1 SUMMARY

These subroutines **generate** an *m* by *n* **random sparse matrix** with user-specified options such as structural nonsingularity and bandedness. The matrix is held in a packed form in a standard sparse matrix format, and there is an option to write it to a file in Rutherford Boeing format (Report RAL-TR-97-031).

ATTRIBUTES — **Version:** 1.1.0. (12 April 2013) **Types:** Real (single, double), Complex (single, double), Integer. **Calls:** FA14, MC56, MC59. **Original date:** July 2001. **Remark:** YM11 is a threadsafe version of YM01 that also includes entries for generating complex valued and integer valued matrices. **Origin:** I.S. Duff, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Initialization of the random sequence and default setting for matrix.

The YM11I/ID/IC/IZ/II entry sets the initial value of a seed for the pseudo-random number generator FA14 to its default value by calling the FA14I/ID entry. It is the normal practice to do this initialization before any use is made of the YM11A/AD/AC/AZ/AI entry and thereafter not disturb its changing value. It is possible to maintain more than one random sequence by using a different initialized seed dedicated to each random sequence. YM11I/ID/IC/IZ/II also sets the parameters (in ICNTL) specifying the characteristics of the matrix being generated. If it is wished to generate a matrix with other characteristics or to set the seed differently, the appropriate entries of ICNTL or ISEED should be changed by the user after the call to YM11I/ID/IC/IZ/II.

The single precision version

```
CALL YM11I(ICNTL,ISEED)
```

The double precision version

```
CALL YM11ID(ICNTL,ISEED)
```

The single precision complex version

```
CALL YM11IC(ICNTL,ISEED)
```

The double precision complex version

```
CALL YM11IZ(ICNTL,ISEED)
```

The integer version

```
CALL YM11II(ICNTL,ISEED)
```

ICNTL is an INTEGER array of length 10 which need not be set by the user. ICNTL is used to define the matrix being generated. The components of ICNTL are defined and set by YM11I/ID/IC/IZ/II as follows:

ICNTL(1) determines whether a band matrix is required (entries clustered round the diagonal). If ICNTL(1) is zero or negative, no clustering is performed. Otherwise, all entries a_{ij} will be such that $|i–j| \leq$ ICNTL(1). If the requested number of entries (NZIN) exceeds the number of entries in the band, then a full band matrix of semibandwidth ICNTL(1) will be produced. ICNTL(1) is set to 0.

ICNTL(2) determines whether only a matrix pattern is produced. If ICNTL(2) is zero, only a matrix pattern is generated. For all other values of ICNTL(2), matrix entries are generated also. ICNTL(2) is set to 1.

ICNTL(3) determines the symmetry of the matrix. If ICNTL(3) is zero, the generated matrix is symmetric. For all other values it is unsymmetric. If M $\neq$ N (see Section 2.2), then the action taken is as if ICNTL(3)

were nonzero. ICNTL(3) is set to 1.

ICNTL(4) determines whether the matrix is structurally singular. If it is set to 1, then the matrix is guaranteed structurally nonsingular. In the unsymmetric non-band case, there will be a set of min(*m*,*n*) distinct entries no two of which lie in the same row or column. In the band and symmetric cases, all of the diagonal will consist of entries. ICNTL(4) is set to 1.

ICNTL(5) is used to determine whether entries within a column are ordered by rows (ICNTL(5)=1) or not (ICNTL(5)≠1). ICNTL(5) is set to 1.

ICNTL(6) is used to determine the range of integers used for an integer-valued matrix. The values will be random integers in the range –ICNTL(6) to +ICNTL(6). ICNTL(6) is set to 100000.

ICNTL(7) is used to determine whether the matrix in Rutherford-Boeing format will be written to a file. If it is negative, or if ICNTL(5)≠1, then no action is taken; otherwise MC56 is used to write the matrix to the file on unit ICNTL(7). ICNTL(7) is set to -1.

ICNTL(8) to ICNTL(10) are not presently used by YM11 but are set to zero by YM11I/ID/IC/IZ/II.

ISEED is an INTEGER variable which need not be set by the user. On return from YM11I/ID/IC/IZ/II, it is set to the initial value of a seed for the random number generator. It will be used in the YM11A/AD/AC/AZ/AI entry to generate random numbers.

