

PŘÍLOHY

Příloha 1: Jednotka vyvíjeného deflačního algoritmu

```

{=====}
{=}          unit DEFLACE;          {=}
{=====}

{ Jednotka pro deflaci komplexni matice matR+jmatI
  -----}
{=====}

Interface
uses GLOBAL;

procedure DEFLA(matR:mat;matI:mat;rad_mat:byte;var nakt:byte;
               var kve:pomQ;var deter:vek);
procedure VynechRadekC(A,B:mat;n,k:byte;var C,D:mat);
procedure VynechSloupecC(A,B:mat;n,k:byte;var C,D:mat);

Implementation
{=====}

procedure MATZERO(var A:mat);    {Vynuluje matici A}

begin fillchar(A,sizeof(A),#0); end;
{-----}
procedure MATIDN(var A:mat;np:integer); {Vytvari jednotkovou matici A}

var i:integer;
begin
  MATZERO(A);
  for i:=1 to np do a[i,i]:=1;
end;
{-----}
procedure VEKTORZERO(var b:vek); {Tato procedura vynuluje vektor b}

begin fillchar(b,sizeof(b),#0); end;
{-----}
procedure MATICEMATICEMULTI(a,b:mat;var c:mat;np:integer);
{procedura nasobeni matice a matici b; vysledkem je matice c}

var i,j,k:integer;
begin
  MATZERO(C);
  for i:=1 to np do
    begin
      for j:=1 to np do
        begin
          for k:=1 to np do c[i,j]:=c[i,j]+a[i,k]*b[k,j];
        end;
      end;
    end;
end;
end;

```

```

{-----}
procedure STOPA(a:mat;var stopa:extended;np:byte);
  {procedura pro vypocet stopy matice a}

  var i: byte;
  begin
    stopa:=0;
    for i:=1 to np do stopa:=stopa+a[i,i];
  end;
{-----}
procedure Faddejev(a:mat;n:byte;var deter:vek);
var i,j:byte;
    en:mat;
begin

  VEKTORZERO(deter);
  deter[n]:=1;MATIDN(en,n);
  for i:=1 to (n-1) do
    begin
      MATICEMATICEMULTI(a,en,en,n);
      STOPA(en,deter[n-i],n); deter[n-i]:=-deter[n-i]/i;
      for j:=1 to n do en[j,j]:=en[j,j]+deter[n-i];
    end;
  MATICEMATICEMULTI(a,en,en,n);
  STOPA(en,deter[0],n); deter[0]:=-deter[0]/n;
end;

{-----}

procedure VynechRadekC(A,B:mat;n,k:byte;var C,D:mat);

var i:byte;
begin
  C:=A;D:=B;
  if (k<n) then
    begin
      for i:=k to (n-1) do begin C[i]:=A[i+1];D[i]:=B[i+1];end;
    end;
end;
{.....}
procedure VynechSloupecC(A,B:mat;n,k:byte;var C,D:mat);

var i,j:byte;
begin
  C:=A;D:=B;
  if (k<n) then
    begin
      for i:=k to (n-1) do
        begin
          for j:=0 to n do begin C[j,i]:=A[j,i+1];D[j,i]:=B[j,i+1];end;
        end;
    end;
end;
{.....}

