320 iSi = iPi - 1
    u = 1
    FOR i = iPi TO iQ
     IF u = 0 THEN
      V = ABS(e(i)) / EpsMach
      IF e2(i) = 0 THEN V = 0
     GOTO 3330
     END IF
     V = e2(i) / u
3330  u = d(i) - x1 - V
     IF u < 0 THEN iSi = iSi + 1
    NEXT i
    ON iSTURM GOTO 3060, 3080, 3200, 3220, 3360
' REFINE INTERVALS
3360 IF iSi >= k THEN GOTO 3400
    xu = x1
    IF iSi >= m1 THEN GOTO 3380
    RV4(m1) = x1
    GOTO 3300
3380 RV4(iSi + 1) = x1
    IF RV5(iSi) > x1 THEN RV5(iSi) = x1
    GOTO 3300
3400 x0 = x1
    GOTO 3300
'K-TH EIGENVALUE FOUND
3420 RV5(k) = x1
    k = k - 1
    IF k >= m1 THEN GOTO 3250
' ORDER EIGENVALUES TAGGED WITH THEIR
' SUBMATRIX ASSOCIATIONS
3900 iSi = iR
    iR = iR + m2 - m1 + 1
    j = 1
    k = m1
    FOR L = 1 TO iR
     IF j <= iSi THEN
      IF k > m2 THEN EXIT FOR
      IF RV5(k) >= W(L) THEN GOTO 3915
      FOR iI = j TO iSi
       i = L + iSi - iI
       W(i + 1) = W(i)
       iND(i + 1) = iND(i)
      NEXT iI
     END IF
     W(L) = RV5(k)
     iND(L) = iTag
     k = k + 1
     GOTO 3920
3915  j = j + 1
3920 NEXT L

3940 IF iQ < n THEN GOTO 3100
    GOTO 3001
' SET ERROR - INTERVAL CANNOT BE FOUND
' CONTAINING EXACTLY THE DESIRED EIGENVALUES
3980 iErr = 3 * n + iSTURM
3001 BL = t1
    BU = t2

END SUB ' End of Tridib
```

```
SUB Reduc (n, a(), b(), dL(), iErr)
'------------------------------------------------------------------
' This subroutine reduces the generalized symmetric eigenproblem
' A*x=(lambda)*b*x, where b is positive definite, to the standard
' symmetric eigenproblem using the cholesky factorization of b.

' This subroutine is a translation of the algol procedure reduc1,
' num. math. 11, 99-110(1968) by martin and wilkinson.
' handbook for auto. comp., vol.ii-linear algebra, 303-314(1971).

' on input
' n is the order of the matrices a and b.  if the cholesky factor L of b
'   is already available, n should be prefixed with a minus sign.
' a and b contain the real symmetric input matrices.  only the
'   full upper triangles of the matrices need be supplied.  if
'   n is negative, the strict lower triangle of b contains,
'   instead, the strict lower triangle of its cholesky factor L.
' dL contains, if n is negative, the diagonal elements of L.

' on output
' a contains in its full lower triangle the full lower triangle
'   of the symmetric matrix derived from the reduction to the
'   standard form.  the strict upper triangle of a is unaltered.
' b contains in its strict lower triangle the strict lower triangle of
'   its cholesky factor L.  the full upper triangle of b is unaltered.
' dL contains the diagonal elements of L.

' ierr is set to
'   zero       for normal return,
'   7*n+1      if b is not positive definite.

' This version dated august 1983.
' FORTRAN code: www.netlib.org/eispack
' ------------------------------------------------------------------
   iErr = 0
   nn = ABS(n)
   IF n > 0 THEN
' form L in the arrays b and dL
    FOR i = 1 TO n
     FOR j = i TO n
      x = b(i, j)
      FOR k = 1 TO i - 1
       x = x - b(i, k) * b(j, k)
      NEXT k
      IF j = i THEN
       IF x <= 0 THEN
' set error - b is not positive definite
        iErr = 7 * n + 1
        EXIT SUB
       END IF
       y = SQR(x)
       dL(i) = y
      ELSE
       b(j, i) = x / y
     END IF
     NEXT j
    NEXT i
   END IF

