INFORMATICA 1/1982

ALGORITHMS FOR THE SOLUTION OF THE GENERALIZED EIGENVALUE PROBLEM

Z. BOHTE, J. GRAD

UDK: 521.643.5

INSTITUTE OF MATHEMATICS, PHYSICS AND MECHANICS
JADRANSKA 19, LJUBLJANA, YUGOSLAVIA

In this paper we deal with generalized eigenvalue problem of matrix A(z) the elements of which are polynomials in z. The well known iterative methods of Muller, Newton and Laguerre for finding the zeros of function f(z) = det A(z) are analysed. Decompositions of matrix A(z) and its derivatives are introduced in order to simplify the computations of the values of f(z) and its first and second derivatives. Comparative analysis gives some indicators about the rate of convergence, computer time and accuracy of the computed eigenvalues for each of the method used. The analyzed methods proved generally to be stable, economical and easily applicable. FORTRAN subroutines and an example of main calling programme are added for Muller's and Laguerre's methods which proved to be more efficient than Newton's.

ALGORITMI ZA REŠEVANJE POSPLOŠENEGA PROBLEMA LASTNIH VREDNOSTI. V članku je obravnavano numerično reševanje posplošenega problema lastnih vrednosti za matriko A(z) z elementi, ki so polinomi spremenljivke z. Primerjane so dobro znane iterativne metode Mullerja, Newtona in Laguerra za računanje ničel polinoma f(z) = det A(z). Vrednosti polinoma f(z) in njegovih odvodov so izračunane na osnovi razcepa matrike A(z). Primerjava metod daje vpogled v hitrost konvergence, porabljen računski čas in natančnost izračunanih lastnih vrednosti za vsako metodo posebej. Analizirane metode so vse numerično stabilne, ekonomične in enostavno uporabne. Podprogrami v FORTRAN-u in primer glavnega programa so dodani za Mullerjevo in Laguerrovo metodo, ki sta se izkazali za bolj učinkoviti od Newtonove.

Introduction

In this paper three numerical methods for the solution of the generalized algebraic eigenvalue problem

$$A(z)x = 0 \tag{1}$$

are described and the programmes of two of them are presented. Matrix A(z) is of the form

$$A(z) = A_o + A_1 z + \dots + A_m z^m \tag{2}$$

where all A_i are real square matrices of order n.

The standard source of this problem is the solution of a system of n linear homogeneous ordinary differential equations of order m.

If the matrix A_m is non-singular then the problem (1) is equivalent to the classical eigenvalue problem

$$\begin{bmatrix} 0 & I & 0 & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ - & - & - & --- & - \\ 0 & 0 & 0 & \dots & I \\ B_o & B_1 & B_2 & \dots & B_{m-1} \end{bmatrix} \cdot \begin{bmatrix} y_o \\ y_1 \\ - \\ y_{m-2} \\ y_{m-1} \end{bmatrix} = z \begin{bmatrix} y_o \\ y_1 \\ - \\ y_{m-2} \\ y_{m-1} \end{bmatrix} \tag{3}$$

of order m.n where $B_i = -A_m^{-1}A_i$. A similar reduction is possible if A_o is non-singular.

Although we have several efficient and stable numerical methods for the solution of the classical eigenvalue problem, for instance, QR algorithm [4], it is not economical to reduce the problem (1) to the standard form (3). Moreover, many practical problems like "flutter" problem in aerodynamics, are such that both A_o and A_m are singular and the reduction to the standard form causes further difficulties [3].

It is therefore considered as more advantageous to work directly with the matrix (2) in all cases. The most natural approach is to determine the zeroes of the polynomial

$$f(z) = \det A(z) \tag{4}$$

using some of the many well known methods.

There are some efficient and stable algorithms for the evaluation of the determinant of the matrix, for instance, Gaussian elimination with pivoting [4]. It has been believed so far that the evaluation of the derivative of the determinant involves much more work than the evaluation of the determinant does, and therefore the interpolation methods which require only function values had been suggested for the solution of the algebraic equation

$$f(z) = 0 \tag{5}$$

see [3], [4]. The most popular of these methods is Muller's method [4] of successive quadratic interpolation. The prejudice against the methods which require the derivatives of the function is probably based on the fact that the derivative of the determinant is a sum of n determinants. However, there can be found in [1] a formula for the logarithmic derivative of the determinant which together with its derivative enable us to solve (5) by using Newton's or Laguerre's method efficiently.

In this paper algorithms for the solution of the generalized eigenvalue problem (1), based on Muller's, Newton's and Laguerre's methods are described and compared with regard to: (i) the number of operations (multiplications and divisions) involved in iteration step, (ii) the number of iterations, and (iii) the computer time used. The computations were performed on a number of typical examples.

Muller's method

Muller's method for the solution of the equation (5) as described in [4] is the following iterative process. From the three previous consecutive approximations to a root of (5): z_{r-2}, z_{r-1}, z_r, the parabola through the points

$$(z_{r-2}, f(z_{r-2})),\ (z_{r-1}, f(z_{r-1})),\ (z_r, f(z_r))$$

is formed and its zero which is the closest to z_r is accepted as the next approximation, say z_{r+1}, to a root. The asymptotic rate of convergence to a simple root is of order 1.84 and at each iteration step only one new function value is required. It is necessary to use complex arithmetic throughout.

The algorithm is as follows.

For r=2, 3, ... we calculate

$$h_r = z_r - z_{r-1},\ k_r = h_r/h_{r-1},\ d_r = 1 + k_r$$

$$g_r = f_{r-2}k_r^2 - f_{r-1}d_r^2 + f_r(k_r + d_r)$$

$$z_{r+1} = z_r - 2f_r h_r d_r/(g_r \pm (g_r^2 - 4f_r d_r k_r(f_{r-2}k_r - f_{r-1}d_r + f_r))^{1/2}) \tag{6}$$

$$f_{r+1} = f(z_{r+1})$$

The sign in the denominator of (6) is chosen so as to give the correction to z_r the smaller absolute magnitude.

The choice of the first three approximations is left to the user. There are no proofs of the global convergence of the iterative process.