```

```
Procedure Defl(var matR,matI:mat;rad_mat:byte;var nakt:byte;var kve:pomQ);
```

```
label POZOR,A,B,C,D,E,F,G,H,AA,KONEC;
```

```
var pom1,pom2:vek;  
    priznakZero:boolean;  
    i,j,k,l:byte;
```

```
{.....}
```

```
procedure ZamenaRadkuC(A,B:mat;n,k,l:byte;var C,D:mat);
```

```
begin
```

```
    C:=A;D:=B;  
    C[k]:=A[l];C[l]:=A[k];  
    D[k]:=B[l];D[l]:=B[k];
```

```
end;
```

```
{.....}
```

```
procedure ZamenaSloupcuC(A,B:mat;n,k,l:byte;var C,D:mat);
```

```
var i:byte;
```

```
begin
```

```
    C:=A;D:=B;  
    for i:=1 to n do  
        begin  
            C[i,k]:=A[i,l];D[i,k]:=B[i,l];C[i,l]:=A[i,k];D[i,l]:=B[i,k];  
        end;
```

```
end;
```

```
{.....}
```

```
procedure ROZVOJ_DET_S;
```

```
var pom:extended;
```

```
    k,l:byte;
```

```
begin
```

```
    if (i>1) then
```

```
        begin
```

```
            for l:=1 to i-1 do
```

```
                begin
```

```
                    pom:=matR[l,i]/matR[i,i];
```

```
                    if (pom<>0) then
```

```
                        begin
```

```
                            for k:=1 to nakt do
```

```
                                begin
```

```
                                    matR[l,k]:=matR[l,k]-pom*matR[i,k];
```

```
                                end;
```

```
                        end;
```

```
                    end;
```

```
                end;
```

```
            if (i<nakt) then
```

```
                begin
```

```
                    for l:=i+1 to nakt do
```

```
                        begin
```

```
                            pom:=matR[l,i]/matR[i,i];
```

```
                            if (pom<>0) then
```

```
                                begin
```

```

        for k:=1 to nakt do
            begin
                matR[l,k]:=matR[l,k]-pom*matR[i,k];
            end;
        end;
    end;
    kve.q:=kve.q*matR[i,i];
    VynechRadekC(matR,matI,nakt,i,matR,matI);
    VynechSloupecC(matR,matI,nakt,i,matR,matI);
    nakt:=nakt-1;
end;
{.....}
procedure ROZVOJ_DET_R;
var pom:extended;
    k,l:byte;
begin
    if (i>1) then
        begin
            for l:=1 to i-1 do
                begin
                    pom:=matR[i,l]/matR[i,i];
                    if (pom<>0) then
                        begin
                            for k:=1 to nakt do
                                begin
                                    matR[k,l]:=matR[k,l]-pom*matR[k,i];
                                end;
                            end;
                        end;
                    end;
                end;
            if (i<nakt) then
                begin
                    for l:=i+1 to nakt do
                        begin
                            pom:=matR[i,l]/matR[i,i];
                            if (pom<>0) then
                                begin
                                    for k:=1 to nakt do
                                        begin
                                            matR[k,l]:=matR[k,l]-pom*matR[k,i];
                                        end;
                                    end;
                                end;
                            end;
                        end;
                    kve.q:=kve.q*matR[i,i];
                    VynechRadekC(matR,matI,nakt,i,matR,matI);
                    VynechSloupecC(matR,matI,nakt,i,matR,matI);
                    nakt:=nakt-1;
                end;
            {.....}

            var pom:extended;
            begin

```

```

kve.q:=1;kve.p:=0;
nakt:=rad_mat;i:=1;

A: if (matI[i,i]<>0) then
{.....}
begin
  for j:= i+1 to nakt do
  begin
    pom1[j]:=matI[i,j]/matI[i,i];
    pom2[j]:=matI[j,i]/matI[i,i];
  end;

  for k:=i+1 to nakt do
  begin
    if (pom1[k]<>0) then
    begin
      if (i>1) then
      begin
        for j:=1 to i-1 do matR[j,k]:=matR[j,k]-pom1[k]*matR[j,i];
      end;
      for j:=i to nakt do
      begin
        matR[j,k]:=matR[j,k]-pom1[k]*matR[j,i];
        matI[j,k]:=matI[j,k]-pom1[k]*matI[j,i];
      end;
    end;
  end;

  for k:=i+1 to nakt do
  begin
    if (pom2[k]<>0) then
    begin
      if (i>1) then
      begin
        for j:=1 to i-1 do matR[k,j]:=matR[k,j]-pom2[k]*matR[i,j];
      end;
      for j:=i to nakt do
      begin
        matR[k,j]:=matR[k,j]-pom2[k]*matR[i,j];
        matI[k,j]:=matI[k,j]-pom2[k]*matI[i,j];
      end;
    end;
  end;

  inc(i);

POZOR: if (i<nakt) then goto A;
        if (((i=nakt) and (matI[i,i]<>0)) or (i>nakt) or (nakt=1))
        then goto KONEC;
end;
{.....}
  if (matR[i,i]=0) then priznakzero:=true else priznakzero:=false;
  j:=i;
B:   inc(j);if (j>nakt) then goto C;

```

```

    if (matI[i,j]=0) then goto B else
    begin
        ZameniSloupcuC(matR,matI,nakt,i,j,matR,matI);
        kve.q:=-kve.q;
        goto A;
    end;
C: j:=i;
D: inc(j);if (j>nakt) then goto E;
    if (matI[j,i]=0) then goto D else
    begin
        ZameniRadkuC(matR,matI,nakt,i,j,matR,matI);
        kve.q:=-kve.q;
        goto A;
    end;
E: if (priznakzero=false) then
    begin
        ROZVOJ_DET_R; goto POZOR;
    end;
    j:=i;
F: inc(j);if (j>nakt) then goto G;
    if (matR[i,j]=0) then goto F else
    begin
        ZameniSloupcuC(matR,matI,nakt,i,j,matR,matI);
        kve.q:=-kve.q;
        if(i<nakt) then
        begin
            for k:=i+1 to nakt do
            begin
                if(matI[k,i]<>0) then
                begin
                    ZameniRadkuC(matR,matI,nakt,i,k,matR,matI);kve.q:=-kve.q;
                    goto A;
                end;
            end;
        end;
        ROZVOJ_DET_R; goto POZOR;
    end;
G: j:=i;
H: inc(j);if (j>nakt) then goto AA;
    if (matR[j,i]=0) then goto H else
    begin
        ZameniRadkuC(matR,matI,nakt,i,j,matR,matI);
        kve.q:=-kve.q;
        if(i<nakt) then
        begin
            for k:=i+1 to nakt do
            begin
                if(matI[i,k]<>0) then
                begin
                    ZameniSloupcuC(matR,matI,nakt,i,k,matR,matI);kve.q:=-kve.q;
                    goto A;
                end;
            end;
        end;
    end;
end;

