P. W. Hemker
P. M. de Zeeuw

A Library of Multigrid Routines
ALGOL 68

Centrum voor Wiskunde en Informatica
Numerieke Wiskunde
1982

```
MGLIB,CM140000,T140,IO340,NP.
ACCOUNT,[redacted]
A68,N,D.
EDITLIB.
RETURN,LGO.
A68,P=LIB/MGLIB,D.
EDITLIB.
RETURN,LGO.
CATALOG,LIB,MGLIB,ID=PHEMKER.
ATTACH,TST,MGTEST,ID=PHEMKER.
SKIPF,TST.
MAP,ON.
A68,I=TST,P=LIB/MGLIB,D.
LGO.
ITEMIZE,LIB,N.
```

```
#   ***************************************************************#
#                                                                  #
#                                                                  #
#                    A LIBRARY OF MULTIGRID ROUTINES               #
#                    PRELIMINARY VERSION                           #
#                                                                  #
#                                                                  #
#                                                                  #
#                                                                  #
#    THIS ALGOL 68 PRELUDE IS A COLLECTION OF PROGRAM MODULES, WHICH #
#    FORM A LIBRARY FOR NUMERICAL EXPERIMENTS WITH MULTIGRID METHODS.#
#    THE PRELUDE WAS DEVELOPED FROM 800918 ON AND IS CONTINUOUSLY    #
#    CHANGING. HOWEVER, AT THIS STAGE (820224), IT HAS TEMPORARILY   #
#    REACHED A STATIONARY STATE AND IS USED FOR TESTING DIFFERENT    #
#    MULTIGRID STRATEGIES. IN THE NEXT VERSIONS IT WILL BE EXTENDED  #
#    AND ADAPTED AT SEVERAL POINTS.                                  #
#                                                                    #
#    SINCE THE PRESENT VERSION WORKS ONLY FOR RECTANGULAR REGIONS    #
#    AND WITHOUT BACK-STORAGE FACILITIES, IT IS STILL SURVEYABLE AND #
#    WRITTEN IN A RATHER READABLE KIND OF ALGOL 68. THEREFORE IT IS  #
#    A CONCISE AND PRECISE DESCRIPTION OF MANY ESSENTIAL PARTS OF    #
#    THE ALGORITHMS.                                                 #
#    OTHER VERSIONS OF THE ROUTINES, FOR MORE GENERAL DOMAINS AND    #
#    WITH THE USE OF DISC STORAGE FACILITIES, ARE ALSO AVAILABLE     #
#    BUT NOT INCLUDED IN THIS LIBRARY.                               #
#    WE KNOW THAT, AT A NUMBER OF SPOTS, CERTAINLY NOT THE OPTIMAL   #
#    ALGORITHM HAS BEEN USED AND THE CODE CAN BE IMPROVED.           #
#    FURTHER WE BELIEVE THAT ANY PIECE OF SOFTWARE OF SOME LENGTH    #
#    CONTAINS AT LEAST ONE BUG.                                      #
#    WITH THIS RESERVE WE GIVE HERE THE PRELIMINARY VERSION OF OUR   #
#    MULTIGRID  LIBRARY.                                             #
#                                                                    #
#                                                                    #
#                                    P.W.HEMKER                      #
#                                    P.M.DE ZEEUW                    #
#                                                                    #
#                                                                    #
#                                                                    #
#                                                                    #
#   ***************************************************************#
                                                        'PR' EJECT 'PR'
#   ***************************************************************#
#                                                                    #
#                                                                    #
#    CONTENTS:                                                       #
#                                                                    #
# MODE DECLARATIONS                                        3  #
# ZERO  AND COPY  OPERATIONS                               4  #
# ELEMENTARY  NET OPERATIONS                               5  #
```



```
MGLIB:      # A MODULAR 2-D MULTIGRID ALGOL 68 PROGRAM #
            # FOR EXPERIMENTAL PURPOSES.  PWH/800918   #
            # VSN 830421                  PF: MGTEXT   #
'BEGIN'

# MODE AND PRIORITY DECLARATIONS #


   'MODE' 'VEC'    = 'REF'[  ]'REAL';
   'MODE' 'NET'    = 'REF'[, ]'REAL';
   'MODE' 'NETMAT' = 'REF'[,,]'REAL';

   'MODE' 'GRID' = 'STRUCT'('INT' LEVEL, 'NET' NET );
   'REAL'  # GLOBAL FOR MODE GRID #
           X0:=0.0, Y0:=0.0, XH0:=1.0, YH0:=1.0;

   'MODE' 'PROBLEM' = 'STRUCT'('BOOL'LINEAR,
             'REF'[,]'REAL' OMEGA,
             'PROC'('REF''REAL','REF''REAL','REF''REAL','REF''REAL',
                    'REAL','REAL'       )'VOID' PRINCIPLE PART,
             'PROC'('REF''REAL','REF''REAL','REF''REAL','REF''REAL',
                    'REAL','REAL','REAL')'VOID' LOWER ORDER,
             'PROC'('REF''REAL',
                    'REAL','REAL'       )'VOID' BOUNDARY PP,
             'PROC'('REF''REAL','REF''REAL',
                    'REAL','REAL','REAL')'VOID' BOUNDAPY LO);

   'PRIO' 'NORM' = 8, 'RESIDUALNORM' = 8;

# ZERO  AND COPY  OPERATIONS    #                      'PR' EJECT 'PR'

   'OP' 'ZERO' = ('VEC' A ) 'VEC':
                 ('FOR' I 'FROM'  'LWB'A 'TO'  'UPB'A
                  'DO' A[I]:= 0.0 'OD'; A             );
```

```
    'OP' 'ZERO' = ('NET' A ) 'NET':
                  ('FOR' I 'FROM' 1'LWB'A 'TO' 1'UPB'A
                   'DO' 'ZERO' A[I,] 'OD'; A          );

    'OP' 'ZERO' = ('NETMAT' A ) 'NETMAT':
                  ('FOR' I 'FROM' 1'LWB'A 'TO' 1'UPB'A
                   'DO' 'ZERO' A[I,,] 'OD'; A          );

    'OP' 'ZERO' = ('GRID' A )'GRID': ('ZERO' (NET'OF'A); A);

    'OP' 'COPY' = ('VEC' N ) 'VEC':
                  ('HEAP' ['LWB'N:'UPB'N]'REAL' :=N);
    'OP' 'COPY' = ('NET' N ) 'NET':
                  ('HEAP' [1'LWB'N:1'UPB'N,2'LWB'N:2'UPB'N]'REAL' :=N);
    'OP'  COPY' = ('NETMAT' N ) 'NETMAT':
                  ('HEAP' [1'LWB'N:1'UPB'N,2'LWB'N:2'UPB'N,
                                   3'LWB'N:3'UPB'N]'REAL' :=N);
    'OP' 'COPY' = ('GRID' A )'GRID':
                  ('NET'N= NET'OF'A; (LEVEL'OF'A,
                   'HEAP' [1'LWB'N:1'UPB'N,2'LWB'N:2'UPB'N]'REAL' :=N));

# ELEMENTARY  NET OPERATIONS    #                       'PR' EJECT 'PR'

    'OP' +      = ('NET' AA,BB )'NET':
    #$B NET+    B$#
                  ( 'INT' L1 = 1'LWB'AA, L2 = 2'LWB'AA,
                          U1 = 1'UPB'AA, U2 = 2'UPB'AA;
                    'HEAP'[L1:U1,L2:U2]'REAL' CC;
                    'FOR' I 'FROM' L1 'TO' U1 'DO'
                    'FOR' J 'FROM' L2 'TO' U2 'DO'
                             CC[I,J]:= AA[I,J] + BB[I,J] 'OD''OD';
                    CC )
    #$E#;
    #E$#
    'OP' -      = ('NET' AA,BB )'NET':
    #$B NET-    B$#
                  ( 'INT' L1 = 1'LWB'AA, L2 = 2'LWB'AA,
                          U1 = 1'UPB'AA, U2 = 2'UPB'AA;
                    'HEAP'[L1:U1,L2:U2]'REAL' CC;
                    'FOR' I 'FROM' L1 'TO' U1 'DO'
                    'FOR' J 'FROM' L2 'TO' U2 'DO'
                             CC[I,J]:= AA[I,J] - BB[I,J] 'OD''OD';
                    CC )
    #$E#;
    #E$#
    'OP' +:=    = ('NET' AA,BB )'NET':
    #$B NET=    B$#
                  ( 'INT' L1 = 1'LWB'AA, L2 = 2'LWB'AA,
                          U1 = 1'UPB'AA, U2 = 2'UPB'AA;
                    'FOR' I 'FROM' L1 'TO' U1 'DO'
                    'FOR' J 'FROM' L2 'TO' U2 'DO'
                             AA[I,J]+:=  BB[I,J] 'OD''OD';
                     AA )
    #$E#;
    #E$#
    'OP' *      = ('REAL' S,'NET' N ) 'NET':
    #$B NET*    B$#
                  ( 'INT' L1 = 1'LWB' N, L2 = 2'LWB'N,
                          U1 = 1'UPB' N, U2 = 2'UPB'N;
                    'HEAP'[L1:U1,L2:U2]'REAL' NN;
                    'FOR' I 'FROM' L1 'TO' U1 'DO'
                    'FOR' J 'FROM' L2 'TO' U2 'DO'
                          NN[I,J]:= S * N[I,J] 'OD' 'OD';
                    NN )
    #$E#;
    #E$#
    'OP' /      = ('NET' G,'REAL' S) 'NET':
    #$B NET/    B$#
                  ( 'REAL' T = 1.0/S; T * G )
    #$E#;
    #E$#
# ELEMENTARY GRID  PERATIONS    #                       'PR' EJECT 'PR'

    'OP' +      = ('GRID' A,B )'GRID':
    #$B GRID+   B$#
                  ( (LEVEL'OF'A /= LEVEL'OF'B ! ERROR );
                    (LEVEL'OF'A, NET'OF'A + NET'OF'B ) )
    #$E#;
    #E$#
    'OP' -      = ('GRID' A,B )'GRID':
    #$B GRID-   B$#
                  ( (LEVEL'OF'A /= LEVEL'OF'B ! ERROR );
                    (LEVEL'OF'A, NET'OF'A - NET'OF'B ) )
    #$E#;
    #E$#
    'OP' -      = ('GRID' A,'PROC'('REAL','REAL')'REAL'SOL )'GRID':
    #$B GRID--  B$#
                  ( 'INT' L:= LEVEL'OF'A; L:= 2**L;
                    'NET' AA= NET'OF'A;
                    'INT' L1 = 1'LWB'AA, L2 = 2'LWB'AA,
                          U1 = 1'UPB'AA, U2 = 2'UPB'AA;
                    'REAL' H1 = XH0/L, H2 = YH0/L;
                    'HEAP'[L1:U1,L2:U2]'REAL' CC;
                    'REAL' X:= X0+L1*H1, Y;
                    'FOR' I 'FROM' L1 'TO' U1 'DO' Y:= Y0+L2*H2;
                    'FOR' J 'FROM' L2 'TO' U2 'DO'
                         CC[I,J]:= AA[I,J] - SOL(X,Y); Y+:= H1 'OD';
                                                       X+:= H2 'OD';
                    (LEVEL'OF'A, CC ) )
    #$E#;
    #E$#
    'OP' +:=    = ('REF''GRID' A,'GRID' B )'REF' 'GRID':
    #$B GRID=   B$#
                  ( (LEVEL'OF'A /= LEVEL'OF'B ! ERROR );
                    NET'OF'A +:= NET'OF'B; A)
    #$E#;
    #E$#
    'OP' *      = ('REAL' S,'GRID' G ) 'GRID':
    #$B GRID*   B$#
                     (LEVEL'OF'G, S*NET'OF'G)
    #$E#;
    #E$#
    'OP' /      = ('GRID' G,'REAL' S) 'GRID':
    #$B GRID/   B$#
                  ( 'REAL' T = 1.0/S; T * G )
    #$E#;
    #E$#
# NORM OPERATORS                #                       'PR' EJECT 'PR'

    'OP' 'NORM' = ('BOOL' TRUE, 'NET' N) []'REAL':
    #$B NRMNT   B$#
                  ('REAL' TEST = ( TRUE ! MAXREAL ! 1.0E+15);
                   'REAL'S:= 0,S1:=0,S2:=0,T;
                   'INT' L1:=  1'LWB'N, B1:= 1'UPB'N,
                         L2:=  2'LWB'N, B2:= 2'UPB'N,NN;
                         NN:= (B1-L1+1)*(B2-L2+1);
                   'FOR' I 'FROM' L1 'TO' B1 'DO'
                   'FOR' J 'FROM' L2 'TO' B2 'DO'
```

```
                ( T:= 'ABS'N[I,J]; T < TEST
                ! S1+:=T; S2+ =T*T; (T>S ! S:=T )
                ! NN-:= 1 )
                'OD' 'OD'; (S1/NN,SQRT(S2/NN),S)  )
  #$E#;
  #E$#

  'OP' 'NORM' = ('NET' N) []'REAL':
  #$B NRM     B$#
        'TRUE''NORM'N
  #$E#;
  #E$#

  'PROC' RESIDUALNORM = ('NETMAT' M,'NET' U,F)[]'REAL':
      # U = ITERAND, F = RIGHT HAND SIDE #
  #$B NRMR    B$#
  'BEGIN' 'INT' L1= 1'LWB'U, L2= 2'LWB'U, L3=-3,
                U1= 1'UPB'U, U2= 2'UPB'U;
        'INT'    NN:= (U1-L1+1)*(U2-L2+1);
        'REAL' TEST = 1.0E+15;
        'REAL' V, S1:= 0, S2:= 0, S3:=0, T;

     'REF'[ ]'REAL' UIM:= U[L1,@L2], UI, UIP:= U[L1,@L2];
     'FOR' I 'FROM' L1 'TO' U1
     'DO' ( UI:= UIP; I = U1 ! 'SKIP' ! UIP:= U[I+1,@L2] );
          'REF' []'REAL' FI = F[I,@L2];
          'REF'[,]'REAL' MI = M[I,@L2,@L3];

          'INT' JM:= L2, JJ, JP:= L2;
          'FOR' J 'FROM' L2 'TO' U2
          'DO' ( JJ:= JP; J=U2 ! 'SKIP' ! JP+:= 1 );
               'REF'[]'REAL' MIJ = MI[JJ,@L3];
               ( MIJ[0] > TEST ! NN-:= 1 !
                         T :=    MIJ[-3]*UIM[JJ] + MIJ[-2]*UIM[JP] +
               MIJ[-1]*UI [JM]  + MIJ[ 0]*UI [JJ] + MIJ[ 1]*UI [JP] +
               MIJ[ 2]*UIP[JM]  + MIJ[ 3]*UIP[JJ] ;
               T := 'ABS' ( FI[JJ] - T );
               S1+:= T; S2+:= T*T; ( T>S3 ! S3:= T ));

               JM:= JJ
          'OD';
          UIM:= UI
     'OD'; (S1/NN,SQRT(S2/NN),S3)
  'END'
  #$E#;
  #E$#
# PROLONGATIONS                 #                       'PR' EJECT 'PR'

  'PROC' LIN INT POL = ('NET' NET )'NET':
  #$B PROL7   B$#
  'BEGIN' 'INT' L1 = 1'LWB'NET,  L2 = 2'LWB'NET,
                B1 = 1'UPB'NET,  B2 = 2'UPB'NET;
        'HEAP'[2*L1:2*B1,2*L2:2*B2]'REAL' FINE;
        'INT' JJ; 'REAL' U2,U3,U4;
             'REF'[]'REAL' UIP= NET[L1,@L2],
                           UPP= FINE[2*L1,@2*L2];
              JJ:= 2*L2; UPP[JJ]:= U4:= UIP[L2];
              'FOR' JP 'FROM' L2+1 'TO' E2
              'DO' U3:= U4; U4:= UIP[JP];
                   UPP[JJ+:=1]:= (U3+U4)/2;
                   UPP[JJ+:=1]:=     U4
              'OD';
        'FOR' IP 'FROM' L1+1 'TO' B1
        'DO' 'REF'[]'REAL' UI  = NET [IP-1  ,@  L2],
                           UIP = NET [IP    ,@  L2],
                           UMM = FINE[2*IP-1,@2*L2],
                           UPP = FINE[2*IP  ,@2*L2];
             JJ:= 2*L2; U2:= UI[L2]; U4:= UIP[L2];
             UMM[JJ]:= (U2+U4)/2;  UPP[JJ]:= U4;
             'FOR' JP 'FROM' L2+1 'TO' B2
             'DO' JJ+:= 1;  U2:= UI [JP];
                  U3:= U4;  U4:= UIP[JP];
                  UMM[JJ] := (U2+U3)/2;
                  UPP[JJ] := (U3+U4)/2;
                  JJ+:= 1;
                  UMM[JJ] := (U2+U4)/2;
                  UPP[JJ] :=     U4
             'OD'
        'OD'; FINE
  'END'
  #$E#;
  #E$#

  'PROC' BIL INT POL = ('NET' NET) 'NET':
  #$B PROL9   B$#
  'BEGIN' 'INT' L1 = 1'LWB'NET,  L2 = 2'LWB'NET,
                B1 = 1'UPB'NET,  B2 = 2'UPB'NET;
        'HEAP'[2*L1:2*B1,2*L2:2*B2]'REAL' FINE;
        'INT' JJ; 'REAL' U1,U2,U3,U4;
             'REF'[]'REAL' UIP= NET[L1,@L2],
                           UPP= FINE[2*L1,@2*L2];
              JJ:= 2*L2; UPP[JJ]:= U4:= UIP[L2];
              'FOR' JP 'FROM' L2+1 'TO' B2
              'DO' U3:= U4; U4:= UIP[JP];
                   UPP[JJ+:=1]:= (U3+U4)/2;
                   UPP[JJ+:=1]:=     U4
              'OD';
        'FOR' IP 'FROM' L1+1 'TO' B1
        'DO' 'REF'[]'REAL' UI  = NET [IP-1  ,@  L2],
                           UIP = NET [IP    ,@  L2],
                           UMM = FINE[2*IP-1,@2*L2],
                           UPP = FINE[2*IP  ,@2*L2];
             JJ:= 2*L2; U2:= UI[L2]; U4:= UIP[L2];
             UMM[JJ]:= (U2+U4)/2;  UPP[JJ]:= U4;
             'FOR' JP 'FROM' L2+1 'TO' B2
             'DO' JJ+:= 1;
                  U1:= U2;  U2:= UI [JP];
                  U3:= U4;  U4:= UIP[JP];
                  UMM[JJ] := (U1+U2+U3+U4)/4;
                  UPP[JJ] := (U3+U4)/2;
                  JJ+:= 1;
                  UMM[JJ] := (U2+U4)/2;
                  UPP[JJ] :=     U4
             'OD'
        'OD'; FINE
  'END'
  #$E#;
  #E$#



#  ORIENTATION:
                L2                      B2
               -------------------------->  Y

       ! L1              JM    JJ    JP
       !
```

```
        !          UIM      O --  O --  O
        !                   !  /  !  /  !
        !          UII      O -- U1 -- U2
        !                   !  /  !  /  !
   X    !          UIP      O -- U3 -- U4
        V B1



        ----------------------> Y
   !    L2                   B2
   ! L1                                        STAR ORIENTATION:
   !          U1 ------  U2                    ----------------> Y
   !           !     /    !                    !                 J
   !           !   /      !                    !          -3   -2
   !          U3 ------  U4                    !     -1    0    1
   ! B1                                        !      2    3
   V                                      X    ! I
                                               V
#                                                          'PR' EJECT 'PR'
   'PROC' SQR INT POL = ('NET' NET )'NET':
   #$B PROLS   B$#
   'IF'   'INT' L1 = 1'LWB'NET,  L2 = 2'LWB'NET,
               B1 = 1'UPB'NET,  B2 = 2'UPB'NET;
          'ODD'(B1-L1) 'OR' 'ODD'(B2-L2)
   'THEN' LIN INT POL (NET)
   'ELSE' 'INT' LL1 = 2*L1, LL2 = 2*L2;
          'HEAP'[LL1:2*B1,LL2:2*B2]'REAL' FINE;

      'INT' JJ, JP;
      'REAL' X1, X2, X3, Y1, Y2, Y3, Z1, Z2, Z3, YY2, YY3, ZZ2, ZZ3;
      'REF'[]'REAL' UI= NET[ L1,@L2], FI= FINE[LL1,];
       FI[LL2]:= X1:= UI[L2]; JJ:= LL2+1;
      'FOR' J 'FROM' L2+1 'BY' 2 'TO' B2-1
      'DO'  X2:= UI[J]; X3:= UI[J+1];
            FI[JJ:JJ+3] :=(  (  3*(X1 + 2*X2) -  X3 )/8, X2,
                             (  -X1 + 3*(2*X2 + X3 ))/8, X3  );
            JJ +:= 4; X1:= X3
      'OD';
      'FOR' II 'FROM' L1+1 'BY' 2 'TO' B1-1
      'DO' 'REF'[ ]'REAL' UIM= NET[II-1,@L2], UII= NET[II  ,@L2],
                          UIP= NET[II+1,@L2];
           'REF'[,]'REAL' FINEI = FINE[2*II-1:2*II+2,@LL2];

            X3:=          UIM[L2]  /8;
            Y3:= ( YY3:= UII[L2] )/4;
            Z3:= ( ZZ3:= UIP[L2] )/8;
            FINEI[,LL2]:= ( 3*(X3+Y3) - Z3, YY3, 3*(Y3+Z3) - X3, ZZ3 );

           'FOR' JJ 'FROM' L2+1 'BY' 2 'TO' B2-1
           'DO'  JP:= JJ+1;       X1:= X3; Y1:= Y3; Z1:= Z3;
                 X2:=          UIM[JJ]  /4; X3:=          UIM[JP]  /8;
                 Y2:= ( YY2:= UII[JJ] )/4; Y3:= ( YY3:= UII[JP] )/4;
                 Z2:= ( ZZ2:= UIP[JJ] )/4; Z3:= ( ZZ3:= UIP[JP] )/8;

                 FINEI[,2*JJ-1:2*JJ+2]:=
                 ((2*(X2+Y1)-Z1+Y2-X3,
                            2*(X2+Y2)-X1+Y1-Z1,
                                       3*(X3+Y2)-Z1,
                                                  3*(X3+Y3)-Z3   ),
                  (2*(Y1+Y2)-X1+X2-X3,   YY2,
                                       2*(Y2+Y3)-Z1+Z2-Z3,   YY3 ),
                                       3*(Z1+Y2)-X3,
                                                  2*(Y2+Z2)-X3+Y3-Z3,
                                                             2*(Z2+Y3)-X3+Y2-Z1,
                                                                        3*(Z3+Y3)-X3   ),
                 (3*(Z1+Z2)-Z3,  ZZ2,  3*(Z3+Z2)-Z1,          ZZ3 ))
        'OD'  'OD';
        FINE
   'FI'
   #$E#;
   #E$#
# RESTRICTIONS                  #                        'PR' EJECT 'PR'

   'PROC' INJECTION = ('NET' NET)'NET':
   #$B RESTR0  B$#
   'BEGIN' 'INT' NL1 = (1'LWB'NET)'OVER'2,  NL2 = (2'LWB'NET)'OVER'2,
                 NU1 = (1'UPB'NET)'OVER'2,  NU2 = (2'UPB'NET)'OVER'2;
           'HEAP'[NL1:NU1,NL2:NU2]'REAL' COARSE;
           'FOR' I 'FROM' NL1 'TO' NU1 'DO'
           'FOR' J 'FROM' NL2 'TO' NU2 'DO'
                 COARSE[I,J]:= NET[2*I,2*J] 'OD' 'OD';
           COARSE
   'END'
   #$E#;
   #E$#

   'PROC' INJ WEIGHT = ('NET' NET)'NET':
   #$B RESTR1  B$#
   'BEGIN' 'INT' NL1 = (1'LWB'NET)'OVER'2,  NL2 = (2'LWB'NET)'OVER'2,
                 NU1 = (1'UPB'NET)'OVER'2,  NU2 = (2'UPB'NET)'OVER'2;
           'HEAP'[NL1:NU1,NL2:NU2]'REAL' COARSE;

            COARSE[NL1,NL2]:= 2.0*NET[2*NL1,2*NL2];
            COARSE[NU1,NL2]:= 2.5*NET[2*NU1,2*NL2];
            COARSE[NL1,NU2]:= 2.5*NET[2*NL1,2*NU2];
            COARSE[NU1,NU2]:= 2.0*NET[2*NU1,2*NU2];
           'FOR' J 'FROM' NL2+1 'TO' NU2-1
           'DO'  COARSE[NL1,J]:= 3.0*NET[2*NL1,2*J];
                 COARSE[NU1,J]:= 3.0*NET[2*NU1,2*J]
           'OD';

           'FOR' I 'FROM' NL1+1 'TO' NU1-1
           'DO'  COARSE[I,NL2]:= 3*NET[2*I,2*NL2];
                 COARSE[I,NU2]:= 3*NET[2*I,2*NU2];
                 'FOR' J 'FROM' NL2+1 'TO' NU2-1
                 'DO' COARSE[I,J]  := 4*NET[2*I,2*J] 'OD'
           'OD'; COARSE
   'END'
   #$E#;
   #E$#
                                                          'PR' EJECT 'PR'
   'PROC' HALF WEIGHT = ('NET' NET)'NET':
   #$B RESTR5  B$#
   'BEGIN' 'INT' L1 = (1'LWB'NET), L2 = (2'LWB'NET),
                 U1 = (1'UPB'NET), U2 = (2'UPB'NET);
           'INT' NL1 = L1'OVER'2,  NL2 = L2'OVER'2,
                 NU1 = U1'OVER'2,  NU2 = U2'OVER'2;
           'HEAP'[NL1:NU1,NL2:NU2]'REAL' COARSE;
           'INT' II,IM,IP, JJ,JM,JP;
           [L2:U2]'REAL' ZERO; 'ZERO' ZERO;
           'FOR' I 'FROM' NL1 'TO' NU1
           'DO'   II:=  2*I; IP:= II+1; IM:= II-1;
                 'REF'[]'REAL' CI = COARSE[I,@NL2],
                               UIM= ( IM<L1 ! ZERO ! NET[IM,@L2]),
                               UII=                   NET[II,@L2] ,
```

```
                        UIP= ( IP>U1 ! ZERO ! NET[IP,@L2]);
            JP:= L2+1;
            CI[NL2]:=      0.5 *(           UIM[L2]+UIM[JP]
                                  +2*UII[L2]+UII[JP]
                                    +UIP[L2]);
            'FOR' J 'FROM' NL2+1 'TO' NU2-1
            'DO'  JJ:= 2*J; JP:= JJ+1; JM:= JJ-1;
                  CI[J]:=  0.5 *(           UIM[JJ]
                                  +UII[JM]+4*UII[JJ]+UII[JP]
                                    +UIP[JJ]        )
            'OD';
            JM:=U2-1;
            CI[NU2]:=      0.5 *(           UIM[U2]
                                  +UII[JM]+2*UII[U2]
                                  +UIP[JM]+  UIP[U2])
      'OD'  ;
      COARSE
'END'
#$E#;
#E$#

'PROC' LIN  WEIGHT = ('NET' NET)'NET':
#$B RESTR7  B$#
'BEGIN' 'INT' L1 = (1'LWB'NET), L2 = (2'LWB'NET),
            U1 = (1'UPB'NET), U2 = (2'UPB'NET);
      'INT' NL1 = L1'OVER'2,  NL2 = L2'OVER'2,
            NU1 = U1'OVER'2,  NU2 = U2'OVER'2;
      'HEAP'[NL1:NU1,NL2:NU2]'REAL' COARSE;
      'INT' II,IM,IP, JJ,JM,JP;
      [L2:U2]'REAL' ZERO; 'ZERO' ZERO;
      'FOR' I 'FROM' NL1 'TO' NU1
      'DO'   II:=  2*I; IP:= II+1; IM:= II-1;
            'REF'[]'REAL' CI = COARSE[I,@NL2],
                        UIM= ( IM<L1 ! ZERO ! NET[IM,@L2]),
                        UII=                  NET[II,@L2] ,
                        UIP= ( IP>U1 ! ZERO ! NET[IP,@L2]);
            JP:= L2+1;
            CI[NL2]:=      0.5 *(           UIM[L2]+UIM[JP]
                                  +2*UII[L2]+UII[JP]
                                    +UIP[L2]);
