NISTIR 5935

The Matrix Market Exchange Formats: Initial Design

Ronald F. Boisvert
Roldan Pozo
Karin A. Remington

U. S. Department of Commerce
Technology Administration
National Institute of Standards and Technology
Computing and Applied Mathematics Laboratory
Gaithersburg, MD 20899 USA

December 1996

The Matrix Market Exchange Formats: Initial Design

Ronald F. Boisvert, Roldan Pozo and Karin A. Remington*
Applied and Computational Mathematics Division
National Institute of Standards and Technology
Gaithersburg, MD 20899 USA.

Abstract

We propose elementary ASCII exchange formats for matrices. Specific instances of the format are defined for dense and sparse matrices with real, complex, integer and pattern entries, with special cases for symmetric, skew-symmetric and Hermitian matrices. Sparse matrices are represented in a coordinate storage format. The overall file structure is designed to allow future definition of other specialized matrix formats, as well as for objects other than matrices.

Contents

*Email: `boisvert@nist.gov`, `kremington@nist.gov`, `pozo@nist.gov`

1 Introduction

The Matrix Market (MM) exchange formats provide a simple mechanism to facilitate the exchange of matrix data. In particular, the objective has been to define a minimal base ASCII file format which can be very easily explained and parsed, but can be easily adapted to applications with a more rigid structure, or extended to related data objects. This file format is the primary format used by the Matrix Market [1], a visual database of matrix test data available on the World Wide Web[1].

The MM exchange format for matrices is really a collection of affiliated formats which share design elements. In our initial specification, two matrix formats are defined.

1. **Coordinate Format**
   A file format suitable for representing general sparse matrices. Only nonzero entries are provided, and the coordinates of each nonzero entry is given explicitly. This will be illustrated in Examples 1 and 2.

2. **Array Format**
   A file format suitable for representing general dense matrices. All entries are provided in a pre-defined (column-oriented) order. This will be illustrated in Example 3.

Several instances of each of these basic formats are defined. These are obtained by specifying an arithmetic field for the matrix entries (i.e., real, complex, integer, pattern) and a symmetry structure which may reduce the size of the data file (i.e. general, symmetric, skew-symmetric, Hermitian).

The MM design overcomes some of the difficulties with existing schemes. The most well known is the Harwell-Boeing (HB) Sparse Matrix Format [2]. Many matrices have been encoded using this format, most notably those comprising the Harwell-Boeing Sparse Matrix Collection (Release I). Unfortunately, the HB specification is somewhat complex, difficult to parse from languages other than Fortran, biased in favor of compressed column representation, and not easily extensible. Some of these factors may have impeded the more widespread use of the HB format.

Recently, a group at NASA Ames proposed an alternate format (GAMFF) [3]. This was motivated mainly due to the need to produce a file format to represent graphs. A specification which included graphs, sparse matrices, and dense matrices was developed, and a large variety of such matrix and graph files were converted to this format[2]. Although the GAMFF format is somewhat simpler and more extensible than HB, its sparse matrix subset remains closely tied to the HB compressed column format.

[1] `http://math.nist.gov/MatrixMarket/`
[2] `http://www.nas.nasa.gov/NAS/DataSets/GAMFF`

In the remainder of this section, we describe the coordinate and array matrix formats informally, providing several examples, while in Section 2, we detail the formal specification of the MM family of formats. In Section 3, we elaborate on the coordinate and array matrix formats. Finally, in Section 4 we outline some of the guiding principles that we used in designing these formats, focusing especially on the features used to assure easy extensibility.

1.1 Coordinate Format for Sparse Matrices

Coordinate format is suitable for representing sparse matrices. Only nonzero entries need be encoded, and the coordinates of each are given explicitly. This is illustrated in the following example of a real general sparse matrix.

Example 1 : MM coordinate format for a real general sparse matrix

Consider the following 5×5 *matrix.*

$$\begin{pmatrix} 1 & 0 & 0 & 6 & 0 \\ 0 & 10.5 & 0 & 0 & 0 \\ 0 & 0 & .015 & 0 & 0 \\ 0 & 250.5 & 0 & -280 & 33.32 \\ 0 & 0 & 0 & 0 & 12 \end{pmatrix}$$

In MM coordinate format this could be represented as follows.

```
%%MatrixMarket matrix coordinate real general
% A 5x5 sparse matrix with 8 nonzeros
5 5  8
1 1     1.0
2 2    10.5
4 2   250.5
3 3     0.015
1 4     6.0
4 4  -280.0
4 5    33.32
5 5    12.0
```