' form the transpose of the upper triangle of inv(L)*a
' in the lower triangle of the array a
   FOR i = 1 TO nn
    y = dL(i)
    FOR j = i TO nn
     x = a(i, j)
     FOR k = 1 TO i - 1
      x = x - b(i, k) * a(j, k)
     NEXT k
     a(j, i) = x / y
    NEXT j
   NEXT i

' pre-multiply by inv(L) and overwrite
   FOR j = 1 TO nn
    FOR i = j TO nn
     x = a(i, j)
     FOR k = j TO i - 1
      x = x - a(k, j) * b(i, k)
     NEXT k
     FOR k = 1 TO j - 1
      x = x - a(j, k) * b(i, k)
     NEXT k
     a(i, j) = x / dL(i)
    NEXT i
   NEXT j

END SUB ' end of Reduc
```

```
SUB rebak (n, b(), dL(), m, z())
'------------------------------------------------------------------
' This subroutine forms the eigenvectors of a generalized symmetric
' eigensystem by back transforming those of the derived symmetric
' matrix determined by REDUC.
'
' on input
'  n is the order of the matrix system.
'  b contains information about the similarity transformation (cholesky
'    decomposition) used in the reduction by  REDUC in its strict
'    lower triangle.
'  dL contains further information about the transformation.
'  m is the number of eigenvectors to be back transformed.
'  z contains the eigenvectors to be back transformed in its first m columns.

' on output
'  z contains the transformed eigenvectors in its first m columns.
'------------------------------------------------------------------
    FOR j = 1 TO m
     FOR i = n TO 1 STEP -1
      x = z(i, j)
      IF i <> n THEN
       i1 = i + 1
       FOR k = i1 TO n
        x = x - b(k, i) * z(k, j)
       NEXT k
      END IF
      z(i, j) = x / dL(i)
     NEXT i
    NEXT j

END SUB ' End of rebak
```

```
SUB reduce (n, mb, a(), b(), iErr)
' calulates invers and of Squre root of B
' Reduces general eigenproblem (A-Lamda*B)*x=0 in standard problem (A-Lamda*I)x=0
' invers and Square root of B
    iErr = 0
    FOR i = 1 TO n
     IF b(i) <= 0 THEN
      iErr = i
      EXIT SUB
     END IF
     b(i) = 1 / SQR(b(i))
    NEXT i
' Reducing
' Calculate B*A
    FOR i = 1 TO n
     FOR j = 1 TO mb
      a(i, j) = b(i) * a(i, j)
     NEXT j
    NEXT i
' Calculate B*A*B
    FOR k = 1 TO n
     FOR j = 1 TO mb
      i = k + j - 1
      IF i > n THEN EXIT FOR
      k1 = k - i + mb
      a(i, k1) = a(i, k1) * b(k)
     NEXT j
    NEXT k

END SUB ' end of reduce
```

```
SUB Bandr (n, mb, a(), d(), e(), e2())
' This subroutine reduces a real symmetric band matrix
' to a symmetric tridiagonal matrix using and optionally
' accumulating orthogonal similarity transformations.

' This subroutine is a translation of the algol procedure bandrd,
' num. math. 12, 231-241(1968) by schwarz.
' handbook for auto. comp., vol.ii-linear algebra, 273-283(1971).

' on input
'  n  is the order of the matrix.
'  mb is the half band width of the matrix, defined as the number of
'     adjacent diagonals, including the principal diagonal, required to
'     specify the non-zero portion of the lower triangle of the matrix.
'  a  contains the lower triangle of the symmetric band input matrix stored
'     as an n by mb array.  its lowest subdiagonal is stored in the last
'     n+1-mb positions of the first column, its next subdiagonal in the last
'     n+2-mb positions of the second column, further subdiagonals similarly,
'     and finally its principal diagonal in the n positions of the last column.
'     contents of storages not part of the matrix are arbitrary.