            'FOR' J 'FROM' NL2+1 'TO' NU2-1
            'DO'  JJ:= 2*J; JP:= JJ+1; JM:= JJ-1;
                  CI[J]:=  0.5 *(           UIM[JJ]+UIM[JP]
                                  +UII[JM]+2*UII[JJ]+UII[JP]
                                  +UIP[JM]+  UIP[JJ]        )
            'OD';
            JM:=U2-1;
            CI[NU2]:=      0.5 *(           UIM[U2]
                                  +UII[JM]+2*UII[U2]
                                  +UIP[JM]+  UIP[U2])
      'OD'; COARSE
'END'
#$E#;
#E$#

'PROC' FULL WEIGHT = ('NET' NET)'NET':
#$B RESTR9  B$#
'BEGIN' 'INT' L1 = (1'LWB'NET), L2 = (2'LWB'NET),
            U1 = (1'UPB'NET), U2 = (2'UPB'NET);
      'INT' NL1 = L1'OVER'2,  NL2 = L2'OVER'2,
            NU1 = U1'OVER'2,  NU2 = U2'OVER'2;
      'HEAP'[NL1:NU1,NL2:NU2]'REAL' COARSE;
      'INT' II,IM,IP, JJ,JM,JP;
      [L2:U2]'REAL' ZERO; 'ZERO' ZERO;
      'FOR' I 'FROM' NL1 'TO' NU1
      'DO'   II:= 2*I; IP:= II+1; IM:= II-1;
            'REF'[]'REAL' CI = COARSE[I,@NL2],
                        UIM= ( IM<L1 ! ZERO ! NET[IM,@L2]),
                        UII=                  NET[II,@L2] ,
                        UIP= ( IP>U1 ! ZERO ! NET[IP,@L2]);
            JP:= L2+1;
            CI[NL2]:=      0.5 *(           UIM[L2]+UIM[JP]
                                  +2*UII[L2]+UII[JP]
                                    +UIP[L2]);
            'FOR' J 'FROM' NL2+1 'TO' NU2-1
            'DO'  JJ:= 2*J; JP:= JJ+1; JM:= JJ-1;
                  CI[J]:=  0.25*(   UIM[JM]+2*UIM[JJ]+  UIM[JP]
                                  +2*UII[JM]+4*UII[JJ]+2*UII[JP]
                                  +  UIP[JM]+2*UIP[JJ]+  UIP[JP])
            'OD';
            JM:= U2-1;
            CI[NU2]:=      0.5 *(           UIM[U2]
                                  +UII[JM]+2*UII[U2]
                                  +UIP[JM]+  UIP[U2])
      'OD'  ;
      COARSE
  'END'
  #$E#;
  #E$#


# FINITE ELEMENT METHOD (FEM)                    #     'PR' EJECT 'PR'
# FEM DISCRETIZATION FOR MG  /PWH800925 / VSN.810311 #

 'PROC' FEM = ('PROBLEM' PROBLEM, 'GRID' U,
              'REF''NET' RHS, 'REF''NETMAT' JAC )'VOID':
   #$B FEM     B$#
 'BEGIN' 'REF'[,]'REAL' NET = NET'OF' U;
         'INT'  L1 = 1'LWB' NET,  L2 = 2'LWB'NET,
                B1 = 1'UPB' NET,  B2 = 2'UPB'NET;
         'HEAP' [L1:B1,L2:B2,-3:3 ]'REAL' STIFF; 'ZERO' STIFF;
         'HEAP' [L1:B1,L2:B2]'REAL' FORCE;       'ZERO' FORCE;
         'REAL' H = XH0/(2**(LEVEL'OF'U)),
                K = YH0/(2**(LEVEL'OF'U));


    'PROC'('REF''REAL','REF''REAL','REF''REAL','REF''REAL',
           'REAL','REAL'       )'VOID' PRINCIPLE PART
                                 = PRINCIPLE PART 'OF' PROBLEM;
    'PROC'('REF''REAL','REF''REAL','REF''REAL','REF''REAL',
           'REAL','REAL','REAL')'VOID' LOWER ORDER
                                 = LOWER ORDER    'OF' PROBLEM;

    []'INT'  ORDER = (0,1,3,-1,0,2,-3,-2,0,
                      0,2,3,-2,0,1,-3,-1,0);

    'REAL' H3 = H/3, K3 = K/3;
    []'REAL' PXT1 = (-1/H,  0 ,+1/H, 0),   PXT2 = ( 0,-1/H,  0 ,+1/H),
             PYT1 = (-1/K,+1/K,  0 , 0),   PYT2 = ( 0,  0 ,-1/K,+1/K);
    'INT'  LB,UB;
    'REAL' XN,XS,YW,YE, AXX,AXY,AYX,AYY,
           BBX,BBY, KK,PE,FF, WEIGHT:= H*K/6;
    [1:4]'REAL' PX,PY,BX,BY,C,F;

    [L2:B2]'REAL' BXI,BXIP,BYI,BYIP,CI,CIP,FI,FIP;
    XS:= X0+L1*H; 'REF' []'REAL' UIP= NET[L1,@L2];
    'FOR'J 'FROM'L2 'TO' B2
    'DO' LOWER ORDER
```

```
      (BXIP[J],BYIP[J],CIP[J],FIP[J],XS,Y0+J*K,UIP[J])
'OD';

'FOR' IP 'FROM' L1+1 'TO' B1
'DO' 'REF' []'REAL' UI = NET[IP-1,@L2],
                    UIP= NET[IP  ,@L2];
     XN:= XS; XS := X0 + IP*H;
     BXI:= BXIP; BYI:= BYIP; CI:=CIP; FI:= FIP;

     'INT'  JJ:= L2;
     'REAL' U1,U2:= UI [L2],
            U3,U4:= UIP[L2];
     YE:= Y0+L2*K;
     LOWER ORDER (BXIP[L2],BYIP[L2],CIP[L2],FIP[L2],XS,YE,U4);

     'FOR' JP 'FROM' L2+1 'TO' B2
     'DO' YW:= YE; YE := Y0 + JP*K;
          U1:= U2; U2 := UI [JP];
          U3:= U4; U4 := UIP[JP];
          LOWER ORDER (BXIP[JP],BYIP[JP],CIP[JP],FIP[JP],XS,YE,U4);

          BX:= (BXI[JJ],BXI[JP],BXIP[JJ],BXIP[JP]);
          BY:= (BYI[JJ],BYI[JP],BYIP[JJ],BYIP[JP]);
          C := (CI [JJ],CI [JP],CIP [JJ],CIP [JP]);
          F := (FI [JJ],FI [JP],FIP [JJ],FIP [JP]);

          'INT' LL:= 0;
          'FOR' TRIANGLE 'TO' 2
          'DO' 'CASE' TRIANGLE
               'IN' (LB:= 1; UB:= 3; PX:= PXT1; PY:= PYT1;
                     PRINCIPLE PART(AXX,AXY,AYX,AYY,XN+H3,YW+K3);
                     BBX:= (BX[1]+BX[2]+BX[3])/3;
                     BBY:= (BY[1]+BY[2]+BY[3])/3                    ),
                    (LB:= 2; UB:= 4; PX:= PXT2; PY:= PYT2;
                     PRINCIPLE PART(AXX,AXY,AYX,AYY,XS-H3,YE-K3);
                     BBX:= (BX[2]+BX[3]+BX[4])/3;
                     BBY:= (BY[2]+BY[3]+BY[4])/3                    )
               'ESAC';
               KK:= HB(AXX,AXY,AYX,AYY,BBX,BBY,H,K);
               AXX*:=3; AXY*:=3; AYX*:=3; AYY*:=3;

               'INT' EI:= 0;
               'FOR' II 'FROM' IP-1 'TO' IP 'DO'
               'FOR' JJ 'FROM' JP-1 'TO' JP 'DO'
               'IF'   EI+:= 1; EI = (TRIANGLE!4,1)
               'THEN' 'SKIP'
               'ELSE' FF:= Q2 * F[EI];
                      'FOR' EJ 'FROM' LB 'TO' UB
                      'DO' LL+:= 1;
                           PE := KK*(BX[EJ]*PX[EI]+BY[EJ]*PY[EI]);
                           FF+:= (Q0+PE)*F[EJ];
                        STIFF[II,JJ,ORDER[LL]] +:=
                        (PX[EJ] *(AXX*PX[EI]+AXY*PY[EI] + BX[EI])+
                         PY[EJ] *(AYX*PX[EI]+AYY*PY[EI] + BY[EI])+
                        C[EJ]*( (EI=EJ!Q1!Q0) + PE ) )*WEIGHT
                      'OD';
                      FORCE[II,JJ] +:= FF*WEIGHT
               'FI'
               'OD' 'OD'
          'OD';

          JJ:= JP
'OD' 'OD';
```

```
    BOUNDARY CONDITIONS ( PROBLEM,U,FORCE,STIFF );

    RHS:= FORCE; JAC:= STIFF
 'END'
   #$E#;
   #E$#
# DIFFERENT OPTIONS FOR DISCRETIZATION           #     'PR' EJECT 'PR'
# BROOKS AND HUGHES ANISOTROPIC DIFFUSION        #

#

'REAL' Q0:= 0, Q1:= 1, Q2:= 1;
'PROC' LUMP = ('BOOL' B)'VOID':
    ( B ! Q0:=0; Q1:=1; Q2:=1 ! Q0:=0.25; Q1:=0.50; Q2:=0.25 );

'REAL' HB FACTOR:= 1/3;
'PROC' HB:= ('REF''REAL' AXX,AXY,AYX,AYY,'REAL' BX,BY,H,K)'REAL':0.0;
#

'PROC' KAPPA = ('REF''REAL' AXX,AXY,AYX,AYY, 'REAL' BX,BY,H,K)'REAL':
   #$B HBW     B$#
'BEGIN' # ## ADDS ARTIFICIAL DIFFUSION AND COMPUTES K # ##
       'REAL' BEB  = (BX*AXX+BY*AYX)*BX +   1.0E-100 +
                     (BX*AXY+BY*AYY)*BY ,
              BB   = BX*BX + BY*BY      ,
              BH   = SQRT(BX*BX*H*H + BY*BY*K*K);
       'REAL' KK   = ( BB=0 ! 0.0 ! BH*HBFACTOR*M(BH*BB/BEB)/BB );
       AXX+:= KK*BX*BX; AXY+:= KK*BX*BY;
       AYX+:= KK*BX*BY; AYY+:= KK*BY*BY; KK
'END'
   #$E#;
   #E$#

'PROC' M = ('REAL' A)'REAL':
   #$B ILINM   B$#
'IF' 'REAL' X, W:= 'ABS'A; W< 0.2
'THEN' W*:= W; ((((( W - 9.9 ) * W + 99.0 ) * W
       - 1039.5 ) * W + 15592.5) * A /46777.5
'ELSE' X:= ( W-1.0)/W;
       ( W< 18.0 ! X +:= 2.0/(EXP(W+W) - 1.0));
       ( A > 0.0 ! X ! -X )
 'FI'
   #$E#;
   #E$#

#
       ---------------------->
  !      J                   Y
  ! I                                          STAR ORIENTATION:
  !           1 -- N --  2                     ----------------> Y
  !           !      /   !                  !                    J
  !           W    /     E                  !
  !           !  /       !                  !          -3   -2
  !           3 -- S --  4                  !     -1    0    1
  ! X                                       !      2    3
  V                                    X    ! I
                                            V
#

# FINITE ELEMENT METHOD (S-U P-G FEM)           #     'PR' EJECT 'PR'
# FEM DISCRETIZATION FOR MG  /PWH800925 / VSN.810311 #
# PIECEWISE CONSTANT COEFFICIENTS         VSN.830107 #
# INCL. STREAMLINE-UPWIND PETROV-GALERKIN            #
```

```
'PROC' FEMD = ('PROBLEM' PROBLEM, 'GRID' U,
               'REF''NET' RHS, 'REF''NETMAT' JAC )'VOID':
  #$B FEMW    B$#
'BEGIN' 'REF'[,]'REAL' NET = NET'OF' U;
        'INT'   L1 = 1'LWB' NET,  L2 = 2'LWB'NET,
                B1 = 1'UPB' NET,  B2 = 2'UPB'NET;
        'HEAP' [L1:B1,L2:B2,-3:3 ]'REAL' STIFF; 'ZERO' STIFF;
        'HEAP' [L1:B1,L2:B2]'REAL' FORCE;       'ZERO' FORCE;
        'REAL' H = XH0/(2**(LEVEL'OF'U)),
               K = YH0/(2**(LEVEL'OF'U));


   'PROC'('REF''REAL','REF''REAL','REF''REAL','REF''REAL',
          'REAL','REAL'       )'VOID' PRINCIPLE PART
                                  = PRINCIPLE PART 'OF' PROBLEM;
   'PROC'('REF''REAL','REF''REAL','REF''REAL','REF''REAL',
          'REAL','REAL','REAL')'VOID' LOWER ORDER
                                  = LOWER ORDER    'OF' PROBLEM;
   'BOOL' LIN = LINEAR'OF'PROBLEM;

   []'INT'  ORDER = (0,1,3,-1,0,2,-3,-2,0,
                     0,2,3,-2,0,1,-3,-1,0);

   'REAL' H3 = H/3, K3 = K/3;
   []'REAL' PXT1 = (-1/H,  0 ,+1/H, 0),   PXT2 = ( 0,-1/H,  0 ,+1/H),
            PYT1 = (-1/K,+1/K,  0 , 0),   PYT2 = ( 0,  0 ,-1/K,+1/K);
   'INT'  LB,UB;
   'REAL' XN,YW,     AXX,AXY,AYX,AYY, WEIGHT:= H*K/6;
   'REAL' XX,YY,UU, BX ,BY ,C  ,F  , KK,PE;
   [1:4]'REAL' PX,PY;

   'FOR' I 'FROM' L1 'TO' B1-1
   'DO' XN:= X0 + I*H;
        'REF' []'REAL' UI = NET[I  ,@L2],
                       UIP= NET[I+1,@L2];
        'REAL' U1,U2:= UI [L2],
               U3,U4:= UIP[L2];

        'FOR' J 'FROM' L2 'TO' B2-1
        'DO' YW:= Y0 + J*K;
             U1:= U2; U2 := UI [J+1];
             U3:= U4; U4 := UIP[J+1];

             'INT' LL:= 0;
             'FOR' TRIANGLE 'TO' 2
             'DO' 'CASE' TRIANGLE
                  'IN' (LB:= 1; UB:= 3;
                        PX:= PXT1;  PY:= PYT1;
                        XX:= XN+H3; YY:= YW+K3;
                        (LIN!'SKIP'!UU:= (U1+U2+U3)/3) ),
                       (LB:= 2; UB:= 4;
                        PX:= PXT2;  PY:= PYT2;
                        XX:= XX+H3; YY:= YY+K3;
                        (LIN!'SKIP'!UU:= (U2+U3+U4)/3) )
                  'ESAC';
                  PRINCIPLE PART(AXX,AXY,AYX,AYY,XX,YY);
                  LOWER ORDER (BX,BY,C,F,XX,YY,UU);
                  KK:= HB(AXX,AXY,AYX,AYY,BX,BY,H,K);
                  AXX*:=3; AXY*:=3; AYX*:=3; AYY*:=3;

                  'INT' EI:= 0;
                  'FOR' II 'FROM' I 'TO' I+1 'DO'
                  'FOR' JJ 'FROM' J 'TO' J+1 'DO'
                  'IF'   EI+:= 1; EI = (TRIANGLE!4,1)
                  'THEN' 'SKIP'
                  'ELSE' PE := KK*(BX*PX[EI]+BY*PY[EI]);
                         'FOR' EJ 'FROM' LB 'TO' UB
                         'DO' LL+:= 1;
                           STIFF[II,JJ,ORDER[LL]] +:=
                           (PX[EJ] *(AXX*PX[EI]+AXY*PY[EI] + BX)+
                            PY[EJ] *(AYX*PX[EI]+AYY*PY[EI] + BY)+
                            C*( (EI=EJ!0.5!0.25) + PE ) )*WEIGHT
                         'OD';
                         FORCE[II,JJ] +:= (1.0+3.0*PE) *F*WEIGHT
                  'FI'
                  'OD' 'OD'
     'OD' 'OD' 'OD';

     BOUNDARY CONDITIONS ( PROBLEM,U,FORCE,STIFF );

     RHS:= FORCE; JAC:= STIFF
  'END'
    #$E#;
    #E$#

# BOUNDARY CONDITION OPTIONS                    #     'PR' EJECT 'PR'

  'PROC' BC PW LIN = ('PROBLEM' PROBLEM, 'GRID' U,
                      'NET' FORCE, 'NETMAT' STIFF )'VOID':
    #$B BCPL    B$#
  'BEGIN' 'REF'[,]'REAL' NET = NET'OF' U;
          'INT'  L1 = 1'LWB' NET,  L2 = 2'LWB'NET,
                 B1 = 1'UPB' NET,  B2 = 2'UPB'NET;
          'REAL' H = XH0/(2**(LEVEL'OF'U)),       SHIFT = 0.001,
                 K = YH0/(2**(LEVEL'OF'U));

     'REF'[,]'REAL' OMEGA        = OMEGA 'OF' PROBLEM ;
     'PROC'('REF''REAL','REF''REAL','REF''REAL','REF''REAL',
            'REAL','REAL'       )'VOID' PRINCIPLE PART
                                    = PRINCIPLE PART 'OF' PROBLEM;
     'PROC'('REF''REAL',
            'REAL','REAL'       )'VOID' BOUNDARY PP
                                    = BOUNDARY PP    'OF' PROBLEM;
     'PROC'('REF''REAL','REF''REAL',
            'REAL','REAL','REAL')'VOID' BOUNDARY LO
                                    = BOUNDARY LO    'OF' PROBLEM;

     'PROC' BOUNDARY SEGMENT = ('REF' 'INT' I,J, 'INT' DIRECTION,
                                'REF' 'REAL' X,Y,BZ,CZ)'VOID':
     'BEGIN' 'INT' IN,JN;
             'REAL' XN,YN,AZN,BZN,CZN,HSTEP,XHALF,YHALF,P,Q,
                    A11,A12,A21,A22,PPP,PLO;
             'INT' AD = 'ABS' DIRECTION,  ND= 'SIGN' DIRECTION;

             ( AD = 1 ! IN := I     ; XN:= X
                      ! IN := I + ND; XN:= X0 + IN*H; HSTEP:= H/2 );
             ( AD = 3 ! JN := J     ; YN:= Y
                      ! JN := J + ( AD ! ND, -ND ! ERROR;0 );
                                    YN:= Y0 + JN*K; HSTEP:= K/2 );
             ( AD = 2 ! HSTEP:= 0.5*SQRT(H*H+K*K) );
             XHALF:= (X+XN)*0.5 - (YN-Y)*SHIFT;
             YHALF:= (Y+YN)*0.5 + (XN-X)*SHIFT;
             PRINCIPLE PART (A11,A12,A21,A22,XHALF,YHALF);
             BOUNDARY PP    (AZN,           XHALF,YHALF);
             BOUNDARY LO    (BZN,CZN, XN, YN, NET[IN,JN]);

            'CASE' AD
            'IN' ( P:=  A11;              Q:=  A12              ),
```

```
              ( P:= (A11+A12+A21+A22)/2; Q:=(-A11+A12-A21+A22)/2 ),
              ( P:=  A22;                Q:= -A21                 )
          'ESAC';
          PPP:= ( P*AZN - Q)/2;          PLO:= P*HSTEP;

          STIFF[ I, J,         0 ] +:= -PPP + PLO*BZ ;
          STIFF[ I, J, DIRECTION] +:=  PPP           ;
          FORCE[ I, J           ] +:=        PLO*CZ  ;
          STIFF[IN,JN,         0 ] +:= +PPP + PLO*BZN;
          STIFF[IN,JN,-DIRECTION] +:= -PPP           ;
          FORCE[IN,JN           ] +:=        PLO*CZN;

          I:= IN; J:= JN; X:= XN; Y:= YN; BZ:= BZN; CZ:= CZN
    'END';

    'IF'    OMEGA :=: 'NIL'
    'THEN' 'INT' I:= L1, J:= L2; 'REAL' X:= X0+I*H, Y:= Y0+J*K, BZ,CZ;
          BOUNDARY LO (BZ,CZ,X,Y,NET[I,J]);
          'WHILE' I<B1 'DO' BOUNDARY SEGMENT(I,J, 3,X,Y,BZ,CZ) 'OD';
          'WHILE' J<B2 'DO' BOUNDARY SEGMENT(I,J, 1,X,Y,BZ,CZ) 'OD';
          'WHILE' I>L1 'DO' BOUNDARY SEGMENT(I,J,-3,X,Y,BZ,CZ) 'OD';
          'WHILE' J>L2 'DO' BOUNDARY SEGMENT(I,J,-1,X,Y,BZ,CZ) 'OD'
    'ELIF' 1'UPB'OMEGA >1
    'THEN' 'INT' I:= 'ROUND'((OMEGA[1,1]-X0)/H ),
                 J:= 'ROUND'((OMEGA[1,2]-Y0)/K ),IN,JN,DIR;
          'REAL' X:= X0+I*H, Y:= Y0+J*K, BZ,CZ;
          BOUNDARY LO (BZ,CZ,X,Y,NET[I,J]);

          'FOR' S 'FROM' 2 'TO' 1'UPB'OMEGA
          'DO'  IN:= 'ROUND'((OMEGA[S,1]-X0)/H );
                JN:= 'ROUND'((OMEGA[S,2]-Y0)/K );
                DIR:= 'SIGN'(JN-J) + 3*'SIGN'(IN-I);
                'WHILE'  BOUNDARY SEGMENT(I,J,DIR,X,Y,BZ,CZ);
                         I /= IN 'OR' J /= JN
                'DO' 'SKIP' 'OD'
          'OD'
    'FI'
'END'
  #$#;
  #E$#


                                                        'PR' EJECT 'PR'
'PROC' BC PW CON = ('PROBI 1' PROBLEM, 'GRID' UG,
                    'NET' FORCE, 'NETMAT' STIFF )'VOID':
  #$B BCPC    B$#
'BEGIN' 'REF'[,]'REAL' NET = NET'OF' UG;
        'INT'  L1 = 1'LWB' NET,  L2 = 2'LWB'NET,
               B1 = 1'UPB' NET,  B2 = 2'UPB'NET;
        'REAL' H = XH0/(2**(LEVEL'OF'UG)),
               K = YH0/(2**(LEVEL'OF'UG));


   'PROC'('REF''REAL','REF''REAL',
          'REAL','REAL' 'REAL')'VOID' BOUNDARY LO
                                    = BOUNDARY LO    'OF' PROBLEM;

   'PROC' BOUNDARY SEGMENT = ('INT' DIRECTION )'VOID':
   'BEGIN' 'REAL' XO:= X, YO:= Y, UO:= U, B, C, W;
           'INT'  IO:= I, JO:= J, ND:= 'SIGN' DIRECTION;

           W:= ( 'ABS' DIRECTION = 1
               ! J+:= ND; Y:= Y0+J*K; K/2
               ! I+:= ND; X:= X0+I*H; H/2
               );

           BOUNDARY LO (B,C,(X+XO)/2,(Y+YO)/2,
               (LIN! 'SKIP' ! ((U:=NET[I,J]) + UO)/2 ));

           # ## DIRICHLET: B=LARGE, C=LARGE*BC       # ##
           # ## NEUMANN  : B=  0.0, C=       H       # ##

           STIFF[IO,JO, 0] +:= W*B;
           FORCE[IO,JO   ] +:= W*C;
           STIFF[I ,J , 0] +:= W*B;
           FORCE[I ,J    ] +:= W*C
   'END';

    'BOOL' LIN = LINEAR'OF'PROBLEM;
    'INT'  I:= L1,     J:= L2;
    'REAL' X:= X0+I*H, Y:= Y0+J*K, U:= NET[I,J];
    'WHILE' I<B1 'DO' BOUNDARY SEGMENT( 3) 'OD';
    'WHILE' J<B2 'DO' BOUNDARY SEGMENT( 1) 'OD';
    'WHILE' I>L1 'DO' BOUNDARY SEGMENT(-3) 'OD';
    'WHILE' J>L2 'DO' BOUNDARY SEGMENT(-1) 'OD'
'END'
  #$E#;
  #E$#


                                                        'PR' EJECT 'PR'
'PROC' BC DIRICH = ('PROBLEM' PROBLEM, 'GRID' U,
                    'NET' FORCE, 'NETMAT' STIFF )'VOID':
  #$B BCDIR   B$#
'BEGIN' 'REF'[,]'REAL' NET = NET'OF' U;
        'INT'  L1 = 1'LWB' NET,  L2 = 2'LWB'NET,
               B1 = 1'UPB' NET,  B2 = 2'UPB'NET;
        'REAL' H = XH0/(2**(LEVEL'OF'U)),
               K = YH0/(2**(LEVEL'OF'U));


   'PROC'('REF''REAL','REF''REAL',
          'REAL','REAL','REAL')'VOID' BOUNDARY LO
                                    = BOUNDARY LO    'OF' PROBLEM;

   'PROC' BOUNDARY SEGMENT = ('INT' DIRECTION )'VOID':
   'BEGIN' 'INT'  ND = 'SIGN' DIRECTION;

           ( 'ABS' DIRECTION = 1
           ! J+:= ND; Y:= Y0+J*K
           ! I+:= ND; X:= X0+I*H
           );

           # ## DIRICHLET: B=LARGE, C=LARGE*BC       # ##
           BOUNDARY LO ( STIFF[I,J,0], FORCE[I,J], X, Y,
                         ( LIN ! 'SKIP' ! NET[I,J]) )

   'END';

    'BOOL' LIN = LINEAR'OF'PROBLEM;
    'INT'  I:= L1,     J:= L2;
    'REAL' X:= X0+I*H, Y:= Y0+J*K;
           BOUNDARY LO ( STIFF[I,J,0], FORCE[I,J], X, Y,
                         ( LIN ! 'SKIP' ! NET[I,J]) );
    'WHILE' I<B1 'DO' BOUNDARY SEGMENT( 3) 'OD';
    'WHILE' J<B2 'DO' BOUNDARY SEGMENT( 1) 'OD';
    'WHILE' I>L1 'DO' BOUNDARY SEGMENT(-3) 'OD';
    'WHILE' J>L2 'DO' BOUNDARY SEGMENT(-1) 'OD'
'END'
```

```
    $E#;
   #E$#

# CALERKIN OPERATOR CONSTRUCTION                #     'PR' EJECT 'PR'

'OP' 'RAP' = ('NETMAT' AFI )'NETMAT':
   #$B RAP     B$#
'BEGIN' 'INT' L1 = (1'LWB'AFI)'OVER'2, U1 = (1'UPB'AFI)'OVER'2,
              L2 = (2'LWB'AFI)'OVER'2, U2 = (2'UPB'AFI)'OVER'2;
        'HEAP' [L1:U1,L2:U2,-3:3]'REAL' ACO; 'ZERO' ACO;
        'INT' TI,TK,TIM,TKM;

       [1:3,1:3,-3:3]'REAL' FINE;
       'REF'[]'REAL'
              A = FINE[1,1,@-3], B = FINE[1,2,@-3], C = FINE[1,3,@-3],
              D = FINE[2,1,@-3], E = FINE[2,2,@-3], F = FINE[2,3,@-3],
              G = FINE[3,1,@-3], H = FINE[3,2,@-3], J = FINE[3,3,@-3];
       'REF''REAL'
              AA =A[ 0], AB =A[ 1], AD =A[ 3],
              BA =B[-1], BB =B[ 0], BC =B[ 1], BD =B[ 2], BE =B[ 3],
              CB =C[-1], CC =C[ 0], CE =C[ 2], CF =C[ 3],
              DA =D[-3], DB =D[-2], DD =D[ 0], DE =D[ 1], DG =D[ 3],
              EB =E[-3], EC =E[-2], ED =E[-1], EE =E[ 0],
                                    EF =E[ 1], EG =E[ 2], EH =E[ 3],
              FC =F[-3], FE =F[-1], FF =F[ 0], FH =F[ 2], FJ =F[ 3],
              GD =G[-3], GE =G[-2], GG =G[ 0], GH =G[ 1],
              HE =H[-3], HF =H[-2], HG =H[-1], HH =H[ 0], HJ =H[ 1],
              JF =J[ 3], JH =J[-1], JJ =J[ 0];

            TI:= 2*L1; TK:= 2*L2;
            FINE[3,3,]:= AFI[TI,TK,];
           # ##JJ# ##ACO[L1,L2,0]:= 4*JJ;
       'FOR' K 'FROM' L2+1 'TO' U2
       'DO' TKM:= TK; TK:= 2*K;
            FINE[3,1:3,]:= AFI[TI,TKM:TK,];
           # ##GG# ##ACO[L1,K-1,0] +:= (GH+HG)*2 + HH;
           # ##JJ# ##ACO[L1,K  ,0] +:= (JH+HJ)*2 + HH +JJ*4;
           # ##G_ # ##ACO[L1,K-1,1] +:= (GH+HJ)*2 + HH;
           # ##JG# ##ACO[L1,K ,-1] +:= (HG+JH)*2 + HH
       'OD';

       'FOR' I 'FROM' L1+1 'TO' U1
       'DO' TIM:= TI; TI:= 2*I; TK:= 2*L2;
            FINE[1:3,3,]:= AFI[TIM:TI,TK,];
           # ##CC# ##ACO[I-1,L2,0] +:= (CF+FC)*2 + FF;
           # ##JJ# ##ACO[I  ,L2,0] +:= (JF+FJ)*2 + FF + JJ*4;
           # #_JJ# ##ACO[I-1,L2,3] +:= (CF+FJ)*2 + FF;
           # ##JC# ##ACO[I ,L2,-3] +:= ( C+JF)*2 + FF;
       'FOR' K 'FROM' L2+1 'TO' U2
       'DO' TKM:= TK; TK:= 2*K;
            FINE[1:3,1:3,]:= AFI[TIM:TI,TKM:TK,];
            'REF'[]'REAL' A = ACO[I-1,K-1,@-3],
                          C = ACO[I-1,K  ,@-3],
                          G = ACO[I  ,K-1,@-3],
                          J = ACO[I  ,K  ,@-3];
           # ##AA# ## A[ 0]+:= BD + DB;
           # ##CC# ## C[ 0]+:= (CE+EC+CF+FC)*2 + EE+FF+BE+EB+EF+FE;
           # ##GG# ## G[ 0]+:= (GE+EG+GH+HG)*2 + EE+HH+DE+ED+EH+HE;
           # ##JJ# ## J[ 0]+:= JJ*4+ (JF+FJ+JH+HJ)*2 + FF+HH+FH+HF;
           # ##AC# ## A[ 1]+:= BE+DB+DE;
           # ##CA# ## C[-1]+:= EB+BD+ED;
           # ##AG# ## A[ 3]+:= BD+DE+BE;
           # ##GA# ## G[-3]+:= DB+ED+EB;
           # ##GC# ## G[-2]+:= (GE+EC)*2 + EE+HE+DE+HF+DB+EF+EB;
           # ##CG# ## C[ 2]+:= (EG+CE)*2 + EE+EH+ED+FH+BD+FE+BE;
           # ##GJ# ## G[ 1]+:= (GH+HJ)*2 + HH+EH+HF+EF;
           # ##JG# ## J[-1]+:= (HG+JH)*2 + HH+HE+FH+FE;
           # ##CJ# ## C[ 3]+:= (CF+FJ)*2 + FF+EH+EF+FH;
           # ##JC# ## J[-3]+:= (FC+JF)*2 + FF+HE+FE+HF
       'OD' 'OD';
       'FOR' I 'FROM' L1 'TO' U1 'DO'
       'FOR' K 'FROM' L2 'TO' U2 'DO'
       'FOR' L 'FROM' -3 'TO'  3 'DO' ACO[I,K,L] *:= 0.25
       'OD' 'OD' 'OD';   ACO
'END'
   #$E#;
   #E$#


#  ORIENTATION:


      ACO = COARSE          K-1           K
                ------------------------->  Y
          !