2.2 Argument list

The single precision version

```
CALL YM11A(M,N,NZIN,NZOUT,IRN,A,JCOLST,IW,ICNTL,KEY,ISEED)
```

The double precision version

```
CALL YM11AD(M,N,NZIN,NZOUT,IRN,A,JCOLST,IW,ICNTL,KEY,ISEED)
```

The single precision complex version

```
CALL YM11AC(M,N,NZIN,NZOUT,IRN,A,JCOLST,IW,ICNTL,KEY,ISEED)
```

The double precision complex version

```
CALL YM11AZ(M,N,NZIN,NZOUT,IRN,A,JCOLST,IW,ICNTL,KEY,ISEED)
```

The integer version

```
CALL YM11AI(M,N,NZIN,NZOUT,IRN,A,JCOLST,IW,ICNTL,KEY,ISEED)
```

M is an INTEGER variable that must be set by the user to *m* the number of rows required in the matrix. This argument is not altered by the subroutine.

N is an INTEGER variable that must be set by the user to *n* the number of columns required in the matrix. This argument is not altered by the subroutine.

NZIN is an INTEGER variable that must be set by the user to the total number of entries required in the matrix. This argument is not altered.

NZOUT is an INTEGER variable that need not be set by the user. It will be set by the subroutine to the total number of entries in the matrix produced. This will usually have the same value as NZIN. It will be different only if NZIN is larger than the maximum number of entries that can be accommodated in the requested structure, or if NZIN is less than min(*m*,*n*) and structural nonsingularity is requested.

IRN is an INTEGER array of length no greater than MAX(NZIN,MIN(M,N)) that need not be set by the user. The first NZOUT entries will be set by the subroutine to the row indices of the entries, ordered so that the entries of a single column are contiguous. The entries of column J precede those of column J+1 (J=1, ..., N-1), and there is no wasted space between columns. Unless ICNTL(5) is set to 1, row indices are not ordered within columns,

except that in the banded or symmetric structurally nonsingular case the diagonal entry comes first. For structurally nonsingular unsymmetric non-banded systems, steps are taken to ensure that the set of distinct entries do not lie in the same relative position in each column.

A is a REAL (DOUBLE PRECISION in the D version, COMPLEX in the C version, COMPLEX*16 in the Z version, or INTEGER in the I version) array that need not be set by the user. If ICNTL(2) is nonzero, A(K) should be of length MAX(NZIN,MIN(M,N)) and will be set by the subroutine to the value of the entry in position K of IRN, K=1, ..., NZOUT. If ICNTL(2) is zero, the array is not accessed by the subroutine.

JCOLST is an INTEGER array of length n+1 that need not be set by the user. It will be set by the subroutine so that JCOLST(J) is the position in arrays IRN and A of the first entry in column J (J=1, ..., N). JCOLST(N+1) will be equal to NZOUT+1.

IW is an INTEGER array of length m+n that will be used as workspace.

ICNTL is an INTEGER array of length 10 that must be set by the user, perhaps by calling YM11I/ID/IC/IZ/II (see Section 2.1). It is not altered.

KEY is a CHARACTER*8 variable that should be set to the key for the matrix if it is to be written to a file as a Rutherford-Boeing matrix. It is only accessed if ICNTL(5) is equal to 1 and ICNTL(7) is positive.

ISEED is an INTEGER variable that holds the generator word for FA14 and should normally be initialized by calling YM11I/ID/IC/IZ/II. It is considered as private and should not normally be changed directly by the user. Its value may be saved and restored by calling FA14C/CD and FA14D/DD, which permits the same matrix to be generated on separate calls. On return, it holds the value from the final call of FA14 by YM11, which is suitable for a fresh call of YM11.

3 GENERAL INFORMATION

Workspace: Array IW is used as workspace (see §2.2).

Use of common: None.

Other routines called directly: Within the package: YM11B/BD/BC/BZ/BI. Outside the package: FA14I/ID, FA14A/AD, and FA14B/BD and, if ICNTL(5) = 1, MC59A/AD/AC/AZ/AI, respectively. If, in addition, ICNTL(7) is non-negative, MC56A/AD/AC/AZ/AI, respectively.

Input/output: None.

Restrictions: None.

4 METHOD

The number of entries in each column is calculated as follows. Counts are initialized to zero unless structural nonsingularity has been requested, in which case min(m,n) of the counts are initialized to 1. A random number from the uniform distribution [1,n], j say, and column j is then a candidate for the next entry. An entry is added to column j only if the current count is less than the maximum permitted in the column. To avoid a bias towards shorter columns in the symmetric and banded cases, the acceptance of the randomly generated value is weighted by the length of the column and by the number of entries already in the column. This generation of random numbers and subsequent action is repeated until the sum of the column counts equals the required number of entries.

Each column is then generated in turn, by generating row indices from a uniform distribution in the appropriate range. For unsymmetric structures, this is $[1, m]$, for banded systems $[\max(1, j–ibw), \min(m, j+ibw)]$ for column j, where ibw is equal to the value of ICNTL(1). For symmetric structures, this is $[j, m]$ for column j.

