D03UBF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

D03UBF performs at each call one iteration of the Strongly Implicit Procedure. It is used to calculate on successive calls the sequence of approximate corrections to the current estimate of the solution when solving a system of simultaneous algebraic equations for which the iterative up-date matrix is of seven-point molecule form on a three-dimensional topologically-rectangular mesh. ('Topological' means that a polar grid, for example, can be used if it is equivalent to a rectangular box.)

2 Specification

```
 SUBROUTINE D03UBF(N1, N2, N3, N1M, N2M, A, B, C, D, E, F, G,
1                  APARAM, IT, R, WRKSP1, WRKSP2, WRKSP3, IFAIL)
 INTEGER           N1, N2, N3, N1M, N2M, IT, IFAIL
 real              A(N1M,N2M,N3), B(N1M,N2M,N3), C(N1M,N2M,N3),
1                  D(N1M,N2M,N3), E(N1M,N2M,N3), F(N1M,N2M,N3),
2                  G(N1M,N2M,N3), APARAM, R(N1M,N2M,N3),
3                  WRKSP1(N1M,N2M,N3), WRKSP2(N1M,N2M,N3),
4                  WRKSP3(N1M,N2M,N3)
```

3 Description

Given a set of simultaneous equations

$$Mt = q \tag{1}$$

(which could be nonlinear) derived, for example, from a finite difference representation of a three-dimensional elliptic partial differential equation and its boundary conditions, the solution t may be obtained iteratively from a starting approximation $t^{(1)}$ by the formulae

$$\begin{aligned} r^{(n)} &= q - Mt^{(n)} \\ Ms^{(n)} &= r^{(n)} \\ t^{(n+1)} &= t^{(n)} + s^{(n)}. \end{aligned}$$

Thus $r^{(n)}$ is the residual of the nth approximate solution $t^{(n)}$, and $s^{(n)}$ is the update change vector.

D03UBF determines the approximate change vector s corresponding to a given residual r, i.e., it determines an approximate solution to a set of equations

$$Ms = r \tag{2}$$

where M is a square $(n_1 \times n_2 \times n_3)$ by $(n_1 \times n_2 \times n_3)$ matrix and r is a known vector of length $(n_1 \times n_2 \times n_3)$. The equations (2) must be of seven-diagonal form:

$$a_{ijk}s_{ij,k-1} + b_{ijk}s_{i,j-1,k} + c_{ijk}s_{i-1,jk} + d_{ijk}s_{ijk} + e_{ijk}s_{i+1,jk} + f_{ijk}s_{i,j+1,k} + g_{ijk}s_{ij,k+1} = r_{ijk}$$

with $i = 1, 2, \ldots, n_1$; $j = 1, 2, \ldots, n_2$ and $k = 1, 2, \ldots, n_3$, provided that $d_{ijk} \neq 0.0$. Indeed, if $d_{ijk} = 0.0$, then the equation is assumed to be:

$$s_{ijk} = r_{ijk}.$$

The calling program supplies the current residual r at each iteration and the coefficients of the seven-point molecule system of equations on which the up-date procedure is based. The routine performs one iteration, using the approximate LU factorization of the Strongly Implicit Procedure with the necessary acceleration parameter adjustment, to calculate the approximate solution s of the system (2). The change s overwrites the residual array, for return to the calling program. The calling program must combine this change stored in r with the old approximation to obtain the new approximate solution for t. It must then recalculate the residuals and, if the accuracy requirements have not been satisfied, commence the next iterative cycle.

Clearly there is no requirement that the iterative up-date matrix passed in the form of the seven-diagonal element arrays A, B, C, D, E, F, G is the same as that used to calculate the residuals, and therefore the one governing the problem. However, the convergence may be impaired if they are not equal. Indeed, if the system of equations (1) is not precisely of the seven-diagonal form illustrated above but has a few additional terms, then the methods of deferred or defect correction can be employed. The residual is calculated by the calling program using the full system of equations, but the up-date formula is based on a seven-diagonal system (2) of the form given above. For example, the solution of a system of eleven-diagonal equations, each involving the combination of terms with $t_{i\pm1,j\pm1,k}$, $t_{i\pm1,j,k}$, $t_{i,j\pm1,k}$, $t_{i,j,k\pm1}$ and t_{ijk} could use the seven-diagonal coefficients on which to base the up-date, provided these incorporate the major features of the equations.