```

```

        ROZVOJ_DET_S; goto POZOR;
    end;
AA: matI[i]:=matR[i];
    fillchar(matR[i],sizeof(matR[i]),#0);
    kve.p:=kve.p-1;
    for j:=1 to i-1 do
        begin
            pom:=matI[i,j]/matI[j,j];
            if (pom<>0) then
                for k:=1 to nakt do matR[i,k]:=matR[i,k]-pom*matR[j,k];
            matI[i,j]:=0;
        end;
    if (matR[i,i]=0) then goto AA else
        begin
            ROZVOJ_DET_R;goto POZOR;
        end;
KONEC: end;
{.....}

procedure UPRAVA_MATICE(var matR:mat;matI:mat;var nakt:byte;var kve:pomQ);

{Procedura, která vrací v proměnné 'matR' redukovanou matici pro QR rozklad. Číslo 'kve.p' udává mocninu
nasobitele celého determinantu:  $p^{kve.p}$ . }

var i,j:byte;

begin
    for i:=1 to nakt do
        begin
            kve.q:=kve.q*matI[i,i];
            if (i>1) then
                begin
                    for j:=1 to i-1 do matR[i,j]:=-matR[i,j]/matI[i,i];
                end;
            matR[i,i]:=-matR[i,i]/matI[i,i];
            if (i<nakt) then
                begin
                    for j:=i+1 to nakt do matR[i,j]:=-matR[i,j]/matI[i,i];
                end;
        end;
    end;

end;
{.....}
procedure DEFLA(matR:mat;matI:mat;rad_mat:byte;var nakt:byte;
                var kve:pomQ;var deter:vek);
begin
    DEFL(matR,matI,rad_mat,nakt,kve);
    if (nakt>0) then
        begin
            UPRAVA_MATICE(matR,matI,nakt,kve);
            if (nakt=1) then begin deter[1]:=1;deter[0]:=matR[1,1];end else
                FADDEJEV(matR,nakt,deter);
        end
        else deter[0]:=1;
end

```

```
end;
```

```
{=====}
end.
```

Příloha 2: Jednotka použitého deflačního algoritmu a výpočtu vlastních čísel obvodové matice

```
{=====}
{=}          unit DEFEIGEN;          {=}
{=====}
```

```
{ Jednotka pro deflaci a vypocet vlastnich cisel matic
-----}
{=====}
```

```
Interface
uses GLOBAL;
```

```
procedure DEFL(N:byte;A:mat;var B:mat;var NB:byte;
               var DS,DL:extended;ABSTOL,RELTOL:double);
procedure EIGEN(N:byte;A:mat;var ER,EI:vek;var IND:byte);
```

```
Implementation
{=====}
```

```
    procedure DEFL(N:byte;A:mat;var B:mat;var NB:byte;
                  var DS,DL:extended;ABSTOL,RELTOL:double);
(*C  =====
C
C *** MATRIX DEFLATION TO ALGEBRAIC EIGENVALUE PROBLEM
C * 22/11/88
C
C * DECLARATIONS*)
var  AN,BN:vek;
     TOL,AH,BH,AAH,ABH,BIG,BIGH{,DABS,DMAX1,DLOG}:extended;
     I,J,K,IZ,JZ,KP1,NP1,KR,NR,JR,IR,IY,JY,NBH:integer;
label 220,260,300,360,400,420,440,500,600,610,620,650,660,670,680,700,740;
label 780,800,840,900,920,RET;
begin
(*C * DETERMINE NORMS OF COLUMNS*)
  for J:=1 to N do
  begin
    AN[J]:=0;
    BN[J]:=0;
    for I:=1 to N do
    begin
      if (abs(A[I,J])>AN[J]) then AN[J]:=ABS(A[I,J]);
      if (abs(B[I,J])>BN[J]) then BN[J]:=ABS(B[I,J]);
    end;
  end;
```

```

end;

(*C *** FORWARD ELIMINATION*)
  DS:=1;
  DL:=0;
  for K:=1 to N do
  begin

(* C * DETERMINE NORM OF A-MATRIX*)
  BIG:=0;
  for J:=K to N do
  begin
    for I:=K to N do
    begin
      IF(ABS(BIG)>ABS(A[I,J])) then GOTO 220;
      BIG:=A[I,J];
      IZ:=I;
      JZ:=J;
220:  end;
    end;

(* C * CHECK NORM*)
    TOL:=RELTOL*AN[JZ];
    IF(ABS(BIG)<=TOL) then GOTO 500;

(* C * INTERCHANGE ROWS AND COLUMNS*)
    IF(IZ=K) then GOTO 260;
    DS:=-DS;
    for J:=1 to N do
    begin
      AH:=A[IZ,J];
      A[IZ,J]:=A[K,J];
      A[K,J]:=AH;
      BH:=B[IZ,J];
      B[IZ,J]:=B[K,J];
      B[K,J]:=BH;
    end;
260: IF(JZ=K) then GOTO 300;
    DS:=-DS;
    for I:=1 to N do
    begin
      AH:=A[I,JZ];
      A[I,JZ]:=A[I,K];
      A[I,K]:=AH;
      BH:=B[I,JZ];
      B[I,JZ]:=B[I,K];
      B[I,K]:=BH;
    end;
    AH:=AN[JZ];
    AN[JZ]:=AN[K];
    AN[K]:=AH;
    BH:=BN[JZ];
    BN[JZ]:=BN[K];
    BN[K]:=BH;