' on output
'  a  has been destroyed, except for its last two columns which
'     contain a copy of the tridiagonal matrix.
'  d  contains the diagonal elements of the tridiagonal matrix.
'  e  contains the subdiagonal elements of the tridiagonal
'     matrix in its last n-1 positions.  e(1) is set to zero.
'     e2 contains the squares of the corresponding elements of e.
'     e2 may coincide with e if the squares are not needed.
'     this version dated september 1989.
' FORTRAN code: http://www.netlib.org/eispack
'------------------------------------------------------------------------------

       dmin = EpsMach
       dminrt = SQR(dmin)
' initialize diagonal scaling matrix
       FOR j = 1 TO n
        d(j) = 1
       NEXT j
       m1 = mb - 1
       IF m1 < 1 THEN
       FOR j = 1 TO n
        d(j) = a(j, mb)
        e(j) = 0
        e2(j) = 0
        NEXT j
       EXIT SUB
       END IF
       IF m1 = 1 THEN GOTO BR800

       n2 = n - 2
       FOR k = 1 TO n2
        Maxr = Min(n - k, m1)

        FOR iR = Maxr TO 2 STEP -1
         kr = k + iR
         mr = mb - iR
         g = a(kr, mr)
         a(kr - 1, 1) = a(kr - 1, mr + 1)
         iUgl = k

         FOR j = kr TO n STEP m1
          j1 = j - 1
          j2 = j1 - 1
          IF g = 0 THEN GOTO BR600
          b1 = a(j1, 1) / g
          b2 = b1 * d(j1) / d(j)
          IF ABS(b1) > 1 THEN
           u = 1 / b1
           s2 = u / (u + b2)
           ELSE
            s2 = 1 / (1 + b1 * b2)
          END IF

          IF s2 >= .5 THEN GOTO BR450
          b1 = g / a(j1, 1)
          b2 = b1 * d(j) / d(j1)
          c2 = 1 - s2
          d(j1) = c2 * d(j1)
          d(j) = c2 * d(j)
          f1 = 2 * a(j, m1)
          f2 = b1 * a(j1, mb)
          a(j, m1) = -b2 * (b1 * a(j, m1) - a(j, mb)) - f2 + a(j, m1)
          a(j1, mb) = b2 * (b2 * a(j, mb) + f1) + a(j1, mb)
          a(j, mb) = b1 * (f2 - f1) + a(j, mb)
```

Bandr sonraki sayfada devam ediyor

Bandr devamı

```
        FOR L = iUgl TO j2
         i2 = mb - j + L
         u = a(j1, i2 + 1) + b2 * a(j, i2)
         a(j, i2) = -b1 * a(j1, i2 + 1) + a(j, i2)
         a(j1, i2 + 1) = u
        NEXT L

        iUgl = j
        a(j1, 1) = a(j1, 1) + b2 * g
        IF j = n THEN GOTO BR500
        MaxL = Min(n - j1, m1)

        FOR L = 2 TO MaxL
         i1 = j1 + L
         i2 = mb - L
         u = a(i1, i2) + b2 * a(i1, i2 + 1)
         a(i1, i2 + 1) = -b1 * a(i1, i2) + a(i1, i2 + 1)
         a(i1, i2) = u
        NEXT L

        i1 = j + m1
        IF i1 <= n THEN g = b2 * a(i1, 1)
        GOTO BR500

BR450:     u = d(j1)
        d(j1) = s2 * d(j)
        d(j) = s2 * u
        f1 = 2 * a(j, m1)
        f2 = b1 * a(j, mb)
        u = b1 * (f2 - f1) + a(j1, mb)
        a(j, m1) = b2 * (b1 * a(j, m1) - a(j1, mb)) + f2 - a(j, m1)
        a(j1, mb) = b2 * (b2 * a(j1, mb) + f1) + a(j, mb)
        a(j, mb) = u
        FOR L = iUgl TO j2
         i2 = mb - j + L
         u = b2 * a(j1, i2 + 1) + a(j, i2)
         a(j, i2) = -a(j1, i2 + 1) + b1 * a(j, i2)
         a(j1, i2 + 1) = u
        NEXT L
        iUgl = j
        a(j1, 1) = b2 * a(j1, 1) + g

        IF j = n THEN GOTO BR500
        MaxL = Min(n - j1, m1)
        FOR L = 2 TO MaxL
         i1 = j1 + L
         i2 = mb - L
         u = b2 * a(i1, i2) + a(i1, i2 + 1)
         a(i1, i2 + 1) = -a(i1, i2) + b1 * a(i1, i2 + 1)
         a(i1, i2) = u
        NEXT L
        i1 = j + m1
        IF i1 <= n THEN
         g = a(i1, 1)
         a(i1, 1) = b1 * a(i1, 1)
        END IF
BR500:    NEXT j