Problems in topologically non-rectangular-box-shaped regions can be solved using the routine by surrounding the region by a circumscribing topologically rectangular box. The equations for the nodal values external to the region of interest are set to zero (i.e., $d_{ijk} = r_{ijk} = 0$) and the boundary conditions are incorporated into the equations for the appropriate nodes.

If there is no better initial approximation when starting the iterative cycle, one can use an array of all zeros as the initial approximation from which the first set of residuals are determined.

The routine can be used to solve linear elliptic equations in which case Q and the arrays A, B, C, D, E, F and G will be unchanged during the iterative cycles. It can also be used for solving nonlinear elliptic equations in which case some or all of these arrays may require updating as each new approximate solution is derived. Depending on the nonlinearity, some under-relaxation of the coefficient and/or source terms may be needed during their recalculations using the new estimates of the solution (see Ames [1]).

The routine can also be used to solve each step of a time-dependent parabolic equation in three space dimensions. The solution at each time step can be expressed in terms of an elliptic equation if the Crank–Nicolson or other form of implicit time integration is used.

Neither diagonal dominance, nor positive-definiteness, of the matrix M or of the up-date matrix formed from the arrays A, B, C, D, E, F, G is necessary to ensure convergence.

For problems in which the solution is not unique, in the sense that an arbitrary constant can be added to the solution, for example Poisson's equation with all Neumann boundary conditions, the calling program should subtract a typical nodal value from the whole solution t at every iteration to keep rounding errors to a minimum for those cases when convergence is slow. For such problems there is generally an associated compatibility condition. For the example mentioned this compatibility condition equates the total net source within the region (i.e., the source integrated over the region) with the total net outflow across the boundaries defined by the Neumann conditions (i.e., the normal derivative integrated along the whole boundary). It is very important that the algebraic equations derived to model such a problem accurately implement the compatibility condition. If they do not, a net source or sink is very likely to be represented by the set of algebraic equations and no steady-state solution of the equations exists.

4 References

[1] Ames W F (1977) *Nonlinear Partial Differential Equations in Engineering* Academic Press (2nd Edition)

[2] Jacobs D A H (1972) The strongly implicit procedure for the numerical solution of parabolic and elliptic partial differential equations *Note RD/L/N66/72* Central Electricity Research Laboratory

[3] Stone H L (1968) Iterative solution of implicit approximations of multi-dimensional partial differential equations *SIAM J. Numer. Anal.* **5** 530–558

[4] Weinstein H G, Stone H L and Kwan T V (1969) Iterative procedure for solution of systems of parabolic and elliptic equations in three dimensions *Industrial and Engineering Chemistry Fundamentals* **8** 281–287

5 Parameters

1: N1 — INTEGER *Input*

On entry: the number of nodes in the first co-ordinate direction, n_1.

Constraint: N1 > 1.

2: N2 — INTEGER *Input*

On entry: the number of nodes in the second co-ordinate direction, n_2.

Constraint: N2 > 1.

3: N3 — INTEGER *Input*

On entry: the number of nodes in the third co-ordinate direction, n_3.

Constraint: N3 > 1.

4: N1M — INTEGER *Input*

On entry: the first dimension of all the three-dimensional arrays, as declared in the (sub)program from which D03UBF is called.

Constraint: N1M $\geq$ N1.

5: N2M — INTEGER *Input*

On entry: the second dimension of all the three-dimensional arrays, as declared in the (sub)program from which D03UBF is called.

Constraint: N2M $\geq$ N2.

6: A(N1M,N2M,N3) — ***real*** array *Input*

On entry: A(i,j,k) must contain the coefficient of $s_{ij,k-1}$ in the (i,j,k)th equation of the system (2) for $i = 1,2,\ldots,$N1; $j = 1,2,\ldots,$N2 and $k = 1,2,\ldots,$N3. The elements of A for $k = 1$ must be zero after incorporating the boundary conditions, since they involve nodal values from outside the box.