          !          FINE     1     2     3
          !                                                  -3    -2
          ! I-1        1      A --  B --  C                    !  /
          !                   !  /  !  /  !              -1  -  0  -  1
          !             2     D --  E --  F                 /  !
          !                   !  /  !  /  !               2     3
          !  I          3     G --  H --  J
   X      V


   THIS ALGORITHM CAN BE EFFECTUATED BY  87 ADDITIONS AND
                                          7 MULTIPLICATIONS BY 0.25
   PER COARSE GRID POINT
#

# SOLVE BY GAUSSIAN ELIMINATION #                          'PR' EJECT 'PR'

   'PROC' SOLVE = ('NETMAT' C, 'NET' U,F)'VOID':
   #$B SOLVE   B$#
   'BEGIN' 'NETMAT' B = C [@1,@1,@3'LWB'C];
           'NET' RHS  = F [@1,@1];
           'INT' N1 = 1'UPB'RHS, N2= 2'UPB'RHS;
           'INT' N = N1*N2, S = 3'UPB'B, N3 = N2-3;
                 # ## S=4 : NINE-STAR ; S=3 : SEVEN-STAR # ##

           'PRIO'  <> = 4;
           'OP' <> = ('VEC' A,B) 'VOID':
           'FOR' I 'FROM' 'LWB' A 'TO' 'UPB' A
           'DO' 'REAL' S = A[I]; A[I]:= B[I]; B[I]:= S 'OD';

           'REF'[,]'REAL' SOLUT = U[@1,@1];
           [1:N, 1:N+1]'REAL' A;
           'VEC' V = A[,N+1];

           'FOR' I 'TO' N1 'DO'
           'FOR' J 'TO' N2 'DO'
           'BEGIN' 'INT' K = (I-1)*N2 + J;
                   'REF'[]'REAL' AK= A[K,], BIJ= B[I,J,@-S];
                   'FOR' L 'TO' N 'DO' AK[L]:= 0.0 'OD';
                   'FOR' Z 'FROM' -S 'TO' S
                   'DO' 'IF'   I= 1 'AND' Z<=-1 'THEN' 'SKIP'
                        'ELIF' I=N1 'AND' Z> 1 'THEN' 'SKIP'
                        'ELIF' J= 1 'AND' (Z=-4 'OR' Z=-1 'OR' Z= 2)
```

```
                    'THEN' 'SKIP'
                'ELIF' J=N2 'AND' (Z= 4 'OR' Z= 1 'OR' Z=-2)
                    'THEN' 'SKIP'
                'ELSE'
                    AK[(Z<-1 ! K+Z-N3 !: Z>1 ! K+Z+N3 ! K+Z)]
                    := BIJ[Z]
                'FI'
            'OD';
            AK[N+1] := RHS[I,J]
    'END' 'OD' 'OD';

    # ##         'FOR' K 'TO' N
            'DO' PRINT(NEWLINE);
                'FOR' L 'TO' N+1
                'DO' PRINT((FLOAT(A[K,L],7,1,2),"  "))
            'OD' 'OD';PRINT((NEWLINE,NEWLINE));
    # ##                                              'PR' EJECT 'PR'
    'FOR' J 'TO' N
    'DO' 'INT' JP1= J+1; 'INT' PJ:= J;
         'REAL' SI,S:= 'ABS' A[J,J];
         'FOR' I 'FROM' JP1 'TO' N
         'DO' (SI:='ABS'A[I,J]) >S ! S:=SI; PJ:=I ) 'OD';
         (J /= PJ ! A[PJ,] <> A[J,] ); S := A[J,J];
         'FOR' I 'FROM' JP1 'TO' N
         'DO' SI:= A[I,J]/S;
              'FOR' K 'FROM' J 'TO' N+1
              'DO' A[I,K] -:= A[J,K]*SI 'OD'
         'OD'
    'OD';
    'FOR' J 'FROM' N 'BY' -1 'TO' 1
    'DO' V[J] /:= A[J,J];

         'FOR' I 'FROM' J-1 'BY' -1 'TO' 1
         'DO' V[I] -:= A[I,J]*V[J] 'OD'
    'OD';

    'FOR' I 'TO' N1
    'DO' 'FOR' J 'TO' N2
         'DO' SOLUT[I,J]:=V[(I-1)*N2 + J] 'OD'
    'OD'
'END'
#$E#;
#E$#
# OPERATOR EVALUATION          #                          'PR' EJECT 'PR'

# !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! #
# EVERYWHERE THE MATRIX DOES NOT DEFINE THE NETMAT M, #
# !!!!!!!!!!!! M SHOULD CONTAIN ZEROES !!!!!!!!!!!!!!!!! #
# !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! #

'PROC' RESIDUAL = ('NETMAT' M, 'NET' U,F )'NET':
#$B RESID   B$#
'BEGIN''INT' L1= 1'LWB'U, L2= 2'LWB'U,
            U1= 1'UPB'U, U2= 2'UPB'U;
   'HEAP'[L1:U1,L2:U2]'REAL' S;

   'REF'[ ]'REAL' UIM:= U[L1,@L2], UI, UIP:= U[L1,@L2];
   'FOR' I 'FROM' L1 'TO' U1
   'DO' ( UI:= UIP; I = U1 ! 'SKIP' ! UIP:= U[I+1,@L2] );
        'REF' []'REAL' SI = S[I,@L2], FI = F[I,@L2];
        'REF'[,]'REAL' MI = M[I,@L2,@-3];

        'INT' JM:= L2, JJ, JP:= L2;
        'FOR' J 'FROM' L2 'TO' U2
        'DO' ( JJ:= JP; J=U2 ! 'SKIP' ! JP+:= 1 );

             'REF'[]'REAL' MIJ = MI[JJ,@-3];
             SI[JJ]:= FI[JJ] -
                      (          MIJ[-3]*UIM[JJ] + MIJ[-2]*UIM[JP] +
             MIJ[-1]*UI [JM]  + MIJ[ 0]*UI [JJ] + MIJ[ 1]*UI [JP] +
             MIJ[ 2]*UIP[JM]  + MIJ[ 3]*UIP[JJ] );

             JM:= JJ
        'OD';
        UIM:= UI
   'OD';S
'END'
#$E#;
#E$#                                                    'PR' EJECT 'PR'
'OP' *  = ('NETMAT' M, 'NET' U)'NET':
#$B NETMAT* B$#
'BEGIN' 'INT' L1= 1'LWB'U, L2= 2'LWB'U, L3= 3'LWB'M,
              U1= 1'UPB'U, U2= 2'UPB'U;
   'HEAP'[L1:U1,L2:U2]'REAL' S;

   'REF'[ ]'REAL' UIM:= U[L1,@L2], UI, UIP:= U[L1,@L2];
   'FOR' I 'FROM' L1 'TO' U1
   'DO' ( UI:= UIP; I = U1 ! 'SKIP' ! UIP:= U[I+1,@L2] );
        'REF' []'REAL' SI = S[I,@L2];
        'REF'[,]'REAL' MI = M[I,@L2,@L3];

        'INT' JM:= L2, JJ, JP:= L2;
        'FOR' J 'FROM' L2 'TO' U2
        'DO' ( JJ:= JP; J=U2 ! 'SKIP' ! JP+:= 1 );

             'REF'[]'REAL' MIJ = MI[JJ,@L3];
             'REF'  'REAL' SIJ = SI[JJ]:=
                               + MIJ[-3]*UIM[JJ] + MIJ[-2]*UIM[JP] +
             MIJ[-1]*UI [JM]  + MIJ[ 0]*UI [JJ] + MIJ[ 1]*UI [JP] +
             MIJ[ 2]*UIP[JM]  + MIJ[ 3]*UIP[JJ] ;


             JM:= JJ
        'OD';
        UIM:= UI
   'OD';S
'END'
#$E#;
#E$#
# RELAXATION METHODS           #                          'PR' EJECT 'PR'

'PROC' DIRICHLET RELAX  = ('PROBLEM' P, 'REF''GRID' UG,
                           'NETMAT'  M, 'NET'        F)'VOID':
      # SUBSTITUTES BOUNDARY VALUES,
        USEFULL IN CASE OF DIRICHLET BOUNDARY CONDITIONS #
#$B RELAXD  B$#
'IF'   'REF'[, ]'REAL' OMEGA = OMEGA'OF'P;
       'REF'[, ]'REAL' U = NET'OF'UG;
       OMEGA :=: 'NIL'
'THEN' 'INT' L1=1'LWB'U, U1=1'UPB'U, L2=2'LWB'U, U2=2'UPB'U;
       'FOR' I 'FROM' L1 'BY' U1-L1 'TO' U1
       'DO' 'FOR' J 'FROM' L2 'TO' U2
            'DO' U[I,J]:= F[I,J]/M[I,J,0] 'OD'
       'OD';
       'FOR' J 'FROM' L2 'BY' U2-L2 'TO' U2
       'DO' 'FOR' I 'FROM' L1 'TO' U1
```

```
            'DO' U[I,J]:= F[I,J]/M[I,J,0] 'OD'
        'OD'
    'ELSE' 'INT'  L = LEVEL'OF'UG;
        'REAL' H = XH0/2**L, K = YH0/2**L;
        'INT'  I:= 'ROUND'((OMEGA[1,1]-X0)/H),
               J:= 'ROUND'((OMEGA[1,2]-Y0)/K), IN,JN,DI,DJ;
        'FOR' S 'FROM' 2 'TO' 1'UPB'OMEGA
        'DO' IN:= 'ROUND'((OMEGA[S,1]-X0)/H); DI:= 'SIGN'(IN-I);
             JN:= 'ROUND'((OMEGA[S,2]-Y0)/K); DJ:= 'SIGN'(JN-J);
             'WHILE' U[I,J]:= F[I,J]/M[I,J,0];
                     I /= IN   'AND' J /= JN
             'DO' I+:= DI; J+:=DJ 'OD'
        'OD'
    'FI'
    #$F#;
    #F$#
# POINT RELAXATION PROCEDURES   #                          'PR' EJECT 'PR'

#'BOOL' SYMMETRIC:= 'FALSE', BACKWARD:= 'FALSE',
        REVERSE  := 'FALSE', ZEBRA   := 'FALSE';
#
    'PROC' PGS RELAX    = ('NETMAT'M, 'NET' U,F)'VOID':
    #$B RELPGS  B$#
    'BEGIN' # ## POINT GAUSS SEIDEL (PGS) # ##
       'INT' L1:= 1'LWB'U, U1:= 1'UPB'U, START1, STEP1, STOP1,
             L2:= 2'LWB'U, U2:= 2'UPB'U, START2, STEP2, STOP2;

       'TO' ( SYMMETRIC ! 2 ! 1)
       'DO' ( BACKWARD ! START1:= U1; STEP1:= -1; STOP1:= L1
                       ! START1:= L1; STEP1:=  1; STOP1:= U1 );
            ( REVERSE  ! START2:= U2; STEP2:= -1; STOP2:= L2
                       ! START2:= L2; STEP2:=  1; STOP2:= U2 );

            'FOR' I 'FROM' START1 'BY' STEP1 'TO' STOP1
            'DO'  'REF'[] 'REAL' FI= F[I,@L2], UIM= U[(I>L1!I-1!I),@L2],
                                 UI= U[I,@L2], UIP= U[(I<U1!I+1!I),@L2];
                  'REF'[,]'REAL' MI= M[I,@L2,@-3];
                  'FOR' J 'FROM' START2 'BY' STEP2 'TO' STOP2
                  'DO' 'INT' JM= (J>L2!J-1!J), JP= (J<U2!J+1!J);
                       'REF'[]'REAL' MIJ = MI[J,@-3];

                       UI[J]:=       ( MIJ[-3]*UIM[J]+MIJ[-2]*UIM[JP]+
                       MIJ[-1]*UI [JM]    -     FI[J]+MIJ[ 1]*UI [JP]+
                       MIJ[ .]*UIP[JM]+MIJ[ 3]*UIP[J] )/   -MIJ[ 0]

                  'OD'
            'OD'  ;

            ( SYMMETRIC! REVERSE:='NOT'REVERSE; BACKWARD:='NOT'BACKWARD)
       'OD'
    'END'
    #$L#;
    #F$#
# LINE  RELAXATION PROCEDURES   #                          'PR' EJECT 'PR'

    'PROC' LGS RELAX    = ('NETMAT'M, 'NET' U,F)'VOID':
    #$B RELLGS  B$#
    'BEGIN' # ## LINE GAUSS SEIDEL (LGS)  # ##
       'INT' ST = ( ZEBRA ! 2 ! 1 );
       'INT' L1:= 1'LWB'U, U1:= 1'UPB'U, START, STEP, STOP;

       'FOR' K 'TO' ( SYMMETRIC 'OR' ZEBRA ! 2 ! 1)
       'DO' ( BACKWARD ! START := U1; STEP := -ST; STOP := L1
                       ! START := L1; STEP :=  ST; STOP := U1 );
            (   ZEBRA
            ! ( SYMMETRIC /= 'ODD'(K+START) ! START+:= 'SIGN'STEP )
            # ## ( SYMMETRIC ! EVEN-ODD ! ODD-EVEN ) HALF STEP # ##);

            'FOR' I 'FROM' START  'BY' STEP  'TO' STOP
            'DO' LINE RELAX ( U[ (I>L1!I-1!I),], U[I,],
                              U[ (I<U1!I+1!I),], F[I,], M[I,,@-3] )
            'OD';
            (SYMMETRIC ! BACKWARD:= 'NOT' BACKWARD )
       'OD'
    'END'
    #$E#;
    #E$#

    'PROC' LINE RELAX = ( 'VEC'UM,U,UP,F,'REF'[,]'REAL'M )'VOID':
    #$B RELLINE B$#
    'BEGIN' 'VEC' B= M[, 1],# ##B[L:K], SUP# ## N = M[,-3], NE= M[,-2],
                  A= M[, 0],# ##A[L:K], DIA# ## S = M[, 3], SW= M[, 2],
                  C= M[,-1];# ##C[L:K], SUB# ##
            # ##NOT EXISTING MATRIX ELEMENTS: C[L]= B[K]=  0 !!!!!!# ##

            'INT' L='LWB'F, K='UPB'F; [L:K]'REAL' AA;
            ( 'UPB'A /= K 'OR' 'UPB'U /= K ! ERROR );
            'INT' I:=L; 'REAL' G:= 0, P; AA[L]:= 1.0;

            'FOR' J 'FROM' L 'TO' K
            'DO' AA[J]:= A[J] - B[I]*    (  P:= C[J]/AA[I] );
                 G    := F[J]        - N[J]*UM[J] -
                         SW[J]*UP[I] - S[J]*UP[J] - G*P;
                 ( J<K ! G  -:=                    NE[J]*UM[J+1] );
                 U[ J]:= G;  I:= J
            'OD';
            'FOR' J 'FROM' K 'BY' -1 'TO' L
            'DO' U [J]:= G := ( U[J] - B[J]*G )/AA[J] 'OD'
    'END'
    #$E#;
    #E$#
# DAMPED JACOBI RELAXATION      #                          'PR' EJECT 'PR'

    'PROC' JAC RELAX    = ('NETMAT'M, 'NET' U,F)'VOID':
    #$B RELJAC  B$#
    'BEGIN' # ## APPROX.OF JACOBIAN = 2 * DIAGONAL OF JACOBIAN # ##
       'INT' L1= 1'LWB'U, L2= 2'LWB'U,
             U1= 1'UPB'U, U2= 2'UPB'U;
       'HEAP'[L1:U1,L2:U2]'REAL' UNEW;
       'REF'[ ]'REAL' UIM:= U[L1,@L2], UI, UIP:= U[L1,@L2];
       'FOR' I 'FROM' L1 'TO' U1
       'DO' ( UI:= UIP; I = U1 ! 'SKIP' ! UIP:= U[I+1,@L2] );
            'REF' []'REAL' FI = F[I,@L2];
            'REF'[,]'REAL' MI:= M[I,@L2,@-3];

            'INT' JM:= L2, JJ, JP:= L2;
            'FOR' J 'FROM' L2 'TO' U2
            'DO' ( JJ:= JP; J=U2 ! 'SKIP' ! JP+:= 1 );

                 'REF'[]'REAL' MIJ = MI[JJ,@-3];
                 UNEW[I,JJ]:=     ( MIJ[-3]*UIM[JJ] + MIJ[-2]*UIM[JP] +
                 MIJ[-1]*UI [JM]  - MIJ[ 0]*UI [JJ] + MIJ[ 1]*UI [JP] +
                 MIJ[ 2]*UIP[JM]  + MIJ[ 3]*UIP[JJ]
                                                - FI [JJ])/(-2.0*MIJ[0]) ;
                 JM:= JJ
            'OD';
            UIM:= UI
       'OD' ;
```

```
      U:= UNEW
    'END'
    #$E#;
    #E$#
# ILU RELAXATION                #                        'PR' EJECT 'PR'

    'PROC' ILU RELAX    = ('REF''NETMAT' M,'NETMAT' A,'NET' U,F)'VOID':
    #$B RELILUX B$#
    'BEGIN' ( M :=: 'NETMAT'('NIL') !
              M :=  ILUDECOMP( A )  );
            # ## THE NOT EFFICIENT VERSION !!! # ##
            'NET' X = RESIDUAL (A,U,F);

       'INT' L1= 1'LWB'X, L2= 2'LWB'X,
             U1= 1'UPB'X, U2= 2'UPB'X;
       'INT' L1P:=L1, U1M:=U1, L2P:=L2, U2M:=U2;
       'REF'[]'REAL' XIM:= X[L1,@L2],XI,XIP:=X[U1,@L2];
       'FOR' I 'FROM' L1P 'TO' U1M
       'DO'  XI:= X[I,@L2];
             'REF'[,]'REAL' MI= M[I,@L2,@-3];
             'INT' JM:= L2, JP:=L2P;
             'FOR' J 'FROM' L2P 'TO' U2M
             'DO'  ( J<U2 ! JP+:=1 );
                   'REF'[]'REAL' MIJ= MI[J,@-3];
                   XI[J]:=              ( MIJ[-3]*XIM[J]+MIJ[-2]*XIM[JP]+
                        MIJ[-1]*XI [JM]    -    XI [J] )/   -MIJ[0];
                   JM:= J
             'OD' ;
             XIM:= XI
       'OD'  ;
       'FOR' I 'FROM' U1M 'BY' -1 'TO' L1P
       'DO'  XI:= X[I,@L2];
             'REF'[,]'REAL' MI= M[I,@L2,@-3];
             'INT' JM:= U2M, JP:= U2;
             'FOR' J 'FROM' U2M 'BY' -1 'TO' L2P
             'DO'  ( J>L2 ! JM-:=1 );
                   'REF'[]'REAL' MIJ= MI[J,@-3];
                   XI[J]:=           -(        -XI [J]+MIJ[ 1]*XI [JP]+
                        MIJ[ 2]*XIP[JM]+MIJ[ 3]*XIP[J]     );
                   JP:= J
             'OD' ;
             XIP:= XI
       'OD'  ;
       'FOR' I 'FROM' L1P 'TO' U1M 'DO'
       'FOR' J 'FROM' L2P 'TO' U2M 'DO'
             U[I,J]+:= X[I,J]  'OD' 'OD'
    'END'
    #$E#;
    #E$#
# PREPARATION OF ILU RELAXATION #                        'PR' EJECT 'PR'

    'PROC' ILU DECOMP   = ('NETMAT' A )'NETMAT':
    #$B ILUDEC  B$#
    'BEGIN' 'NETMAT' M = A[@1,@1,@-3];
            'INT' II = 1'UPB'M, JJ= 2'UPB'M;
            'HEAP'[1:II,1:JJ,-3:3]'REAL' AA; 'ZERO' AA;

            'REF' [,]'REAL' MM3 = M[,,-3],  L3 = AA[,,-3],
                            MM2 = M[,,-2],  L2 = AA[,,-2],
                            MM1 = M[,,-1],  L1 = AA[,,-1],
                            M0  = M[,, 0],  L0 = AA[,, 0],

                            M1  = M[,, 1],  U1 = AA[,, 1],
                            M2  = M[,, 2],  U2 = AA[,, 2],
                            M3  = M[,, 3],  U3 = AA[,, 3];

            'REF' [ ]'REAL' MM11 = MM1[1,],     L11 = L1[1,],
                            M01  = M0 [1,],     L01 = L0[1,],
                            M11  = M1 [1,@2],   U11 = U1[1,@2];

            'REAL' L01JM:= L01[1]:= M01[1];

            'FOR' J 'FROM' 2 'TO' JJ
            'DO'   L01JM := L01[J]:= M01[J] -
                 ( L11[J]:= MM11[J] ) * .
                 ( U11[J]:= M11 [J] / L01JM # ## I.E. L01[J-1]# ## )
            'OD';

            'FOR' I 'FROM' 2 'TO' II
            'DO''INT' IM = I-1;
                'REF' [ ]'REAL'
                 M0I  = M0 [I,], L0I = L0[I,], L0IM = L0[IM,],
                 MM1I = MM1[I,], L1I = L1[I,], L1IM = L1[IM,@0],
                 MM2I = MM2[I,], L2I = L2[I,],
                 MM3I = MM3[I,], L3I = L3[I,],
                 M1I  = M1[I ,],
                 M1IM = M1[IM,], U1I = U1[I,], U1IM = U1[IM,  ],
                 M2IM = M2[IM,@0],             U2IM = U2[IM,@0],
                 M3IM = M3[IM,  ],             U3IM = U3[IM,  ];

                'FOR' J 'TO' JJ-1
                'DO'  L2I[J]  := MM2I[J] - U1IM[J] *
                    ( L3I[J]  := MM3I[J]           );
                      U2IM[J] := ( M2IM[J] - L1IM[J] *
                    ( U3IM[J] := M3IM[J] / L0IM[J] ) )/L0IM[J+1]
                'OD';

                L3I [JJ]:= MM3I[JJ];
                U3IM[JJ]:= M3IM[JJ]/ L0IM[JJ];
                L0I [ 1]:= M0I [ 1] - L3I[1]*U3IM[1] - L2I[1]*U2IM[1];

                'FOR' J 'FROM' 2 'TO' JJ
                'DO' 'INT' JM  = J - 1;
                     L0I[J]  := M0I [J] -
                    ( L1I[J]  := MM1I[J] - L3I[J]*U2IM[JM] )*
                    ( U1I[JM] :=(M1I[JM] - L2I[JM]*U3IM[J])/L0I[JM] )
                                - ( J=JJ ! 0.0 ! L2I[J]*U2IM[J] )
                                -               L3I[J]*U3IM[J]
                'OD'
            'OD';
            AA[@(1'LWB'A),@(2'LWB'A),@-3]
    'END'
    #$E#;
    #E$#
# ILLU RELAXATION               #                        'PR' EJECT 'PR'

   'PROC' SOLL = ('INT' I, 'NETMAT' DEC, 'NET' R)'VOID':
    #$B SOLL    B$#
   'BEGIN' 'REF'[]'REAL' L = DEC[I, ,-1],
                         D = DEC[I, , 0],
                         U = DEC[I, , 1],
                         Z = R  [I, ];
          'INT' LL = 'LWB' Z, UU = 'UPB' Z;

     'FOR' J 'FROM' LL+1          'TO' UU 'DO' Z[J]+:= L[J]*Z[J-1] 'OD';
     'FOR' J 'FROM' LL            'TO' UU 'DO' Z[J]/:= D[J]        'OD';
     'FOR' J 'FROM' UU-1 'BY' -1 'TO' LL 'DO' Z[J]+:= U[J]*Z[J+1] 'OD'
   'END'
```

```
  #$E#;
  #E$#

'PROC' ILLU RELAX = ('REF''NETMAT' DEC,'NETMAT'JAC, 'NET' U,F)'VOID':
 #$B ILLUR   B$#
'BEGIN' 'INT' L1= 1'LWB'U , U1= 1'UPB'U, L2= 2'LWB'U, U2= 2'UPB'U;
        ( 'NETMAT'(DEC) :=: 'NETMAT'('NIL') ! ILLUDEC (JAC,DEC) );
        [L1:U1,L2:U2]'REAL' DU,RH;

        RH:= RESIDUAL(JAC,U,F);
        SOLL(L1,DEC,RH);
        'FOR' I 'FROM' L1+1 'TO' U1
        'DO' 'FOR' J 'FROM' L2 'TO' U2
             'DO' RH[I,J]-:= JAC[I,J,-3]*RH[I-1,J  ] +
                    ( J<U2 ! JAC[I,J,-2]*RH[I-1,J+1] ! 0.0 )
             'OD';
             SOLL(I,DEC,RH)
        'OD';
        DU[U1,]:=RH[U1,];
        'FOR' I 'FROM' U1-1 'BY' -1 'TO' L1
        'DO' 'FOR' J 'FROM' L2 'TO' U2
             'DO' DU[I,J] := JAC[I,J, 3]*DU[I+1,J  ] +
                    ( J>L2 ! JAC[I,J, 2]*DU[I+1,J-1] ! 0.0 )
             'OD';
             SOLL(I,DEC,DU);
             'FOR' J 'FROM' L2 'TO' U2
             'DO' DU[I,J] := RH[I,J] - DU[I,J] 'OD'
        'OD';

        'FOR' I 'FROM' L1 'TO' U1 'DO'
        'FOR' J 'FROM' L2 'TO' U2 'DO'
              U[I,J]+:= DU[I,J]
        'OD' 'OD'
'END'
 #$E#;
 #E$#

'PROC' ILLUDEC = ('NETMAT' JAC, 'REF''NETMAT' DECOMP )'VOID':
 #$B ILLUD   B$#
'BEGIN' 'INT' L1= 1'LWB'JAC, U1= 1'UPB'JAC,
              L2= 2'LWB'JAC, U2= 2'UPB'JAC;
        'INT' IP;
        'REAL' DD,LL,II,L DINV U;
        [L2:U2,-1:+1]'REAL' D;
        [L2:U2,-2:+2]'REAL' DINV;
        [L2:U2,-1:+2]'REAL' L DINV;
        'HEAP'[L1:U1,L2:U2,-1:+1]'REAL' DEC;

        D[L2:U2,-1:+1]:= JAC[L1,L2:U2,-1:+1];

        DD:= DEC[L1,L2,0]:= D[L2,0];
        'FOR' J 'FROM' L2 'TO' U2-1
        'DO' DEC[L1,J  ,+1]:=     -D[J  ,+1]/DD;
             DEC[L1,J+1,-1]:= LL:=-D[J+1,-1]/DD;
             DEC[L1,J+1, 0]:= DD:= D[J+1, 0] + D[J,1]*LL
        'OD';

        'FOR' I 'FROM' L1 'TO' U1-1
        'DO'  IP:= I+1;
              DINV[U2,0]:= II:= 1.0/DEC[I,U2,0];
             'FOR' J 'FROM' U2-1 'BY' -1 'TO' L2
             'DO' DINV[ J,0]:= II:= 1.0/DEC[I, J,0] +
                               II * DEC[I,J,1]*DEC[I,J+1,-1]
             'OD';
             'FOR' K 'TO' 2 'DO'
             'FOR' J 'FROM' U2 'BY'-1 'TO' L2+K 'DO'
                   DINV[J  ,-K]:= DINV[J    ,1-K]*DEC[I,J-K+1,-1];
                   DINV[J-K, K]:= DINV[J-K+1,K-1]*DEC[I,J-K  ,+1]
             'OD' 'OD';

             'FOR' K 'FROM'           -1  'TO'           2    'DO'
             'FOR' J 'FROM' L2+(K=-1!1!0) 'TO' U2-(K=2!2!1)
             'DO'  L DINV[J ,K]:= JAC[IP,J ,-3]*DINV[J  ,K  ] +
                                  JAC[IP,J ,-2]*DINV[J+1,K-1]
             'OD';
                                                              ( K<1 !