In this case we have to calculate the value of

$$f = f(z) = \det A(z)$$

at each iteration step what can be done most efficiently by triangular decomposition of the matrix A(z) using partial pivoting as follows. For the given value of z we first calculate the elements of the matrix

$$A = A(z) \tag{7}$$

what requires $m.n^2$ operations. Then we decompose A into the form

$$PA = LU \tag{8}$$

where L is unit lower triangular matrix, U is upper triangular matrix, and P is the corresponding permutation matrix [4]. This step requires $n^3/3 - n/3$ operations. Finally we compute

$$f = \pm \det U = \pm u_{11}u_{22} \cdots u_{nn}$$

what requires n-1 operations. Together we must perform

$$op_M = m.n^2 + n^3/3 + 2n/3-1$$

operations for the calculation of f.

Newton's method

The well known Newton's method for the solution of (5) is the iterative process

$$z_{r+1} = z_r - f(z_r)/f'(z_r), \text{ for } r=1,2,\ldots$$

At each iteration step we evaluate the correction of the current approximation to a root as the reciprocal of the logarithmic derivative of the function. From [1], [2] we have the formula

$$f'(z)/f(z) = \mathrm{tr}(A^{-1}(z)A'(z)) = \mathrm{tr}\ X \tag{9}$$

where $A'(z)$ denotes the derivative of the matrix $A(z)$ with respect to z.

An efficent algorithm for calculating the expression (9) is as follows.

First we calculate the matrices

$$A = A(z), \quad B = A'(z) \tag{10}$$

which requires $(2m-1)n^2$ operations. Then we decompose A as in (8) obtaining

$$PA = LU \tag{11}$$

which requires $n^3/3 - n/3$ operations, compute the matrices Y and X which satisfy

$$LY = PB \tag{12}$$

$$UX = Y \tag{13}$$

As we need only the trace of X, we can use formulae

$$x_{ik} = (y_{ik} - \sum_{j=i+1}^{n} u_{ij}x_{jk})/u_{ii} \tag{14}$$

for $i=n, n-1, \ldots, 1$ and $k=1,2, \ldots, i$. These two steps require together $(n^3/2 - n^2/2) + (n^3/6 + n^2/2 + n/3)$ operations. So we can calculate

$$f'(z)/f(z) = \sum_{i=1}^{n} x_{ii} \tag{15}$$

with

$$op_N = (2m-1)n^2 + n^3 \tag{16}$$

operations.

Although the asymptotic rate of convergence of the Newton's method to a simple root is 2 there seem to be little advantage in using Newton's method against Muller's. There appear to be almost three times as many operations per iteration step in Newton's method than in Muller's.

Laguerre's method

The quite famous Laguerre's method requires apart from the function value also the values of its first two derivatives. The asymptotic rate of convergence to a simple root is 3. Perhaps the most attractive feature from the users point of view is its stability in global convergence for any value of the first approximation.

The iterative process is defined by [4]

$$z_{r+1} = z_r - n/(S_1 \pm ((n-1)(nS_2 - S_1^2))^{1/2})$$

for $r=1,2, \ldots$, where

$$S_1 = f'(z_r)/f(z_r)$$

$$S_2 = (f'(z_r)^2 - f(z_r)f''(z_r))/f(z_r)^2$$

The sign in the denominator is chosen so as to give the correction to z_r the smaller absolute magnitude. The form of S_1 which is suitable for numerical evaluation is defined by (9) and (10) - (15), while the formula for S_2 is derived from (9) by differentiation, see [2],

$$S_2 = tr((A^{-1}(z_r)A'(z_r))^2) - tr(A^{-1}(z_r)A''(z_r)) \tag{17}$$

The algorithm for evaluation of S_2 demands in addition to the starting operations for S_1, i.e. (10) and (11), the following computations:

We must first calculate the matrix

$$C = A''(z) \tag{18}$$

with additional $(m-2)n^2$ operations. Then we calculate the trace of the matrix $W = A^{-1}C$ by steps

$$LZ = PC \tag{19}$$

$$UW = Z \tag{20}$$

$$tr\, W = \sum_{i=1}^{n} w_{ii} \tag{21}$$

which require $(n^3/2 - n^2/2) + (n^3/6 + n^2/2 + n/3) + (n - 1)$ operations. The first term in (17) is found by calculating first the full matrix X by (12) and (13) with n^3 operations and then the trace of X^2 as

$$tr(X^2) = \sum_{i=1}^{n} \sum_{k=1}^{n} x_{ik}x_{ki} \tag{22}$$

which requires additional n^2 operations.

At each iteration step we must perform

$$op_L = (3m-3)n^2 + 2n^3 + n^2$$

operations for the calculation of S_1 and S_2. This number is almost twice as large as (16) for the Newton's method.

Numerical tests

The programmes in FORTRAN programming language were written for all three algorithms and tested on CDC CYBER 72 computer. Single precision complex arithmetic was used with $t=48$, where t is the number of significant digits in the mantissa of a binary floating-point number. The iterative process was stopped after either the computed correction of the computed approximation to the eigenvalue was less than or equal to ε or if $|u_{ii}| \le \varepsilon$, for $1 \le i \le n$, occured in (8) or (11). The value for ε was chosen as $10.\|A(z)\|_E \cdot 2^{-t}$.

The implicit deflation of $f(z)$ was applied in the programmes in order to prevent the iterative process to converge to one of the previously computed eigenvalues, say $x_1, x_2, \ldots, x_m$. This introduces the following modifications of algorithms:

$$f(z) \quad \text{into} \quad f(z)/\prod_{i=1}^{m}(z-x_i) \qquad (23)$$

$$f'(z)/f(z) \quad \text{into} \quad f'(z)/f(z)-\sum_{i=1}^{m}(z-x_i)^{-1}$$

$$S_1 \quad \text{into} \quad S_1-\sum_{i=1}^{m}(z-x_i)^{-1} \quad \text{and} \qquad (24)$$

$$S_2 \quad \text{into} \quad S_2-\sum_{i=1}^{m}(z-x_i)^{-2} \qquad (25)$$

Some typical examples of (1) were computed with multiple eigenvalues either of finite value or in the infinity. The values of the parameters m and n of $A(z)$ were on intervals $1 \le m \le 2$ and $3 \le n \le 12$ respectively. No breakdown of the iterative process occured regardless the algorithm used. All three methods proved to be stable and accurate. It was found that the previous computed eigenvalue and the values round it were the most suitable starting values for computing the next eigenvalue.