The first line contains the `%%MatrixMarket` *header followed by a sequence of keywords. These indicate that the object being represented is a matrix in coordinate format, and that the numeric data is represented in real general form. (By general we mean that the storage format is not taking advantage of any symmetry properties.) The second line is a comment. (Zero or more such lines may appear at this point in the file, each beginning with a %.) The next line contains three integers which give the number of rows, columns,*

Figure 1: A version of Example 1 illustrating free-format features.

```
%%MatrixMarket  MATRIX    Coordinate    Real General
% -----------------------------------------------------------------
%               Same matrix as in Example 1
% -----------------------------------------------------------------
%
%  See http://math.nist.gov/MartrixMarket for more information.

    5  5        8

1 1  1.0
2 2       10.5
3 3             1.5e-2
4 4                     -2.8E2
5 5                               12.
     1      4      6
     4      2      250.5
     4      5      33.32
```

and nonzeros (`L`*) in the matrix. This is followed by exactly* `L` *lines, each containing two integers followed by a floating-point number. The integers provide the (i,j) matrix coordinates for the floating-point value which follows (indices in each dimension start at 1 as in Fortran). Only nonzero entries need be listed.*

The Matrix Market format is flexible: items are case insensitive and blank-delimited; extra blanks between items are ignored, and the nonzero entries can be listed in any order. The data is designed to be easy to read using free-format input in Fortran, or `scanf` in C. Note also that this means that the MM representation of a sparse matrix is not unique. Figure 1 illustrates a valid MM representation of the matrix of Example 1 which looks quite different. Nevertheless, the simple Matrix Market rules allow this version to be read just as easily.

Variants of the coordinate format are defined for matrices with complex and integer entries, as well as for those in which only the position of the nonzero entries is prescribed (pattern matrices). Additional variants are defined for cases in which symmetries can be used to significantly reduce the size of the data: symmetric, skew-symmetric and Hermitian. In these cases only entries in the lower triangular portion are supplied. (In the skew-symmetric case the diagonal entries are zero, and hence they too are omitted.) An example of a complex Hermitian sparse matrix follows.

Example 2 : MM coordinate format for a complex Hermitian sparse matrix

Consider the following 5×5 *complex matrix.*

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 10.5 & 0 & 250.5 - 22.22i & 0 \\ 0 & 0 & .015 & 0 & 0 \\ 0 & 250.5 + 22.22i & 0 & -280 & -33.32i \\ 0 & 0 & 0 & 33.32i & 12 \end{pmatrix}$$

In MM coordinate format this could be represented as follows.

```
%%MatrixMarket matrix coordinate complex hermitian
5 5  7
1 1    1.0      0
2 2   10.5      0
4 2  250.5     22.22
3 3    1.5e-2   0
4 4   -2.8e2    0
5 5   12.       0
5 4    0       33.32
```

The first line indicates the file contains a matrix in coordinate format represented in complex Hermitian form. This format is the same as in Example 1, except that two floating-point numbers appear in each matrix entry, giving the real and imaginary parts of the matrix entries. Also, only the entries in the lower triangular portion of the matrix are listed since those in the upper triangle are known by symmetry (i.e., $a_{ji} = \overline{a_{ij}}$*, where the bar denotes the complex conjugate).*

1.2 Array Format for Dense Matrices

The coordinate matrix format is a very simple scheme for representing nonzero matrix entries. Less index data can be used when the nonzero entries appear in regular ways. The simplest case is when all entries (both zero and nonzero) are explicitly stored in a fixed order. This is particularly suitable for representing dense matrices. The *array* format in Matrix Market is such a format based on column-major order. An example follows.

Example 3 : MM array format for a real dense matrix

Consider the following 4×3 *real matrix.*

$$\begin{pmatrix} 1.0 & 5.0 & 9.0 \\ 2.0 & 6.0 & 10.0 \\ 3.0 & 7.0 & 11.0 \\ 4.0 & 8.0 & 12.0 \end{pmatrix}$$

In MM array format this could be represented as follows.

```
%%MatrixMarket matrix array real general
% A 4x3 dense matrix
4 3
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
```

As before, the first line indicates that the file represents a matrix in array format which is real and general. The second line is a comment. The third gives the number of rows and columns. This is followed by all matrix entries, one per line, in column-major order.