                   L DINV[U2,K]:= JAC[IP,U2,-3]*DINV[U2 ,K  ]       )
             'OD';

             'FOR' K 'FROM'           -1  'TO'           1    'DO'
             'FOR' J 'FROM' L2+(K=-1!1!0) 'TO' U2-(K=1!1!0)
             'DO'  L DINV U := L DINV[J,K  ]*JAC[I,J+K  ,3];
                                                          (J+K<U2 !
                   L DINV U+:= L DINV[J,K+1]*JAC[I,J+K+1,2]        );
                   D[J,K]    :=   JAC[IP,J,K] - L DINV U
             'OD' 'OD';

             DD:= DEC[IP,L2,0]:= D[L2,0];
             'FOR' J 'FROM' L2 'TO' U2-1
             'DO' DEC[IP,J  ,+1]:=     -D[J  ,+1]/DD;
                  DEC[IP,J+1,-1]:= LL:=-D[J+1,-1]/DD;
                  DEC[IP,J+1, 0]:= DD:= D[J+1, 0] + D[J,1]*LL
        'OD' 'OD';
        DECOMP:= DEC
  'END'
   #$E#;
   #E$#

# OTHER POINTWISE RELAXATION    #                        'PR' EJECT 'PR'
#
   'INT' TH RELAX STRATEGY := 1;
#
   'PROC' TH RELAX     = ('NETMAT'M, 'NET' U,F)'VOID':
   #$B RELTH   B$#
   'BEGIN' 'INT' L1= 1'LWB'U, L2= 2'LWB'U,
                 U1= 1'UPB'U, U2= 2'UPB'U,
                 START= 'ABS' TH RELAX STRATEGY;

      'FOR' COLOR 'FROM' START
                  'BY' 'SIGN' TH RELAX STRATEGY
                  'TO'  START + 2
      'DO''REF'[ ]'REAL' UIM:= U[L1,@L2], UI, UIP:= U[L1,@L2];
          'FOR' I 'FROM' L1 'TO' U1
          'DO' ( UI:= UIP; I = U1 ! 'SKIP' ! UIP:= U[I+1,@L2] );
               'REF' []'REAL' FI = F[I,@L2];
               'REF'[,]'REAL' MI:= M[I,@L2,@-3];

               'INT' JM:= L2, JP:= U2;
               'FOR' J 'FROM' L2+((COLOR+I)'MOD'3)'BY' 3 'TO' U2
               'DO' ( J=L2 ! 'SKIP' ! JM:= J-1 );
                    ( J=U2 ! 'SKIP' ! JP:= J+1 );

                   'REF'[]'REAL' MIJ = MI[J,@-3];
                UI[J]  :=        ( MIJ[-3]*UIM[J] + MIJ[-2]*UIM[JP] +
                MIJ[-1]*UI [JM]  -         FI [J] + MIJ[ 1]*UI [JP] +
                MIJ[ 2]*UIP[JM]  + MIJ[ 3]*UIP[J] )/   -MIJ[ 0]
               'OD';
```

```
                UIM:= UI
      'OD''OD'
   'END'
   #$E#;
   #E$#
                                                    'PR' EJECT 'PR'
#  'PROC' FOUR COLOR RELAX = 'REF'[,]'INT':
        'HEAP'[1:4,1:2]'INT' := ((0,1),(1,0),(1,1),(0,0));
   'REF' [,]'INT' CH RELAX STRATEGY:= FOUR COLOR RELAX;
#
   'PROC' CH  RELAX    = ('NETMAT'M, 'NET' U,F)'VOID':
   #$B RELCH   B$#
   'BEGIN' 'INT' L1= 1'LWB'U, L2= 2'LWB'U,
                 U1= 1'UPB'U, U2= 2'UPB'U;
           [L2:U2]'REAL' DUMMY; 'ZERO' DUMMY;

      'FOR' S 'TO' 1'UPB' CH RELAX STRATEGY
      'DO' 'INT' C1 = CH RELAX STRATEGY[S,1],
                 C2 = CH RELAX STRATEGY[S,2];
          'REF'[ ]'REAL' UIM, UI, UIP;
          'FOR' I 'FROM' L1+C1 'BY' 2 'TO' U1
          'DO'  UI:= U[I,@L2];
                UIM:= ( I=L1 ! DUMMY ! U[I-1,@L2]);
                UIP:= ( I=U1 ! DUMMY ! U[I+1,@L2]);
               'REF' []'REAL' FI = F[I,@L2];
               'REF'[,]'REAL' MI:= M[I,@L2,@-3];

               'INT' JM:= L2, JP:= L2;
               'FOR' J 'FROM' L2+C2 'BY' 2 'TO' U2
               'DO' ( J=U2 ! 'SKIP' ! JP:= J+1 );

                    'REF'[]'REAL' MIJ = MI[J,@-3];
                UI[J] :=        ( MIJ[-3]*UIM[J] + MIJ[-2]*UIM[JP] +
                MIJ[-1]*UI [JM]  -         FI [J] + MIJ[ 1]*UI [JP] +
                MIJ[ 2]*UIP[JM]  + MIJ[ 3]*UIP[J] )/   -MIJ[ 0]     ;

                    JM:= JP
               'OD';
               UIM:= UIP
      'OD''OD'
   'END'
   #$E#;
   #E$#
# GRID STYLE PROCEDURES          #                    'PR' EJECT 'PR'

   'PROC' SOLVE DIRECTLY = ('REF''GRID' U, 'GRID' F)'VOID':
   #$B SOLVED  B$#
         SOLVE (JACOBIAN[LEVEL'OF'F],NET'OF'U,NET'OF'F)
   #$E#;
   #E$#

   'OP' 'NORM' = (             'GRID'A) []'REAL':
   #$B NRMG    B$#
                 'TRUE''NORM'A
   #$E#;
   #E$#
   'OP' 'NORM' = ('BOOL' TRUE, 'GRID'A) []'REAL':
   #$B NRMGRID B$#
                 TRUE 'NORM'NET'OF'A
   #$E#;
   #E$#
   'OP''ERRORNORM'     = ('GRID'   U )[]'REAL':
   #$B NRMGERR B$#
                      'TRUE'  'NORM' (U- SOLUTION)
   #$E#;
   #E$#
   'OP''RESIDUALNORM' = ('GRID' FG, UG )[]'REAL':
   #$B NRMRESD B$#
        RESIDUALNORM(JACOBIAN[LEVEL'OF'FG],NET'OF'FG,NET'OF'UG)
   #$E#;
   #E$#

   'OP' *  = ('NETMAT' M, 'GRID' U)'NET':
   #$B NET**   B$#
           M*NET'OF'U
   #$E#;
   #E$#

   'PROC' RESIDUAL GRID= ('GRID' U,F )'GRID':
   #$B RESGR   B$#
                       F - 'LH' U
   #$E#;
   #E$#

   'OP' 'LH' = ('GRID' U)'GRID':
   #$B LH      B$#
               (LEVEL'OF'U, JACOBIAN[LEVEL'OF'U]*(NET'OF'U))
   #$E#;
   #E$#


# CENTRAL PROCEDURES GRID STYLE #                        'PR' EJECT 'PR'

   # ********************************************************** #
   # *********    GLOBAL PARAMETERS FOR GRID STYLE         ******** #
   # ********************************************************** #
#
   'PROC' PROLONGATE := ('GRID' A)'GRID':
                        (LEVEL'OF'A+1,LIN INT POL(NET'OF'A)) ;
   'PROC' INTERPOLATE:= ('GRID' A)'GRID':
                        (LEVEL'OF'A+1,SQR INT POL(NET'OF'A)) ;
   'PROC' RESTRICTBAR:= ('GRID' A)'GRID':
                        (LEVEL'OF'A-1,LIN WEIGHT(NET'OF'A))  ;
   'PROC' RESTRICT   := ('GRID' A)'GRID':
                        (LEVEL'OF'A-1,INJECTION (NET'OF'A))  ;

   'PROC' RELAX      := ('REF''GRID' U, 'GRID' F)'VOID':
          ('INT' L = LEVEL'OF' F; ( L /= LEVEL'OF'U ! ERROR);
            ILU! ILURELAX (DECOMP[L], JACOBIAN[L],
                                      NET'OF'U , NET'OF'F   )
                ! LGSRELAX ( JACOBIAN[L],NET'OF'U,NET'OF'F)   );
#
   'PROC' MG FAS = ('INT' L, 'REF'[]'GRID' U,F)'VOID':
   #$B MGFAS   B$#
   'IF'   L = 0
   'THEN' SOLVE DIRECTLY (U[0],F[0])
                                                        ;REP(0,0)
   'ELSE' 'TO' P 'DO' RELAX(U[L],F[L]) 'OD';
                                                         REP(1,L);
          'GRID' Y = RESTRICT ( U[L] );
           # ## OR  Y = 'COPY' U[L-1]; # ##
           F[L-1]:= 'LH' Y + RESTRICTBAR(
                            RESIDUAL GRID ( U[L], F[L] ))/MU;
                                                         REP(2,L-1);
          'TO' ( L=1 ! 1 ! S ) 'DO' MG FAS (L-1,U,F) 'OD';
           U[L] +:= MU * PROLONGATE (U[L-1]-Y);
                                                         REP(3,L);
```

```
        'TO' Q 'DO' RELAX(U[L],F[L]) 'OD'
                                                ;REP(4,L)
'FI'
#$E#;
#ES#

'PROC' MG CS= ('INT' L, 'REF'[]'GRID' U,F)'VOID':
#$B MGCS    B$#
'IF'   L = 0
'THEN' SOLVE DIRECTLY (U[0],F[0])
                                                ;REP(0,0)
'ELSE' 'TO' P 'DO' RELAX(U[L],F[L]) 'OD';
                                                 REP(1,L);
       'IF'   S > 0
       'THEN' F[L-1]:= RESTRICTBAR( RESIDUAL GRID( U[L], F[L] ));
              'ZERO' U[L-1];
                                                 REP(2,L-1);
            'TO' (L=1 ! 1 ! S ) 'DO' MG CS(L-1,U,F) 'OD';
            U[L] +:= PROLONGATE (U[L-1])
                                                ;REP(3,L)
                                               'FI';
       'TO' Q 'DO' RELAX(U[L],F[L]) 'OD'
                                                ;REP(4,L)
'FI'
#$E#;
#ES#

'PROC' FMGG     =('PROBLEM' PROBLEM, 'REF'[]'GRID' U,F)'VOID':
#$B FMGG    B$#
'BEGIN' # ## EXPECTS [0:M]'GRID' U; U[0] = STARTAPPROX.    # ##
        'BOOL' LIN = LINEAR'OF'PROBLEM;
        'INT'  L   = 'UPB'U;
        [1:3]'REAL' RES;
        JACOBIAN:= 'HEAP'[0:L]'NETMAT';
        DECOMP  := 'HEAP'[0:L]'NETMAT';
                   'FOR' I 'FROM' 0 'TO' L
                   'DO' LEVEL'OF'F[I]:= I;
                        DECOMP    [I]:= 'NETMAT'('NIL')
                   'OD';

        DISCRETIZATION (PROBLEM, U[0],NET'OF'F[0],JACOBIAN[0]);
        SOLVE DIRECTLY (U[0],F[0]);
        GOON FMGG ( U[0], F[0], RES );
                                                 REP (12,0);
        'FOR' K 'TO' L
        'DO' U[K]:= INTERPOLATE (U[K-1]);
                                                 REP (11,K);
            DISCRETIZATION (PROBLEM,U[K],NET'OF'F[K],JACOBIAN[K]);
            ( PR >= 0
            ! DIRICHLET RELAX(PROBLEM,U[K],JACOBIAN[K],NET'OF'F[K]);
              'TO' PR 'DO' RELAX(U[K],F[K]) 'OD'
            );
                                                 REP (13,K);
            'TO' R  'WHILE'
                 ( LIN ! MG CS (K,U,F) ! MG FAS (K,U,F));
                 GOON FMGG ( U[K], F[K], RES )
            'DO' 'SKIP' 'OD'
                                                ;REP (12,K)
       'OD'
'END'
#$E#;
#ES#

'PROC' GOON FMGG1    = ('GRID' U,F, 'REF'[]'REAL' RES) 'BOOL':
   #$B FMGG1   B$#
   'BEGIN' 'INT' K = LEVEL'OF'U;
           'GRID'  RESID = RESIDUAL GRID (U,F);
           []'REAL' RESN = 'FALSE''NORM' RESID;
           PRINT((NEWLINE," LEVEL ",WHOLE(K,0)));
           FL('ERRORNORM'U);FL(RESN);
           ( K > 0 ! FX(RES,RESN) );
           RES:= RESN; RESN[3] > 1.0E-12
   'END'
   #$E#;
   #ES#
# CENTRAL PROCEDURES NEW STYLE  #                        'PR' EJECT 'PR'

   # ************************************************************ #
   # *********    GLOBAL PARAMETERS FOR THE MG-ALGORITHM   ******** #
   # ************************************************************ #
#
   'BOOL' ILU:= 'FALSE';
   'INT' P:= 1, S:= 2, Q:= 1, R, PR:= 0; R:= S; 'REAL'MU:= 1.0;

   'OP'  'I'   = ('NET' N)'NET': SQR INT POL(N);
   'OP'  'P'   = ('NET' N)'NET': LIN INT POL(N);
   'OP'  'R'   = ('NET' N)'NET': INJECTION  (N);
   'OP'  'RBAR'= ('NET' N)'NET': LIN WEIGHT (N);

   'PROC' RELAX NET :=
         ('REF''NETMAT' D,'NET' M, 'NET'U,F )'VOID': LGS RELAX(M,U,F);
#
   'PROC' MLA FAS= ('INT'L,'REF'[]'NETMAT'D,J,'REF'[]'NET' U,F)'VOID':
   #$B MLAFAS  B$#
   'IF'   L = 0
   'THEN' SOLVE (J[0],U[0],F[0])                        ;REP(0,0)
   'ELSE' 'TO'P 'DO' RELAX NET(D[L],J[L],U[L],F[L])'OD';REP(1,L);
          'NET' Y = 'R' U[L];  # ## OR 'COPY' U[L-1] # ##
           F[L-1]:= J[L-1]*U[L-1] +
                    'RBAR' (RESIDUAL (J[L],U[L],F[L])/MU);
                                                        REP(2,L-1);
          ( L=1
          ! SOLVE (J[0],U[0],F[0])
          ! 'TO' S 'DO' MLA FAS(L-1,D,J,U,F) 'OD'
          );
           U[L] +:= MU * 'P' (U[L-1]-Y);                REP(3,L);

          'TO'Q 'DO' RELAX NET(D[L],J[L],U[L],F[L])'OD';REP(4,L)
   'FI'
   #$E#;
   #ES#

   'PROC' MLA CS = ('INT'L,'REF'[]'NETMAT'D,J,'REF'[]'NET' U,F)'VOID':
   #$B MLACS   B$#
   'IF'   L = 0
   'THEN' SOLVE (J[0],U[0],F[0])                        ;REP(0,0)
   'ELSE' 'TO'P 'DO' RELAX NET(D[L],J[L],U[L],F[L])'OD';REP(1,L);
          F[L-1]:= 'RBAR' RESIDUAL (J[L],U[L],F[L]);
          'ZERO' U[L-1];                                REP(2,L-1);
          ( L=1
          ! SOLVE (J[0],U[0],F[0])
          ! 'TO' S 'DO' MLA CS (L-1,D,J,U,F) 'OD'
          );

          U[L] +:= 'P' U[L-1];                          REP(3,L);
          'TO'Q 'DO' RELAX NET(D[L],J[L],U[L],F[L])'OD';REP(4,L)
   'FI'
   #$E#;
```

```
    #E$#
                                                    'PR' EJECT 'PR'

    'PROC' FMG      =('PROBLEM' PROBLEM, 'REF'[]'GRID' UG)'VOID':
    #$B FMG      B$#
     BEGIN' # ## EXPECTS [0:M]'GRID' UG; UG[0]= STARTAPPROX. # ##
            'BOOL' LIN = LINEAR'OF'PROBLEM;
            'INT'  L   = 'UPB'UG;
            [1:3]'REAL' RES;
            [0:L]'NETMAT' JAC, DEC;
                       'FOR' I 'FROM' 0 'TO' L
                       'DO' DECOMP[I]:= 'NETMAT'('NIL') 'OD';
            [0:L]'NET' F; 'REF'[]'NET' U= NET'OF'UG;

            DISCRETIZATION(PROBLEM, UG[0],F[0],JAC[0]);
            SOLVE (JAC[0],U[0],F[0]);
            GOON FMG (JAC[0], U[0], F[0], RES );
                                                    REP (12,0);

           'FOR' K 'TO' L
           'DO' UG[K]:= 'GRID'(K,'I' U[K-1]);
                                                    REP (11,K);
                DISCRETIZATION (PROBLEM,UG[K],F[K],JAC[K]);
                ( PR >= 0
                ! DIRICHLET RELAX (PROBLEM,UG[K],JAC[K],F[K]);
                  'TO'PR'DO' RELAX NET(DEC[K],JAC[K],U[K],F[K])'OD'
                );
                                                    REP (13,K);
                'TO' R  'WHILE'
                     ( LIN ! MLA CS (K,DEC,JAC,U,F)
                           ! MLA FAS(K,DEC,JAC,U,F));
                     GOON FMG (JAC[K], U[K], F[K], RES )
                'DO' 'SKIP' 'OD'
                                                    ;REP (12,K)
           'OD'
    'END'
    #$E#;
    #E$#
# NAG CONTRIBUTION : THE MORE EFFICIENT VERSION  #     'PR' EJECT 'PR'

    'PROC' ADD INT POL = ('NET' FINE,NET )'VOID':
    #$B AIP      B$#
    'BEGIN' 'INT' L1 = 1'LWB'NET,  L2 = 2'LWB'NET,
                  B1 = 1'UPB'NET,  B2 = 2'UPB'NET;
            'INT' JJ; 'REAL' U2,U3,U4;
                 'REF'[]'REAL' UIP= NET[L1,@L2],
                               UPP= FINE[2*L1,@2*L2];
                  JJ:= 2*L2; UPP[JJ]+:=(U4:=UIP[L2]);
                  'FOR' JP 'FROM' L2+1 'TO' B2
                  'DO' U3:= U4; U4:= UIP[JP];
                       UPP[JJ+:=1]+:= (U3+U4)/2;
                       UPP[JJ+:=1]+:=     U4
                  'OD';
            'FOR' IP 'FROM' L1+1 'TO' B1
            'DO' 'REF'[]'REAL' UI  = NET [IP-1  ,@  L2],
                               UIP = NET [IP    ,@  L2],
                               UMM = FINE[2*IP-1,@2*L2],
                               UPP = FINE[2*IP  ,@2*L2];
                 JJ:= 2*L2; U2:= UI[L2]; U4:= UIP[L2];
                 UMM[JJ]:= (U2+U4)/2;  UPP[JJ]:= U4;
                 'FOR' JP 'FROM' L2+1 'TO' B2
                 'DO' JJ+:= 1;  U2:= UI [JP];
                      U3:= U4;  U4:= UIP[JP];
                      UMM[JJ]+:= (U2+U3)/2;
                      UPP[JJ]+:= (U3+U4)/2;
                      JJ+:= 1;
                      UMM[JJ]+:= (U2+U4)/2;
                      UPP[JJ]+:=     U4
                 'OD'
           'OD'
    'END'
    #$E#;
    #E$#
                                                    'PR' EJECT 'PR'

    'PROC' WEIGHT  RES = ('NETMAT'M,'NET'U,F,COARSE)'VOID':
    #$B WR       B$#
    'BEGIN' 'INT' L1 = (1'LWB'F), L2 = (2'LWB'F),
                  U1 = (1'UPB'F), U2 = (2'UPB'F);
            'INT' NL1 = L1'OVER'2,  NL2 = L2'OVER'2,
                  NU1 = U1'OVER'2,  NU2 = U2'OVER'2;
            'BOOL' U IS ZERO = 'FALSE';
            'INT' II,IM,IP, JJ,JM,JP;
            [L2:U2]'REAL' SIM,SII,SIP,ZERO; 'ZERO' ZERO; 'ZERO' SIM;

      'FOR' I 'FROM' NL1 'TO' NU1
      'DO' II:=  2*I; IP:= II+1; IM:= II-1;
         'REF'[]'REAL' CI = COARSE[I,@NL2];
         'IF' U IS ZERO
         'THEN' SII:= F[II,@L2]; SIP:= F[IP,@L2]
         'ELSE'
           'FOR' I 'FROM' II 'TO' IP
           'WHILE' I <= U1
           'DO' 'REF' []'REAL' UIM = (I=L1!ZERO!U[I-1,@L2]),
                               UI  =           U[I  ,@L2] ,
                               UIP = (I=U1!ZERO!U[I+1,@L2]),
                               SI  = (I=II!SII!SIP),
                               FI  = F[I,@L2];
                'REF'[,]'REAL' MI = M[I,@L2,@-3];

                'INT' JM, JJ:= L2, JP:= L2;
                'FOR' J 'FROM' L2 'TO' U2
                'DO' (JM:= JJ; JJ:= JP; J=U2 ! 'SKIP' ! JP+:= 1 );

                     'REF'[]'REAL' MIJ = MI[JJ,@-3];
                     SI[JJ]:=
                     FI[JJ] -      ( MIJ[-3]*UIM[JJ]+MIJ[-2]*UIM[JP]+
                     MIJ[-1]*UI [JM]+MIJ[ 0  UI [JJ]+MIJ[ 1]*UI [JP]+
                     MIJ[ 2]*UIP[JM]+MIJ[ 3]*UIP[JJ] )
                'OD'
           'OD'
         'FI';

           JM:= JP:= L2+1;
           CI[NL2]:=     (         SIM[L2]+SIM[JP]
                                   +SII[JP]
                                  +SIP[L2]        )*0.5 + SII[L2];
           'FOR' J 'FROM' NL2+1 'TO' NU2-1
           'DO'  JM:= JP; JJ:= JM+1; JP:= JJ+1;
                 CI[J]:= (         SIM[JJ]+SIM[JP]
                           +SII[JM]        +SII[JP]
                           +SIP[JM]+SIP[JJ]        )*0.5 + SII[JJ]
           'OD';
           CI[NU2]:=     (         SIM[U2]
                           +SII[JM]
                           +SIP[JM]+SIP[U2]        )*0.5 + SII[U2];
           SIM:= SIP
      'OD'
```

```
'END'
#$E#;
#E$#
                                                    'PR' EJECT 'PR'

'PROC' REDUCE = ('NETMAT' AFI,'REF''NETMAT' ACO,
                 'NET' FFI,'REF''NET' FCO,UCO)'VOID':
#$B REDUCE  B$#
'BEGIN' 'INT' L1 = (1'LWB'AFI)'OVER'2, U1 = (1'UPB'AFI)'OVER'2,
              L2 = (2'LWB'AFI)'OVER'2, U2 = (2'UPB'AFI)'OVER'2;
              ACO:='HEAP' [L1:U1,L2:U2,-3:3]'REAL';
              FCO:='HEAP' [L1:U1,L2:U2     ]'REAL';
              UCO:='HEAP' [L1:U1,L2:U2     ]'REAL'; 'ZERO' UCO;
    'REAL' Q= 0.25;
    'INT' TI,TIP,TK,TKP;
    'REAL' FFB,FFD,FFE;

    [1:3,1:3,-3:3]'REAL' FINE;
    'REF'[]'REAL'
           A = FINE[1,1,@-3], B = FINE[1,2,@-3], C = FINE[1,3,@-3],
           D = FINE[2,1,@-3], E = FINE[2,2,@-3], F = FINE[2,3,@-3],
           G = FINE[3,1,@-3], H = FINE[3,2,@-3], J = FINE[3,3,@-3];
    'REF''REAL'
           AA =A[ 0], AB =A[ 1], AD =A[ 3],
           BA =B[-1], BB =B[ 0], BC =B[ 1], BD =B[ 2], BE =B[ 3],
           CB =C[-1], CC =C[ 0], CE =C[ 2], CF =C[ 3],
           DA =D[-3], DB =D[-2], DD =D[ 0], DE =D[ 1], DG =D[ 3],
           EB =E[-3], EC =E[-2], ED =E[-1], EE =E[ 0],
                                 EF =E[ 1], EG =E[ 2], EH =E[ 3],
           FC =F[-3], FE =F[-1], FF =F[ 0], FH =F[ 2], FJ =F[ 3],
           GD =G[-3], GE =G[-2], GG =G[ 0], GH =G[ 1],
           HE =H[-3], HF =H[-2], HG =H[-1], HH =H[ 0], HJ =H[ 1],
           JF =J[-3], JH =J[-1], JJ =J[ 0];

    'ZERO' FCO[L1,]; 'ZERO' ACO[ L1,  ,];

    'FOR' I 'FROM' L1 'TO' U1-1
    'DO' TI:= I+I; TIP:= TI+2;

    FCO[I+1,L2]:= 0; 'ZERO' ACO[I+1,L2,];

    'FOR' K 'FROM' L2 'TO' U2-1
    'DO' TK:= K+K; TKP:= TK+2;
           FFD:= FFI[TI+1,TK  ];
           FFB:= FFI[TI  ,TK+1];
           FFE:= FFI[TI+1,TK+1];
         ((FCO[I  ,K  ]+:= FFD+FFB)*:= 0.5)+:= FFI[TI,TK];
           FCO[I+1,K  ] := FFD+FFE;
           FCO[I  ,K+1]+:= FFB+FFE;
        FINE[1:3,1:3,]:= AFI[TI:TIP,TK:TKP,];
        'REF'[]'REAL' A = ACO[I  ,K  ,@-3],
                      C = ACO[I  ,K+1,@-3],
                      G = ACO[I+1,K  ,@-3],
                      J = ACO[I+1,K+1,@-3];
       # ##AA# ##((A[ 0]+:=
                   (AB+BA+AD+DA)*2+ BB+DD+BD+DB )*:=Q)+:=AA;
       # ##CC# ##  C[ 0]+:= (CE+EC+CB+BC)*2+ EE+BB+BE+EB+EF+FE;
       # ##GG# ##  G[ 0]+:= (GE+EG+GD+DG)*2+ EE+DD+DE+ED+EH+HE;
       # ##JJ# ##  J[ 0] := FH+HF;
       # ##AC# ##( A[ 1]+:= (AB+BC)*2+ BB+BE+DB+DE)*:=Q;
       # ##CA# ##( C[-1]+:= (BA+CB)*2+ BB+EB+BD+ED)*:=Q;
       # ##AG# ##( A[ 3]+:= (AD+DG)*2 +DD+BD+DE+BE)*:=Q;
       # ##GA# ##( G[-3]+:= (DA+GD)*2 +DD+DB+ED+EB)*:=Q;
       # ##GC# ##( G[-2] := (GE+EC)*2 + EE+HE+DE+HF+DB+EF+EB)*:=Q;
       # ##CG# ##( C[ 2] := (EG+CE)*2 + EE+EH+ED+FH+BD+FE+BE)*:=Q;
       # ##GJ# ##  G[ 1] := EH+HF+EF;
       # ##JG# ##  J[-1] := HE+FH+FE;
       # ##CJ# ##  C[ 3] := EH+EF+FH;
       # ##JC# ##  J[-3] := HE+FE+HF
    'OD';
           FFD:= FFI[TI+1,TKP ];
         ((FCO[I  ,U2]+:= FFD)*:=0.5)+:= FFI[TI,2*U2];
           FCO[I+1,U2] := FFD;
        FINE[1:3,1,]:= AFI[TI:TIP,TKP,];
        'REF'[]'REAL' A = ACO[I  ,U2,@-3],
                      G = ACO[I+1,U2,@-3];
       # ##AA# ##((A[ 0]+:= (AD+DA)*2 + DD)*:= Q)+:=AA;