In the table

	Muller		Newton		Laguerre	
i	No	T	No	T	No	T
1	55	.57	41	.58	17	.45
2	0		0		0	
3	11	.13	19	.28	11	.30
4	0		0		0	
5	3	.04	4	.06	2	.06
6	0		0		0	
7	16	.17	41	.61	3	.08
8	1	.02	2	.04	2	.06
ALL	86	.93	107	1.57	35	.95

the number of iteration steps (No) and the computer time used (T in sec) are shown for each of the computed eigenvalue z_i, $i=1,\ldots,8$, of the system

$$(A_o + zA_1 + z^2A_2)x = 0$$

where $A_2 = I$, and

$$A_1 = \begin{bmatrix} 0 & -3 & 0 & -1 \\ 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \end{bmatrix}, \quad A_o = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -2 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

The system has a triple eigenvalue at $\pm$ i and a double eigenvalue at 0 [5]. Somehow similar results were obtained with other examples.

Three more examples are

1) $A(z) = A_o + A_1 z + A_2 z^2$

$$A_o = \begin{bmatrix} 1 & -1 & 1 \\ -15 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad A_1 = \begin{bmatrix} -2 & 1 & -1 \\ 3 & 0 & 1 \\ 1 & 0.5 & 0 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0.25 & 0 \\ -1 & 0 & 1 \end{bmatrix}.$$

z_i:

1.00000000000
2.08866333896
-0.256555796702 ± 0.896010203022 i
-2.91609433069
11.3405425851

2) $A(z) = A_o + A_1 z$

$$A_o = \begin{bmatrix} -1 & -3 & -3 & -3 & -3 & -3 \\ -3 & -4 & -3.1 & -3.1 & -3.1 & -3.1 \\ -3 & -3.1 & 2.8 & 3.8 & 3.8 & 3.8 \\ -3 & -3.1 & 3.8 & 9.8 & 10.7 & 10.7 \\ -3 & -3.1 & 3.8 & 10.7 & 12.6 & 14.6 \\ -3 & -3.1 & 3.8 & 10.7 & 14.6 & 15.6 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & -1 & -1 \\ 1 & 0 & -1 & -2 & -2 & -2 \\ 1 & 0 & -1 & -2 & -3 & -3 \\ 1 & 0 & -1 & -2 & -3 & -2 \end{bmatrix}$$

z_i:

0.908770404173 ± 1.93967680102 i
0.931536974557 ± 1.97197662562 i
4.18245919165
6.13692605089

3) $A(z) = A_o + A_1 z + A_2 z^2$

$$A_o = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1.5 & 1 & 0 & 0 \\ 0 & 0 & -8 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.4 & 1 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

z_i:

-1.09579878411

$-1.08865049880 \pm 1.04888469390\, i$

$-0.541227886923 \pm 1.51715942168\, i$

$$z_6 - z_{14} \approx K \cdot 10^7, \quad |K| \approx 1$$

The last nine eigenvalues lie in the infinity.

Muller´s and Laguerre´s methods were also tested on DEC 1091 computer with t=27. In one case when some of the elements of A(z) differed up to 10^{13} in their absolute values Muller´s method failed to compute correctly some of the eigenvalues while all the eigenvalues obtained by Laguerre´s method were correct.

References

1. BODEWIG, E. Matrix Calculus. North-Holland Publishing Company, Amsterdam, 1959.

2. BOHTE, Z. On Algorithms for the Calculation of Derivatives of the Determinant. Proc. ALGORITHMS 79 Symposium on Algorithms, Strbske Pleso, Czechoslovakia, 1979, pp. [illegible]-110.

3. PETERS, G., AND WILKINSON, J.H. Ax = λBx and the Generalized Eigenproblem. SIAM J. Numer. Anal. 7, 4(1970), 479-492.

4. WILKINSON, J.H. The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

5. LANCASTER, P. Lambda-matrices and Vibrating Systems. Pergamon Press, Oxford, London, Edinburgh, New York, Toronto, Paris, Braunschweig, 1966.

Programmes

1. Comments

In this chapter two subroutines EIGENM and EIGENL for computing the eigenvalues of a general matrix (2) are presented. An example of a possible main (calling) programme for both subroutines is also given at the end of the chapter. In EIGENM the eigenvalues are computed by Muller´s method while in EIGENL by Laguerre´s method. If A(z) in (2) satisfies $m \leq 3$ then the subroutines can be used such as they are presented. For $m > 3$ some changes in the subroutines are necessary as explained at the end of the programmes.

The subroutines EIGENM and EIGENL are completely selfcontained (EIGENM is composed of four subroutines EIGENM, F, ALAMDA, LU and EIGENL is composed of ten subroutines EIGENL, ALAMDA, LU, DERIV1, DERIV2, LØWER, UPPER, UPPER2, TRACE, TRXSQ) and communication to them is solely through their argument lists and CØMMØN statements. The entrace to the subroutine EIGENM is achieved by

```
  CØMMØN /MATRIX/ AO(ND,ND), A1(ND,ND), A2(ND,
 1ND)
  CALL EIGENM (M, N, ND, T, INDIC, LAM, PERMUT,
 1A, L, U)
```

The entrance to the subroutine EIGENL is achieved by

```
  CØMMØN /MATRIX/ AO(ND, ND), A1(ND,ND), A2(ND,
 1ND), A3(ND,ND)
  CALL EIGENL (M, N, ND, T, INDIC, LAM, PERMUT,
 1A, D, L, U)
```

The meaning of the parameters is described in the following lines.

