1 SUMMARY

This subroutine **triangulates an arbitrary set of points in the plane.** The subroutine is suited to grid generation for finite-element analysis and for the construction of contour plots.

ATTRIBUTES — Version: 1.0.0. **Types:** GA06A, GA06AD. **Calls:** FD05. **Original date:** May 1989. **Origin:** J. A.Scott, Harwell and S. W.Sloan, University of Newcastle, New South Wales.

2 HOW TO USE THE PACKAGE

2.1 The argument list

The single precision version

```
CALL GA06A(NUMPTS,N,LIST,COORD,NUMTRI,IV,IT,LIW,IWORK,IFLAG)
```

The double precision version

```
CALL GA06AD(NUMPTS,N,LIST,COORD,NUMTRI,IV,IT,LIW,IWORK,IFLAG)
```

NUMPTS is an INTEGER variable which must be set to the total number of data points. It is assumed that the data points are numbered 1,2,..., NUMPTS. This argument is not altered by the subroutine.

N is an INTEGER variable which must be set to the number of points which are to be triangulated. This argument is not altered by the subroutine. **Restrictions:** N ≤ NUMPTS and N ≥ 3.

LIST is an INTEGER array of length at least N. On entry, if N ≠ NUMPTS, LIST must contain a list of the points in the data set to be triangulated. On exit, LIST is unchanged. If N = NUMPTS, LIST is not accessed.

COORD is a REAL (DOUBLE PRECISION in the D version) array of dimension (2, NUMPTS). If data point I is to be included in the triangulation, then on entry COORD(1,I) and COORD(2,I) must contain the x- and y-coordinates of the point I, respectively. On exit, COORD is unchanged.

NUMTRI is an INTEGER variable which need not be set on entry. On exit, NUMTRI is set to the number of triangles created by the subroutine. Note that NUMTRI will not exceed 2*N+1.

IV is an INTEGER array of dimension (3, 2*N+1) which need not be set on entry. On exit, the vertices of triangle J are given, in anticlockwise order, in IV(I,J), I=1,2,3 (1 ≤ J ≤ NUMTRI).

IT is an INTEGER array of dimension (3, 2*N+1) which need not be set on entry. On exit, the triangles adjacent to triangle J are given, in anticlockwise order, in IT(I,J), I=1,2,3 (1 ≤ J ≤ NUMTRI). A zero indicates that there is no adjacent triangle.

LIW is an integer variable which must be set by the user to the length of the array IWORK. This argument is not altered by the subroutine. **Restriction:** LIW ≥ MAX(3*N+1, NUMPTS).

IWORK is an INTEGER array of length LIW. This array is used by the subroutine as workspace.

IFLAG is an INTEGER variable which need not be set on entry. On exit, IFLAG = 0 indicates that the subroutine has performed successfully. Negative values indicate an error and positive values indicate a warning (see § 2.3).

2.2 Common blocks

One common block is used. The common block is:

The single precision version

```
COMMON/ GA06D/ LP, MP
```

The double precision version

```
COMMON/ GA06DD/ LP, MP
```

where the parameters are given default values by a block data subprogram GA06E/ED.

LP is an INTEGER variable used as the unit number of the device to which error messages are sent. The default value is 6. Error messages can be suppressed by setting LP ≤ 0. This parameter is not altered by the subroutine.

MP is an INTEGER variable used as the unit number of the device to which warning messages are sent. The default value is 6. Warning messages can be suppressed by setting MP ≤ 0. This parameter is not altered by the subroutine.

2.3 Errors and diagnostic messages

Successful return from GA06A/AD is indicated by a value of IFLAG equal to 0. Possible nonzero values of IFLAG are given below. In each case an identifying message is output on unit LP (error messages) or unit MP (warning messages).

–1 – N < 3. Immediate return with input parameters unchanged.
–2 – N > NUMPTS. Immediate return with input parameters unchanged.
–3 – Defined length of integer workspace too small. Immediate return with an error message giving the value of LIW that will guarantee sufficient workspace.
–4 – All the points to be triangulated have the same x and/or y coordinates.
+1 – The user has input entries in LIST outside the range 1 ≤ LIST(I) ≤ NUMPTS. The subroutine ignores these entries.
+2 – The user has input duplicate entries in LIST. The subroutine ignores these duplicates. If LIST contains duplicate entries and entries which are out of range, IFLAG = 2 is returned.
+3 – The triangulation generated by the subroutine contains triangles in which either at least 2 of the vertices are coincident or all 3 vertices are colinear. Details of the first 10 such triangles are printed on unit MP.
+4 – Some of the data points supplied by the user for triangulation do not appear in the triangulation created by the subroutine. This may occur if the data points are (almost) colinear. The number of points which do not appear in the triangulation is returned in IWORK(1).

3 GENERAL INFORMATION

Workspace: An integer array IWORK of length LIW is used by the subroutine as workspace.

Use of common: The subroutine uses the common block GA06D/DD; see § 2.2.