       # ##GG# ##  G[ 0]+:= (GD+DG)*2 + DD;
       # ##GA# ##( G[-3]+:= (GD+DA)*2 + DD)*:=Q;
       # ##AG# ##( A[ 3]+:= (AD+DG)*2 + DD)*:=Q;
           G[-2] := G[ 1]:= 0.0
    'OD';

    'FOR' K 'FROM' L2 'TO' U2-1
    'DO' TK:= K+K; TKP:= TK+2;
           FFB:= FFI[2*U1,TK+1];
         ((FCO[U1,K  ]+:= FFB)*:=0.5)+:= FFI[2*U1,TK];
           FCO[U1,K+1]+:= FFB;
        FINE[1,1:3,]:= AFI[TIP,TK:TKP,];
        'REF'[]'REAL' A = ACO[U1,K  ,@-3],
                      C = ACO[U1,K+1,@-3];
       # ##AA# ##((A[ 0]+:= (AB+BA)*2 + BB)*:= Q)+:=AA;
       # ##CC# ##  C[ 0]+:= (CB+BC)*2 + BB;
       # ##CA# ##( C[-1]+:= (CB+BA)*2 + BB)*:=Q;
       # ##AC# ##( A[ 1]+:= (AB+BC)*2 + BB)*:=Q;
           C[ 2] := C[ 3]:= 0.0
    'OD';
          (FCO[U1,U2]  *:=0.5)+:=FFI[2*U1,2*U2];
       # ##AA# ##(ACO[U1,U2,0]*:=Q)+:=AFI[2*U1,2*U2,0]
  'END'
  #$E#;
  #E$#
# PRELIMINARY NAG MGM ROUTINE   #                    'PR' EJECT 'PR'

       # !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! #
       # EVERYWHERE THE MATRIX DOES NOT DEFINE THE NETMAT M, #
       # !!!!!!!!!!! M SHOULD CONTAIN ZEROES !!!!!!!!!!!!!!!! #
       # !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! #

 'PROC' MGM = ('REF'[]'NETMAT' LH, 'REF'[]'NET' UH,FH,
               'INT' ITMAX,P,Q,S,T, 'REF'[]'NETMAT' ILLU,
               'REF''INT' ITUSED,
               'PROC'('INT','NETMAT','NET','NET')'BOOL' GO ON MGM,
               'PROC'('INT','STRING')'VOID' FAIL)'VOID':
  #$B MGM     B$#
  'BEGIN' 'INT' L='UPB'UH, R = S;
          'REF'[]'NETMAT' ILLUD =
             ( ILLU :=: 'REF'[]'NETMAT'('NIL')
             ! 'LOC'[0:L]'NETMAT' ! ILLU );

    'PROC' MG = ('INT'L)'VOID':
    'IF'   L = 0
    'THEN' ILLU RELAX(ILLUD[0],LH[0],UH[0],FH[0])
    'ELSE' 'TO' P 'DO' ILLU RELAX(ILLUD[L],LH[L],UH[L],FH[L])'OD';
           FH[L-1]:= LIN WEIGHT( RESIDUAL (LH[L],UH[L],FH[L]) );
           # ## WEIGHT  RES (LH[L],UH[L],FH[L],FH[L-1]);   # ##
           'ZERO' UH[L-1];
           'TO' (L=1!T!S) 'DO' MG (L-1) 'OD';
```

```
          UH[L] +:= LIN INT POL ( UH[L-1]);
          # ## ADD INT POL (UH[L],UH[L-1]);                    # ##
          'TO' Q 'DO' ILLU RELAX(ILLUD[L],LH[L],UH[L],FH[L])'OD'
    'FI';

          'INT' ERR =
          (  'LWB' UH /= 0 'OR' 'LWB'FH /= 0 'OR' 'LWB'LH /= 0
                           'OR' 'UPB'FH /= L 'OR' 'UPB'LH /= L ! 1
          !: 'NETMAT' LL = LH[L];
              3'LWB'LL /=-3 'OR' 3'UPB'LL /= 3                 ! 2
          !: S <= 0 'OR' S > 3  'OR' T <= 0                    ! 3
          !: ITMAX<0 'OR' P<0 'OR' Q<0                         ! 4
          !: 'LWB'ILLUD /= 0 'OR' 'UPB'ILLUD /=L               ! 5
          !  0  );
          ( ERR>0 ! FAIL ( ERR," MGM "));

          'IF' ITUSED < 0
          'THEN' # ## CREATE GALERKIN APPROXIMATIONS # ##
                 'FOR' I 'FROM' L 'BY' -1 'TO' 1
                 'DO' REDUCE(LH[I],LH[I-1],FH[I],FH[I-1],UH[I-1])
                 'OD';
                 ITUSED:= 0
          'FI';


          'IF' ITUSED = 0
          'THEN' 'FOR'K 'FROM' 0 'TO' L
                 'DO' ILLUDEC (LH[K],ILLUD[K]) 'OD';

                 # ## APPLY FULL MULTIGRID # ##
                 'TO' T 'DO' MG(0) 'OD';
                 'FOR' K 'TO' L-1
                 'DO'  UH[K]:= SQR INT POL (UH[K-1]);
                       'TO' R  'DO'  MG (K) 'OD'
                 'OD'; UH[L]:= SQR INT POL (UH[L-1]);
                 GOON MGM (ITUSED,LH[L],UH[L],FH[L])
          'FI';

          'TO' ITMAX
          'WHILE'  MG (L); ITUSED +:= 1;
                   GOON MGM (ITUSED, LH[L], UH[L], FH[L])
          'DO' 'SKIP' 'OD'
    'END'
    #$E#;
    #E$#

# BLACK-BOX SOLUTION PROCEDURE  #                          'PR' EJECT 'PR'

    'PROC' RAP FMGG =('PROBLEM' PROBLEM, 'REF'[]'GRID' U,F)'VOID':
    #$B RAPFMGG B$#
    'BEGIN'# ## EXPECTS [0:M]'GRID' U,F;                                  # ##
           # ## AND     U[0]='GRID'( 0,'REF'[L1:U1,L2:U2]'REAL') # ##
           'INT' L = 'UPB'U;
           ( 'NOT' LINEAR'OF'PROBLEM ! FAIL ( 1, "RAP FMGG") );
           ( 'LWB'U  /= 0 'OR' 'LWB'F  /= 0 'OR' 'UPB'F  /= L
                                  ! FAIL ( 2, "RAP FMGG") );

           [0:L]'NETMAT' JAC;

           'NETMAT' N0 = NET'OF'(U[0]); 'INT' TTL = 2**L;
           DISCRETIZATION (PROBLEM,
                'GRID'( L,'LOC'[(1'LWB'N0)*TTL:(1'UPB'N0)*TTL,
                               (2'LWB'N0)*TTL:(2'UPB'N0)*TTL]'REAL'),
                          NET'OF'F[L], JACOBIAN[L]);
               MGM(JAC,NET'OF'U,NET'OF'F,
                   100,1,0,1,1,'NIL','LOC''INT',MGM GOON,FAIL)
    'END'
    #$E#;
    #E$#

# OPERATORS WITH ASYMMETRIC WEIGHTS                #     'PR' EJECT 'PR'
# FOR EXPERIMENTAL PURPOSES.        PWH/830314   #

#  [-3:3]'REAL' WASYM,PASYM,RASYM;
                PASYM[@1]:= (0.5,0.5,0.5,1.0,0.5,0.5,0.5);
                RASYM    := WASYM    := PASYM;
#
    'PROC' WEIGHT = ('NET' NET)'NET':
    #$B RESTRW  B$#
    'BEGIN' 'INT' L1 = (1'LWB'NET), L2 = (2'LWB'NET),
                  U1 = (1'UPB'NET), U2 = (2'UPB'NET);
            'INT' NL1 = L1'OVER'2,  NL2 = L2'OVER'2,
                  NU1 = U1'OVER'2,  NU2 = U2'OVER'2;
            'REAL'M3=WASYM[ 3],M2=WASYM[ 2],M1=WASYM[ 1],
                  W3=WASYM[-3],W2=WASYM[-2],W1=WASYM[-1],W0=WASYM[ 0];
            'HEAP'[NL1:NU1,NL2:NU2]'REAL' COARSE;
            'INT' II,IM,IP, JJ,JM,JP;
            [L2:U2]'REAL' ZERO; 'ZERO' ZERO;
            'FOR' I 'FROM' NL1 'TO' NU1
            'DO'   II:=  2*I; IP:= II+1; IM:= II-1;
                   'REF'[]'REAL' CI = COARSE[I,@NL2],
                                 UIM= ( IM<L1 ! ZERO ! NET[IM,@L2]),
                                 UII=                  NET[II,@L2] ,
                                 UIP= ( IP>U1 ! ZERO ! NET[IP,@L2]);
                   JP:= L2+1;
                   CI[NL2]:=                    W3*UIM[L2]+W2*UIM[JP]
                                               +W0*UII[L2]+M1*UII[JP]
                                               +M3*UIP[L2] ;
                   'FOR' J 'FROM' NL2+1 'TO' NU2-1
                   'DO'  JJ:= 2*J; JP:= JJ+1; JM:= JJ-1;
                         CI[J]:=               W3*UIM[JJ]+W2*UIM[JP]
                                    +W1*UII[JM]+W0*UII[JJ]+M1*UII[JP]
                                    +M2*UIP[JM]+M3*UIP[JJ]
                   'OD';
                   JM:=U2-1;
                   CI[NU2]:=                    W3*UIM[U2]
                                    +W1*UII[JM]+W0*UII[U2]
                                    +M2*UIP[JM]+M3*UIP[U2]
            'OD'; COARSE
    'END'
    #$E#;
    #E$#

# GALERKIN OPERATOR CONSTRUCTION                   #     'PR' EJECT 'PR'

'OP' 'RASP' = ('NETMAT' AFI )'NETMAT':
   #$B RAPW    B$#
'BEGIN' 'INT' L1 = (1'LWB'AFI)'OVER'2, U1 = (1'UPB'AFI)'OVER'2,
              L2 = (2'LWB'AFI)'OVER'2, U2 = (2'UPB'AFI)'OVER'2;
        'HEAP' [L1:U1,L2:U2,-3:3]'REAL' ACO; 'ZERO' ACO;
        'INT' TI,TK,TIM,TKM;
        'REAL'M3=PASYM[-3],M2=PASYM[-2],M1=PASYM[-1],
              W3=PASYM[ 3],W2=PASYM[ 2],W1=PASYM[ 1],W0=PASYM[ 0],
              Q3=RASYM[-3],Q2=RASYM[-2],Q1=RASYM[-1],
              P3=RASYM[ 3],P2=RASYM[ 2],P1=RASYM[ 1],P0=RASYM[ 0];

      [1:3,1:3,-3:3]'REAL' FINE;
```

```
'REF'[]'REAL'
       A = FINE[1,1,@-3], B = FINE[1,2,@-3], C = FINE[1,3,@-3],
       D = FINE[2,1,@-3], E = FINE[2,2,@-3], F = FINE[2,3,@-3],
       G = FINE[3,1,@-3], H = FINE[3,2,@-3], J = FINE[3,3,@-3];
'REF''REAL'
       AA =A[ 0], AB =A[ 1], AD =A[ 3],
       BA =B[-1], BB =B[ 0], BC =B[ 1], BD =B[ 2], BE =B[ 3],
       CB =C[-1], CC =C[ 0], CE =C[ 2], CF =C[ 3],
       DA =D[-3], DB =D[-2], DD =D[ 0], DE =D[ 1], DG =D[ 3],
       EB =E[-3], EC =E[-2], ED =E[-1], EE =E[ 0],
                             EF =E[ 1], EG =E[ 2], EH =E[ 3],
       FC =F[-3], FE =F[-1], FF =F[ 0], FH =F[ 2], FJ =F[ 3],
       GD =G[-3], GE =G[-2], GG =G[ 0], GH =G[ 1],
       HE =H[-3], HF =H[-2], HG =H[-1], HH =H[ 0], HJ =H[ 1],
       JF =J[-3], JH =J[-1], JJ =J[ 0];

     TI:= 2*L1; TK:= 2*L2;
     FINE[3,3,]:= AFI[TI,TK,];
    # ##JJ# ##ACO[L1,L2,0]:= P0*JJ*W0;
'FOR' K 'FROM' L2+1 'TO' U2
'DO' TKM:= TK; TK:= 2*K;
     FINE[3,1:3,]:= AFI[TI,TKM:TK,];
    # ##GG# ##ACO[L1,K-1,0] +:= P0*GH*M1+P1*HG*W0+P1*HH*M1;
    # ##JJ# ##ACO[L1,K  ,0] +:= P0*JH*W1+Q1*HJ*W0+Q1*HH*W1
                              +P0*JJ*W0;
    # ##GJ# ##ACO[L1,K-1,1] +:= P0*GH*W1+P1*HJ*W0+P1*HH*W1;
    # ##JG# ##ACO[L1,K ,-1] +:= Q1*HG*W0+P0*JH*M1+Q1*HH*M1
'OD';

'FOR' I 'FROM' L1+1 'TO' U1
'DO' TIM:= TI; TI:= 2*I; TK:= 2*L2;
     FINE[1:3,3,]:= AFI[TIM:TI,TK,];
    # ##CC# ##ACO[I-1,L2,0] +:= P0*CF*M3+P3*FC*W0+P3*FF*M3;
    # ##JJ# ##ACO[I  ,L2,0] +:= P0*JF*W3+Q3*FJ*W0
                              +Q3*FF*W3+P0*JJ*W0;
    # ##CJ# ##ACO[I-1,L2,3] +:= P0*CF*W3+P3*FJ*W0+P3*FF*W3;
    # ##JC# ##ACO[I ,L2,-3] +:= Q3*FC*W0+P0*JF*M3+Q3*FF*M3;
'FOR' K 'FROM' L2+1 'TO' U2
'DO' TKM:= TK; TK:= 2*K;
     FINE[1:3,1:3,]:= AFI[TIM:TI,TKM:TK,];
     'REF'[]'REAL' A = ACO[I-1,K-1,@-3],
                   C = ACO[I-1,K  ,@-3],
                   G = ACO[I  ,K-1,@-3],
                   J = ACO[I  ,K  ,@-3];
    # ##AA# ## A[ 0]+:= P1*BD*M3+P3*DB*M1;
    # ##CC# ## C[ 0]+:= P0*CE*M2+P2*EC*W0+P0*CF*M3
                       +P3*FC*W0+P2*EE*M2+P3*FF*M3
                       +Q1*BE*M2+P2*EB*W1+P2*EF*M3+P3*FE*M2;
    # ##GG# ## G[ 0]+:= P0*GE*W2+Q2*EG*W0+P0*GH*M1
                       +P1*HG*W0+Q2*EE*W2+P1*HH*M1
                       +Q3*DE*W2+Q2*ED*W3+Q2*EH*M1+P1*HE*W2;
    # ##JJ# ## J[ 0]+:= P0*JJ*W0+P0*JF*W3+Q3*FJ*W0
                       +P0*JH*W1+Q1*HJ*W0+Q3*FF*W3
                       +Q1*HH*W1+Q3*FH*W1+Q1*HF*W3;
    # ##AC# ## A[ 1]+:= P1*BE*M2+P3*DB*W1+P3*DE*M2;
    # ##CA# ## C[-1]+:= P2*EB*M1+Q1*BD*M3+P2*ED*M3;
    # ##AG# ## A[ 3]+:= P1*BD*W3+P3*DE*W2+P1*BE*W2;
    # ##GA# ## G[-3]+:= Q3*DB*M1+Q2*ED*M3+Q2*EB*M1;
    # ##GC# ## G[-2]+:= P0*GE*M2+Q2*EC*W0+Q2*EE*M2
                       +P1*HE*M2+Q3*DE*M2+P1*HF*M3
                       +Q3*DB*W1+Q2*EF*M3+Q2*EB*W1;
    # ##CG# ## C[ 2]+:= P2*EG*W0+P0*CE*W2+P2*EE*W2
                       +P2*EH*M1+P2*ED*W3+P3*FH*M1
                       +Q1*BD*W3+P3*FE*W2+Q1*BE*W2;
    # ##GJ# ## G[ 1]+:= P0*GH*W1+P1*HJ*W0+P1*HH*W1
                       +Q2*EH*W1+P1*HF*W3+Q2*EF*W3;
    # ##JG# ## J[-1]+:= Q1*HG*W0+P0*JH*M1+Q1*HH*M1
                       +Q1*HE*W2+Q3*FH*M1+Q3*FE*W2;
    # ##CJ# ## C[ 3]+:= P0*CF*W3+P3*FJ*W0+P3*FF*W3
                       +P2*EH*W1+P2*EF*W3+P3*FH*W1;
    # ##JC# ## J[-3]+:= Q3*FC*W0+P0*JF*M3+Q3*FF*M3
                       +Q1*HE*M2+Q3*FE*M2+Q1*HF*M3
     'OD' 'OD';  ACO
'END'
   #$E#;
   #E$#


#  ORIENTATION:


      ACO = COARSE          K-1           K
                ------------------------->  Y
                  !
                  !          FINE     1     2     3
                  !                                                -3    -2
                  ! I-1         1      A --  B --  C                 !  /
                  !                    !  /  !  /  !           -1  -  0  -  1
                  !               2    D --  E --  F              /  !
                  !                    !  /  !  /  !            2     3
                  !  I          3      G --  H --  J
   X    V
#

# COLLECTION OF PRIMARY TESTPROBLEMS                #     'PR' EJECT 'PR'

#  'PROBLEM' STANDARD = 'PROBLEM' ('TRUE','NIL',
             ('REF''REAL' AXX,AXY,AYX,AYY, 'REAL' X,Y)'VOID':
                          (AXX:=A11; AXY:=A12; AYX:=A21; AYY:=A22 ),
             ('REF''REAL' A,B,C,F, 'REAL' X,Y,U)'VOID':
                          (A  := B1; B  := B2; C  := C0;
                                        F:= STANDARD RHS(X,Y) ),
             ('REF''REAL' A, 'REAL' X,Y)'VOID':          (A:= 0.0 ),
             ('REF''REAL' B,C, 'REAL' X,Y,U)'VOID':
                          (B:= 1.0E+40;  C:= B * DIRICHLET(X,Y) ) );
   'PROC' STANDARD RHS := ('REAL' X,Y)'REAL': S0 + X*S1 + Y*S2;
   'PROC' DIRICHLET    := ('REAL' X,Y)'REAL': X*X + Y*Y;
   'PROC' SOLUTION     := ('REAL' X,Y)'REAL': X*X + Y*Y;

   'REAL' A11:=1,A12:=0,A21:=0,A22:=1,B1:= 0,B2:=0,C0:=0,
                                       S0:=-4,S1:=0,S2:=0;
#

   'PROC' ANISOTROPIC  = ('REAL' EPSILON,ALFA)'VOID':
   #$B ANISOP  B$#
          ('REAL' C = COS(ALFA), S = SIN(ALFA);
           A11:= EPSILON*C*C + S*S; A12:= A21:= (EPSILON-1)*C*S;
           A22:= EPSILON*S*S + C*C;
           B1:= B2:= C0:= S1:= S2:= 0;     S0:= -2*(EPSILON+1) )
   #$E#;
   #E$#
   'PROC' CONVECTION   = ('REAL' EPSILON,ALFA)'VOID':
   #$B CONVEC  B$#
          (A11:= A22:= EPSILON; A12:= A21:= 0;
           B1 := COS(ALFA); B2 := SIN(ALFA); C0:= 0;
           S0:= -4*EPSILON; S1:= 2*B1; S2:= 2*B2 )
   #$E#;
   #E$#
```

```
    'PROC' POISSON  = 'VOID':
    #$B POISON  B$#
    ANISOTROPIC(1,0)
    #$E#;
    #E$#
# OUTPUT PROCEDURES             #                    'PR' EJECT 'PR'

    'OP' 'PRINT' = ('NET' NET)'VOID':
    #$B PRNET   B$#
                  ('INT' L1 = 1'LWB'NET,  L2 = 2'LWB'NET,
                         B1 = 1'UPB'NET,  B2 = 2'UPB'NET;
                   PRINT((NEWLINE,NEWLINE,L1,B1,L2,B2));
                   'FOR' I 'FROM' L1 'TO' B1
                   'DO' PRINT(NEWLINE);
                        'FOR' J 'FROM' L2 'TO' B2
                        'DO'  PRINT((FLOAT(NET[I,J],11,4,3),"  "))
                   'OD' 'OD'; PRINT(NEWLINE))
    #$E#;
    #E$#
    'OP' 'PRINT' = ('NETMAT' NET)'VOID':
    #$B PRNETM  B$#
                  ('INT' L1 = 1'LWB'NET,  L2 = 2'LWB'NET,
                         B1 = 1'UPB'NET,  B2 = 2'UPB'NET;
                   PRINT((NEWLINE,NEWLINE,  L1,B1,L2,B2));
                   'FOR' I 'FROM' L1 'TO' B1
                   'DO' PRINT(NEWLINE);
                       'FOR' J 'FROM' L2 'TO' B2
                       'DO' PRINT(NEWLINE);
                           'FOR' K 'FROM' -3 'TO' +3
                           'DO' PRINT((FLOAT(NET[I,J,K],12,6,2),"  "))
                   'OD''OD''OD';PRINT(NEWLINE))
    #$E#;
    #E$#
    'OP' 'PRINT' = ('GRID' A)'VOID':
    #$B PRGRID  B$#
                  ( PRINT((NEWLINE,NEWLINE," LEVEL = ",LEVEL'OF'A));
                   'PRINT' NET'OF'A )
    #$E#;
    #E$#
    'PROC' FL    = ([]'REAL' R)'VOID':
    #$B FL      B$#
                  ('FOR' I 'TO' 'UPB' R
                   'DO' PRINT((" ",FLOAT(R[I],12,6,2))) 'OD')
    #$E#;
    #E$#
    'PROC' FX    = ([]'REAL' R,S)'VOID':
    #$B FX      B$#
                  ('FOR' I 'TO' 'UPB' R
                   'DO' PRINT((" ",FIXED(R[I]/S[I],6,2))) 'OD')
    #$E#;
    #E$#
    'PROC' SPC   = 'VOID':
    #$B SPC     B$#
                  (COLLECT GARBAGE; PRINT((
          NEWLINE,"GARBAGE:",COLLECTIONS,GARBAGE,COLLECT SECONDS,
          NEWLINE,"SPACE  :",PROGSIZE,STACKSIZE,HEAPSIZE,AVAILABLE,
                             PROGSIZE+STACKSIZE+HEAPSIZE+AVAILABLE,
          NEWLINE,"TIME   :",CLOCK )))
    #$E#;
    #E$#
    'OP''ERRORPRINT' = ('GRID' U ) 'VOID':
    #$B PRERR   B$#
    'PRINT'(U-SOLUTION)
    #$E#;
    #E$#

# GLOBAL PARAMETERS             #                    'PR' EJECT 'PR'

    # ********************************************************** #
    # *********    GLOBAL PARAMETERS FOR THE MG-ALGORITHM   ******** #
    # ********************************************************** #

    'PROC' PROLONGATE := ('GRID' A)'GRID': (ERROR; A);
    'PROC' INTERPOLATE:= ('GRID' A)'GRID': (ERROR; A);
    'PROC' RESTRICTBAR:= ('GRID' A)'GRID': (ERROR; A);
    'PROC' RESTRICT   := ('GRID' A)'GRID': (ERROR; A);
    'PROC' RELAX      := ('REF''GRID' U, 'GRID' F)'VOID': (ERROR);

    # ********************************************************** #

    'BOOL' ILU:= 'FALSE';
    'INT' P:= 1, S:= 2, Q:= 1, R, PR:= 0; R:= S; 'REAL'MU:= 1.0;

    'OP'  'I'   = ('NET' N)'NET': SQR INT POL(N);
    'OP'  'P'   = ('NET' N)'NET': LIN INT POL(N);
    'OP'  'R'   = ('NET' N)'NET': INJECTION  (N);
    'OP'  'RBAR'= ('NET' N)'NET': LIN WEIGHT (N);

    'PROC' RELAX NET :=
                ('REF''NETMAT' D,'NETMAT' M, 'NET'U,F )'VOID': 'SKIP';

    # ********************************************************** #
    # *********    GLOBAL PARAMETERS FOR THE RELAXATION     ******** #
    # ********************************************************** #

    'BOOL' SYMMETRIC:= 'FALSE', BACKWARD:= 'FALSE',
           REVERSE  := 'FALSE', ZEBRA   := 'FALSE';

    'INT' TH RELAX STRATEGY := 1;

    [,]'INT' FOUR COLOR RELAX = ((0,1),(1,0),(1,1),(0,0));
    'FLEX'[1:1,1:2]'INT' CH RELAX STRATEGY:= FOUR COLOR RELAX;

    # ********************************************************** #
    # *********    GLOBAL PARAMETERS FOR ASYMM.RESTRICTONS   ******* #
    # ********************************************************** #

    [-3:3]'REAL' WASYM,PASYM,RASYM;
                 PASYM[@1]:= (0.5,0.5,0.5,1.0,0.5,0.5,0.5);
                 RASYM    := WASYM    := PASYM;

    # ********************************************************** #
    # *********    COMMON GLOBAL PARAMETERS     ****************** #
    # ********************************************************** #

    'REF'[]'NETMAT' JACOBIAN,DECOMP;

    'PROC' DISCRETIZATION      := ('PROBLEM' PROBLEM, 'GRID' U,
           'REF''NET' RHS, 'REF''NETMAT' JAC )'VOID':
           'SKIP';
    'PROC' BOUNDARY CONDITIONS := ('PROBLEM' PROBLEM, 'GRID' U,
           'NET' RHS, 'NETMAT' JAC )'VOID':
           'SKIP';

    'REAL' Q0:= 0, Q1:= 1, Q2:= 1;
    'PROC' LUMP = ('BOOL' B)'VOID':
        ( B ! Q0:=0; Q1:=1; Q2:=1 ! Q0:=0.25; Q1:=0.50; Q2:=0.25 );
```

```
'REAL' HB FACTOR:= 1/3;
'PROC' HB:= ('REF''REAL' AXX,AXY,AYX,AYY,'REAL' BX,BY,H,K)'REAL':0.0;

   'PROC' GOON FMGG   := ('GRID' U,F, 'REF'[]'REAL' RES)'BOOL': 'TRUE';
   'PROC' GOON FMG    := ('NETMAT' JAC, 'NET' U,F, 'REF'[]'REAL' RES)
                         'BOOL': 'TRUE';
   'PROC' MGM GOON:=('INT' ITNUM,'NETMAT'LH,'NET'UH,FH)'BOOL':'TRUE';
   'PROC' REP := ('INT' NUMBER, LEVEL)'VOID': 'SKIP';

   'PROC' FAIL= ('INT' N,'STRING' TEXT)'VOID':
   ( PRINT((NEWLINE,TEXT,N,NEWLINE)); ERROR);

   'BOOL' MONIT:= 'FALSE';

                                                    'PR' EJECT 'PR'

   # ********************************************************** #
   # ***************** GLOBAL PARAMETERS   ********************** #
   # ************ FOR PRIMARY TESTPROBLEMS     ******************** #
   # ********************************************************** #

   'PROBLEM' STANDARD = 'PROBLEM' ('TRUE','NIL',
            ('REF''REAL' AXX,AXY,AYX,AYY, 'REAL' X,Y)'VOID':
                        (AXX:=A11; AXY:=A12; AYX:=A21; AYY:=A22 ),
            ('REF''REAL' A,B,C,F, 'REAL' X,Y,U)'VOID':
                        (A  := B1; B  := B2; C  := C0;
                                          F:= STANDARD RHS(X,Y) ),
            ('REF''REAL' A, 'REAL' X,Y)'VOID':          (A:= 0.0 ),
            ('REF''REAL' B,C, 'REAL' X,Y,U)'VOID':
                        (B:= 1.0E+40;  C:= B * DIRICHLET(X,Y) ) );
   'PROC' STANDARD RHS := ('REAL' X,Y)'REAL': S0 + X*S1 + Y*S2;
   'PROC' DIRICHLET    := ('REAL' X,Y)'REAL': X*X + Y*Y;
   'PROC' SOLUTION     := ('REAL' X,Y)'REAL': X*X + Y*Y;

   'REAL' A11:=1,A12:=0,A21:=0,A22:=1,B1:= 0,B2:=0,C0:=0,
                                   S0:=-4,S1:=0,S2:=0;


'PR' PROG 'PR' 'SKIP'
'END'
LIBRARY(LIB,OLD)
ADD(*,LGO)
FINISH.
ENDRUN.
```

```
PWH,CM120000,T50,NP.
ACCOUNT,[redacted]
ATTACH,MGLIB,MGLIB,ID=PHEMKER.
A68,D,P=MGLIB/MGLIB.
LGO.