```

```
(* C * ELIMINATION*)
300: IF(K=N) then GOTO 420;
    KP1:=K+1;
    for J:=KP1 to N do
    begin
        AH:=A[K,J]/BIG;
        AAH:=ABS(AH);
        IF (AAH<=ABSTOL) then GOTO 360;
        AN[J]:=AN[J]+AN[K]*AAH;
        BN[J]:=BN[J]+BN[K]*AAH;
        for I:=KP1 to N do A[I,J]:=A[I,J]-A[I,K]*AH;
        for I:=1 to N do B[I,J]:=B[I,J]-B[I,K]*AH;
360: end;
    for I:=KP1 to N do
    begin
        AH:=A[I,K]/BIG;
        IF(ABS(AH)<=ABSTOL) then GOTO 400;
        for J:=1 to N do B[I,J]:=B[I,J]-B[K,J]*AH;
400: end;
    end;

(*C * COMPUTE DETERMINANT AND MULTIPLY BY NEGATIVE INVERSE MATRIX*)
420:NB:=N;
440:for I:=1 to NB do
    begin
        IF(A[I,I]<0) then DS:=-DS;
        DL:=DL+LN(ABS(A[I,I]));
        for J:=1 to NB do B[I,J]:=-B[I,J]/A[I,I];
    end;
    GOTO RET;

(*C *** BACKWARD ELIMINATION*)
500:NB:=K-1;
    NP1:=N+1;
    K:=0;
600:K:=K+1;
610:KR:=NP1-K;
    NR:=N-NB;

(*C * DETERMINE NORM OF B-MATRIX*)
    BIG:=0;
    for J:=K to NR do
    begin
        JR:=NP1-J;
        for I:=K to NR do
        begin
            IR:=NP1-I;
            IF(ABS(BIG)>ABS(B[IR,JR])) then GOTO 620;
            BIG:=B[IR,JR];
            IZ:=IR;
            JZ:=JR;
620: end;
        end;
    end;
```

```
(*C * CHECK NORM*)
  IF(NB>0) then GOTO 650;
  TOL:=RELTOL*BN[JZ];
  GOTO 670;
650:BIGH:=0;
  for I:=1 to NB do
  begin
    BH:=B[I,JZ]/A[I,I];
    IF(ABS(BIGH)>ABS(BH)) then GOTO 660;
    BIGH:=BH;
    IY:=I;
660:end;
  TOL:=RELTOL*AN[IY]*ABS(BIGH);
670:IF(ABS(BIG)>TOL) then GOTO 680;
  IF(NB=0) then GOTO 920;
  BIG:=BIGH;
  TOL:=RELTOL*BN[JZ];
  IF(ABS(BIG*A[IY,IY])<=TOL) then GOTO 920;
  JY:=JZ;
  IZ:=IY;
  JZ:=IY;
  KR:=NB;

(*C * INTERCHANGE ROWS AND COLUMMS*)
  A[IY,IY]:=A[NB,NB];
  AN[IY]:=AN[NB];
680:IF(IZ=KR) then GOTO 700;
  DS:=-DS;
  for J:=K to N do
  begin
    JR:=NP1-J;
    BH:=B[IZ,JR];
    B[IZ,JR]:=B[KR,JR];
    B[KR,JR]:=BH;
  end;
700:IF(JZ=KR) then GOTO 740;
  DS:=-DS;
  for I:=K to N do
  begin
    IR:=NP1-I;
    BH:=B[IR,JZ];
    B[IR,JZ]:=B[IR,KR];
    B[IR,KR]:=BH;
  end;
  BH:=BN[JZ];
  BN[JZ]:=BN[KR];
  BN[KR]:=BH;

(*C * REDUCTION IF NECESSARY*)
740:IF(KR<>NB) then GOTO 800;
  NBH:=NB;
  NB:=NB-1;
  IF(NB=0) then GOTO 610;
```

```

KR:=NP1-K;
for J:=1 to NB do
begin
  BH:=B[J,JY]/A[J,J]/BIG;
  ABH:=ABS(BH);
  IF(ABH<=ABSTOL) then GOTO 780;
  BN[NBH]:=BN[NBH]+BN[J]*ABH;
  for I:=1 to KR do B[I,NBH]:=B[I,NBH]+B[I,J]*BH;
780:end;
GOTO 610;