Other routines called directly: Subroutines internal to the package are GA06B/BD and GA06C/CD. GA06A/AD also references the block data subprogram GA06E/ED. In addition, GA06A/AD calls FD05A/AD.

Input/output: Error messages on unit LP (LP ≤ 0 suppresses them), and warning messages on unit MP (MP ≤ 0 suppresses them).

Restrictions:

N ≥ 3,

N ≤ NUMPTS,

LIW ≥ MAX(3*N+1, NUMPTS).

4 METHOD

The algorithm constructs Delaunay triangulations in the plane. A Delaunay triangulation can conveniently be described in terms of the corresponding Dirichlet tessellation. For an arbitrary set of points in the plane, the Dirichlet tessellation divides the plane into a collection of polygonal regions whose boundaries are the perpendicular bisectors of the lines joining the neighbouring points. Each polygon is associated with a single data point. Any location within a given polygon is closer to the polygon data point than to any other data point. The Delaunay triangulation that corresponds to the Dirichlet tessellation is constructed by connecting all data points that share a polygon boundary.

The algorithm computes the Delaunay triangulation by introducing each point, one at a time, into an existing Delaunay triangulation, which is then updated.

The coordinates of the points to be triangulated are first normalised. The triangulation process is then started by selecting three points to form a 'supertriangle' which completely encompasses all the points to be triangulated. A new point P is introduced into the triangulation by finding an existing triangle which encloses P and forming three new triangles by connecting P to each of the vertices of this triangle. The existing triangulation is then updated to a Delaunay triangulation using the swapping algorithm of Lawson (1977). In this procedure all the triangles adjacent to the edges opposite P are placed on a stack. Each triangle is unstacked, one at a time, and a check is made to see if P lies inside its circumcircle. If this is the case, the triangle containing the vertex P and the adjacent triangle form a convex quadrilateral with the diagonal drawn in the wrong direction, and it must be replaced by the alternative diagonal to preserve the structure of the Delaunay triangulation. Once the swap is completed, any triangles which are now opposite P are added to the stack. The next triangle is then unstacked and the process repeated until the stack is empty, resulting in a new Delaunay triangulation containing the point P. After all the points have been introduced, any triangle containing one or more of the supertriangle vertices is removed.

For full details of the algorithm the user is referred to Sloan (1987).

References

Lawson, C. L. (1977) Software for C^1 interpolation. In: Rice, J. (ed) *Mathematical Software III*, Academic Press, 161-194.

Sloan, S. W. (1987) A fast algorithm for constructing Delaunay triangulations in the plane. *Adv. Eng. Software*, **9**, 34-55.

5 EXAMPLE OF USE

The following program provides an example of the use of GA06A.

```
C  SIMPLE CODE TO TEST THE DELAUNAY TRIANGULATION ROUTINE OF SLOAN
      PARAMETER(NPMAX=20)
      INTEGER I, J, IFLAG, LIW, NUMTRI, N, LP, MP,
     *        LIST(NPMAX), IWORK(3*NPMAX+1), IV(3,2*NPMAX+1),
     *        IT(3,2*NPMAX+1)
      REAL COORD(2,NPMAX)
      COMMON / GA06D/ LP, MP
C
C  READ IN DATA
C
      READ(5,*)  NUMPTS, N
      LIW = MAX(3*N+1,NUMPTS)
      DO 10 I = 1, NUMPTS
           READ(5,*) COORD(1,I), COORD(2,I)
   10 CONTINUE
      IF(N.NE.NUMPTS) READ(5,*) (LIST(I), I= 1,N)
C
C  CALL TRIANGULATION ROUTINE
```

```
      CALL GA06A(NUMPTS,N,LIST,COORD,NUMTRI,IV,IT,LIW,IWORK,IFLAG)
      WRITE(6,998) NUMTRI
      WRITE(6,997)
      DO 20 J = 1, NUMTRI
         WRITE(6,996) J, (IV(I,J), I=1,3), (IT(I,J), I=1,3)
   20 CONTINUE
C
998   FORMAT(/' THE NUMBER OF TRIANGLES IS ',I8)
997   FORMAT(/'    TRIANGLE    VERTICES        ADJACENT',
     *       /'             (ANTICLOCKWISE)   TRIANGLES')
996   FORMAT(3X,I5,2X,3I5,2X,3I5)
      STOP
      END
```

The following are suitable input data:–

```
10    7
0.0   0.0
1.0   0.0
1.5   2.0
0.5   0.5
0.8   3.0
3.0   2.2
2.0   0.0
2.5   2.0
1.5  -0.5
1.5   2.5
1  2  4  7  8  9  10
```

and produce the following output:

```
 THE NUMBER OF TRIANGLES IS        7

    TRIANGLE    VERTICES         ADJACENT
             (ANTICLOCKWISE)   TRIANGLES
       1      9    2    1      5    4    0
       2      7    4    2      7    4    5
       3     10    1    4      0    4    6
       4      1    2    4      1    2    3
       5      7    2    9      2    1    0
       6      8   10    4      0    3    7
       7      8    4    7      6    2    0
```