 'BEGIN'  # MGTESTPWH: TESTPROGRAMMA VOOR MGTEXTPWH  PWH820815 #


          # !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! #
          # EVERYWHERE THE MATRIX DOES NOT DEFINE THE NETMAT M, #
          # !!!!!!!!!!!! M SHOULD CONTAIN ZEROES !!!!!!!!!!!!!!!! #
          # !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! #

  'PROC' MGM = ('REF'[]'NETMAT' LH, 'REF'[]'NET' UH,FH,
                'INT' ITMAX,P,Q,S,T, 'REF'[]'NETMAT' ILLU,
                'REF''INT' ITUSED,
                'PROC'('INT','NETMAT','NET','NET')'BOOL' GO ON MGM,
                'PROC'('INT','STRING')'VOID' FAIL)'VOID':
   #$B MGM     B$#
   'BEGIN' 'INT' L='UPB'UH, R = S;
           'REF'[]'NETMAT' ILLUD =
              ( ILLU :=: 'REF'[]'NETMAT'('NIL')
              ! 'LOC'[0:L]'NETMAT' ! ILLU );

     'PROC' MG = ('INT'L)'VOID':
     'IF'   L = 0
     'THEN' ILLU RELAX(ILLUD[0],LH[0],UH[0],FH[0])
     'ELSE' 'TO' P 'DO' ILLU RELAX(ILLUD[L],LH[L],UH[L],FH[L])'OD';
            FH[L-1]:= LIN WEIGHT( RESIDUAL (LH[L],UH[L],FH[L]) );
            # ## WEIGHT  RES (LH[L],UH[L],FH[L],FH[L-1]);   # ##
            'ZERO' UH[L-1];
            'TO' (L=1!T!S) 'DO' MG (L-1) 'OD';

            UH[L] +:= LIN INT POL ( UH[L-1]);
            # ## ADD INT POL (UH[L],UH[L-1]);                # ##
            'TO' Q 'DO' ILLU RELAX(ILLUD[L],LH[L],UH[L],FH[L])'OD'
     'FI';

          'INT' ERR =
          (  'LWB' UH /= 0 'OR' 'LWB'FH /= 0 'OR' 'LWB'LH /= 0
                           'OR' 'UPB'FH /= L 'OR' 'UPB'LH /= L ! 1
          !: 'NETMAT' LL = LH[L];
              3'LWB'LL /=-3 'OR' 3'UPB'LL /= 3                  ! 2
          !: S <= 0 'OR' S > 3  'OR' T <= 0                     ! 3
          !: ITMAX<0 'OR' P<0 'OR' Q<0                          ! 4
          !: 'LWB'ILLUD /= 0 'OR' 'UPB'ILLUD /=L                ! 5
          !  0  );
          ( ERR>0 ! FAIL ( ERR," MGM "));

          'IF' ITUSED < 0
          'THEN' # ## CREATE GALERKIN APPROXIMATIONS # ##
                 'FOR' I 'FROM' L 'BY' -1 'TO' 1
                 'DO' REDUCE(LH[I],LH[I-1],FH[I],FH[I-1],UH[I-1])
                 'OD';
                 ITUSED:= 0
          'FI';


          'IF' ITUSED = 0
          'THEN' 'FOR'K 'FROM' 0 'TO' L
                 'DO' ILLUDEC (LH[K],ILLUD[K]) 'OD';


  #  #
                       'NETMAT' LUD = ILLUD[L];
                       'INT' QQQ = 3'LWB'LUD, QQZ = (1'UPB'LUD)'OVER'2;
     (MONITOR!         'PRINT' LUD [,,QQQ  ];
                       'PRINT' LUD [,,QQQ+1];
                       'PRINT' LUD [,,QQQ+2]
     );
     PRINT((NEWLINE,LUD[QQZ,QQZ,QQQ],NEWLINE,LUD[QQZ,QQZ,QQQ+1],
                                     NEWLINE,LUD[QQZ,QQZ,QQQ+2],
                                     NEWLINE,LUD[QQZ,QQZ,QQQ+2],

   #  #
                  # ## APPLY FULL MULTIGRID # ##
                  'TO' T 'DO' MG(0) 'OD';
                  'FOR' K 'TO' L-1
                  'DO'  UH[K]:= SQR INT POL (UH[K-1]);
                       'TO' R  'DO'  MG (K) 'OD'
                  'OD'; UH[L]:= SQR INT POL (UH[L-1]);
                  GOON MGM (ITUSED,LH[L],UH[L],FH[L])
            'FI';

            'TO' ITMAX
            'WHILE'  MG (L); ITUSED +:= 1;
                     GOON MGM (ITUSED, LH[L], UH[L], FH[L])
            'DO' 'SKIP' 'OD'
    'END'
    #$E#;
    #E$#

'FOR' AAA 'FROM' 0 'TO' 7
'DO'  PRINT((NEWPAGE,AAA));
      'REAL' ALFA:= AAA*PI/8;
      [1:3,1:2]'REAL' RAND1 := ((0,1),(0,0),(1,0)),
                      RAND2 := ((0,0),(1,0),(1,1));
      'PROBLEM' PRN         := STANDARD;
                OMEGA'OF'PRN:= RAND2;

'INT' M = 4;
DISCRETIZATION:= FEM;  BOUNDARY CONDITIONS:= BC PW LIN;
ANISOTROPIC(1.0E-9,ALFA);
JACOBIAN:= 'HEAP'[0:M]'NETMAT';
[0:M]'GRID' UG;
[0:M]'NET'  F;  'REF'[]'NET' U = NET'OF'UG;
'INT' USED;
[1:3]'REAL' NO,NN:= (1,1,1);

MGM GOON:= (('INT' IT,'NETMAT'L, 'NET' U,F)'BOOL':
            (PRINT((NEWLINE,NEWLINE," MGM GOON", IT));
             (IT=0!'ZERO' U;
                   U:= NET'OF'(
                       'GRID' (M,U) -
                       (('REAL'X,Y)'REAL': -X*(1-X)*Y*(1-Y)*1.0E6 )
                            );
                   F:= L*U; 'ZERO' U;
                   PRINT((NEWLINE,L[1,1,] ))      );
                        NO:= NN;
             PRINT((NEWLINE,NN:=RESIDUALNORM(L,U,F),NEWLINE ));
             FX(NN,NO); FX(NO,NN);
             'TRUE'                                )    );
UG[M]:= 'GRID'(M,('HEAP'[0:2**M,0:2**M]'REAL'A;'ZERO'A;A));

DISCRETIZATION(PRN,UG[M],F[M],JACOBIAN[M]);

MGM(JACOBIAN,U,F,
    3,0,1,1,1,'NIL',USED:= -1,
    MGM GOON,FAIL)
```

```
'OD'
'END'
```

```
PWH,CM120000,T50,NP.
ACCOUNT,[redacted]
ATTACH,MGLIB,MGLIB,ID=PHEMKER.
A68,D,P=MGLIB/MGLIB.
LGO.
 'BEGIN'   # MGTESTPWH: TESTPROGRAMMA VOOR MGTEXTPWH  PWH820815 #


          # !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! #
          # EVERYWHERE THE MATRIX DOES NOT DEFINE THE NETMAT M, #
          # !!!!!!!!!!! M SHOULD CONTAIN ZEROES !!!!!!!!!!!!!!! #
          # !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! #

   'PROC'  MGM = ('REF'[]'NETMAT' LH, 'REF'[]'NET' UH,FH,
                  'INT' ITMAX,P,Q,S,T, 'REF'[]'NETMAT' ILLU,
                  'REF''INT' ITUSED,
                  'PROC'('INT','NETMAT','NET','NET')'BOOL' GO ON MGM,
                  'PROC'('INT','STRING')'VOID' FAIL)'VOID':
    #$B MGM      B$#
    'BEGIN' 'INT' L='UPB'UH, R = S;
            'REF'[]'NETMAT' ILLUD =
               ( ILLU :=: 'REF'[]'NETMAT'('NIL')
               ! 'LOC'[0:L]'NETMAT' ! ILLU );

      'PROC' MG = ('INT'L)'VOID':
      'IF'    L = 0
      'THEN'  ILLU RELAX(ILLUD[0],LH[0],UH[0],FH[0])
      'ELSE'  'TO' P 'DO' ILLU RELAX(ILLUD[L],LH[L],UH[L],FH[L])'OD';
              FH[L-1]:= LIN WEIGHT( RESIDUAL (LH[L],UH[L],FH[L]) );
              # ## WEIGHT  RES (LH[L],UH[L],FH[L],FH[L-1]);    # ##
              'ZERO' UH[L-1];
              'TO' (L=1!T!S) 'DO' MG (L-1) 'OD';

              UH[L] +:= LIN INT POL ( UH[L-1]);
              # ## ADD INT POL (UH[L],UH[L-1]);                # ##
              'TO' Q 'DO' ILLU RELAX(ILLUD[L],LH[L],UH[L],FH[L])'OD'
      'FI';

            'INT' ERR =
            (  'LWB' UH /= 0 'OR' 'LWB'FH /= 0 'OR' 'LWB'LH /= 0
                               'OR' 'UPB'FH /= L 'OR' 'UPB'LH /= L ! 1
            !: 'NETMAT' LL = LH[L];
                 3'LWB'LL /=-3 'OR' 3'UPB'LL /= 3                ! 2
            !: S <= 0 'OR' S > 3  'OR' T <= 0                    ! 3
            !: ITMAX<0 'OR' P<0 'OR' Q<0                         ! 4
            !: 'LWB'ILLUD /= 0 'OR' 'UPB'ILLUD /=L               ! 5
            !  0  );
            ( ERR>0 ! FAIL ( ERR," MGM "));

            'IF' ITUSED < 0
            'THEN' # ## CREATE GALERKIN APPROXIMATIONS # ##
                   'FOR' I 'FROM' L 'BY' -1 'TO' 1
                   'DO' REDUCE(LH[I],LH[I-1],FH[I],FH[I-1],UH[I-1])
                   'OD';
                   ITUSED:= 0
            'FI';


            'IF' ITUSED = 0
            'THEN' 'FOR'K 'FROM' 0 'TO' L
                   'DO' ILLUDEC (LH[K],ILLUD[K]) 'OD';


   #  #
                        'NETMAT' LUD = ILLUD[L];
                        'INT' QQQ = 3'LWB'LUD, QQZ = (1'UPB'LUD)'OVER'2;
     (MONITOR!          'PRINT' LUD [,,QQQ  ];
                        'PRINT' LUD [,,QQQ+1];
                        'PRINT' LUD [,,QQQ+2]
     );
     PRINT((NEWLINE,LUD[QQZ,QQZ,QQQ],NEWLINE,LUD[QQZ,QQZ,QQQ+1],
                                     NEWLINE,LUD[QQZ,QQZ,QQQ+2],
                                     NEWLINE,LUD[QQZ,QQZ,QQQ+2],
   #  #
                   # ## APPLY FULL MULTIGRID # ##
                   'TO' T 'DO' MG(0) 'OD';
                   'FOR' K 'TO' L-1
                   'DO'  UH[K]:= SQR INT POL (UH[K-1]);
                         'TO' R  'DO'  MG (K) 'OD'
                   'OD'; UH[L]:= SQR INT POL (UH[L-1]);
                   GOON MGM (ITUSED,LH[L],UH[L],FH[L])
            'FI';

            'TO' ITMAX
            'WHILE'  MG (L); ITUSED +:= 1;
                     GOON MGM (ITUSED, LH[L], UH[L], FH[L])
            'DO' 'SKIP' 'OD'
    'END'
    #$E#;
    #E$#

'FOR' AAA 'FROM' 0 'TO' 7
'DO'  PRINT((NEWPAGE,AAA));
      'REAL' ALFA:= AAA*PI/8;
      [1:3,1:2]'REAL' RAND1 := ((0,1),(0,0),(1,0)),
                      RAND2 := ((0,0),(1,0),(1,1));
      'PROBLEM' PRN          := STANDARD;
                OMEGA'OF'PRN:= RAND2;

'INT' M = 4;
DISCRETIZATION:= FEM;  BOUNDARY CONDITIONS:= BC PW LIN;
ANISOTROPIC(1.0E-9,ALFA);
JACOBIAN:= 'HEAP'[0:M]'NETMAT';
[0:M]'GRID' UG;
[0:M]'NET'  F;  'REF'[]'NET' U = NET'OF'UG;
'INT' USED;
[1:3]'REAL' NO,NN:= (1,1,1);

MGM GOON:= (('INT' IT,'NETMAT'L, 'NET' U,F)'BOOL':
            (PRINT((NEWLINE,NEWLINE," MGM GOON", IT));
             (IT=0!'ZERO' U;
                   U:= NET'OF'(
                       'GRID' (M,U) -
                       (('REAL'X,Y)'REAL': -X*(1-X)*Y*(1-Y)*1.0E6 )
                               );
                   F:= L*U; 'ZERO' U;
                   PRINT((NEWLINE,L[1,1,] ))      );
                            NO:= NN;
             PRINT((NEWLINE,NN:=RESIDUALNORM(L,U,F),NEWLINE ));
             FX(NN,NO); FX(NO,NN);
             'TRUE'                                 )     );
UG[M]:= 'GRID'(M,('HEAP'[0:2**M,0:2**M]'REAL'A;'ZERO'A;A));

DISCRETIZATION(PRN,UG[M],F[M],JACOBIAN[M]);

MGM(JACOBIAN,U,F,
    3,0,1,1,1,'NIL',USED:= -1,
    MGM GOON,FAIL)
```

```
'OD'
'END'
```

```
IEQLIB:
'BEGIN' # AUTOMATIC SOLUTION OF FREDHOLM EQUATIONS OF THE SECOND KIND #

   'MODE' 'NAGFAIL' = 'PROC'('INT','STRING')'VOID';
   'MODE' 'VEC' = 'REF' [ ] 'REAL';
   'MODE' 'MAT' = 'REF' [,] 'REAL';

   'OP' ** = ('REAL'X,Y) 'REAL': ( X <= 0.0 ! 0.0 ! EXP(Y*LN(X)) );

   'OP' + = ('VEC' A,B) 'VEC' :
   ( 'INT' L='LWB'A, U='UPB'A; 'VEC' C = 'HEAP'[L:U]'REAL';
     'FOR' I 'FROM' L 'TO' U 'DO' C[I]:= A[I] + B[I] 'OD';
     C ) # VEC + VEC #;

   'OP' - = ('VEC' A,B) 'VEC' :
   ( 'INT' L='LWB'A, U='UPB'A; 'VEC' C = 'HEAP'[L:U]'REAL';
     'FOR' I 'FROM' L 'TO' U 'DO' C[I]:= A[I] - B[I] 'OD';
     C ) # VEC - VEC #;

   'OP' / = ('VEC' A, 'REAL' B) 'VEC' :
   ( 'INT' L='LWB'A, U='UPB'A; 'VEC' C = 'HEAP'[L:U]'REAL';
     'FOR' I 'FROM' L 'TO' U 'DO' C[I]:= A[I] / B  'OD';
     C ) # VEC / REAL #;

   'OP' * = ('REAL' B, 'VEC' A) 'VEC' :
   ( 'INT' L='LWB'A, U='UPB'A; 'VEC' C = 'HEAP'[L:U]'REAL';
     'FOR' I 'FROM' L 'TO' U 'DO' C[I]:= B * A[I] 'OD';
     C ) # REAL * VEC #;

   'OP' * = ('VEC' A, B) 'REAL' :
   ( 'INT' L='LWB'A, U='UPB'A; 'REAL' C:= 0.0;
     'FOR' I 'FROM' L 'TO' U 'DO' C +:= A[I]*B[I] 'OD';
     C ) # VEC * VEC #;

   'OP' * = ('MAT' A, 'VEC' B) 'VEC' :
   ( 'INT' L=1'LWB'A, U=1'UPB'A; 'VEC' C = 'HEAP'[L:U]'REAL';
     'FOR' I 'FROM' L 'TO' U 'DO' C[I] := A[I, ]*B[ ] 'OD';
     C ) # MAT * VEC #;

   'OP' 'COPY' = ('VEC' U) 'VEC' :
   ( 'INT'L = 'LWB'U, UP = 'UPB'U; 'VEC' C = 'HEAP'[L:UP] 'REAL';
     'FOR' I 'FROM' L 'TO' UP 'DO' C[I]:= U[I] 'OD';
     C  )  ;

   'PROC' PRVEC = ('VEC' X) 'VOID':
   (  PRINT((" VEC BOUNDS ",'LWB'X,'UPB'X,NEWLINE));
     'FOR' I 'FROM' 'LWB'X 'TO' 'UPB'X
     'DO'  PRINT(X[I]) 'OD';
      PRINT(NEWLINE) );
                                                          'PR' EJECT 'PR'
#'PROC' PINJECT = ('INT' L, 'PROC'('REAL')'REAL'F) 'VEC':
 ( 'INT' NL= N(L);          'HEAP'[0:NL]'REAL' YL;
   'REAL' A:= INTERVAL[1]; 'REAL' H = (INTERVAL[2] - A)/NL;
   YL[0]:= F(A);
   'FOR' I 'TO' NL
```

```
   'FOR' I 'FROM' L+1 'TO' U 'DO' ( A[I]<S ! S:=A[I] ) 'OD';
    S );

'PROC' NORM = ('VEC' A)'REAL':
( 'INT' L='LWB'A, U='UPB'A; 'REAL'S:='ABS'A[L];
   'FOR' I 'FROM' L+1 'TO' U
   'DO' 'REAL' B = 'ABS' A[I]; ( B>S ! S:=B ) 'OD';
    S );
                                                          'PR' EJECT 'PR'

   'PROC' SOLVE DIRECTLY = ('MAT' JACOBIAN, 'VEC' UM, RHSM) 'VOID':
   'BEGIN'       # GAUSSIAN ELIMINATION #
   ( 1'UPB'JACOBIAN /= 'UPB' RHSM 'OR' 2'UPB'JACOBIAN/= 'UPB'RHSM !
    ERROR);
   'MAT' JB= JACOBIAN['AT'1,'AT'1];
   'INT' N = 'UPB' JB;
    [1:N,1:N+1]'REAL' A;
   'VEC' V = A[,N+1]; A[,1:N]:= JB; V:= RHSM['AT'1];
   'FOR' J 'TO' N
   'DO' 'INT' JP1= J+1; 'INT' PJ:= J;
        'REAL' SI,S:= 'ABS' A[J,J];
        'FOR' I 'FROM' JP1 'TO' N
        'DO' ((SI:='ABS'A[I,J]) >S ! S:=SI; PJ:=I ) 'OD';
        'IF'  J /= PJ
        'THEN''REAL' T;
              'FOR' K 'TO' N+1
              'DO' T:= A[PJ,K]; A[PJ,K]:= A[J,K]; A[J,K]:=T 'OD'
        'FI'  ;
         S := A[J,J];
        'FOR' I 'FROM' JP1 'TO' N
        'DO' SI:= A[I,J]/S;
             'FOR' K 'FROM' J 'TO' N+1
             'DO' A[I,K] -:= A[J,K]*SI 'OD'
        'OD'
   'OD';
   'FOR' J 'FROM' N 'BY' -1 'TO' 1
   'DO' V[J] /:= A[J,J];
        'FOR' I 'FROM' J-1 'BY' -1 'TO' 1
        'DO' V[I] -:= A[I,J]*V[J] 'OD'
   'OD';
   UM:= V['AT'(1'LWB'JACOBIAN)]
   'END' # SOLVE DIRECTLY # ;

   'PROC' ERRORX = 'VOID': ERROR;
   'BOOL' MONITOR:= 'TRUE';
                                                          'PR' EJECT 'PR'
'PROC' SOL INT EQN =([]'REAL'INTERVAL AB,
                     'PROC'('REAL','REAL','REAL')'REAL' KK,
                     'PROC'('REAL'              )'REAL' FORCEY,
                     'REAL' TOL, 'BOOL' TRAPEZ,
                     'REF''INT' N0, 'INT' N0UPPER,NLUPPER,
                     'REF''VEC' UM, 'REF''REAL' ERROR, 'NAGFAIL' FAIL)
                     'VOID' :
'BEGIN'

   'PROC' DETERMINE V1 = # ('REF''REAL'NUMV1,DENV1,V1,'REF''VEC'UM,
```

```
          DENV1:= NUMV1; NUMV1:= NORM(UMOLD-UM);
          'IF'  NUMV1 > MIN((TOL,1.0E-8))
          'THEN'(IT > 1 ! V1OLD:= V1 );
                V1:= NUMV1/DENV1;
                IT<3 'OR' 'ABS'(V1-V1OLD) > 0.02*V1OLD
          'ELSE'( IT=1 ! V1*:= RATIO ); V1OLD:=V1; 'FALSE'
          'FI'
    'DO'  'SKIP'   'OD';
    V1:= 1.02*MAX((V1OLD,V1))                );

   'PROC' EXAMINE CONVERGENCE = 'BOOL':
          #('REAL'V1,'INT'LEVELS,'REF'[]'INT'GAMMA)#
   ('BOOL' CONV;  GAMMA[1]:= 0;
    'FOR'  II 'TO' LEVELS+3
    'WHILE''IF'   II <= LEVELS
          'THEN' 'FOR' I 'FROM' 2 'TO' II 'DO' GAMMA[I]:= 3 'OD'
          'ELIF' II=LEVELS+1 'THEN' GAMMA[2]:= 4
          'ELIF' II=LEVELS+2 'THEN' GAMMA[2]:= 5
          'ELIF' II=LEVELS+3 'THEN' ( LEVELS>2 ! GAMMA[3]:= 4 )
          'FI'   ;
          'REAL' VP:= V1,WP:= V1;
          'IF'   CONV:= ( WP*WP <= VP )
          'THEN' 'FOR'   P 'FROM' 2 'TO' LEVELS
                 'WHILE' VP *:= RATIO;
                         WP  := VP+WP**GAMMA[P]*(VP+NORMT);
                         CONV:= ( WP*WP <= VP )
                 'DO'    'SKIP'    'OD'
          'FI';
          ('NOT' CONV) 'AND' N0 = N0UPPER
    'DO'  'SKIP'  'OD'
                       ;(MONITOR ! PRINT(NEWLINE);
                       'FOR' II 'TO' LEVELS
                       'DO' PRINT((WHOLE(GAMMA[II],4))) 'OD');
    CONV          );

 # 'PROC' V1 ON LEVEL = ('INT'M) 'REAL':
   ('REAL' VP:= V1, WP:= V1;
    'FOR' P 'FROM' 2 'TO' M
    'DO' VP:= RATIO*VP; WP:= VP+WP**GAMMA[P]*(VP+NORMT) 'OD';
     WP      );
 #
 'PROC' N = ('INT'L)'I' ':( N0 * 2**L );

#'PROC' LEVEL = ('INT'NB) 'INT' :
 ('INT'  S:= N(0), L:= 0;
  'WHILE'S < NB 'DO' L+:= 1; S:= 2*S 'OD';
  'IF'   S > NB 'THEN' ERRORX 'FI'; L );
#
 'PROC' ZERO = ('INT'L) 'VEC':
 ('INT  NL = N(L); 'VEC' BB = 'HEAP'[0:NL] 'REAL';
  'FOR  I 'FROM' 0 'TO' NL 'DO' BB[I]:= 0.0 'OD' ;BB);

 'PROC' PROJECT FORCEY = ('INT' L ) 'VEC':
 ( 'INT' NL= N(L);          'HEAP'[0:NL]'REAL' YL;
   'REAL' A:= INTERVALAB[1]; 'REAL' H = (INTERVALAB[2] - A)/NL;
   YL[0]:= FORCEY(A);
   'FOR' I 'TO' NL
   'DO' YL[I]:= FORCEY( A+:=H ) 'OD'; YL );

 'PROC' RESTRICT = ('VEC' VP) 'VEC':
 ( 'INT' NQ = 'UPB' VP 'OVER' 2;
   'VEC' VQ = 'HEAP'[0:NQ]'REAL';
   'FOR' I 'FROM' 0 'TO' NQ 'DO' VQ[I]:= VP[2*I] 'OD'; VQ);

  'PROC' EVALUATE JACOBIAN = ('INT' M, 'VEC' UM )'MAT':
  'BEGIN' 'INT' NM = N(M);       [0:NM]'REAL' UMD := UM;
          'VEC' QM = QAD(M,UM);   'HEAP'[0:NM,0:NM]'REAL' JAC;
          'FOR' I 'FROM' 0 'TO' NM
          'DO' 'REAL' DELTA = MAX(( UM[I]*0.001, 0.001));
                UMD[I]  +:= DELTA;
                JAC[,I]  := (QM - QAD(M,UMD))/DELTA;
                JAC[I,I]+:= 1.0;
                UMD[I]   := UM[I]
          'OD'; JAC
  'END' # EVALUATE JACOBIAN # ;

                                                        'PR' EJECT 'PR'

'PROC'LIN INT = ('VEC' VP) 'VEC':
( 'INT' NP = 'UPB' VP;
  'VEC' VQ = 'HEAP' [0:2*NP] 'REAL';
  VQ[0]:= VP[0];
  'FOR' I 'TO' NP
  'DO'  VQ[2*I]  := VP[I];
        VQ[2*I-1]:= 0.5*(VP[I-1] + VP[I])
  'OD'  ;
   VQ   );

'PROC' TRAP  = (#TO LEVEL # 'INT' P, 'VEC' V) 'VEC':
('INT' NP = N(P), NQ =  'UPB' V;
 'VEC' VP = 'HEAP' [0:NP] 'REAL';
 'INT' ST = ( NP=NQ ! 1 !: 2*NP=NQ ! 2 ! ERRORX; 0 );
 'REAL' A = INTERVALAB[1], B = INTERVALAB[2];
 'REAL' H = (B-A)/NQ;
 'FOR' I 'FROM' 0 'BY' ST 'TO' NQ
 'DO' 'REAL' S, TJ; 'REAL' TI = A + I*H;
      S := KK(TI,TJ:=A,V[0])/2;
      'FOR' J 'TO' NQ-1
      'DO'  S +:= KK(TI,TJ+:=H,V[J]) 'OD';
      S +:=  KK(TI,TJ+H,V[NQ])/2;
      VP[I'OVER'ST] := S*H
 'OD';
 VP ) ;


'PROC'CUB INT = ('VEC' VP) 'VEC':
( 'INT' NP = 'UPB' VP; 'INT' NQ = 2*NP;
  'VEC' VQ = 'HEAP' [0:NQ] 'REAL';
  VQ[0]   := VP[0];
  VQ[1]   := (5.0*(VP[0]+3.0*VP[1]-VP[2])+VP[3])/16.0;
  VQ[NQ-2]:= VP[NP-1];
  VQ[NQ-1]:= (5.0*(VP[NP]+3.0*VP[NP-1]-VP[NP-2])+VP[NP-3])/16.0;
  VQ[NQ]  := VP[NP];
  'FOR' I 'TO' NP-2
  'DO' VQ[2*I]  := VP[I];
       VQ[2*I+1]:= (-VP[I-1]+9.0*(VP[I]+VP[I+1])-VP[I+2])/16.0
  'OD' ;
  VQ   ) ;

'PROC' SIMP  = (#TO LEVEL # 'INT' P, 'VEC' V) 'VEC':
(  'INT' NP = N(P), NQ =  'UPB' V;
   'VEC' VP = 'HEAP' [0:NP] 'REAL';
   'INT' ST = ( NP=NQ ! 1 !: 2*NP=NQ ! 2 ! ERRORX; 0 );
   'REAL' A = INTERVALAB[1], B = INTERVALAB[2], W43 = 4/3, W23 = 2/3;
   'REAL' H = (B-A)/NQ;
   'FOR' I 'FROM' 0 'BY' ST 'TO' NQ
   'DO' 'REAL' S, TJ; 'REAL' TI = A + I*H;
```

```
        S := KK(TI,TJ:=A,V[0])/3;
        'FOR' J 'TO' NQ-1
        'DO'  S +:= ('ODD' J ! W43 ! W23) * KK(TI,TJ+:=H,V[J]) 'OD';
         S +:=  KK(TI,TJ+H,V[NQ])/3;
        VP[I'OVER'ST] := S*H
    'OD';
    VP ) ;

'PROC'('VEC')'VEC'  PROLONGATE = ( TRAPEZ ! LININT ! CUBINT );
'REAL' NORMT                   = ( TRAPEZ !      6 !     24 );
'PROC'('INT','VEC') 'VEC' QAD  = ( TRAPEZ !   TRAP !   SIMP );
'REAL' ALFA                    = ( TRAPEZ !      2 !      4 );

                                                        'PR' EJECT 'PR'
  'PROC' MGCYCLE = ('INT' M, SIGMA, 'REF''VEC' UM,'VEC' RHSM,
                   'BOOL' UM IS ZERO) 'VOID':
  'IF'   M = 0
  'THEN' SOLVE DIRECTLY(JACOBIAN,UM,RHSM)
  'ELSE' 'BOOL' UZ:= UM IS ZERO;
         'FOR' IT 'TO' SIGMA
         'DO' 'VEC' RM   = ( UZ ! RHSM ! RHSM-UM+QAD(M,UM));