7: B(N1M,N2M,N3) — ***real*** array *Input*

On entry: B(i,j,k) must contain the coefficient of $s_{i,j-1,k}$ in the (i,j,k)th equation of the system (2) for $i = 1,2,\ldots,$N1; $j = 1,2,\ldots,$N2 and $k = 1,2,\ldots,$N3. The elements of B for $j = 1$ must be zero after incorporating the boundary conditions, since they involve nodal values from outside the box.

8: C(N1M,N2M,N3) — ***real*** array *Input*

On entry: C(i,j,k) must contain the coefficient of $s_{i-1,j,k}$ in the (i,j,k)th equation of the system (2), for $i = 1,2,\ldots,$N1; $j = 1,2,\ldots,$N2 and $k = 1,2,\ldots,$N3. The elements of C for $i = 1$ must be zero after incorporating the boundary conditions, since they involve nodal values from outside the box.

9: D(N1M,N2M,N3) — ***real*** array *Input*

On entry: D(i,j,k) must contain the coefficient of s_{ijk}, the 'central' term, in the (i,j,k)th equation of the system (2), for $i = 1,2,\ldots,$N1; $j = 1,2,\ldots,$N2 and $k = 1,2,\ldots,$N3. The elements of D are checked to ensure that they are non-zero. If any element is found to be zero, the corresponding algebraic equation is assumed to be $s_{ijk} = r_{ijk}$. This feature can be used to define the equations for nodes at which, for example, Dirichlet boundary conditions are applied, or for nodes external to the problem of interest, by setting D$(i,j,k) = 0.0$ at appropriate points. The corresponding value of r_{ijk} is set equal to the appropriate value, namely the difference between the prescribed value of t_{ijk} and the current value in the Dirichlet case, or zero at an external point.

10: E(N1M,N2M,N3) — ***real*** array *Input*

On entry: E(i,j,k) must contain the coefficient of $s_{i+1,j,k}$ in the (i,j,k)th equation of the system (2), for $i = 1,2,\ldots$,N1; $j = 1,2,\ldots$,N2 and $k = 1,2,\ldots$,N3. The elements of E for $i =$ N1 must be zero after incorporating the boundary conditions, since they involve nodal values from outside the box.

11: F(N1M,N2M,N3) — ***real*** array *Input*

On entry: F(i,j,k) must contain the coefficient of $s_{i,j+1,k}$ in the (i,j,k)th equation of the system (2), for $i = 1,2,\ldots$,N1; $j = 1,2,\ldots$,N2 and $k = 1,2,\ldots$,N3. The elements of F for $j =$ N2 must be zero after incorporating the boundary conditions, since they involve nodal values from outside the box.

12: G(N1M,N2M,N3) — ***real*** array *Input*

On entry: G(i,j,k) must contain the coefficient of $s_{i,j,k+1}$ in the (i,j,k)th equation of the system (2), for $i = 1,2,\ldots$,N1; $j = 1,2,\ldots$,N2 and $k = 1,2,\ldots$,N3. The elements of G for $k =$ N3 must be zero after incorporating the boundary conditions, since they involve nodal values from outside the box.

13: APARAM — ***real*** *Input*

On entry: the iteration acceleration factor. A value of 1.0 is adequate for most typical problems. However, if convergence is slow, the value can be reduced, typically to 0.2 or 0.1. If divergence is obtained, the value can be increased, typically to 2.0, 5.0 or 10.0.

Constraint: $0.0 <$ APARAM $\leq ((\mathrm{N1}-1)^2 + (\mathrm{N2}-1)^2 + (\mathrm{N3}-1)^2)/3.0$.

14: IT — INTEGER *Input*

On entry: the iteration number. It must be initialised, but not necessarily to 1, before the first call, and should be incremented by one in the calling program for each subsequent call. The routine uses this counter to select the appropriate acceleration parameter from a sequence of nine, each one being used twice in succession. (Note that the acceleration parameter depends on the value of APARAM).

15: R(N1M,N2M,N3) — ***real*** array *Input/Output*

On entry: the current residual r_{ijk} on the right-hand side of the (i,j,k)th equation of the system (2), $i = 1,2,\ldots$,N1; $j = 1,2,\ldots$,N2 and $k = 1,2,\ldots$,N3.

On exit: these residuals are overwritten by the corresponding components of the solution s of the system (2), i.e., the changes to be made to the vector T to reduce the residuals supplied.