BR600:   NEXT iR

'Rescale to avoid underflow or overflow
       IF (k MOD 64) = 0 THEN
        FOR j = k TO n
         IF d(j) >= dmin THEN GOTO BR650
         MaxL = Min(mb + 1 - j, 1)
         FOR L = MaxL TO m1
          a(j, L) = dminrt * a(j, L)
         NEXT L
         IF j <> n THEN
          MaxL = Min(n - j, m1)
          FOR L = 1 TO MaxL
           i1 = j + L
           i2 = mb - L
           a(i1, i2) = dminrt * a(i1, i2)
          NEXT L
         END IF
         a(j, mb) = dmin * a(j, mb)
         d(j) = d(j) / dmin
BR650:    NEXT j
       END IF
      NEXT k
```

```
' Form square root of scaling matrix
BR800:  FOR j = 2 TO n
         e(j) = SQR(d(j))
        NEXT j
        u = 1
        FOR j = 2 TO n
         a(j, m1) = u * e(j) * a(j, m1)
         u = e(j)
         e2(j) = a(j, m1) ^ 2
         a(j, mb) = d(j) * a(j, mb)
         d(j) = a(j, mb)
         e(j) = a(j, m1)
        NEXT j
        d(1) = a(1, mb)
        e(1) = 0
        e2(1) = 0

END SUB ' end of Bandr
```

```
SUB bandv (n, mb, a(), m, W(), z(), iErr)
'------------------------------------------------------------------
' this subroutine finds those eigenvectors of a real symmetric
' band matrix corresponding to specified eigenvalues, using inverse iteration

' on input
'  n is the Order of the matrix.
'  nb is the half band width of the matrix a.
'  a contains the lower triangle of the symmetric band input
'    matrix stored as an n by mb array.  its lowest subdiagonal
'    is stored in the last n+1-mb positions of the first column,
'    its next subdiagonal in the last n+2-mb positions of the
'    second column, further subdiagonals similarly, and finally
'    its principal diagonal in the n positions of column mb.
'  m is the number of specified eigenvalues.
'  w contains the m known eigenvalues in ascending order.

' on output
'  a and w are unaltered.
'  z contains the associated set of orthogonal eigenvectors.
'    any vector which fails to converge is set to zero
'  iErr is set to
'          zero    for normal return,
'          -iR     if the eigenvector corresponding to the iR-th
'                  eigenvalue fails to converge.

' rv and rv6 are temporary storage arrays.  note that rv is
'           of dimension at least n*(2*mb-1).

' Subprograms called: Pythag, Sign, EpsMach

' this version dated august 1983.
' FORTRAN code: http://www.netlib.org/eispack
'------------------------------------------------------------------
    DIM RV(n * (2 * mb - 1)), RV6(n)

    iErr = -1
    IF m <= 0 THEN EXIT SUB
    iErr = 0
    x0 = 0
    m1 = mb - 1
    m21 = m1 + mb

' compute aNorm of matrix
     aNorm = 0
     FOR j = 1 TO mb
      V = 0
      FOR i = mb + 1 - j TO n
       V = V + ABS(a(i, j))
      NEXT i
      IF V > aNorm THEN aNorm = V
     NEXT j

' eps2 is the criterion for grouping,
' eps3 replaces zero pivots and equal roots are modified by eps3,
' eps4 is taken very small to avoid overflow
      IF aNorm = 0 THEN aNorm = 1
      Eps2 = .001 * aNorm
      eps3 = ABS(aNorm) * (EpsMach# * aNorm)
      uk = SQR(n)
      eps4 = uk * eps3
      iGroup = 0

' find vectors by inverse iteration
    FOR iR = 1 TO m
     its = 1
     x1 = W(iR)

' look for close or coincident roots
     IF ABS(x1 - x0) >= Eps2 THEN
      iGoup = 0
     ELSE
      iGroup = iGroup + 1
      IF x1 - x0 <= 0 THEN x1 = x0 + eps3
     END IF

' expand matrix, subtract eigenvalue, and initialize vector
     FOR i = 1 TO n
      ij = i + Min(0, i - m1) * n
      kj = ij + mb * n
      ij1 = kj + m1 * n