The elements of matrices $A_o, A_1, A_2, \ldots, A_m$ in (2) are to be stored in the first N rows and columns of the two dimensional arrays A0, A1, A2, A3,... .

M, where $M \geq 1$, is the degree of z-matrix A(z) in (2).

N, where $N \geq 1$, is the order of matrices $A_o, A_1, A_2, \ldots, A_M$.

ND, where $ND \geq N$, defines the first dimension of the two dimensional arrays A0, A1, A2,.. .., A,D,L,U and the dimension of the one dimensional array PERMUT. Dimension of the one dimensional arrays INDIC and LAM must be equal to or greater than ND*M.

T is the number of binary digits in the

mantissa of a single precision floating-point number. T is of integer type.

The array INDIC indicates the success of the subroutine as follows

value of INDIC(I)	eigenvalue (I)
0	not found after 100 iteration steps
1	found

LAM is one dimensional complex array. The computed eigenvalues will be found in the first N*M places.

The arrays PERMUT, A, D, L, U is the working storage.

The main (calling) programme should contain the following declaration statements

```
 INTEGER, M,N,T,INDIC,PERMUT
 REAL A0,A1,A2,A3
 CØMPLEX LAM,A,L,U
 CØMMØN /MATRIX/ A0(ND,ND), A1(ND,ND),
1A2(ND,ND),A3(ND,ND)
 DIMENSIØN INDIC (ND*M),LAM(ND*M),PERMUT
1(ND),A(ND,ND+1),L(ND,ND+1),U(ND,ND+1)
 EQUIVALENCE (A(1),L(1),U(1))
```

and the following two additional declaration statements for the subroutine EIGENL

```
 CØMPLEX D
 DIMENSIØN D(ND,ND)
```

Note that the dimensions of the arrays in DIMENSIØN and CØMMØN statements of the main programme and in CØMMØN statements within subroutines must be constants of integer type. See the example in 5 where ND was replaced by 12, ND+1 by 13 and ND*M by 36.