16: WRKSP1(N1M,N2M,N3) — ***real*** array *Workspace*
17: WRKSP2(N1M,N2M,N3) — ***real*** array *Workspace*
18: WRKSP3(N1M,N2M,N3) — ***real*** array *Workspace*
19: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, N1 < 2,
or N2 < 2,
or N3 < 2.

IFAIL = 2

On entry, N1M < N1,
or N2M < N2.

IFAIL = 3

On entry, APARAM $\leq$ 0.0.

IFAIL = 4

On entry, APARAM > ((N1−1)^2 + (N2−1)^2 + (N3−1)^2)/3.0.

7 Accuracy

The improvement in accuracy for each iteration, i.e., on each call, depends on the size of the system and on the condition of the up-date matrix characterised by the seven-diagonal coefficient arrays. The ultimate accuracy obtainable depends on the above factors and on the ***machine precision***. However, since the routine works with residuals and the up-date vector, the calling program can calculate the residuals from extended precision values of the function, source term and equation coefficients if greater accuracy is required. The rate of convergence obtained with the Strongly Implicit Procedure is not always smooth because of the cyclic use of nine acceleration parameters. The convergence may become slow with very large problems. The final accuracy obtained can be judged approximately from the rate of convergence determined from the changes to the dependent variable T and in particular the change on the last iteration.

8 Further Comments

The time taken by the routine is approximately proportional to N1 × N2 × N3 for each call.

When used with deferred or defect correction, the residual is calculated in the calling program from a different system of equations to those represented by the seven-point molecule coefficients used by the routine as the basis of the iterative up-date procedure. When using deferred correction the overall rate of convergence depends not only on the items detailed in Section 7 but also on the difference between the two coefficient matrices used.

Convergence may not always be obtained when the problem is very large and/or the coefficients of the equations have widely disparate values. The latter case may be associated with a nearly ill-conditioned matrix.

9 Example

To solve Laplace's equation in a rectangular box with a non-uniform grid spacing in the x, y, and z co-ordinate directions and with Dirichlet boundary conditions specifying the function on the surfaces of the box equal to

$$e^{(1.0+x)/y(n_2)} \times \cos(\sqrt{2}y/y(n_2)) \times e^{(-1.0-z)/y(n_2)}.$$

Note that this is the same problem as that solved in the example for D03ECF. The differences in the maximum residuals obtained at each iteration between the two test runs are explained by the fact that in D03ECF the residual at each node is normalised by dividing by the central coefficient, whereas this normalisation has not been used in the example program for D03UBF.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     D03UBF Example Program Text
*     Mark 19 Revised. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          N1, N2, N3, N1M, N2M, NITS
      PARAMETER        (N1=4,N2=5,N3=6,N1M=N1,N2M=N2,NITS=10)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             ADEL, APARAM, ARES, DELMAX, DELMN, RESMAX, RESMN,
     +                 ROOT2
      INTEGER          I, IFAIL, IT, J, K
*     .. Local Arrays ..
      real             A(N1M,N2M,N3), B(N1M,N2M,N3), C(N1M,N2M,N3),
     +                 D(N1M,N2M,N3), E(N1M,N2M,N3), F(N1M,N2M,N3),
     +                 G(N1M,N2M,N3), Q(N1M,N2M,N3), R(N1M,N2M,N3),
     +                 T(N1M,N2M,N3), WRKSP1(N1M,N2M,N3),
     +                 WRKSP2(N1M,N2M,N3), WRKSP3(N1M,N2M,N3), X(N1),
     +                 Y(N2), Z(N3)
*     .. External Subroutines ..
      EXTERNAL         D03UBF
*     .. Intrinsic Functions ..
      INTRINSIC        ABS, COS, EXP, MAX, real, SQRT
*     .. Data statements ..
      DATA             X(1), X(2), X(3), X(4)/0.0e0, 1.0e0, 3.0e0,
     +                 6.0e0/
      DATA             Y(1), Y(2), Y(3), Y(4), Y(5)/0.0e0, 1.0e0, 3.0e0,
     +                 6.0e0, 10.0e0/
      DATA             Z(1), Z(2), Z(3), Z(4), Z(5), Z(6)/0.0e0, 1.0e0,
     +                 3.0e0, 6.0e0, 10.0e0, 15.0e0/
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D03UBF Example Program Results'
      WRITE (NOUT,*)
      ROOT2 = SQRT(2.0e0)
      APARAM = 1.0e0
*     Set up difference equation coefficients, source terms and
*     initial approximation
      DO 60 K = 1, N3
         DO 40 J = 1, N2
            DO 20 I = 1, N1
               IF ((I.NE.1) .AND. (I.NE.N1) .AND. (J.NE.1)
     +             .AND. (J.NE.N2) .AND. (K.NE.1) .AND. (K.NE.N3)) THEN
*                 Specification for internal nodes
                  A(I,J,K) = 2.0e0/((Z(K)-Z(K-1))*(Z(K+1)-Z(K-1)))
                  G(I,J,K) = 2.0e0/((Z(K+1)-Z(K))*(Z(K+1)-Z(K-1)))
                  B(I,J,K) = 2.0e0/((Y(J)-Y(J-1))*(Y(J+1)-Y(J-1)))
                  F(I,J,K) = 2.0e0/((Y(J+1)-Y(J))*(Y(J+1)-Y(J-1)))
                  C(I,J,K) = 2.0e0/((X(I)-X(I-1))*(X(I+1)-X(I-1)))
                  E(I,J,K) = 2.0e0/((X(I+1)-X(I))*(X(I+1)-X(I-1)))
                  D(I,J,K) = -A(I,J,K) - B(I,J,K) - C(I,J,K) - E(I,J,K)
     +                       - F(I,J,K) - G(I,J,K)
                  Q(I,J,K) = 0.0e0
                  T(I,J,K) = 0.0e0
               ELSE
*                 Specification for boundary nodes
```