```

```

(*C * ELIMINATION*)
800:IF(BIG<0) then DS:=-DS;
  DL:=DL+LN(ABS(BIG));
  IF(K=N) then GOTO 900;
  KP1:=K+1;
  for J:=KP1 to N do
  begin
    JR:=NP1-J;
    BH:=B[KR,JR]/BIG;
    ABH:=ABS(BH);
    IF(ABH<=ABSTOL) then GOTO 840;
    BN[JR]:=BN[JR]+BN[KR]*ABH;
    for I:=KP1 to N do
    begin
      IR:=NP1-I;
      B[IR,JR]:=B[IR,JR]-B[IR,KR]*BH;
    end;
840:end;
  IF(K<>NR) then GOTO 600;
900:IF(NB<>0) then GOTO 440;
  GOTO RET;

```

```

(*C * ERROR EXIT*)
920:DS:=0;
RET: END;
(*-----*)

```

procedure EIGEN(N:byte;A:mat;var ER,EI:vek;var IND:byte);

(* =====

C

C *** FIND EIGENVALUES OF REAL MATRIX

C * 22/11/88

C

C * DECLARATIONS*)

var NOCONV,NOLAST:boolean;

MM,L,LL,IT,NAM1,NA,NB,MM1,MP1,MP2,I,J,K,M,JJ,LOW,UPP,EXC:integer;

B,B2,C,F,G,P,Q,R,S,T,W,X,Y,Z,ZERO:extended;

label 100,150,170,180,190,210,220,230,240,250,260,270,300,310,320,370,380;

label 390,400,410,430,450,470,490,500,510,520,540,560,570,580,590,RET;

begin

```

exc:=1;
(**** REDUCE NORM OF MATRIX BY SIMILARITY TRANSFORMATION*)
  B:=2;
  B2:=B*B;
  LOW:=1;
  UPP:=N;
  GOTO 150;

(* * INTERCHANGE ROWS AND COLUMNS*)
100: for I:=1 to UPP do
  begin
    F:=A[I,J];
    A[I,J]:=A[I,M];
    A[I,M]:=F;
  end;
  for I:=LOW to N do
  begin
    F:=A[J,I];
    A[J,I]:=A[M,I];
    A[M,I]:=F;
  end;
  if (EXC=2) then GOTO 180;

(*C * SEARCH FOR ROWS ISOLATING EIGENVALUE AND PUSH THEM DOWN*)
  IF(UPP=1) then GOTO 310;
  dec(UPP);
150: for JJ:=1 to UPP do
  begin
    J:=UPP+1-JJ;
    for I:=1 to UPP do IF((I<>J) AND (A[J,I]<>0)) then GOTO 170;
    M:=UPP;
    EXC:=1;
    GOTO 100;
170: end;
  GOTO 190;

(*C * SEARCH FOR COLUMNS ISOLATING EIGENVALUE AND PUSH THEM LEFT*)
180: INC(LOW);
190: for J:=LOW to UPP do
  begin
    for I:=LOW to UPP do IF((I<>J) AND (A[I,J]<>0)) then GOTO 210;
    M:=LOW;
    EXC:=2;
    GOTO 100;
210: end;

(*C * BALANCE SUBMATRIX*)
220: NOCONV:=FALSE;
  for I:=LOW to UPP do
  begin
    R:=0;
    C:=0;
    for J:=LOW to UPP do
    begin