If structural nonsingularity without bandedness or symmetry is requested, further indices corresponding to a random selection of min(m,n) integers from [1,m] are placed in min(m,n) columns selected at random from [1,n]. If structural nonsingularity with band clustering or symmetry is requested, the diagonal is included in the structure..

If the entries in a column are required to be ordered by rows, this is done through a call to MC59. If, in addition, ICNTL(7) is positive, then the matrix is written to unit ICNTL(7) in Rutherford Boeing format with the title "Random matrix generated by YM11" and the key given by the parameter KEY.

In all cases where matrix values are requested, these are generated in a final pass from a uniform distribution in the range [−1,1]. If the matrix is integer valued, entries are generated as random integers in the range between −ICNTL(6) and +ICNTL(6).

5 EXAMPLE OF USE

Two 20×20 random matrices may be generated by the following code

```
      INTEGER MAXN,MAXNZ
      PARAMETER (MAXN=20,MAXNZ=160)
      INTEGER IRN(MAXNZ),IW(MAXN,2)
      INTEGER JCOLST(MAXN+1)
      INTEGER ICNTL(10),IX,KASE,M,N,NZIN,NZOUT
      CHARACTER*8 KEY
      DOUBLE PRECISION A(1)

      CALL YM11ID(ICNTL,IX)
C Only generate pattern
      ICNTL(2) = 0

      DO 50 KASE = 1,2
        READ (5,FMT=100) M,N,NZIN,ICNTL(1),ICNTL(3)
        WRITE (6,FMT=110) M,N

  100   FORMAT (5I4)
  110   FORMAT (/'  The ',I4,' by ',I4,' matrix pattern')

        CALL YM11AD(M,N,NZIN,NZOUT,IRN,A,JCOLST,IW,ICNTL,KEY,IX)
        CALL PRMAT(M,N,NZOUT,IRN,JCOLST)

   50 CONTINUE
      STOP

      END
      SUBROUTINE PRMAT(M,N,NZ,IRN,JCOLST)
      CHARACTER MAT(40000)
      INTEGER M,N,NZ,MN,J,K,I,IROW,III
      INTEGER  IRN(NZ)
      INTEGER JCOLST(N)
      MN=M*N
      DO 17 J=1,MN
        MAT(J)='.'
   17 CONTINUE
      DO 6 J=1,N
        K=NZ
        IF (J .LT. N) K=JCOLST(J+1)-1
        DO 7 I=JCOLST(J),K
          IROW=IRN(I)
          III=(IROW-1)*N+J
          MAT(III)='X'
    7   CONTINUE
    6 CONTINUE
      WRITE (6,6001)
 6001 FORMAT(' ')
      DO 20 I=1,MN,N
        WRITE(6,6000)(MAT(J),J=I,I+N-1)
```

```
 6000   FORMAT(' ',60A2)
  20  CONTINUE
      RETURN
      END
```

run on the data

```
    20  20  60   5   0
    10  20  60  -1   1
```

This would generate the following output

```
    The   20 by   20 matrix pattern

    X . . . . . . . . . . . . . . . . . . .
    X X . . . . . . . . . . . . . . . . . .
    X X X . . . . . . . . . . . . . . . . .
    X X . X . . . . . . . . . . . . . . . .
    X . X . X . . . . . . . . . . . . . . .
    . X . . X X . . . . . . . . . . . . . .
    . X X X . X X . . . . . . . . . . . . .
    . . . . . . . X . . . . . . . . . . . .
    . . . X . . . . X . . . . . . . . . . .
    . . . . . X . X . X . . . . . . . . . .
    . . . . . X . X . . X . . . . . . . . .
    . . . . . . . . X X . X . . . . . . . .
    . . . . . . . . X X . X X X . . . . . .
    . . . . . . . . . X X . . X . . . . . .
    . . . . . . . . . X X . . . X . . . . .
    . . . . . . . . . . X . X . . X . . . .
    . . . . . . . . . . . X . X X . X . . .
    . . . . . . . . . . . . . . X X . X . .
    . . . . . . . . . . . . . . . . X . X .
    . . . . . . . . . . . . . . . X . X X X

    The   10 by   20 matrix pattern

    . . . . . . . X . . . . . . X X . . . X
    X . X X X . . . . . . . . X . . . . . X
    X . . . . X . X X . . . . . . X X . . .
    X X . X X X X X . . X . . . X . . . . .
    . . . X X X X . . . . . X . . . . . X X
    . X . . X . . . X . . . X . X . X X . .
    X . . . . . . . X X . . X X . X . . . .
    . X . . X . . . X X . . . X . . . . . .
    . X . X . . . . . . . . . . X . . X X .
    . . . . . . . . X X . . X . . X X . . .
```