Again, variants for complex and integer data, as well as for symmetric, skew-symmetric and Hermitian matrices are also defined. For the symmetric cases, matrix entries are given in packed form, i.e., only entries in the lower triangle appear. In the skew-symmetric case the diagonal entries are zero and hence also are omitted.

2 Specification of the Base MM File Format

All Matrix Market exchange format files contain three sections, which must appear in order.

1. **Header.** The first line must follow the following template:

   `%%MatrixMarket` *object format [qualifier ...]*

   The first 15 characters must be `%%MatrixMarket` followed by at least one blank. This is followed by an object type, a format type, and zero or more qualifiers. Each of these is a single word[3] The object type indicates the mathematical object (e.g., vector, matrix, directed graph) which is stored in the file. The format type indicates the format used to store the object (e.g., coordinate, array). Qualifiers are used to indicate special properties of the format (e.g. field and symmetry). The number of qualifiers and their meanings depend upon the object type.

[3] A word is simply a sequence of non-blank printable ASCII characters.

2. **Comments.** Zero or more lines of comments (each with first character `%`).
3. **Data.** The remainder of the file contains the data which represents the object. The details of the format depend upon the type of object. For simplicity, each data entry should occupy one line. (Of course, a single data entry may be comprised of more than one number.)

MM files are intended to be easy to parse without an overly restrictive specification. In particular, the data should be easy to read using free format input in Fortran, or the standard `scanf` in C. The following additional rules apply.

- All lines are limited to 1024 characters (for purposes of line buffering).
- Blank lines may appear anywhere after the first line.
- Numeric data on a given line is separated by one or more blanks; leading and trailing blanks are ignored.
- Real data items must be in floating-point decimal format, optionally utilizing the E-format exponential notation common to C and Fortran.
- All indices are 1-based. That is, all rows and columns are numbered starting with 1.
- Character data may be in either upper or lower case (i.e., readers must be case insensitive.)

3 Specification of MM Formats for Matrices

In this section we describe the two fundamental Matrix Market formats for matrices: coordinate and array. All matrices have two additional qualifiers: field and symmetry. Their values and meanings are given in Tables 1 and 2, respectively. The following indicates the meaningful header combinations for MM matrices.

$$\texttt{\%\%MatrixMarket}\quad\texttt{matrix}\quad \begin{bmatrix}\texttt{coordinate}\\ \texttt{array}\end{bmatrix} \begin{bmatrix}\texttt{real}\\ \texttt{integer}\\ \texttt{complex}\end{bmatrix} \begin{bmatrix}\texttt{general}\\ \texttt{symmetric}\\ \texttt{skew-symmetric}\end{bmatrix}$$

$$\texttt{\%\%MatrixMarket}\quad\texttt{matrix}\quad \begin{bmatrix}\texttt{coordinate}\\ \texttt{array}\end{bmatrix} \quad\texttt{complex}\quad \texttt{Hermitian}$$

$$\texttt{\%\%MatrixMarket}\quad\texttt{matrix}\quad \texttt{coordinate}\quad \texttt{pattern}\quad \begin{bmatrix}\texttt{general}\\ \texttt{symmetric}\end{bmatrix}$$

3.1 Coordinate Formats for Sparse Matrices

The following template illustrates the data section of MM files for matrices in coordinate format:

Table 1: **First matrix qualifier (Field).** Determines the type and number of values listed for each matrix entry.

Code	Interpretation
Real	The matrix is real. Matrix entries are represented by a single float.
Complex	The matrix is complex. Matrix entries are represented by two floats, the first giving the real part and the second the imaginary part.
Integer	The matrix has only integer entries. Matrix entries are represented by a single integer.
Pattern	Only the matrix nonzero pattern is provided. Matrix entries are omitted (only the nonzero pattern is supplied).

```
M N L
I J A(I,J)
I J A(I,J)
   .
   .
I J A(I,J)
```

The first line contains exactly three integers giving the number of rows (M) and columns (N) in the matrix, as well as the number of entries which are given in the file (L). This is followed by exactly L lines, each containing two integers and an entry value. The first integer is the row index of the entry (I) and the second is the column index (J). Indices are 1-based, that is, the entry in the upper left corner of the matrix is indexed as A(1,1) (as opposed to A(0,0), as it would be in C). These are followed by the value of the entry a_{ij} (A(I,J)). The format of the entry values depends upon the first qualifier (field); see Table 1.

Entries not explicitly provided in the file are considered to be zero, except for those known by symmetry; see Table 2.