```

```

        IF(J=I) then GOTO 230;
        R:=R+ABS(A[I,J]);
        C:=C+ABS(A[J,I]);
230:  end;
        G:=R/B;
        F:=1;
        S:=C+R;
240:  IF(C>=G) then GOTO 250;
        F:=F*B;
        C:=C*B2;
        GOTO 240;
250:  G:=R*B;
260:  IF(C<G) then GOTO 270;
        F:=F/B;
        C:=C/B2;
        GOTO 260;
270:  IF(((R+C)/F)>=0.95*S) then GOTO 300;
        G:=1/F;
        for J:=LOW to N do A[I,J]:=A[I,J]*G;
        for J:=1 to UPP do A[J,I]:=A[J,I]*F;
        NOCONV:=TRUE;
300:  end;
        IF(NOCONV) then GOTO 220;

(*C *** REDUCE MATRIX TO UPPER HESSENBERG FORM*)
310:  for M:=LOW to UPP do
        begin
            IF((M=LOW) OR (M=UPP)) then GOTO 370;
            MM1:=M-1;
            MP1:=M+1;

(* C * DETERMINE MAXIMA IN COLUMNS*)
            X:=0;
            for J:=M to UPP do
                begin
                    IF(ABS(A[J,MM1])<ABS(X)) then GOTO 320;
                    X:=A[J,MM1];
                    I:=J;
320:  end;

(* C * INTERCHANGE ROWS AND COLUMNS*)
            for J:=MM1 to N do
                begin
                    Y:=A[I,J];
                    A[I,J]:=A[M,J];
                    A[M,J]:=Y;
                end;
            for J:=1 to UPP do
                begin
                    Y:=A[J,I];
                    A[J,I]:=A[J,M];
                    A[J,M]:=Y;
                end;

```

```

(* C * REDUCE SUBMATRIX*)
  IF(X=0) then GOTO 370;
  for I:=MP1 to UPP do
  begin
    A[I,MM1]:=A[I,MM1]/X;
    for J:=M to N do A[I,J]:=A[I,J]-A[I,MM1]*A[M,J];
    for J:=1 to UPP do A[J,M]:=A[J,M]+A[I,MM1]*A[J,I];
  end;
370: end;

(*C *** FIND EIGENVALUES OF UPPER HESSENBERG MATRIX*)
  IND:=0;
  T:=0;
  ZERO:=1;
380: ZERO:=ZERO/2;
  IF((ZERO/2+1)>1) then GOTO 380;

(*C * SAVE INSULATED EIGENVALUES*)
  for I:=1 to N do
  begin
    IF((I>=LOW) AND (I<=UPP)) then GOTO 390;
    inc(IND);
    ER[I]:=A[I,I];
    EI[I]:=0;
390: end;
  NB:=UPP;

(*C * FIND NEXT EIGENVALUES*)
400: IF(NB<LOW) then goto RET;
  NA:=NB-1;
  NAM1:=NA-1;
  IT:=0;

(*C * SEARCH FOR SINGLE SMALL SUBDIAGONAL ELEMENT*)
410: for LL:=LOW to NB do
  begin
    L:=NB+LOW-LL;
    IF(L=LOW) then GOTO 430;
    IF(ABS(A[L,L-1])<=ZERO*(ABS(A[L-1,L-1])+ABS(A[L,L]))) then GOTO 430;
  end;

(*C * CHECK LOCATION*)
430: X:=A[NB,NB];
  IF(L=NB) then GOTO 570;
  Y:=A[NA,NA];
  W:=A[NA,NB]*A[NB,NA];
  IF(L=NA) then GOTO 580;

(*C * CHECK COUNTER*)
  IF(IT=30) then goto RET;
  IF((IT<>10) AND (IT<>20)) then GOTO 450;

(*C * FORM EXCEPTIONAL SHIFT*)
  T:=T+X;

```

```

for I:=LOW to NB do A[I,I]:=A[I,I]-X;
S:=ABS(A[NA,NAM1])+ABS(A[NB,NA]);
X:=0.75*S;
Y:=X;
W:=-0.4375*S*S;

```

(*C * SEARCH FOR TWO CONSECUTIVE SMALL SUBDIAGONAL ELEMENTS*)

```

450: for MM:=L to NAM1 do
  begin
    M:=NAM1+L-MM;
    Z:=A[M,M];
    R:=X-Z;
    S:=Y-Z;
    P:=A[M,M+1]+(R*S-W)/A[M+1,M];
    Q:=A[M+1,M+1]-Z-R-S;
    R:=A[M+2,M+1];
    S:=ABS(P)+ABS(Q)+ABS(R);
    P:=P/S;
    Q:=Q/S;
    R:=R/S;
    IF(M=L) then GOTO 470;
    IF(ABS(A[M,M-1])*(ABS(Q)+ABS(R))<=ZERO*
      (ABS(A[M-1,M-1])+ABS(A[M+1,M+1])+ABS(Z))*ABS(P)) then GOTO 470;
  end;