2. Subroutine EIGENM - Muller's method

The subroutine is composed of four subroutines EIGENM, F, ALAMDA and LU.

```
      SUBRØUTINE EIGENM(M,N,ND,T,INDIC,LAM,
     1PERMUT,A,L,U)
      INTEGER M,N,ND,T,INDIC,PERMUT,I,IND,
     1ITERST, NM, R
      REAL EPS,EPS100, IM, RE
      REAL A0,A1,A2,A3
      CØMPLEX LAM,A,L,U,DENØM1,DENØM2,DR,FR,
     1FRM1,FRM2,GR,HR,HRM1,KR,ZR,ZRM1,ZRM2
      CØMMØN /MATRIX/ A0(ND,ND),A1(ND,ND),
     1A2(ND,ND),A3(ND,ND)
      DIMENSIØN INDIC(1),LAM(1),PERMUT(1),
     1A(ND,1),L(ND,1),U(ND,1)
      NM = N*M
      ITERST = 102
      DØ 1 I=1,NM
         INDIC(I) = 0
    1 CØNTINUE
      I = 1
C     THE MULLER ITERATIVE PRØCESS.
C     DETERMINATIØN ØF THREE STARTING
C     CØNSECUTIVE APPRØXIMATIØNS TØ  THE
C     CØMPUTED EIGENVALUE LAM(I),FØR
C     I=1,2,...NM.
    2 ZR = (1.0,0.0)*FLØAT(I)
    3 ZRM1 = 0.5*ZR
      ZRM2 = 0.1*ZR
C     THE INITIAL CØMPUTATIØNS ØF THE NEXT
C     APPRØXIMATIØN Z(R+1) TØ LAM(I),FØR R=2.
      R = 2
      CALL F(ZRM2,FRM2,EPS,I,IND,M,N,ND,T,LAM,
     1PERMUT,A,L,U)
         IF(IND.EQ.1)GØ TØ 5
      CALL F(ZRM1,FRM1,EPS,I,IND,M,N,ND,T,LAM,
     1PERMUT,A,L,U)
         IF(IND.EQ.1)GØ TØ 6
      CALL F(ZR,FR,EPS,I,IND,M,N,ND,T,LAM,
     1PERMUT,A,L,U)
         IF(IND.EQ.1)GØ TØ 7
C
      HRM1 = ZRM1-ZRM2
      HR   = ZR  -ZRM1
      KR   = HR/HRM1
C     THE MAIN LØØP FØR  THE CØMPUTATIØN ØF
C     THE NEXT APPRØXIMATIØN Z(R+1) TØ LAM(I),
C     FØR R=2,3,...,ITERST.
    4   DR=1.0+KR
        GR=FRM2*KR**2 - FRM1*DR**2 + FR*(KR+DR)
        DENØM2 = CSQRT(GR**2-4.*FR*DR*KR*
     1  (FRM2*KR-FRM1*DR+FR))
        DENØM1 = GR+DENØM2
        DENØM2 = GR-DENØM2
        IF(CABS(DENØM2).GT.CABS(DENØM1))DENØM1
     1  = DENØM2
        KR = -2.0*FR*DR/DENØM1
        HR = HR*KR
        GR = ZR
        ZR = ZR+HR
        FRM2 = FRM1
        FRM1 = FR
        CALL F(ZR,FR,EPS,I,IND,M,N,ND,T,LAM,
     1  PERMUT,A,L,U)
C
        IF(IND.EQ.1)GØ TØ 7
        IF(CABS(ZR-GR).LE.10.*EPS)GØ TØ 7
```

```
      R = R+1
      IF(R.LT.ITERST)GØ TØ 4
      LAM(I) = ZR
      GØ TØ 9
    5 ZR = ZRM2
      GØ TØ 7
    6 ZR = ZRM1
    7 LAM(I) = ZR
      INDIC(I) = 1
      I = I+1
C     COMPUTE THE CØMPLEX CØNJUGATE VALUE ØF
C     LAM(I).
      RE = REAL(ZR)
      IM = AIMAG(ZR)
      EPS100 = 100.0*EPS
      IF(ABS(IM).LE.EPS100)GØ TØ 8
      LAM(I) = CØNJG(ZR)
      INDIC(I) = 1
      I =I+1
C
    8 IF(NM.LT.I)GØ TØ 9
C     DETERMINATIØN ØF THE STARTING APPRØXIM
C     ATIØN TØ THE CØMPUTED EIGENVALUE LAM
C     (I+1).
      ZR = 0.9*ZR
      RE = ABS(RE)+ABS(IM)
      IF(RE.LE.EPS100.ØR.1./RE.LE.EPS100)GØ
     1TØ 2
      GØ TØ 3
C
    9 CØNTINUE
      RETURN
      END

      SUBRØUTINE F(Z,FUNC,EPS,I,IND,M,N,ND,T,
     1LAM,PERMUT,A,L,U)
      INTEGER I,IND,M,N,ND,T,PERMUT,IM1,J,K
      REAL EPS,FK
      CØMPLEX Z,FUNC,LAM,A,L,U,C
      DIMENSIØN LAM(1),PERMUT(1),A(ND,1),
     1L(ND,1),U(ND,1)
C
C     CØMPUTATIØN ØF FUNC = F(Z) = DET A(Z) =
C     DET A. MATRIX A IS DECØMPØSED INTØ THE
C     FØRM
C                             PA = LU
C     WHERE L IS UNIT LØWER TRIANGULAR MATRIX,
C     U IS UPPER TRIANGULAR MATRIX AND P IS
C     THE CØRRESPØNDING PERMUTATIØN MATRIX
C     (SEE WILKINSØN).
C
C     CALCULATIØN ØF A(Z).
      CALL ALAMDA (Z,M,N,ND,A)
C     CALCULATIØN ØF THE SMALL PØSITIVE
C     NUMBER EPS.
      C = (0.,0.)
```

```
      DØ 1 J=1,N
         DØ 1 K=1,N
            C = C + A(J,K)**2
    1 CØNTINUE
      EPS = CABS(CSQRT(C))*2.0**(-T)
C     CØMPUTATIØN ØF THE MATRICES L AND U,
C     WHERE PA = L*U.
      CALL LU(EPS, IND,N,ND,PERMUT,A,L,U)
      IF(IND.EQ.1)GØ TØ 4
C
      J = -IND
      K = (-1)**J
      FK = FLØAT(K)
      FUNC = CMPLX(FK,0.0)
      DØ 2 K=1,N
         FUNC = FUNC*U(K,K)
    2 CØNTINUE
C     MØDIFICATIØN ØF FUNC = F(Z) INTØ F(Z)/
C     ((Z-X(1))(Z-X(2))...(Z-X(I-1)) ACCØRDING
C     TØ EQUATIØN (23) ØF THIS PAPER.
      IM1 = I-1
      IF(IM1.EQ.0)GØ TØ 4
      DØ 3 J=1,IM1
         FUNC = FUNC/(Z-LAM(J))
    3 CØNTINUE
    4 CØNTINUE
      RETURN
      END

      SUBRØUTINE ALAMDA(Z,M,N,ND,A)
      INTEGER M,N,ND,I,J,K
      REAL A0,A1,A2,A3
      CØMPLEX Z,A,AIJ
      CØMPLEX Z2,Z3
      CØMMØN /MATRIX/ A0(ND,ND),A1(ND,ND),
     1A2(ND,ND),A3(ND,ND)
      DIMENSIØN A(ND,1)
C
C     THIS SUBRØUTINE CALCULATES THE ELEMENTS
C     ØF MATRIX A(Z), WHICH ARE FUNCTIØNS ØF Z,
C     FØR A GIVEN VALUE ØF Z.
      Z2 = Z**2
      Z3 = Z**3
      CØNTINUE
      DØ 44 I=1,N
         DØ 44 J=1,N
            AIJ = A0(I,J)
            DØ 33 K=1,M
               GØ TØ (1,2,3),K
    1          AIJ = AIJ+A1(I,J)*Z
               GØ TØ 33
    2          AIJ = AIJ+A2(I,J)*Z2
               GØ TØ 33
    3          AIJ = AIJ+A3(I,J)*Z3
   33       CØNTINUE
            A(I,J) = AIJ
```

```
   44 CONTINUE
      RETURN
      END

      SUBROUTINE LU (EPS,IND,N,ND,PERMUT,A,L,
     1U)
      INTEGER IND,N,ND,PERMUT,I,K,NP1,R,RC,
     1RM1,RP1,T
      REAL EPS,SRC,STABS
      COMPLEX A,L,U,ST,URR
      DIMENSION PERMUT(1),A(ND,1),L(ND,1),
     1U(ND,1)
C
C     THIS SUBROUTINE PERFORMS THE
C     DECOMPOSITION OF MATRIX (PA) INTO L*U,
C     WHERE L IS UNIT LOWER TRIANGULAR MATRIX,
C     U IS UPPER TRIANGULAR MATRIX AND P IS
C     THE CORRESPONDING PERMUTATION MATRIX.
C     PARTIAL PIVOTING IS INTRODUCED (SEE J.H.
C     WILKINSON, PAGE 225).
      IND = 0
      DO 1 I=1,N
         PERMUT(I) = I
    1 CONTINUE
C
      NP1 = N+1
      DO 11 R=1,N
         RM1 = R-1
         RP1 = R+1
         RC = R
         SRC = 0
         DO 4 T=R,N
            ST = A(T,R)
            IF(RM1.EQ.0)GO TO 3
C
            DO 2 K=1,RM1
               ST = ST-L(T,K)*U(K,R)
    2       CONTINUE
    3       A(T,NP1) = ST
            STABS = CABS(ST)
            IF(SRC.GE.STABS)GO TO 4
            SRC = STABS
            RC  = T
    4    CONTINUE
         IF(SRC.GT.EPS)GO TO 5
         IND = 1
         RETURN
C
    5    IF(R.EQ.RC)GO TO 7
         T = PERMUT(R)
         PERMUT(R) = PERMUT(RC)
         PERMUT(RC) = T
C     THE FOLLOWING STATEMENT(IND=IND-1) MAY
C     BE TAKEN OUT IN EIGENL.
         IND = IND-1
         DO 6 K=1,NP1
            ST = A(R,K)
            A(R,K) = A(RC,K)
            A(RC,K) = ST
    6    CONTINUE
    7    URR = A(R,NP1)
         U(R,R) = URR
         IF(R.EQ.N)GO TO 11
         DO 10 T=RP1,N
            L(T,R) = A(T,NP1)/URR
            ST = A(R,T)
            IF(RM1.EQ.0)GO TO 9
C
            DO 8 K=1,RM1
               ST = ST - L(R,K)*U(K,T)
    8       CONTINUE
    9       U(R,T) = ST
   10    CONTINUE
   11 CONTINUE
      RETURN
      END
```