              'VEC' UMM1:= ZERO(M-1);
              'VEC' RMM1 = RESTRICT(RM);
              'VEC' FMM1 = RMM1-QAD(M-1,RMM1)+QAD(M-1,RM);
              MGCYCLE ( M-1,GAMMA[M],UMM1,FMM1,'TRUE');
              UM:= UM + RM + PROLONGATE(UMM1-RMM1);
              UZ:= 'FALSE'
         'OD'
  'FI'   ;

  'BOOL' RAPIDCONVERGENCE;
  'INT'  LEVELS:= 'ROUND'(LN(NLUPPER/N0)/LN(2.0));
  'HEAP'[1:LEVELS]'INT'GAMMA; 'FOR'J'TO'LEVELS 'DO' GAMMA[J]:=2 'OD';
  'REAL' RATIO = 0.5**ALFA;
  'REAL' V1:=0.5,DENV1,NUMV1,V2:=0.5,DENV2,NUMV2:=1.0; ERROR:= MAXREAL;
  'VEC'  RHS, UMM1, UMOLD; 'MAT' JACOBIAN;


  'FOR'  LOOP
  'WHILE'UM:= ZERO(0); RHS:= PROJECT FORCEY (0);
         JACOBIAN:= EVALUATE JACOBIAN(0,UM);
         SOLVE DIRECTLY(JACOBIAN,UM,RHS);
         'IF'   LOOP>1
         'THEN' DENV2:=NUMV2; NUMV2:=NORM(RESTRICT(UM)-UMM1);
                (LOOP>2 ! V2:= MAX((RATIO,MIN((0.5,NUMV2/DENV2)))) );
                ERROR:=(V2/(1-V2))*NUMV2
         'FI';
         'IF'   ERROR>TOL
         'THEN' RHS  :=  PROJECT FORCEY(1);
                UMM1 := 'COPY' UM;
                UM   := PROLONGATE(UMM1);
                DETERMINE V1 #(NUMV1,DENV1,V1,UM,RHS)# ;
                RAPIDCONVERGENCE:= EXAMINE CONVERGENCE
                                  # (V1,LEVELS,GAMMA) #
         'FI';
         ERROR>TOL 'AND' ('NOT' RAPIDCONVERGENCE) 'AND' N0UPPER >= 2*N0
  'DO'   N0*:= 2; LEVELS-:= 1 'OD'  ;


  'IF'   'NOT' RAPIDCONVERGENCE
  'THEN'( ERROR > TOL ! FAIL(10,"SOL INT EQN") )
                      ;(MONITOR ! PRINT((NEWLINE,
                      " MULTIGRID CONVERGENCE TOO SLOW ")))
  'ELIF' ERROR  > TOL
  'THEN' 'FOR'  M 'TO' LEVELS
         'WHILE'DENV2:=NUMV2;  NUMV2:= NORM(RESTRICT(UM)-UMM1);
                'REAL' RT    := MIN((RATIO,V2));
                'REAL' WM    = # V1 ON LEVEL(M) #
                               ('REAL' VP:= V1, WP:= V1;
                                'FOR' P 'FROM' 2 'TO' M
                                'DO' VP:= RATIO*VP;
                                     WP:= VP+WP**GAMMA[P]*(VP+NORMT)
                                'OD'; WP );
                'REAL' V1M   := WM;
                'FOR'  IMAX 'TO' 5
                'WHILE'NUMV1 > 0.1*((1.0-V1M)/V1M)*(RT/(1.0-RT))*NUMV2
                       # EXTRA ITERATION: #
                'DO'   DENV1:= NUMV1;
                       UMOLD:= 'COPY' UM;
                       MGCYCLE(M,1,UM,RHS,'FALSE');
                       NUMV1:= NORM(UM-UMOLD);
                       NUMV2:= NORM(RESTRICT(UM)-UMM1);
                       V1M  := MIN((WM,NUMV1/DENV1))
                'OD'   ;
                V2   := MAX((RATIO,MIN((0.5,NUMV2/DENV2))));
                ERROR:= (V2/(1-V2))*NUMV2;
                ERROR>TOL 'AND' M<LEVELS
         'DO'   RHS  := PROJECT FORCEY (M+1);
                UMM1 := 'COPY' UM;
                UM   := PROLONGATE(UMM1);
                UMOLD:= 'COPY' UM;
                MGCYCLE(M+1,1,UM,RHS,'FALSE');
                NUMV1:= NORM(UM-UMOLD)
         'OD'
  'FI'

  'END' # SOLVE INT EQ #   ;

'PR' PROG 'PR' 'SKIP'
'END'
 'BEGIN'
 # EEN AANTAL TESTPROBLEMEN. ( RAPPORT NW 99/80 ) #
 # PROBLEM PARAMETERS #
 'REAL' B, BB, PAR, EB, LINV;
 'PROC' ('REAL')'REAL' SOL;
 'INT' KEVAL:= 0;

 []'PROC''VOID' PROBLEM  = (
        'VOID':      # CASE( I )   #
         ( KK := ('REAL'S,T,Y)'REAL':
          (KEVAL+:= 1;
           (S < T ! -Y*LINV*S*(1.0-T) ! -Y*LINV*T*(1.0-S)) );
           FORCEY :=   ('REAL'S )'REAL':
          ('REAL'R  = PAR;
           'REAL'SR:= S**R;
           'REAL'T1 = SR*(1.0-S);
                SR:= S*SR;
           'REAL'T2 = (SR-1)/(R+1)+(2-SR*(S+1))/(R+2)+(S*SR-1)/(R+3);
           R*R*(T1-LINV*S*T2) );
           SOL:= ('REAL'X)'REAL':
          ( 'REAL'R = PAR; R*R*(1.0-X)*X**R )       )       ,

       'VOID':      # CASE( II )  #
       (  KK := ('REAL'S,T,U)'REAL':
          (KEVAL+:= 1; LINV*U*COS(PI*S*T));
           SOL:= ('REAL'X)'REAL': EXP(X)*COS(7.0*X);
           FORCEY:= ('REAL'X)'REAL':
          ('REAL' ALFA = PI*X-7.0, BETA = PI*X+7.0;
```

```
        SOL(X)-0.5*LINV*
       ((EB*(COS(ALFA*B)+ALFA*SIN(ALFA*B))-1.0)/(1.0+ALFA*ALFA)+
        (EB*(COS(BETA*B)+BETA*SIN(BETA*B))-1.0)/(1.0+BETA*BETA))
       )          ) ,

    'VOID':       # CASE( III ) #
    (  KK := ('REAL'S,T,Y)'REAL':
      (KEVAL+:=1; PAR/(PAR*PAR+(S-T)*(S-T))*LINV*Y );
       FORCEY :=    ('REAL'S )'REAL':
      ('REAL'T1 = LN((PAR*PAR+(1.0-S)**2)/(PAR*PAR+S*S));
       'REAL'T2 = ARCTAN((1.0-S)/PAR)+ARCTAN(S/PAR);
       'REAL'T3 = PAR+PAR*(S-0.4)*T1+(0.06-PAR*PAR-0.8*S+S*S)*T2;
       -LINV*T3+0.06+S*(S-0.8) );
       SOL:= ('REAL' X)'REAL': 0.06+X*(X-0.8)      ) ,

    'VOID':
     ( KK := ('REAL'S,T,Y)'REAL':
      (KEVAL+:= 1; Y );
       FORCEY :=    ('REAL'S )'REAL':
                    ( 0.2<-S ! 1.0 !: 0.0<=S ! 5*S ! 0.0 );
       SOL:= ('REAL'X)'REAL': FORCEY(X) + BB      )                );

'INT' PRNR:= 2;
PAR := 5.0;
LINV:= -1.42;
B   := 2.0;

N0 := 8;
TOL:= 1.0E-5;
PROBLEM:= CUB INT;  NORMT:= 24;  Q:= SIMP;  ALFA:= 4;

INT:= (0.0,B); EB:= EXP(B); BB := (B-0.1)/(1-B);
'REAL' TIME:= CLOCK;
PROBLEM[PRNR];
'TO' 10 'DO' PRINT((NEWLINE," PROBLEM    ",WHOLE(PRNR,0))) 'OD';
PRINT((NEWLINE,NEWLINE," INTERVAL= ",INTERVAL));
PRINT((NEWLINE," LINV    = ",LINV));
PRINT((NEWLINE," PAR     = ",PAR));
PRINT((NEWLINE," TOL     = ",TOL,NEWLINE));

'REAL'ERROR; 'VEC' RESULT;
'IF'   SOLVE INT EQ (N0,32,256,TOL,ALFA,NORMT,RESULT,ERROR)
'THEN' PRINT((NEWLINE," USEFULL RESULT ! ",NEWLINE));
       'INT' NL  = 'UPB' RESULT;
       'INT' LEV = #LEVEL(NL)# 'ROUND'(LN(NL/N0)/LN(2.0));

       PRINT((NEWLINE," N0      = ",N0));
       PRINT((NEWLINE," LEVEL   = ",LEV));
       PRINT((NEWLINE," NL      = ",NL));
       PRINT((NEWLINE," KEVAL   = ",KEVAL,NEWLINE));

       PRINT((NEWLINE," ESTIMATED ERROR = ",ERROR));
       'VEC' ERR = RESULT - PINJECT(LEV,SOL);
       PRINT((NEWLINE,"    ACTUAL ERROR = ",NORM(ERR),NEWLINE));
       PRINT((NEWLINE,"     COEFFICIENT = ",KEVAL/(NL*NL)));
       PRINT((NEWLINE,NEWLINE," RESULT : "));  PRVEC(RESULT)
    #; PRINT((NEWLINE,NEWLINE," ERR    : "));  PRVEC(ERR) #
'ELSE' PRINT((NEWLINE," USELESS RESULT ! ",NEWLINE));
       PRINT((NEWLINE," N0      = ",N0));
       PRINT((NEWLINE," KEVAL   = ",KEVAL,NEWLINE));

       PRINT((NEWLINE," ESTIMATED ERROR = ",ERROR))
'FI'   ;
PRINT((NEWLINE," CP TIME = ",CLOCK-TIME))
'END'
'BEGIN'

# DETERMINATIE VAN DE MAXIMAAL SLECHTSTE CONVERGENTIE SNELHEID. #

'PROC' F = ('REAL'X,ALFA,[]'INT'GAMMA,'INT'LEVELS) 'BOOL':
'BEGIN'
    'BOOL'BB:= 'TRUE';
    'REAL'VP:= X;
    'REAL'WP:= X;
    'IF' (WP*WP) > VP 'THEN' BB:= 'FALSE' 'FI';
    'IF' BB 'THEN'
       'FOR' P 'FROM' 2 'TO' LEVELS 'WHILE'
          VP:= RATIO*VP;
          WP:= VP+(WP**GAMMA[P])*(VP+ALFA);
          BB:= ( WP*WP <= VP );
          BB
          'DO' 'SKIP' 'OD'
    'FI';
 BB
'END'  ;

'REAL'A,B,C,EPS,RATIO,ALFA; 'INT' I,J;
[1:5]'INT' GAMMA:= (0,3,3,3,3);
[1:4]'REAL' ARRALF; ARRALF:= (1.0,6.0,24.0,100.0);
[1:4]'REAL' ARRAT ; ARRAT := (6.25E-2,2.5E-1,5.0E-1,7.5E-1);
EPS:= 1.0E-8;

'FOR' I 'TO' 2 'DO'
RATIO:= ARRAT[I];

'FOR' J 'FROM' 2 'TO' 3 'DO'
ALFA := ARRALF[J];

    PRINT((NEWPAGE,"                                RATIO = ",RATIO,
           " ALFA = ",ALFA));
    'FOR' LEVELS 'TO' 'UPB'GAMMA 'DO'
    A:= 0.0;
    B:= 1.0;

    PRINT((NEWLINE,"***************************",NEWLINE,
           " LEVELS = ",LEVELS,"   GAMMA =   "));
    'FOR' LL 'TO' LEVELS 'DO' PRINT((WHOLE(GAMMA[LL],4))) 'OD';
    'IF' F(A,ALFA,GAMMA,LEVELS) 'AND' 'NOT'
         F(B,ALFA,GAMMA,LEVELS) 'THEN'
       'WHILE'
          C:= (A+B)/2.0;
          'IF' F(C,ALFA,GAMMA,LEVELS) 'THEN' A:= C 'ELSE' B:= C 'FI';
          (B-A) > EPS
          'DO' 'SKIP' 'OD'
    'FI'   ;
    PRINT((NEWLINE," A = ",A,NEWLINE," B = ",B))
    'OD'
'OD'
'OD'
'END'
```

```
FALIB,CM200000,IO77,T140,NP.
ACCOUNT[redacted]
FTN.
A68,N,D.
EDITLIB.
CATALOG,NEW,FALIB,ID=PHEMKER.
RETURN,NEW,LGO.
ATTACH,FALIB,ID=PHEMKER.
ATTACH,IMSL.
LIBRARY,IMSL.
ATTACH,T,FATEST,ID=PHEMKER.
SKIPF,T,1.
A68,I=T,P=FALIB/FALIB.
LGO.
RETURN,LGO.
COPYSBF.
      SUBROUTINE EVV(M,BMAX)
      COMPLEX M(4,4)
      INTEGER IJOB,IER
      REAL WK(40)
      COMPLEX A(4,4),W(4),Z(4,4)
      IJOB = 2

      DO 11 J=1,4
      DO 11 I=1,4
         A(I,J) = M (J,I)
   11 CONTINUE

      CALL EIGCC(A,4,4,IJOB,W,Z,4,WK,IER)
      BMAX= 0.0
      IF (WK(1).GE.100.0) BMAX= 1.0E10
      DO 14 J= 1,4
      BABS = CABS(W(J))
      IF (BABS.GE.BMAX) BMAX = BABS
   14 CONTINUE
      RETURN
      END
          IDENT     UERTST
          ENTRY     UERTST
 UERTST   BSS       1
          MESSAGE   MESS,LOCAL,RECALL
          EQ        UERTST
 MESS     DATA      8LUERTST
          END
 FALIB:
 'BEGIN' # FATEXT: NIEUWE VERSIE FA VOOR TLA /PWH830105 #
         # MGANAL01/PWH800112,        PF= MGANAL      #
         #                                VSN 810116 #
         # COMBINES: CEVAL/PWH800108, MCLEV/PWH801219 #
         # FOURIER ANALYSE ETC.      MCFOUR/PWH801203 #
         #                                VSN 801209 #

  'MODE' 'STAR' = [-1:1,-1:1]'REAL';
  'MODE' 'CAP'  = [1:4,1:4]'COMPL';
  'MODE' 'CAPD' = [1:4]'COMPL';

  'PROC' MAXEVAL = ('REF'[,]'COMPL' M)'REAL':
  'BEGIN' 'PROC' IMSL = ('REF''COMPL'M,
                         'REF''REAL' MAX)'VOID':
          'PR' XREF A68FTN,EVV 'PR' 'SKIP';
          'REAL' MAX; IMSL(M[1,1],MAX); MAX
  'END' ;

  'PRIO' 'PLOT' = 1;
  'OP' 'PLOT' = ('INT' N, 'REF'[]'REAL' RSD)'VOID':
  # PLAATJE VAN FOURIER GETRANSFORMEERDE          PWH830223  #
  # E.G.    16'PLOT'(DECILUSTAR(ANISOP(EPS,PI*PHII)))         #
  'BEGIN' [-N:N,-N:N]'REAL' A; 'REAL' M:= 1.0E-100;
     'STRING' S;

     'FOR' I 'TO' N 'DO'
     'FOR' J 'TO' N 'DO'
       'CAPD' C:= RELFT (RSD,I*PI/N,J*PI/N);
       A[I  ,J  ]:= 'ABS' C[1];
       A[I-N,J  ]:= 'ABS' C[2];
       A[I  ,J-N]:= 'ABS' C[3];
       A[I-N,J-N]:= 'ABS' C[4];
       'FOR' K 'TO' 4
       'DO' ('ABS'C[K]>M ! M:='ABS'C[K] ) 'OD'
     'OD' 'OD';

     A[,-N]:= A[, N];
     A[-N,]:= A[N, ];    ( PLOTABS ! M:= 1.0 );

     'FOR' I 'FROM' -N 'TO' N 'DO'  S:= " ";
     'FOR' J 'FROM' -N 'TO' N 'DO'  S+:= WHOLE(100*A[I,J]/M,-3)
                                    'OD'; PRINT((NEWLINE,S))
                                    'OD'; PRINT((M,NEWLINE,NEWLINE))
  'END';

  'BOOL' PLOTABS:= 'FALSE';

                                                          'PR' EJECT 'PR'
  'INT' MAZEN:= 16; 'REAL' MAZE:= 1.00E-10;

  'PROC' SUP = ( 'PROC'('REAL','REAL')'REAL' FUNC,
                 'REF'  'REAL'  XMAX, YMAX )'REAL':
  'BEGIN' # NEW VERSION :    PWH/830520      #
        'INT' II:= 0,JJ:= 0,ZI,ZJ;
        'REAL' H:= PI/(2*MAZEN), M, MAX:= 0.0;
        'FOR' I 'FROM' 1-MAZEN 'TO' MAZEN 'DO'
        'FOR' J 'FROM' 1-MAZEN 'TO' MAZEN 'DO'
               M:= FUNC(I*H,J*H);
               (M>MAX ! MAX:=M; II:=I; JJ:=J)
        'OD' 'OD';
        XMAX:= II*H; YMAX:= JJ*H;

        'WHILE' H > MAZE
        'DO' H/:= 2; ZI:= II*:= 2; ZJ:= JJ*:=2;

             'FOR' I 'FROM' ZI-3 'TO' ZI+3 'DO'
             'FOR' J 'FROM' ZJ-3 'TO' ZJ+3 'DO'
             'IF' 'ODD'I 'OR' 'ODD'J
             'THEN' M:= FUNC(I*H,J*H);
                    (M>MAX ! MAX:=M; II:=I; JJ:=J)
             'FI' 'OD' 'OD';
             XMAX:= II*H; YMAX:= JJ*H;

             # PERIODICITY ON THE TORUS !!!! #
             ('ABS' XMAX>PI/2 ! XMAX-:= PI*'SIGN'XMAX);
             ('ABS' YMAX>PI/2 ! YMAX-:= PI*'SIGN'YMAX)
        'OD'; MAX
   'END' # SUP # ;


                                                          'PR' EJECT 'PR'
  'PROC' TLA FUNC = ('REAL'PH,TH) 'REAL':
   ( (PH=0!TH=0!'FALSE') ! 0.0 ! MAXEVAL( TLA(PH,TH) ) );
```

```
'PROC' TLA CAP FUNC = ('REAL'PH,TH) 'REAL':
 ( (PH=0!TH=0!'FALSE') ! 0.0 ! MAXEVAL( TLA CAP (PH,TH) ) );

'REAL' PIH= PI/2;

'PROC' SMO FUNC = ('REAL'PH,TH)  REAL':
'BEGIN' 'REAL' P:= PH, T:= TH;
        ( P > PIH ! P-:= PIH !: P <-PIH ! P +:= PIH );
        ( T > PIH ! T-:= PIH !: T <-PIH ! T +:= PIH );
        'CAPD' A = RELFT(RELSTARDEC,P,T);
        'REAL' S,M:=0;
        # ALS (PH,TH) OVER HET BINNENGEBIED LOOPT,        #
        # KRYGEN WE HET MAXIMUM OVER HET BUITENGEBIED !!! #
        'FOR' I 'FROM' 2 'TO' 4
        'DO' S:= 'ABS' A[I]; (S>M!M:=S) 'OD';
        M
'END';

'PROC' SMO ZEBRA FUNC = ('REAL'PH,TH) 'REAL':
'BEGIN' 'REAL' P:= PH, T:= TH;
        ( P > PIH ! P-:= PIH !: P <-PIH ! P +:= PIH );
        ( T > PIH ! T-:= PIH !: T <-PIH ! T +:= PIH );
        'CAP' B    = RELFTCAP(RELSTARDEC,P,T);
        'CAP' A    := B                       ;
        'FOR' I 'FROM' 2 'TO' PP+QQ
        'DO'  A:= A*B  'OD'                   ;
        'REAL' S   := 'ABS' A[2,2]            ,
               M   := 'ABS' (A[3,3]+A[4,4])   ;
        # ALS (PH,TH) OVER HET BINNENGEBIED LOOPT,        #
        # KRYGEN WE HET MAXIMUM OVER HET BUITENGEBIED !!! #
        (S>M!(PP+QQ)'ROOT'S!(PP+QQ)'ROOT'M)
'END';


                                                  'PR' EJECT 'PR'

'OP'   'ROOT' = ('INT' P   ,'REAL' A)         'REAL':
( A< -1.0E-12 ! ( P'MOD'2=1!-EXP(LN('ABS'A)/P)
                          ! ERROR;0.0
                )
              ! ( A     <=0!       0.0
                          ! EXP(LN(     A)/P)
                )
);

'PRIO' 'ROOT' = 8;

'OP' + = ('CAPD' B)'CAP':
'BEGIN' 'HEAP' 'CAP' A;
        'FOR' I 'TO' 4 'DO'
        'FOR' J 'TO' 4 'DO'
            A[I,J]:= (I=J!B[I]!0.0) 'OD' 'OD'; A
'END';

'OP' '* M' = ('CAP' M )'CAP':
'BEGIN' 'HEAP''CAP' N;
       'FOR' I 'TO' 4 'DO'
       'FOR' J 'TO' 4 'DO'
       'BEGIN' 'COMPL' S:= 0;
               'FOR' K 'TO' 4 'DO' S+:= M[K,J]*'CONJ'M[K,I] 'OD';
               N[I,J]:= S
       'END' 'OD' 'OD'; N
'END' ;

'OP' 'MMH' = ('CAP' M )'CAP':
'BEGIN' 'HEAP''CAP' N;
       'FOR' I 'TO' 4 'DO'
       'FOR' J 'TO' 4 'DO'
       'BEGIN' 'COMPL' S:= 0;
               'FOR' K 'TO' 4 'DO' S+:= M[I,K]*'CONJ'M[J,K] 'OD';
               N[I,J]:= S
       'END' 'OD' 'OD'; N
'END' ;

'OP' * = ('CAP' M1,M2)'CAP':
'BEGIN' 'HEAP''CAP' N;
       'FOR' I 'TO' 4 'DO'
       'FOR' J 'TO' 4 'DO'
       'BEGIN' 'COMPL' S:= 0;
               'FOR' K 'TO' 4 'DO' S+:= M1[I,K]*M2[K,J] 'OD';
               N[I,J]:= S
       'END' 'OD' 'OD'; N
'END' ;

'OP' * = ('CAPD' M1,'CAP' M2)'CAP':
'BEGIN' 'HEAP''CAP' N;
       'FOR' I 'TO' 4 'DO'
       'FOR' J 'TO' 4 'DO'
        N[I,J]:=  M1[I]*M2[I,J]
       'OD' 'OD'; N
'END' ;

'OP' * = ('CAP' M1,'CAPD' M2)'CAP':
'BEGIN' 'HEAP'[1:4,1:4]'COMPL' N;
       'FOR' I 'TO' 4 'DO'
       'FOR' J 'TO' 4 'DO'
        N[I,J]:= M1[I,J]*M2[J]
       'OD' 'OD'; N
'END' ;

'OP' * = ('CAPD' M1,M2)'CAPD':
'BEGIN' 'CAPD' N;
       'FOR' I 'TO' 4 'DO'
        N[I]:= M1[I]*M2[I]
       'OD'; N
'END' ;

'OP' ** = ('CAPD' M1,M2)'CAP':
'BEGIN' 'HEAP''CAP' N;
       'FOR' I 'TO' 4 'DO'
       'FOR' J 'TO' 4 'DO'
        N[I,J]:= M1[I]*M2[J]
       'OD' 'OD'; N
'END' ;

'OP' 'PRINT' = ('COMPL' V)'VOID':
 PRINT(( FLOAT('RE' V,10,4,2)," +I* ",FLOAT('IM' V,10,4,2),"   "));

'OP' 'PRINT' = ('CAPD' V)'VOID':
( PRINT(NEWLINE); 'FOR' J 'TO' 4 'DO' 'PRINT'V[J] 'OD');

'OP''PRINT' = ('CAP' M)'VOID':
( PRINT(NEWLINE); 'FOR' I 'TO' 4 'DO' 'PRINT'M[I,] 'OD');

'OP' 'PRINT' = ('STAR' SR)'VOID':
'BEGIN''FOR' I 'FROM' -1 'TO' 1 'DO' 'STRING' S:= " ";
       'FOR' J 'FROM' -1 'TO' 1 'DO' S +:= FL(SR[I,J])
```

```
                                'OD';PRINT((NEWLINE,S ))
                                'OD';PRINT((NEWLINE   ))
'END'       ;

'PROC' FL = ('REAL' A)'STRING': FLOAT(A,11,5,2)+"   ";

                                                        'PR' EJECT 'PR'
# GLOBAL PARAMETERS #

'BOOL' SIGMA:='FALSE', BAR:= 'FALSE', SHOW := 'FALSE';
'INT' PP:=1, QQ:=0;
'REF'[]'REAL' RSTARDEC, PSTARDEC, AFSTARDEC, ACSTARDEC, RELSTARDEC;

'PROC' TLA = ('REAL' PH,TH) 'REF''CAP':
'BEGIN' 'CAPD' RELP,RELQ,
                P   := STARFT(PSTARDEC,PH,TH),
                R   := STARFT(RSTARDEC,PH,TH),
                RELX:= RELFT (RELSTARDEC,PH,TH),
                AH  := STARFT(AFSTARDEC,PH,TH);
        'COMPL'  AHH := STARFT(ACSTARDEC,2*PH,2*TH)[1] + 1.0E-300;
        'HEAP' 'CAP' M;

        'FOR' I 'TO' 4
        'DO' RELP[I]:= RELX[I]**PP;
             (BAR!P!R)[I] *:= AH[I]/AHH;
             RELQ[I]:= RELX[I]**QQ
        'OD';

        'FOR' I 'TO' 4
        'DO' 'FOR' J 'TO' 4
             'DO' M[I,J] :=   RELQ[I]
                            * ( (I=J!1!0) - P[I]*R[J] )
                            * RELP[J]
        'OD' 'OD';
        ( SIGMA ! M:= 'MHM' M ! M )
'END' ;
                                                        'PR' EJECT 'PR'
'PROC' TLA CAP = ('REAL' PH,TH) 'REF''CAP':
'BEGIN' 'CAPD'   P   := STARFT(PSTARDEC,PH,TH),
                 R   := STARFT(RSTARDEC,PH,TH),
                 AH  := STARFT(AFSTARDEC,PH,TH);
        'COMPL'  AHH := STARFT(ACSTARDEC,2*PH,2*TH)[1] + 1.0E-300;
        'CAP'    REL := + RELFT(RELSTARDEC,PH,TH);
        'HEAP''CAP' M;

        'FOR' I 'TO' 4
        'DO' (BAR!P!R)[I] *:= AH[I]/AHH  'OD';

        'FOR' I 'TO' 4 'DO'
        'FOR' J 'TO' 4 'DO'
           M[I,J] := (I=J!1!0) - P[I]*R[J] 'OD' 'OD';

        'FOR' I 'TO' PP 'DO' M:= M*REL 'OD';
        'FOR' I 'TO' QQ 'DO' M:= REL*M 'OD';
        ( SIGMA ! M:= 'MHM' M ! M )
'END' ;
                                                        'PR' EJECT 'PR'
'PROC' DECSTAR = ('STAR' S)'REF'[]'REAL':
('HEAP'[1:9]'REAL'A :=
       (-S[1,1]+S[-1,-1] ,-S[1,0]+S[-1,0],-S[1,-1]+S[-1,1],
        -S[0,1]+S[ 0,-1] , S[0,0]        , S[0,1]+S[ 0,-1],
         S[1,-1]+S[-1,1] , S[1,0]+S[-1,0], S[1,1]+S[-1,-1]);
        'BOOL' SYM:= 'TRUE';
        'FOR' I 'TO' 4 'DO' ( A[I]/=0 ! SYM:= 'FALSE' ) 'OD';
 ( SYM ! A[5:9@0] ! A[@-4] ) );

'PROC' FOURDECSTAR = ([]'STAR' ST )[][]'REAL':
       ( DECSTAR( ST[1] ), DECSTAR( ST[2] ),  # VOLGORDE :     (X ,Y ),#
         DECSTAR( ST[3] ), DECSTAR( ST[4] )); # (X+,Y),(X,Y+),(X+,Y+) #

'PROC' STARFT = ([]'REAL'A,'REAL'PH,TH)'CAPD':
'BEGIN' 'REAL' A0:= A[0], A1:= A[1]*COS(TH), A2:= A[2]*COS(PH-TH),
                          A3:= A[3]*COS(PH), A4:= A[4]*COS(PH+TH);
        'REAL' R1 = A0 + A1 + A2 + A3 + A4,
               R2 = A0 + A1 - A2 - A3 - A4,
               R3 = A0 - A1 - A2 + A3 - A4,
               R4 = A0 - A1 + A2 - A3 + A4;
'IF' 'LWB'A = 0
'THEN' (R1,R2,R3,R4)
'ELSE'  A1:= A[-1]*SIN(TH); A2:= A[-2]*SIN(PH-TH);
        A3:= A[-3]*SIN(PH); A4:= A[-4]*SIN(PH+TH);
        ( R1 +* ( A1 + A2 + A3 + A4 ),
          R2 +* ( A1 - A2 - A3 - A4 ),
          R3 +* (-A1 - A2 + A3 - A4 ),
          R4 +* (-A1 + A2 - A3 + A4 ))
'FI' 'END' # STARFT #;

'PROC' RELFT:= ([]'REAL'A,'REAL'PH,TH)'CAPD': ILUSTARFT(A,PH,TH);

'PROC' RELFTCAP:= ([]'REAL'A,'REAL'PH,TH)'CAP': ZEBRASTARFT(A,PH,TH);