```

(*C * RESET SUBDIAGONAL ELEMENTS*)

```

470: MP2:=M+2;
  for I:=MP2 to NB do
    begin
      A[I,I-2]:=0;
      IF(I<>MP2) then A[I,I-3]:=0;
    end;

```

(*C * DOUBLE QR STEP*)

```

  for K:=M to NA do
    begin
      NOLAST:=(K<>NA);
      IF(K=M) then GOTO 490;
      P:=A[K,K-1];
      Q:=A[K+1,K-1];
      R:=0;
      IF(NOLAST) then R:=A[K+2,K-1];
      X:=ABS(P)+ABS(Q)+ABS(R);
      IF(X=0) then GOTO 560;
      P:=P/X;
      Q:=Q/X;
      R:=R/X;
490:  S:=SQRT(P*P+Q*Q+R*R);
      IF(P<0) then S:=-S;
      IF(K=M) then GOTO 500;
      A[K,K-1]:=-S*X;
      GOTO 510;
500:  IF(M<>L) then A[K,K-1]:=-A[K,K-1];
510:  P:=P+S;

```

```

X:=P/S;
Y:=Q/S;
Z:=R/S;
Q:=Q/P;
R:=R/P;

```

```

(* C * ROW MODIFICATION*)
  for J:=K to NB do
  begin
    P:=A[K,J]+Q*A[K+1,J];
    IF(NOT NOLAST) then GOTO 520;
    P:=P+R*A[K+2,J];
    A[K+2,J]:=A[K+2,J]-P*Z;
520:   A[K+1,J]:=A[K+1,J]-P*Y;
    A[K,J]:=A[K,J]-P*X;
  end;

```

```

(* C * COLUMN MODIFICATION*)
  J:=K+3;if(J>NB) then J:=NB;
  for I:=L to J do
  begin
    P:=X*A[I,K]+Y*A[I,K+1];
    IF(NOT NOLAST) then GOTO 540;
    P:=P+Z*A[I,K+2];
    A[I,K+2]:=A[I,K+2]-P*R;
540:   A[I,K+1]:=A[I,K+1]-P*Q;
    A[I,K]:=A[I,K]-P;
  end;
560: end;
  inc(IT);
  GOTO 410;

```

```

(*C * ONE ROOT FOUND*)
570: inc(IND);
  ER[NB]:=X+T;
  EI[NB]:=0;
  NB:=NA;
  GOTO 400;

```

```

(*C * TWO ROOTS FOUND*)
580: IND:=IND+2;
  P:=(Y-X)/2;
  Q:=P*P+W;
  Z:=SQRT(ABS(Q));
  X:=X+T;
  IF(Q<=0) then GOTO 590;

```

```

(*C * REAL PAIR*)
  IF(P<0) then Z:=-Z;
  Z:=P+Z;
  ER[NA]:=X+Z;
  ER[NB]:=X-W/Z;
  EI[NA]:=0;
  EI[NB]:=0;

```

```
NB:=NAM1;
GOTO 400;
```

```
(*C * COMPLEX PAIR*)
```

```
590: ER[NA]:=X+P;
    ER[NB]:=ER[NA];
    EI[NA]:=Z;
    EI[NB]:=-Z;
    NB:=NAM1;
    GOTO 400;
```

```
RET:end;
end.
```