3. Subroutine EIGENL - Laguerre's method

The subroutine is composed of ten subroutines EIGENL,ALAMDA (see 2.), LU(see 2.), DERIV1, DERIV2, LOWER, UPPER, UPPER2, TRACE and TRXSQ.

```
      SUBROUTINE EIGENL(M,N,ND,T,INDIC,LAM,
     1PERMUT,A,D,L,U)
      INTEGER M,N,ND,T,INDIC,PERMUT,I,IM1,IND,
     1ITERST,J,K,NM,NM1,R
      REAL EPS,EPS100,IM,RE
      REAL A0,A1,A2,A3
      COMPLEX LAM,A,D,L,U,CEPS,DENOM1,DENOM2,
     1S1,S2,TRACEX,TRACEW,TRACX2,ZR,ZRP1
      COMMON /MATRIX/ A0(ND,ND),A1(ND,ND)
     1A2(ND,ND),A3(ND,ND)
      DIMENSION INDIC(1),LAM(1),PERMUT(1),
     1A(ND,1),D(ND,1),L(ND,1),U(ND,1)
C
      NM = N*M
      ITERST = 101
      DO 1 I=1,NM
         INDIC(I)=0
    1 CONTINUE
      I=1
C     THE LAGUERRE ITERATIVE PROCESS.
C     DETERMINATION OF THE STARTING
C     APPROXIMATION TO THE FIRST COMPUTED
C     EIGENVALUE LAM(1).
      ZR = (1.,1.)
    2 NM1 = NM-I
C     COMPUTATION OF THE NEXT EIGENVALUE LAM(I).
      R = 1
C
C     THE MAIN LOOP FOR THE COMPUTATION OF THE
```

```
C       NEXT APPROXIMATION Z(R+1) TO LAM(I).
C       CALCULATION OF THE EQUATIONS (10) - (22)
C       OF THIS PAPER.
C       CALCULATION OF A(Z).
      3 CALL ALAMDA(ZR,M,N,ND,A)
C       CALCULATION OF THE SMALL POSITIVE NUMBER
C       EPS.
        CEPS = 0
        DO 4 J=1,N
           DO 4 K=1,N
              CEPS = CEPS+A(J,K)**2
      4 CONTINUE
        EPS = CABS(CSQRT(CEPS))*2.0**(-T)
        EPS100 = 100.*EPS
C       COMPUTATION OF THE MATRICES L AND U,
C       WHERE PA = L*U.
        CALL LU(EPS,IND,N,ND,PERMUT,A,L,U)
        IF(IND.EQ.1)GO TO 8
C       COMPUTATION OF THE TRACE OF MATRIX X
C       AND THE TRACE OF X**2.
        CALL DERIV1(ZR,M,N,ND,PERMUT,D)
        CALL LOWER(N,ND,D,L)
        CALL UPPER(N,ND,D,U)
        CALL TRACE(N,ND,D,TRACEX)
        CALL TRXSQ(N,ND,D,TRACX2)
C       COMPUTATION OF THE TRACE OF MATRIX W.
        TRACEW = 0.0
        IF(M.LT.2)GO TO 5
        CALL DERIV2(ZR,M,N,ND,PERMUT,D)
        CALL LOWER(N,ND,D,L)
        CALL UPPER2 (N,ND,D,U)
        CALL TRACE(N,ND,D,TRACEW)
C
      5 S1 = TRACEX
        S2 = TRACX2-TRACEW
C       MODIFICATIONS OF S1 AND S2 ACCORDING
C       TO THE EQUATIONS (24) AND (25) OF THIS
C       PAPER.
        IM1 = I-1
        IF(IM1.EQ.0)GO TO 7
        DO 6 J=1,IM1
           CEPS = 1.0/(ZR-LAM(J))
           S1 = S1-CEPS
           S2 = S2-CEPS**2
      6 CONTINUE
C
      7 CEPS = CSQRT(NM1*((NM1+1)*S2-S1**2))
        DENOM1 = S1+CEPS
        DENOM2 = S1-CEPS
        IF(CABS(DENOM1).GT.CABS(DENOM2))DENOM2=
       1DENOM1
        ZRP1 = ZR-(NM1+1)/DENOM2
C
        IF(CABS(ZRP1-ZR).LE.10.*EPS)GO TO 9
        ZR = ZRP1
        R = R+1
        IF(R.LT.ITERST)GO TO 3
        LAM(I) = ZR
        GO TO 11
C
      8 ZRP1 = ZR
      9 LAM(I) = ZRP1
        INDIC(I) = 1
C       DETERMINATION OF THE STARTING
C       APPROXIMATION TO THE NEXT COMPUTED
C       EIGENVALUE LAM(I+1) AND COMPUTATION OF
C       THE COMPLEX CONJUGATE VALUE OF LAM(I).
        ZR = 0.9*ZRP1
        I = I+1
        RE = REAL(ZRP1)
        IM =AIMAG(ZRP1)
        IF(ABS(IM).LE.1.0E-3*ABS(RE))ZR=0.5*
       1CMPLX(RE,RE)
        RE = ABS(RE)+ABS(IM)
        IF(RE.LE.EPS100.OR.1./RE.LE.EPS100)
       1ZR=(1.,1.)*FLOAT(I)
        IF(ABS(IM).LE.EPS100)GO TO 10
        LAM(I) = CONJG(ZRP1)
        INDIC(I) = 1
        I = I+1
C
     10 IF(I.LE.NM)GO TO 2
     11 CONTINUE
        RETURN
        END

        SUBROUTINE DERIV1(Z,M,N,ND,PERMUT,A)
        INTEGER M,N,ND,PERMUT,I,II,J,K,MM1
        REAL A0,A1,A2,A3
        COMPLEX Z,A,AIJ