      FOR j = 1 TO m1
       IF ij <= m1 THEN
        IF ij <= 0 THEN
         RV(ij1) = 0
         ij1 = ij1 + n
        END IF
       ELSE
       RV(ij) = a(i, j)
       END IF
       ij = ij + n
       iI = i + j
       IF iI <= n THEN
        RV(kj) = a(iI, mb - j)
        kj = kj + n
       END IF
      NEXT j

      RV(ij) = a(i, mb) - x1
      RV6(i) = eps4
     NEXT i

' elimination with interchanges
     IF m1 <> 0 THEN
      FOR i = 1 TO n
       iI = i + 1
       Maxk = Min(i + m1 - 1, n)
       Maxj = Min(n - i, m21 - 2) * n
       FOR k = i TO Maxk
        kj1 = k
        j = kj1 + n
        jj = j + Maxj
        FOR kj = j TO jj STEP n
         RV(kj1) = RV(kj)
         kj1 = kj
        NEXT kj
        RV(kj1) = 0
       NEXT k

       IF i <> n THEN
        u = 0
        Maxk = Min(i + m1, n)
        Maxj = Min(n - iI, m21 - 2) * n
        FOR j = i TO Maxk
         IF ABS(RV(j)) >= ABS(u) THEN
          u = RV(j)
          k = j
         END IF
        NEXT j

        j = i + n
        jj = j + Maxj
        IF k <> i THEN
         kj = k
         FOR ij = i TO jj STEP n
          V = RV(ij)
          RV(ij) = RV(kj)
          RV(kj) = V
          kj = kj + n
         NEXT ij
        END IF

        IF u <> 0 THEN
         FOR k = iI TO Maxk
          V = RV(k) / u
          kj = k
          FOR ij = j TO jj STEP n
           kj = kj + n
           RV(kj) = RV(kj) - V * RV(ij)
          NEXT ij
         NEXT k
        END IF

       END IF

      NEXT i

     END IF
```

Bandv sonraki sayfada devam ediyor

Bandv devamı

```
' back substitution
600  FOR i = n TO 1 STEP -1
      Maxj = Min(n - i + 1, m21)
      IF Maxj <> 1 THEN
       ij1 = i
       j = ij1 + n
       jj = j + (Maxj - 2) * n
       FOR ij = j TO jj STEP n
        ij1 = ij1 + 1
        RV6(i) = RV6(i) - RV(ij) * RV6(ij1)
       NEXT ij
      END IF

      V = RV(i)
      IF ABS(V) < eps3 THEN V = Sign(eps3, V)
      RV6(i) = RV6(i) / V
     NEXT i

' orthogonalize with respect to previous members of iGroup
     xu = 1
     FOR jj = 1 TO iGroup
      j = iR - iGroup - 1 + jj
      xu = 0
      FOR i = 1 TO n
       xu = xu + RV6(i) * z(i, j)
      NEXT i

      FOR i = 1 TO n
       RV6(i) = RV6(i) - xu * z(i, j)
      NEXT i
     NEXT jj

     aNorm = 0
     FOR i = 1 TO n
      aNorm = aNorm + ABS(RV6(i))
     NEXT i

' choose a new starting vector
     IF aNorm < .1 THEN
      IF its < n THEN
       its = its + 1
       xu = eps4 / (uk + 1)
       RV6(1) = eps4

       FOR i = 2 TO n
        RV6(i) = xu
       NEXT i

       RV6(its) = RV6(its) - eps4 * uk
       GOTO 600
      ELSE

' set error - non-converged eigenvector
       iErr = -iR
       xu = 0
       GOTO 870
      END IF
     END IF

' normalize so that sum of squares is 1
     u = 0
     FOR i = 1 TO n
      u = Pythag(u, RV6(i))
     NEXT i
     xu = 1 / u

870  FOR i = 1 TO n
      z(i, iR) = RV6(i) * xu
     NEXT i

     x0 = x1

    NEXT iR

END SUB ' end of Bandv
```

```
FUNCTION Min (ia, ib)
' Returns minimum of two integers
 IF ia < ib THEN Min = ia ELSE Min = ib
END FUNCTION ' end of Min
```