3.2 Array Formats for Dense Matrices

In array format all matrix entries are explicitly listed (zeros and nonzeros alike) and have a predefined order, allowing the indexing data to be dropped. In MM files, the entries are listed in column-major order, i.e. sorted first by column, then by rows within the column.

The first line of the data section of MM array format files contain two integers: M and N, giving the number of rows and columns in the matrix, respectively. This is followed by matrix entry values given one per line. The representation for the value depends upon

Table 2: **Second matrix qualifier (Symmetry).** Determines how to interpret (specifically, how to replicate) the matrix entries.

Code	Interpretation
General	The matrix has no symmetry properties, or symmetry is not used to reduce the number of matrix entries. (All non-square matrices are general.)
Symmetric	Square matrix with $a_{ij} = a_{ji}$. Only entries on or below the matrix diagonal are provided in the file. The entries above the main diagonal are known by symmetry.
Skew-symmetric	Square matrix with $a_{ij} = -a_{ji}$. Only entries below the main diagonal are stored in the file. The entries on the main digonal are zero and those above the main diagonal are known by symmetry.
Hermitian	Square complex matrix with $a_{ij} = \overline{a_{ji}}$. Only entries on or below the matrix diagonal are provided in the file. The entries above the main diagonal are known by symmetry.

the first qualifier (field); see Table 1.

The entries are listed in column-major order. The number of entries depends upon the second qualifier (symmetry); see Table 2. In Table 3 we illustrate the ordering of entries in two ways, (1) by a Fortran implicit loop and (2) by example with a 3×3 array of numbers indicating the order in which the elements in the corresponding positions would appear in the file.

4 Extending the Base MM Formats

The concepts of simplicity and extensibility are central to the MM format. As the examples of the previous section illustrate, the MM format allows great freedom in many formatting details, and requires few syntactic embellishments. We expect that this simplicity will provide a result more amenable to routine matrix exchange. Extensibility in the MM format comes in three forms: Structured Comments, Format Specializations and Object and Format Types.

4.1 Structured Comments

The comment section of the Matrix Market format is typically used to present further documentation and information about the data object stored in the file. There are no

Figure 2: A version of Example 1 with structured comments. This is a suggested use only — Matrix Market does not define any semantics nor impose any rules on the layout of structured comments.

```
%%MatrixMarket matrix coordinate real general
%%Harwell-Boeing collection
%
%HB FILE_NAME    example1.mtx
%HB KEY          MMEXMPL1
%HB OBJECT       matrix
%HB FORMAT       coordinate
%HB QUALIFIERS   real general
%HB DESCRIPTION  Unsymmetric matrix from Example 1 of MM Format report
%HB CONTRIBUTOR  R. Boisvert (boisvert@nist.gov)
%HB DATE         June 24, 1996
%HB PRECISION    4
%HB DATA_LINES   9
%
% This illustrates an example use of structured comments (above) and
% a fixed format for the numerical data (below).  It also sorts entries
% first by column and then by row.
%
%
    5     5     8
    1     1   1.000e+00
    2     2   1.050e+01
    3     3   1.500e-02
    1     4   6.000e+00
    2     4   2.505e+02
    4     4  -2.800e+02
    4     5   3.332e+01
    5     5   1.200e+01
```

Figure 3: A verbose (self-documenting) version of Example 1 with structured comments . This is a suggested use only — Matrix Market does not define any semantics nor impose any rules on the layout of these comments.

```
%%MatrixMarket matrix coordinate real general
%
%<#-begin format-section>
%
% This ASCII file represents a sparse MxN matrix with L nonzeros
% in the following Matrix-Market format:
%
% +----------------------------------------------+
% |%%MatrixMarket matrix coordinate real general | <--- header line
% |%                                             | <--+
% |% comments                                    |    |-- 0 or more comments
% |%                                             | <--+
% |    M N  L                                    | <--- rows, cols, entries
% |    I1  J1  A(I1, J1)                         | <--+
% |    I2  J2  A(I2, J2)                         |    |
% |    I3  J3  A(I3, J3)                         |    |-- L lines
% |        . . .                                 |    |
% |    IL JL  A(IL, JL)                          | <--+
% +----------------------------------------------+
%
% Indices are 1-based, i.e. A(1,1) is the first element of a matrix.
%
% See http://math.nist.gov/MatrixMarket/formats for further details
% and software (C, Fortran, Matlab, Java) to process these files.
%
%<#-end format-section>
%-------------------------------------------------------------------
    5     5     8
    1     1   1.000e+00
    2     2   1.050e+01
    3     3   1.500e-02
    1     4   6.000e+00
    2     4   2.505e+02
    4     4  -2.800e+02
    4     5   3.332e+01
    5     5   1.200e+01
```