        COMPLEX Z2,Z3
        COMMON /MATRIX/ A0(ND,ND),A1(ND,ND)
       1A2(ND,ND),A3(ND,ND)
        DIMENSION A(ND,1),PERMUT(1)
C
C       THIS SUBROUTINE CALCULATES THE ELEMENTS
C       OF THE FIRST DERIVATIVE OF MATRIX A(Z),
C       FOR A GIVEN VALUE OF Z.
        Z2 = 2.*Z
        Z3 = 3.*Z**2
        CONTINUE
        MM1 = M-1
        DO 55 II=1,N
           I = PERMUT(II)
           DO 55 J=1,N
              AIJ = A1(I,J)
              IF(M.LT.2)GO TO 44
              DO 33 K=1,MM1
                 GO TO (2,3),K
      2          AIJ = AIJ+A2(I,J)*Z2
                 GO TO 33
      3          AIJ = AIJ+A3(I,J)*Z3
```

```
33        CØNTINUE
44        A(II,J)=AIJ
55 CØNTINUE
   RETURN
   END

   SUBRØUTINE DERIV2(Z,M,N,ND,PERMUT,A)
   INTEGER M,N,ND,PERMUT,I,II,J,K,MM2
   REAL A0,A1,A2,A3
   CØMPLEX Z,A,AIJ
   CØMPLEX Z2,Z3
   CØMMØN /MATRIX/ A0(ND,ND),A1(ND,ND),
  1A2(ND,ND),A3(ND,ND)
   DIMENSIØN A(ND,1),PERMUT(1)
C
C     THIS SUBRØUTINE CALCULATES THE ELEMENTS
C     ØF THE SECØND DERIVATIVE ØF MATRIX A(Z),
C     FØR A GIVEN VALUE ØF Z.
   Z3 = 6.*Z
   CØNTINUE
   MM2 = M-2
   DØ 55 II=1,N
      I = PERMUT(II)
      DØ 55 J=1,N
         AIJ = 2.*A2(I,J)
         IF(M.LT.3)GØ TØ 44
         DØ 33 K=1,MM2
            GØ TØ (3,3),K
 3          AIJ = AIJ+A3(I,J)*Z3
33       CØNTINUE
44       A(II,J) = AIJ
55 CØNTINUE
   RETURN
   END

   SUBRØUTINE LØWER(N,ND,A,L)
   INTEGER N,ND,I,IM1,J,K
   CØMPLEX A,L,AIJ
   DIMENSIØN A(ND,1),L(ND,1)
C
C     THIS SUBRØUTINE CALCULATES THE ELEMENTS
C     ØF MATRIX M WHICH SATISFIES THE RELATIØN
C     L*M=A, WHERE L IS UNIT LØWER TRIANGULAR
C     MATRIX. M IS STØRED INTØ ARRAY A.
   DØ 2 I=2,N
      IM1 = I-1
      DØ 2 J=1,N
         AIJ = (0.,0.)
         DØ 1 K=1,IM1
            AIJ = AIJ+L(I,K)*A(K,J)
 1       CØNTINUE
         A(I,J)=A(I,J)-AIJ
 2 CØNTINUE
   RETURN
   END

   SUBRØUTINE UPPER(N,ND,A,U)
   INTEGER N,ND,I,II,IP1,J,K
   CØMPLEX A,U,AIJ
   DIMENSIØN A(ND,1),U(ND,1)
C
C     THIS SUBRØUTINE CALCULATES THE ELEMENTS
C     ØF MATRIX M WHICH SATISFIES THE RELATIØN
C     U*M=A, WHERE U IS UPPER TRIANGULAR
C     MATRIX. M IS STØRED INTØ ARRAY A.
   DØ 3 II=1,N
      I = N+1-II
      IP1 = I+1
      DØ 3 J=1,N
         AIJ = (0.,0.)
         IF(I.EQ.N)GØ TØ 2
         DØ 1 K=IP1,N
            AIJ = AIJ+U(I,K)*A(K,J)
 1       CØNTINUE
 2       A(I,J) = (A(I,J)-AIJ)/U(I,I)
 3 CØNTINUE
   RETURN
   END

   SUBRØUTINE TRACE(N,ND,A,TR)
   INTEGER N,ND,I
   CØMPLEX A,TR
   DIMENSIØN A(ND,1)
C
C     THIS SUBRØUTINE CALCULATES THE TRACE
C     ØF MATRIX A.
   TR = (0.,0.)
   DØ 1 I=1,N
      TR = TR+A(I,I)
 1 CØNTINUE
   RETURN
   END

   SUBRØUTINE TRXSQ(N,ND,A,TR)
   INTEGER N,ND,I,K
   CØMPLEX A,TR
   DIMENSIØN A(ND,1)
C
C     THIS SUBRØUTINE CALCULATES THE TRACE ØF
C     A**2.
   TR = (0.,0.)
   DØ 1 I=1,N
      DØ 1 K=1,N
         TR = TR+A(I,K)*A(K,I)
 1 CØNTINUE
   RETURN
   END

   SUBRØUTINE UPPER2 (N,ND,A,U)
   INTEGER N,ND,I,II,IP1,J,K
   CØMPLEX A,U,AIJ
   DIMENSIØN A(ND,1),U(ND,1)
```

```
C
C     THIS SUBRØUTINE CALCULATES THE
C     ELEMENTS OF THE LØWER TRIANGULAR ØF
C     MATRIX M. M SATISFIES THE RELATIØN U*M=A,
C     WHERE U IS UPPER TRIANGULAR MATRIX. M
C     IS STØRED INTØ ARRAY A.
      DØ 3 II=1,N
         I = N+1-II
         IP1 = I+1
         DØ 3 J=1,I
            AIJ = (0.,0.)
            IF(I.EQ.N)GØ TØ 2
            DØ 1 K=IP1,N
               AIJ = AIJ+U(I,K)*A(K,J)
    1       CØNTINUE
    2       A(I,J)=(A(I,J)-AIJ)/U(I,I)
    3 CØNTINUE
      RETURN
      END
```