Příloha 3: Jednotka pro hledání kořenů polynomu spolu s testovacím příkladem

```
{*****}
{*}          unit POLYNOM;          {*}
{*****}
interface
```

```
uses GLOBAL,KOMPLEX;
```

```
procedure KORENY(a:vekc;m:byte;var roots:vekc;polish:boolean);
```

```
implementation
```

```
procedure LAGUER(a:vekc;m:integer;var x:complex;eps:real;polish:boolean);
{procedura na reseni korenu polynomu}
```

```
label KONEC;
const epss=1e-20;
    maxit=100;
var i,iter:integer;
    err,dxold,abx,cdx:extended;
    b,d,f,g,g2,h,dx,sq,gp,gm,cdum,x1:complex;
```

```
{.....}
```

```
begin
```

```
dxold:=CABS(x);
for iter:=1 to maxit do
begin
    b:=a[m+1];
    err:=CABS(b);
    d:=zeroc;f:=zeroc;
    abx:=CABS(x);
    for i:=m downto 1 do
begin
    NASOBC(x,f,f);SECTIC(f,d,f);
    NASOBC(d,x,d);SECTIC(d,b,d);
```

```

    NASOBC(b,x,b);SECTIC(b,a[i],b);
    err:=CABS(b)+abx*err;
end;
err:=epss*err;
if (CABS(b)<=err) then
begin
    dx:=zeroc; goto KONEC;
end
else
begin
    CDIV(d,b,g);
    NASOBC(g,g,g2);
    NASOBRC(-2,f,h);CDIV(h,b,h);SECTIC(h,g2,h);
    NASOBRC(m,h,h);ODECTIC(h,g2,h);NASOBRC(m-1,h,h);CSQRT(h,sq);
    SECTIC(g,sq,gp);
    ODECTIC(g,sq,gm);
    if (CABS(gp)<CABS(gm)) then gp:=gm;
    cdum.re:=m;cdum.im:=0;CDIV(cdum,gp,dx);
end;
ODECTIC(x,dx,x1);
if ((x.re=x1.re) and (x.im=x1.im)) then goto KONEC;
x:=x1;
cdx:=CABS(dx);
if ((iter>6) and (cdx>=dxold)) then goto KONEC;
dxold:=cdx;
if (not polish) then
    if (CABS(dx)<=eps*CABS(x)) then goto KONEC;
end;
writeln('pause in routine LAGUER - too many iterations');readln;
KONEC: end;
{-----}
procedure KORENY(a:vekc;m:byte;var roots:vekc;polish:boolean);
{vyhledá vsechny koreny polynomu pomoci procedury LAGUER}
label SKOK;
const eps=1e-20;
var i,j,jj:integer;
    dum:extended;
    b,c,x:complex;
    ad:vekc;
begin
    ad:=a;
    for j:=m downto 1 do
        begin
            x:=zeroc;
            LAGUER(ad,j,x,eps,false);
            if(abs(x.im)<=(2*eps*abs(x.re))) then x.im:=0;
            roots[j]:=x;
            b:=ad[j+1];
            for jj:=j downto 1 do
                begin
                    c:=ad[jj];
                    ad[jj]:=b;
                    NASOBC(x,b,b);SECTIC(b,c,b);
                end;
            end;
        end;
    end;

```

```

    end;
  if (polish) then
    begin
      for j:=1 to m do LAGUER(a,m,roots[j],eps,true);
    end;
  for j:=2 to m do
    begin
      x:=roots[j];
      for i:=j-1 downto 1 do
        begin
          if (roots[i].re<=x.re) then goto SKOK;
          roots[i+1]:=roots[i];
        end;
      i:=0;
    SKOK: roots[i+1]:=x;
    end;
  end;
  {-----}

end.
{-----}

```

Program TEST;

uses

CRT, POLYNOM, GLOBAL;

var

rad, i : byte;

b, x : vekc;

upresni : boolean;

begin

ClrScr;

{... sestaveni zkusebniho polynomu = zadani obsahu parametru procedury ...}

{... zpusob zadavani:

polynom a[1]+a[2]x+a[3]x^2+ .. +a[n+1]x^n

n.....rad

a[i]..koeficienty polynomu, mohou byt obecne komplexni

v nasem prikladu

a[i]=b[i]; b je komplexni vektor;jsou-li koeficienty jen realne, pak se zada a[i]=b[i].re, b[i].im=0

}

rad:=4;

fillchar(b,sizeof(b),#0);

b[1].re:=1;b[2].re:=4;b[3].re:=6;b[4].re:=4;b[5].re:=1;

{nyni je zadan polynom $x^4+4x^3+6x^2+4x+1$, ktery ma ctynasobny koren $x=-1$

}

upresni := true;

{parametrem upresni=true se docili iterativniho zpresneni korenu}

{nepotrebujeme-li zpresneni, pak upresni=false

}

```

{... volani procedury ...}
KORENY(b,rad,x,upresni);
{v komplexnim vektoru x jsou ulozeny vsechny koreny polynomu vzestupne
 podle velikosti x[1],x[2],...,x[n+1]
}

{... vypis vysledku ...}
for i:=1 to rad do writeln(x[i].re:25:20,x[i].im:25:20);
repeat until keypressed;
end.

```

Pozn.: Jednotka globálních typů a konstant GLOBAL:

```

{=====}
{=}          unit GLOBAL;          {=}
{=====}

{ Jednotka globalnich promennych a typu
  -----}
{=====}
Interface
const
  n=50;
type
  vek=array[0..n] of extended;
mat=array[0..n] of vek;
vekint=array[0..n] of integer;
matint=array[0..n] of vekint;

  {-----komplexni aritmetika-----}
  complex=record
    Re: extended;
    Im: extended
  end;
vekc=array[0..n] of complex;
matc=array[0..n] of vekc;
{-----konec komplexnich typu-----}
const
  imagc:complex=(re:0;im:1);
  zeroc:complex=(re:0;im:0);
  jednac:complex=(re:1;im:0);
  {-----konec komplexnich konstant-----}

Implementation
{=====}
end.

```