```
                  A(I,J,K) = 0.0e0
                  B(I,J,K) = 0.0e0
                  C(I,J,K) = 0.0e0
                  E(I,J,K) = 0.0e0
                  F(I,J,K) = 0.0e0
                  G(I,J,K) = 0.0e0
                  D(I,J,K) = 0.0e0
                  Q(I,J,K) = EXP((X(I)+1.0e0)/Y(N2))*COS(ROOT2*Y(J)
     +                       /Y(N2))*EXP((-Z(K)-1.0e0)/Y(N2))
                  T(I,J,K) = 0.0e0
               END IF
   20       CONTINUE
   40    CONTINUE
   60 CONTINUE
*     Iterative loop
      WRITE (NOUT,*) 'Iteration      Residual                   Change'
      WRITE (NOUT,*)
     +  '   No      Max.        Mean           Max.      Mean'
      WRITE (NOUT,*)
      DO 200 IT = 1, NITS
         RESMAX = 0.0e0
         RESMN = 0.0e0
         DO 120 K = 1, N3
            DO 100 J = 1, N2
               DO 80 I = 1, N1
                  IF (D(I,J,K).NE.0.0e0) THEN
*                    Seven point molecule formula
                     R(I,J,K) = Q(I,J,K) - A(I,J,K)*T(I,J,K-1) - B(I,J,
     +                          K)*T(I,J-1,K) - C(I,J,K)*T(I-1,J,K) -
     +                          D(I,J,K)*T(I,J,K) - E(I,J,K)*T(I+1,J,K)
     +                          - F(I,J,K)*T(I,J+1,K) - G(I,J,K)*T(I,J,
     +                          K+1)
                  ELSE
*                    Explicit equation
                     R(I,J,K) = Q(I,J,K) - T(I,J,K)
                  END IF
                  ARES = ABS(R(I,J,K))
                  RESMAX = MAX(RESMAX,ARES)
                  RESMN = RESMN + ARES
   80          CONTINUE
  100       CONTINUE
  120    CONTINUE
         RESMN = RESMN/(real(N1*N2*N3))
         IFAIL = 0
*
         CALL D03UBF(N1,N2,N3,N1M,N2M,A,B,C,D,E,F,G,APARAM,IT,R,WRKSP1,
     +               WRKSP2,WRKSP3,IFAIL)
*
*        Update the dependent variable
         DELMAX = 0.0e0
         DELMN = 0.0e0
         DO 180 K = 1, N3
            DO 160 J = 1, N2
               DO 140 I = 1, N1
                  T(I,J,K) = T(I,J,K) + R(I,J,K)
                  ADEL = ABS(R(I,J,K))
                  DELMAX = MAX(DELMAX,ADEL)
                  DELMN = DELMN + ADEL
  140          CONTINUE
  160       CONTINUE
  180    CONTINUE
         DELMN = DELMN/(real(N1*N2*N3))
         WRITE (NOUT,99999) IT, RESMAX, RESMN, DELMAX, DELMN
*        Convergence tests here if required
  200 CONTINUE
*     End of iterative loop
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Table of calculated function values'
      WRITE (NOUT,*)
     +  'K  J   (I      T   ) (I      T   ) (I      T   ) (I      T   )'
      WRITE (NOUT,*)
      WRITE (NOUT,99998) ((K,J,(I,T(I,J,K),I=1,N1),J=1,N2),K=1,N3)
      STOP
*
99999 FORMAT (1X,I5,4(2X,e11.4))
99998 FORMAT ((1X,I1,I3,1X,4(1X,I3,2X,F8.3)))
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D03UBF Example Program Results