4. Modifications

For matrix A(z) satisfying (2), where m>3, some of the statements must be replaced with other statements as shown below for the case m≤5:

1) Statements

```
      REAL A0,A1,A2,A3
      CØMMØN /MATRIX/ A0(ND,ND),A1(ND,ND),
     1A2(ND,ND),A3(ND,ND)
```

in the main programme, EIGENM, ALAMDA, EIGENL, DERIV1 and DERIV2 by

```
      REAL A0,A1,A2,A3,A4,A5
      CØMMØN /MATRIX/ A0(ND,ND),A1(ND,ND),
     1A2(ND,ND),A3(ND,ND),A4(ND,ND),A5(ND,ND)
```

2) Statement CØMPLEX Z2,Z3 in ALAMDA,DERIV1 and DERIV2 by

```
      CØMPLEX Z2,Z3,Z4,Z5
```

3) Statement CØNTINUE

(i) in ALAMDA by

```
      Z4 = Z**4
      Z5 = Z**5
```

(ii) in DERIV1 by

```
      Z4 = 4.*Z**3
      Z5 = 5.*Z**4
```

(iii) in DERIV2 by

```
      Z4 = 4.*3.*Z**2
      Z5 = 5.*4.*Z**3
```

4) Statement GØ TØ(1,2,3),K in ALAMDA by

```
      GØ TØ(1,2,3,4,5),K
```

5) Statement GØ TØ(2,3),K in DERIV1 by

```
      GØ TØ(2,3,4,5),K
```

6) Statement GØ TØ(3,3),K in DERIV2 by

```
      GØ TØ(3,4,5),K
```

7) Statement 33 CØNTINUE in ALAMDA,DERIV1 and DERIV2 by

```
      GØ TØ 33
    4 AIJ = AIJ+A4(I,J)*Z4
      GØ TØ 33
    5 AIJ = AIJ+A5(I,J)*Z5
   33 CØNTINUE
```

5. Main - calling programme

In the following lines an example of a possible calling programme for subroutines EIGENM and EIGENL is presented. It finds the eigenvalues of A(z) in (2) with m≤3 and n≤12. The programme stops after reading value zero for n.

```
      INTEGER INDIC,PERMUT,M,N,T,I,J,K,MN,MP1
      REAL A0,A1,A2,A3
      CØMPLEX LAM,A,L,U
      CØMMØN /MATRIX/ A0(12,12),A1(12,12),
     1A2(12,12),A3(12,12)
      EQUIVALENCE (A(1),L(1),U(1))
      DIMENSIØN INDIC(36),LAM(36),PERMUT(12),
     1A(12,13),L(12,13),U(12,13)
      CØMPLEX D
      DIMENSIØN D(12,12)
C
    1 READ 2,N,M,T
    2 FØRMAT(3I5)
    3 FØRMAT(8F10.0)
      IF(N.EQ.0)GØ TØ 12
C     THE PRØGRAMME STØPS AFTER READING ZERØ
C     FØR N.
      MN = M*N
      MP1 = M+1
      DØ 8 I=1,MP1
         GØ TØ (4,5,6,7),I
    4    READ 3,((A0(K,J),J=1,N),K=1,N)
         GØ TØ 8
    5    READ 3,((A1(K,J),J=1,N),K=1,N)
         GØ TØ 8
    6    READ 3,((A2(K,J),J=1,N),K=1,N)
         GØ TØ 8
    7    READ 3,((A3(K,J),J=1,N),K=1,N)
    8 CØNTINUE
```

```
C
      PRINT 9,((A0(K,J),J=1,N),K=1,N)
    9 FØRMAT (1H ,6E16.7)
      PRINT 9,((A1(K,J),J=1,N),K=1,N)
      IF(M.GT.1)PRINT 9,((A2(K,J),J=1,N),
     1K=1,N)
      IF(M.GT.2)PRINT 9,((A3(K,J),J=1,N),
     1K=1,N)
C
      CALL EIGENM(M,N,12,T,INDIC,LAM,PERMUT,
     1A,L,U)
C
      PRINT 10,(INDIC(I),LAM(I),I=1,MN)
   10 FØRMAT(1H0,10X,6H INDIC,10X,9HREAL PART,
     112X,10H IMAG.PART,///(1H ,12X,I2,5X,
     1E19.12,4X,E19.12))
      DØ 11 I=1,MN
         LAM(I) = 0.0
   11 CØNTINUE
C
      CALL EIGENL(M,N,12,T,INDIC,LAM,PERMUT,
     1A,D,L,U)
      PRINT10,(INDIC(I),LAM(I),I=1,MN)
      GØ TØ 1
C
   12 CØNTINUE
      END
```


CENIK OGLASOV

Ovitek - notranja stran (za letnik 1982)

2 stran	28.000 din
3 stran	21.000 din

Vmesne strani (za letnik 1982)

1/1 stran	13.000 din
1/2 strani	9.000 din

Vmesne strani za posamezno številko

1/1 stran	5.000 din
1/2 strani	3.300 din

Oglasi o potrebah po kadrih (za posamezno številko)

2.000 din

Razen oglasov v klasični obliki so zaželjene tudi krajše poslovne, strokovne in propagandne informacije in članki. Cene objave tovrstnega materiala se bodo določale sporazumno.

ADVERTIZING RATES

Cover page (for all issues of 1982)

2nd page	1300 $
3rd page	1000 $

Inside pages (for all issues of 1982)

1/1 page	790 $
1/2 page	520 $

Inside pages (individual issues)

1/1 page	260 $
1/2 page	200 $

Rates for classified advertizing:

each ad	66 $

In addition to advertisment, we welcome short business or product news, notes and articles. The related charges are negotiable.