'PROC' FOURSTARFT = ([][]'REAL' A, 'REAL'PH,TH) 'CAP':
'BEGIN' 'HEAP'  'CAP'   MM, LLAM;
        'REF'[,]'COMPL' M = MM[@0,@0], LAM = LLAM[@0,@0];

        'FOR' P 'TO' 4 'DO' LLAM[P,]:= STARFT(A[P],PH,TH) 'OD';

        'FOR' R1 'FROM' 0 'TO' 1 'DO' 'FOR' R2 'FROM' 0 'TO' 1 'DO'
        'FOR' K1 'FROM' 0 'TO' 1 'DO' 'FOR' K2 'FROM' 0 'TO' 1 'DO'
        'BEGIN' 'COMPL' S:= 0;
                'FOR' P1 'FROM' 0 'TO' 1 'DO'
                'FOR' P2 'FROM' 0 'TO' 1 'DO'
                      S +:=  ( K1=R1 ! 1 !: P1=0 ! 1 ! -1 ) *
                             ( K2=R2 ! 1 !: P2=0 ! 1 ! -1 ) *
                             LAM[ 2*P2+P1, 2*K2+K1]
                OD' 'OD';
               M[2*R2+R1,2*K2+K1]:= S/4
        'END' 'OD' 'OD' 'OD' 'OD';  MM
'END';
                                                        'PR' EJECT 'PR'
'BOOL' BACKWARD GS:= 'FALSE';

'PROC' DEC GS STAR = ( 'STAR'M) 'REF'[]'REAL':
'BEGIN' 'HEAP'[-4:4]'REAL'A; A[@1]:=
        ( M[-1,-1],M[-1,0],M[-1,1],
          M[ 0,-1],M[ 0,0],M[ 0,1],
          M[ 1,-1],M[ 1,0],M[ 1,1] ); A
'END'   ;

'PROC' GS STAR FT = ([]'REAL' A, 'REAL' PH,TH)'CAPD':
'BEGIN' 'REAL' AM4C:= COS( PH+TH), AM4S:= SIN( PH+TH),
               AM3C:= COS( PH   ), AM3S:= SIN( PH   ),
               AM2C:= COS( PH-TH), AM2S:= SIN( PH-TH),
               AM1C:= COS(    TH), AM1S:= SIN(    TH);
        'REAL' AP4C:= A[ 4]*AM4C,  AP4S:= A[ 4]*AM4S,
               AP3C:= A[ 3]*AM3C,  AP3S:= A[ 3]*AM3S,
               AP2C:= A[ 2]*AM2C,  AP2S:= A[ 2]*AM2S,
               AP1C:= A[ 1]*AM1C,  AP1S:= A[ 1]*AM1S;
```

```
        AM4C*:= A[-4]; AM3C*:= A[-3]; AM2C*:= A[-2]; AM1C*:= A[-1];
        AM4S*:= A[-4]; AM3S*:= A[-3]; AM2S*:= A[-2]; AM1S*:= A[-1];

       'IF' BACKWARD GS
       'THEN'
       (
         -(  (     AM4C+AM3C+AM2C+AM1C)+*( AM4S+AM3S+AM2S+AM1S)  )/
          (  (A[0]+AP4C+AP3C+AP2C+AP1C)+*(-AP4S-AP3S-AP2S-AP1S)  )  ,

         -(  (    -AM4C-AM3C-AM2C+AM1C)+*(-AM4S-AM3S-AM2S+AM1S)  )/
          (  (A[0]-AP4C-AP3C-AP2C+AP1C)+*( AP4S+AP3S+AP2S-AP1S)  )  ,

         -(  (    -AM4C+AM3C-AM2C-AM1C)+*(-AM4S+AM3S-AM2S-AM1S)  )/
          (  (A[0]-AP4C+AP3C-AP2C-AP1C)+*( AP4S-AP3S+AP2S+AP1S)  )  ,

         -(  (     AM4C-AM3C+AM2C-AM1C)+*( AM4S-AM3S+AM2S-AM1S)  )/
          (  (A[0]+AP4C-AP3C+AP2C-AP1C)+*(-AP4S+AP3S-AP2S+AP1S)  )
       )
       'ELSE'
       (
         -(  (     AP4C+AP3C+AP2C+AP1C)+*(-AP4S-AP3S-AP2S-AP1S)  )/
          (  (A[0]+AM4C+AM3C+AM2C+AM1C)+*(+AM4S+AM3S+AM2S+AM1S)  )  ,

         -(  (    -AP4C-AP3C-AP2C+AP1C)+*( AP4S+AP3S+AP2S-AP1S)  )/
          (  (A[0]-AM4C-AM3C-AM2C+AM1C)+*(-AM4S-AM3S-AM2S+AM1S)  )  ,

         -(  (    -AP4C+AP3C-AP2C-AP1C)+*( AP4S-AP3S+AP2S+AP1S)  )/
          (  (A[0]-AM4C+AM3C-AM2C-AM1C)+*(-AM4S+AM3S-AM2S-AM1S)  )  ,

         -(  (     AP4C-AP3C+AP2C-AP1C)+*(-AP4S+AP3S-AP2S+AP1S)  )/
          (  (A[0]+AM4C-AM3C+AM2C-AM1C)+*( AM4S-AM3S+AM2S-AM1S)  )
       )
       'FI'
'END'    ;

                                                        'PR' EJECT 'PR'
'PROC'('STAR')'REF'[]'REAL' DEC ZEBRA STAR = DEC GS STAR ;

'PROC' ZEBRA STAR FT = ([]'REAL' A, 'REAL' PH,TH)'CAP':
'BEGIN'
        'REAL' AM4C  := COS(PH+TH)   , AM4S := SIN(PH+TH)          ,
               AM3C  := COS(PH   )   , AM3S := SIN(PH   )          ,
               AM2C  := COS(PH-TH)   , AM2S := SIN(PH-TH)          ,
               AM1C  := COS(   TH)   , AM1S := SIN(   TH)          ;
        'REAL' AP4C  := A[4]*AM4C    , AP4S := A[4]*AM4S           ,
               AP3C  := A[3]*AM3C    , AP3S := A[3]*AM3S           ,
               AP2C  := A[2]*AM2C    , AP2S := A[2]*AM2S           ,
               AP1C  := A[1]*AM1C    , AP1S := A[1]*AM1S           ;
        AM4C*:= A[-4];  AM3C*:= A[-3]; AM2C*:= A[-2]; AM1C*:= A[-1];
        AM4S*:= A[-4];  AM3S*:= A[-3]; AM2S*:= A[-2]; AM1S*:= A[-1];

        'COMPL' A00 =
                     (  (    -AM4C-AM3C-AM2C-AP2C-AP3C-AP4C)
                      +*(    -AM4S-AM3S-AM2S+AP2S+AP3S+AP4S)  )  /
                     (  (A[0]+AM1C+AP1C                    )
                      +*(     AM1S-AP1S                    )  )    ,
               A01 =
                     (  (     AM4C-AM3C+AM2C+AP2C-AP3C+AP4C)
                      +*(     AM4S-AM3S+AM2S-AP2S+AP3S-AP4S)  )  /
                     (  (A[0]-AM1C-AP1C                    )
                      +*(    -AM1S+AP1S                    )  )    ;
        'COMPL'  E1 = A00*A00      ,
                 E2 = A01*A01      ;
        'COMPL'  E11= 0.5*(E1+A00) ,
                 E22= 0.5*(E1-A00) ,
                 E33= 0.5*(E2+A01) ,
                 E44= 0.5*(E2-A01) ;
       (  (E11,E11,0  ,0  ),
          (E22,E22,0  ,0  ),
          (0  ,0  ,E33,E33),
          (0  ,0  ,E44,E44)
       )
'END';

                                                        'PR' EJECT 'PR'
'INT'  ILUDECN1 := 5,  ILUDECN2 := 200;
'REAL' ILUDECTOL:= 1.0E-7; 'BOOL' ILUDECOK:= 'TRUE';

'PROC' ILUDEC = ('STAR' M, 'REF'[]'REAL' L,U)'REAL':
'BEGIN' 'REAL' TOL= ILUDECTOL;     ILUDECOK:= 'TRUE';
       'REAL'       M0:= M[0,0],
                    M1:= M[0, 1], MM1:= M[0,-1],
                    M2:= M[1,-1], MM2:= M[-1,1],
                    M3:= M[1, 0], MM3:= M[-1,0];
       'BOOL' NEG= M0<0;
                    ( NEG ! M0:=-M0; M1 :=- M1; M2 :=- M2; M3 :=- M3;
                                     MM1:=-MM1; MM2:=-MM2; MM3:=-MM3);
       'REF''REAL' LO = L[0]:=  M0,
                    L1 = L[1]:= MM1, U1 = U[1]:= M1,
                    L2 = L[2]:= MM2, U2 = U[2]:= M2,
                    L3 = L[3]:= MM3, U3 = U[3]:= M3,
                    L4 = L[4]      , U4 = U[4];
       'REAL' Z,ERR:= 1.0E100;

       'PRIO' <:= = 1;
       'OP'   <:= = ('REF''REAL' X,'REAL'A)'VOID':
       ('REAL' XX=X; X:= A; ERR +:= 'ABS'((X-XX)/(TOL+'ABS'X)) );

       Z := M0/2;
       LO:= 'ABS' Z + SQRT(Z*Z - L1*M1);
       # SQRT(<0) ALLEEN MOGELIJK BIJ  M0**2 < M1*MM1 !!!!!! #
       U1:= M1/LO;
       U3:= M3/LO;

       'TO' ILUDECN1 'WHILE' ERR>TOL
       'DO'    ERR:= 0.0;
               L2   <:= MM2 - U1*L3;
               U2   <:=( M2 - L1*U3)/LO;
               L1   <:= MM1 - U2*L3;
               U1   <:=( M1 - L2*U3)/LO;
               LO   <:=  M0 - L1*U1 - L2*U2 - L3*U3;
               U3   <:=  M3/LO
#$#            ;         DECN+:=1
       'OD';

       L4:=  U1*L2; U4:= L1*U2;

       'TO' ILUDECN2 'WHILE' ERR>TOL
       'DO'    ERR:= 0.0;
               Z     :=     LO -  U3*L3;
               L1   <:=(MM1*LO -  M2*L3)/Z;
               L2   <:=(MM2*LO -  M1*L3)/Z;
               U1   <:=( M1    - MM2*U3)/Z;
               U2   <:=( M2    - MM1*U3)/Z;
               Z     :=( M0 - L1*U1 - L2*U2)/2;
               LO   <:= 'ABS' Z + SQRT(Z*Z - L3*M3);
               U3   <:=  M3/LO
#$#            ;         DECN+:=1
```

```
        'OD';

        ILUDECOK:= ERR<= TOL;
        ( MONI 'AND' 'NOT' ILUDECOK
        ! PRINT((NEWLINE," ILUDEC !!!  ",ERR,TOL,
                 NEWLINE,M,NEWLINE,L,U,NEWLINE))
        );
        (NEG!LO:=-LO;L1:=-L1;L2:=-L2;L3:=-L3);
        L4:= U1*L2; U4:= L1*U2; ERR
'END'           ;

'PROC' DEC ILU STAR = ('STAR' S)'REF'[]'REAL':
'BEGIN' [0:4]'REAL' LL,UU;  ILUDEC(S,LL,UU);
        'REAL' L4 = LL[4],U4 = UU[4], E = 1.0E-9;
        'HEAP'[-4:4]'REAL' A;
         A  [-3:3@1] :=  (-S[1,0]+S[-1,0],-S[1,-1]+S[-1,1],
        -S[0,1]+S[ 0,-1] , S[0,0]        , S[0,1]+S[ 0,-1],
         S[1,-1]+S[-1,1] , S[1,0]+S[-1,0]);
         ( 'ABS'L4<E 'AND' 'ABS'U4<E
         ! A[-4]:= A[ 4]:= 0; A[0]+:= 1;
           ( MONI ! PRINT((NEWLINE," N.B. DECILUSTAR"))   )
         ! A[-4]:= L4-U4; A[ 4]:= L4+U4
         ); A
'END'       ;

'PROC' ILU STAR FT = ([]'REAL' A, 'REAL' PH,TH)'CAPD':
'BEGIN' 'REAL' A4C =  A[ 4]*COS(PH-2*TH),
               A4S =  A[-4]*SIN(PH-2*TH),
               A1C =  A[ 1]*COS(     TH),
               A1S =  A[-1]*SIN(     TH),
               A2C =  A[ 2]*COS(PH  -TH),
               A2S =  A[-2]*SIN(PH  -TH),
               A3C =  A[ 3]*COS(PH),
               A3S =  A[-3]*SIN(PH);
   (( A4C +* A4S )/ ( ( A[0]+A1C+A2C+A3C+A4C ) +* ( A1S+A2S+A3S+A4S ))
   ,(-A4C +*-A4S )/ ( ( A[0]+A1C-A2C-A3C-A4C ) +* ( A1S-A2S-A3S-A4S ))
   ,( A4C +* A4S )/ ( ( A[0]-A1C-A2C+A3C+A4C ) +* (-A1S-A2S+A3S+A4S ))
   ,(-A4C +*-A4S )/ ( ( A[0]-A1C+A2C-A3C-A4C ) +* (-A1S+A2S-A3S-A4S ))
   )
'END'    ;
                                                    'PR' EJECT 'PR'

  'PROC' ILLUDEC = ([]'REAL' JAC, 'REF'[]'REAL' DECOMP )'VOID':
  'BEGIN' 'REAL' TOL = ILUDECTOL;
          'REAL' DD,LL,II,L DINV U,S,ERR:= 1.0E100;

          'PRIO' <:= = 1;
          'OP'   <:= = ('REF''REAL' X,'REAL'A)'VOID':
          ('REAL' XX=X; X:= A; ERR +:= 'ABS'((X-XX)/(TOL+'ABS'X)) );

          'HEAP'[-1:+1]'REAL' D;
          [-2:+2]'REAL' DINV;
          [-1:+2]'REAL' L DINV;
          [-1:+1]'REAL' DEC;

          D[-1:+1]:= JAC[-1:+1];

          'TO' ILUDECN2 'WHILE' ERR>TOL
          'DO' LL:= D[0]/2;
               S := LL*LL-D[1]*D[-1];
               ( S<0 ! 'GOTO' EXEPTION );
               DD:= LL + SQRT(S);
               LL:= -D[-1]/DD;
               DEC [0]:= DD; DEC[-1]:= LL; DEC[1]:= -D[+1]/DD;
               DINV[0]:= (1/DEC[0])/(1.0-DEC[1]*DEC[-1]);
               'FOR' K 'TO' 2
               'DO'  DINV[-K]:= DINV[1-K]*DEC[-1];
                     DINV[ K]:= DINV[K-1]*DEC[+1]
               'OD';

               'FOR' K 'FROM' -1  'TO' 2
               'DO'  L DINV[K]:= JAC[-3]*DINV[K  ] +
                                 JAC[-2]*DINV[K-1]
               'OD';

               ERR:= 0;
               'FOR' K 'FROM' -1  'TO' 1
               'DO'  L DINV U := L DINV[K  ]*JAC[3] +
                                 L DINV[K+1]*JAC[2];
                     D[K] <:=   JAC[K] - L DINV U
               'OD'
         'OD';

      EXEPTION:
         ( MONI 'AND' 'NOT' (ILUDECOK:= ERR <= TOL )
         ! PRINT((NEWLINE," ILUDEC !!!  ",ERR,TOL,
                  NEWLINE,JAC,NEWLINE,DECOMP,NEWLINE))
         );
         DECOMP:= D
'END';

'PROC' DECILLUSTAR = ('STAR' STAR )'REF'[]'REAL':
'BEGIN'[-3:3]'REAL' JAC;
       JAC[-3:-2]:= STAR[-1, 0:1];
       JAC[-1: 1]:= STAR[ 0,-1:1];
       JAC[ 2: 3]:= STAR[ 1,-1:0];
       'HEAP'[-3:6]'REAL' C;
       C  [-3: 3]:= JAC [-3:3];
       ILLUDEC(JAC,C[4:6@-1]);
       C
'END';

'PROC' ILLU STAR FT = ([]'REAL' C, 'REAL' P,T )'CAPD':
( ILLUFT(C,P,T   ),ILLUFT(C,P+PI,T   ),
  ILLUFT(C,P,T+PI),ILLUFT(C,P+PI,T+PI)  );

'PROC' ILLUFT = ([]'REAL' C, 'REAL' P,T )'COMPL':
'BEGIN'[]'REAL' A = C[-3:3@-3], D = C[4:6@-1];
       'REAL'  SP = SIN(T  ), CP = COS(T  ),
               ST = SIN(P  ), CT = COS(P  ),
               SZ = SIN(P-T), CZ = COS(P-T);

        'COMPL'
           DB = ( D[ 0]+ (D[ 1]+D[-1])*CP )+*( (D[ 1]-D[-1])*SP ),
           CB = ( A[ 0]+ (A[ 1]+A[-1])*CP )+*( (A[ 1]-A[-1])*SP ),
           LB = ( A[-3]+ (A[-2]      )*CP )+*( (A[-2]      )*SP ),
           UB = ( A[ 3]+ (      A[ 2])*CP )+*( (     -A[ 2])*SP ),

           LPU= ( (A[3]+A[-3])*CT + (A[2]+A[-2])*CZ ) +*
                ( (A[3]-A[-3])*ST + (A[2]-A[-2])*SZ );

        (LB*UB+DB*(DB- CB))/
        (LB*UB+DB*(DB+LPU))
'END';

                                                    'PR' EJECT 'PR'
                          # SEVEN-POINT SIGESTAR #
```

```
#'PROC' SIGESTARS=('STAR'S,
                   'REF'[,]'STAR' PROLN,RESTR,DIF,CGC)'VOID':
 'BEGIN' 'REAL' MD1,MD2,MD3,MD4;
         'FOR' I 'FROM' 0 'TO' 1 'DO'
         'FOR' J 'FROM' 0 'TO' 1
         'DO' PROLN[I,J]:= 'HEAP''STAR';
              RESTR[I,J]:= 'HEAP''STAR';
              DIF  [I,J]:= 'HEAP''STAR';
              CGC  [I,J]:= 'HEAP''STAR'
         'OD' 'OD';

          MD1:= MD2:= MD3:= MD4:= -S[0,0];
          PROLN[@1,@1][@1,@1]:=
              (( UNITY,  ((         0,        0,       0    ),
                         (    S[0,-1]/MD2,   1,  S[0,1]/MD2),
                         (         0,        0,       0    )) ),
               ( ((0,S[-1,0]/MD3,0),
                  (0,     1     ,0),
                  (0,S[ 1,0]/MD3,0 )) ,
                        ((           0,        0,S[-1,1]/MD4),
                         (           0,        1,          0),
                         (S[ 1,-1]/MD4,        0,          0)) ));
          RESTR[@1,@1][@1,@1]:=
              ((  ((          0  ,S[-1,0]/MD3,S[-1,1]/MD4),
                   (S[ 0,-1]/MD2,     1     ,S[ 0,1]/MD2),
                   (S[ 1,-1]/MD4,S[ 1,0]/MD3,        0  )) ,
                  UNITY ), ( UNITY, UNITY ));
          DIF[@1,@1][@1,@1]:=
       ((     ((0,0,0),   (0,0,0),   (0,0,0)) ,
          ((     0     ,-S[-1,0]    ,-S[-1,1]),
           (     0     ,-S[ 0,0]-MD2,     0   ),
           (-S[ 1,-1],-S[ 1,0]     ,     0   )) ),
        ( ((     0     ,       0    ,-S[-1,1]),
           (-S[ 0,-1],-S[ 0,0]-MD3,-S[ 0,1]),
           (-S[ 1,-1],       0     ,     0   ))  ,
          ((     0     ,-S[-1,0]    ,     0   ),
           (-S[ 0,-1],-S[ 0,0]-MD4,-S[ 0,1]),
           (     0     ,-S[ 1,0]    ,     0   ))  ));

          CGC[@1,@1][@1,@1]:=
          (( ((                0  , S[-1,0]**2/MD3, S[-1,1]**2/MD4),
              ( S[ 0,-1]**2/MD2, S[ 0,0]       , S[ 0,1]**2/MD2),
              ( S[ 1,-1]**2/MD4, S[ 1,0]**2/MD3,            0  )),
             ((0,0,0),(0,-MD2,0),(0,0,0))                        ),
           (  ((0,0,0),(0,-MD3,0),(0,0,0))                       ,
              ((0,0,0),(0,-MD4,0),(0,0,0))                       ));

          CGC[0,0][0,0]+:= 2*( S[0, 1]* S[ 0,-1] /MD2 +
                               S[1,-1]* S[-1, 1] /MD4 +
                               S[1, 0]* S[-1, 0] /MD3  )
 'END';     #
                                                        'PR' EJECT 'PR'
 'STAR' UNITY,LAPLACE,LAPLACEC,HALFW,FULLW,LINW;
    UNITY  [@1,@1] := ((0, 0,0),( 0,1, 0),(0, 0,0));
    LAPLACE[@1,@ ] := ((0,-1,0),(-1,4,-1),(0,-1,0));
    LAPLACEC[@1,@1]:= ((0,-1/4,0),(-1/4,1,-1/4),(0,-1/4,0));
    HALFW  [@1,@1] := ((  0 ,1/8,  0 ),(1/8,1/2,1/8),(  0 ,1/8,  0 ));
    FULLW  [@1,@1] := ((1/16,1/8,1/16),(1/8,1/4,1/8),(1/16,1/8,1/16));
    LIN W  [@1,@1] := ((  0 ,1/8,1/8 ),(1/8,1/4,1/8),(1/8 ,1/8,  0 ));

 'PROC' M = ('REAL' A)'REAL':
 'IF' 'REAL' X, W:= 'ABS'A; W< 0.2
 'THEN' W*:= W; ((((  W - 9.9 ) * W + 99.0 ) * W
        - 1039.5 ) * W + 15592.5) * A /46777.5
 'ELSE' X:= ( W-1.0)/W;
        ( W< 18.0 ! X +:= 2.0/(EXP(W+W) - 1.0));
        ( A > 0.0 ! X ! -X )
  'FI'        ;

 'PROC' ILIN = ('REAL' EPSOH, ALF)'STAR':
 'BEGIN' 'REAL' S = 0.5*SIN(ALF)/EPSOH, C= 0.5*COS(ALF)/EPSOH;
         'HEAP' 'STAR' A; A[@1,@1]:=
 ((           0 ,              -1-C*M(C)-C  ,            0 ),
  ( -1-S*M(S)-S ,  2*( 2+S*M(S)+C*M(C)   ), -1-S*M(S)+S ),
  (            0 ,             -1-C*M(C)+C  ,            0 ));A
 'END'       ;


 'PROC' FEM = ('REAL' A11,A12,A22,A1,A2, H)'STAR':
 'BEGIN' 'REAL' X= H/6;
         'HEAP' 'STAR' A; A[@1,@1]:=
         ((     0         ,  -A11-A12+X*(-2*A1-A2), A12+X*(-A1+A2) ),
          (-A22-A12+X*(-A1-2*A2),
                            (A11+A12+A22)*2,
                                              -A22-A12+X*(A1+2*A2) ),
          ( A12+X*(A1-A2),  -A11-A12+X*( 2*A1+A2),               0 ));
         A
 'END';

 'PROC' ANISOP  = ('REAL' EPS,PHI) 'STAR':
 'BEGIN' 'REAL' S= SIN(PHI), C= COS(PHI);
         FEM( EPS*C*C+S*S, (EPS-1)*S*C, EPS*S*S+C*C, 0,0,0)
 'END';

 'PROC' CONVDIF = ('REAL' EPSOH,PHI) 'STAR':
 'BEGIN' 'REAL' S= SIN(PHI), C= COS(PHI);
         FEM( EPSOH, 0, EPSOH, C, S, 1)
 'END'       ;
                                                        'PR' EJECT 'PR'
 RSTARDEC := PSTARDEC:= DECSTAR(LIN W);
 AFSTARDEC:=           DECSTAR(LAPLACE);
 ACSTARDEC:=           DECSTAR(LAPLACEC);
#$# 'INT' DECN:= 0; 'BOOL' MONI:= 'TRUE';
 'PR' PROG 'PR' 'SKIP'
#$#              ;PRINT((NEWLINE," DECN : ",DECN,NEWLINE))
#'FOR' K 'FROM' 0 'TO' 1
 'DO' 'FOR' L 'FROM' 0 'TO' 8
      'DO' PRINT((NEWLINE," EPS: ",0.0001**K," PHI: ",L,"*PI/8"));
           'PRINT' ANISOP (0.0001**K,L*PI/8);
           'PRINT' CONVDIF(0.0001**K,L*PI/8)
 'OD' 'OD'


 'REAL' EPS,X,Y;
  RELFT:= ILUSTARFT;
 'FOR' P 'FROM' 3 'TO' 5
 'DO' 'FOR' L 'FROM' 0 'TO' 7
      'DO' EPS:= 0.1**P;
           PRINT((NEWLINE," EPS: ",EPS," L ",L));
           'STAR' S= ILIN(EPS,L*PI/4);
           'PRINT' S;
           RELSTARDEC:= DECILUSTAR(S);
           PRINT((NEWLINE,SUP(SMOFUNC,X,Y)));
           PRINT((" X ",X," Y",Y,NEWLINE))
 'OD' 'OD'
#'END'
LIBRARY(NEW,NEW)
ADD(*,LGO)
```

```
FINISH.
ENDRUN.
( # PLAATJE VAN FOURIER GETRANSFORMEERDE      PWH830223  #

    'REAL' PHII,EPS:= 0.01;
    'FOR' P 'FROM' 0 'TO' 6
    'DO' PHII:= P/16;
         PRINT((NEWLINE," PHI = PI* ",FIXED(PHII,6,5),
                        "     EPS = ",FLOAT(EPS,8,2,3),NEWLINE));
         12'PLOT'(DECILUSTAR(ANISOP(EPS,PI*PHII)))
    'OD'
 )
'BEGIN' #FATESTJE PWH/830113 #
 ILUDECOK:= 'TRUE';
#'PROC' MCOMP = ('REAL' PH,TH)'COMPL':
 ('REAL' CT=COS(TH), ST= SIN(TH),
         CP=COS(PH), SP= SIN(PH);
  ((CP+CT)+*(SP+ST))/ ( 4 - ((CP+CT)+*(SP-ST)) ));

 RELFT:= 'PROC' ('REAL' PH,TH)'CAPL':
 ((MCOMP(PH,TH),MCOMP(PH+PI,TH),MCOMP(PH,TH+PI),MCOMP(PH+PI,TH+PI)));
#

                                    MAZE:= 1.0E-5;
 'REAL' EPS,X,Y; 'INT' NARCS=12;
  RELFT:= ILLU STAR FT;
 'FOR' L 'FROM' 0 'TO' NARCS-1
 'DO' 'FOR' P 'FROM' 0 'TO' 5
      'DO' EPS:= 0.1**P;
           PRINT((NEWLINE," EPS: ",FL(EPS)," L ",FL(L)));
           'STAR' S= ANISOP(EPS,L*PI/NARCS);
           RELSTARDEC:= DEC ILLU STAR(S);
           'IF' ILUDECOK
           'THEN' PRINT(("   ",SUP(SMO FUNC,X,Y)));
                  PRINT((" X ",FL(X/PI)," Y",FL(Y/PI),NEWLINE))
           'ELSE' PRINT((" ILUDEC FAILURE ",NEWLINE))
           'FI'
 'OD' 'OD'
'END'
```

```
FANAL,CM140000,T77,NP.
ACCOUNT,[redacted]
ATTACH,FALIB,ID=PHEMKER.
A68,P=FALIB/FALIB.
ATTACH,IMSL.
LIBRARY,IMSL.
LGO.
'BEGIN' #FATEST PWH/830107 #
#'FOR' K 'FROM' 0 'TO' 1
 'DO' 'FOR' L 'FROM' 0 'TO' 8
      'DO' PRINT((NEWLINE," EPS: ",0.0001**K," PHI: ",L,"*PI/8"));
           'PRINT' ANISOP (0.0001**K,L*PI/8);
           'PRINT' CONVDIF(0.0001**K,L*PI/8)
 'OD' 'OD'
#

 RSTARDEC := PSTARDEC:= DECSTAR(LIN W);
 AFSTARDEC:=            DECSTAR(LAPLACE);
 ACSTARDEC:=            DECSTAR(LAPLACEC);
 PP:= QQ:= 1; SIGMA:= 'FALSE';

 'REAL' EPS,X,Y;
  RELFT:= ILUSTARFT;
 'FOR' P 'FROM' 0 'TO' 1
 'DO' 'FOR' L 'FROM' 0 'TO' 1
      'DO' EPS:= 0.33333333333**P;
           PRINT((NEWLINE," EPS: ",FL(EPS)," L ",L));
           'STAR' S= ANISOP(EPS,L*PI/2);
           RELSTARDEC:= DECILUSTAR(S);
           PRINT((NEWLINE,SUP(TLA FUNC,X,Y)));
           PRINT((" X ",FL(X/PI)," Y",FL(Y/PI),NEWLINE))
 'OD' 'OD'
 'END'
```