 Iteration      Residual                   Change
    No      Max.         Mean           Max.      Mean

     1   0.1822E+01   0.4847E+00   0.1822E+01   0.6173E+00
     2   0.8585E-02   0.9369E-03   0.1970E-01   0.1895E-02
     3   0.3168E-02   0.7783E-04   0.1496E-02   0.5819E-04
     4   0.4085E-04   0.2179E-05   0.3848E-04   0.1931E-05
     5   0.7820E-05   0.3999E-06   0.5481E-05   0.2312E-06
     6   0.2246E-06   0.1524E-07   0.2333E-06   0.1093E-07
     7   0.2219E-07   0.1669E-08   0.2222E-07   0.9131E-09
     8   0.2841E-08   0.1820E-09   0.1969E-08   0.9337E-10
     9   0.6696E-09   0.4762E-10   0.5873E-09   0.2450E-10
    10   0.7848E-10   0.4908E-11   0.5863E-10   0.2671E-11

 Table of calculated function values
 K  J   (I      T   ) (I      T   ) (I      T   ) (I      T   )

 1  1    1     1.000    2     1.105    3     1.350    4     1.822
 1  2    1     0.990    2     1.094    3     1.336    4     1.804
 1  3    1     0.911    2     1.007    3     1.230    4     1.661
 1  4    1     0.661    2     0.731    3     0.892    4     1.205
 1  5    1     0.156    2     0.172    3     0.211    4     0.284
 2  1    1     0.905    2     1.000    3     1.221    4     1.649
 2  2    1     0.896    2     0.990    3     1.210    4     1.632
 2  3    1     0.825    2     0.912    3     1.114    4     1.503
 2  4    1     0.598    2     0.662    3     0.809    4     1.090
 2  5    1     0.141    2     0.156    3     0.190    4     0.257
 3  1    1     0.741    2     0.819    3     1.000    4     1.350
 3  2    1     0.733    2     0.811    3     0.991    4     1.336
 3  3    1     0.675    2     0.747    3     0.913    4     1.230
 3  4    1     0.490   2     0.543   3     0.664   4     0.892
 3  5    1     0.116   2     0.128   3     0.156   4     0.211
 4  1    1     0.549   2     0.607   3     0.741   4     1.000
 4  2    1     0.543   2     0.601   3     0.734   4     0.990
 4  3    1     0.500   2     0.554   3     0.677   4     0.911
 4  4    1     0.363   2     0.402   3     0.492   4     0.661
 4  5    1     0.086   2     0.095   3     0.116   4     0.156
 5  1    1     0.368   2     0.407   3     0.497   4     0.670
 5  2    1     0.364   2     0.403   3     0.492   4     0.664
 5  3    1     0.335   2     0.371   3     0.454   4     0.611
 5  4    1     0.243   2     0.270   3     0.330   4     0.443
 5  5    1     0.057   2     0.063   3     0.077   4     0.105
 6  1    1     0.223   2     0.247   3     0.301   4     0.407
 6  2    1     0.221   2     0.244   3     0.298   4     0.403
 6  3    1     0.203   2     0.225   3     0.274   4     0.371
 6  4    1     0.148   2     0.163   3     0.199   4     0.269
 6  5    1     0.035   2     0.038   3     0.047   4     0.063
```