Table 3: Ordering of entries for matrices in array format

Symmetry	**Fortran implicit loop**	**3×3 example**
General	`(( A(I,J), I=1,M), J=1,N )`	$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$
Symmetric and Hermitian	`(( A(I,J), I=J,N), J=1,N )`	$\begin{bmatrix} 1 & & \\ 2 & 4 & \\ 3 & 5 & 6 \end{bmatrix}$
Skew-symmetric	`(( A(I,J), I=J+1,N), J=1,N-1 )`	$\begin{bmatrix} & & \\ 1 & & \\ 2 & 3 & \end{bmatrix}$

implied rules of what the comments should contain. However it is useful to include information such as the matrix origin, its mathematical properties, references, etc.

In particular, one could utilize a rigid convention for describing this information, so that it can be machine parsable. An example of this is shown in Figure 2. The specification of such conventions are completely user-defined and are outside the Matrix Market specifications. This does, however, allow one to record specialized information in a Matrix Market file without changing the basic format (or processing tools). Input routines that know about the special encoded comments can utilize these to extract this information. Other routines will interpret these lines as comments and ignore them.

4.2 Format Specializations.

Few requirements are imposed on the formatting of the numerical data in an MM file. In particular applications we expect that conventions will be adopted which will allow the files to be more highly structured so that they are more uniform. This might be done to obtain unique representations or to simply allow the data to be be more easily viewed with a text editor. This is also illustrated Figure 2. We expect to develop *pretty printer* applications which read arbitrarily formatted MM files and convert them to a more visually-pleasing standardized form.

4.3 Extending the MM Type Code

The MM type code provides the most powerful method of extending the MM file format into new domains. (In Example 1 this is the string `matrix coordinate real general`.) The first part of this code, the *object type*, defines what mathematical object the file represents. In our initial specification we have defined only one, the matrix. We expect to provide specifications for other types of objects in the future; examples we are considering include graphs, grids and vectors. We expect that others will find additional object types necessary within their own applications.

Each object may have multiple *format types* associated with it. This is given as the second part of the matrix type code. For the initial specification of MM we define two alternate formats for a matrix: coordinate and array. Note that *any* matrix could be represented by either of the original two formats. Our philosophy is that a new matrix format need not be defined unless (a) it can save at least one-half the storage of the best alternative, or (b) it supports a format of keen interest to a large community. Examples of other specialized formats which might be reasonable to include are an elemental matrix format (for use in finite element applications), band matrix format, and a Toeplitz matrix format.

The third part of the type code, the *qualifiers*, is a sequence of words which serve to flag additional properties of the data file. Their number and definition depends upon the object type. For matrices two qualifiers are defined; the first indicates the arithmetic field from which data entries are drawn (i.e., real, complex, integer or pattern), and the second indicates symmetry properties which have been used to reduce the amount of data in the file (e.g. symmetric, Hermitian, skew symmetric).

Acknowledgements

John Lewis and Roger Grimes of Boeing Computer Services read several drafts of this document and provided insightful comments which lead to improvements in the design. Iain Duff of the Rutherford Appleton Laboratories and Daniel Pierce at Boeing Computer Services provided additional comments which lead to further improvements in the presentation.

References

[1] Ronald F. Boisvert, Roldan Pozo, Karin Remington, Richard F. Barrett, and Jack J. Dongarra, Matrix Market : A Web Resource for Test Matrix Collections, in *The Quality of Numerical Software: Assessment and Enhancement*, Chapman & Hall, London, 1997, pp. 125-137.

[2] Iain S. Duff, Roger G. Grimes and John G. Lewis, Users' Guide for the Harwell-Boeing Sparse Matrix Collection (Release I), Technical Report TR/PA/92/86, CERFACS, Lyon, France, October 1992.
URL: `ftp://ftp.cerfacs.fr/pub/harwell_boeing/userguide.ps.Z`.

[3] Jason Y. Zien, Horst D. Simon and Alex C. Woo, GAMFF – A Graph and Matrix File Format, NAS Report, NASA Ames Research Center, Moffett Field, CA, November 9, 1995. URL: `http://www.nas.nasa.gov/NAS/DataSets/GAMFF/gamffpaper.ps`