Design of an interactive matrix calculator

by CLEVE MOLER
University of New Mexico
Albuquerque, New Mexico

INTRODUCTION

MATLAB is an interactive computer program that serves as a convenient "laboratory" for computations involving matrices. It provides easy access to matrix software developed by the LINPACK and EISPACK projects [1-3]. The capabilities range from standard tasks such as solving simultaneous linear equations and inverting matrices, through symmetric and nonsymmetric eigenvalue problems, to fairly sophisticated matrix tools such as the singular value decomposition.

It is expected that one of MATLAB's primary uses will be in the classroom. It should be useful in introductory courses in applied linear algebra, as well as more advanced courses in numerical analysis, matrix theory, statistics and applications of matrices to other disciplines. In nonacademic settings, MATLAB can serve as a "desk calculator" for the quick solution of small problems involving matrices.

The program is written in Fortran and is designed to be readily installed under any operating system which permits interactive execution of Fortran programs. The resources required are fairly modest. There are about 6000 lines of Fortran source code, including the LINPACK and EISPACK subroutines used. With proper use of overlays, it is possible to run the system on a minicomputer with only 32K bytes of memory.

The size of the matrices that can be handled in MATLAB depends upon the amount of storage that is set aside when the system is compiled on a particular machine. We have found that an allocation of 4000 words for matrix elements is usually quite satisfactory. This provides room for several 20 by 20 matrices, for example. One implementation on a virtual memory system provides 50,000 elements. Since most of the algorithms used access memory in a sequential fashion, the large amount of allocated storage causes no difficulties.

In some ways, MATLAB resembles SPEAKEASY [4] and, to a lesser extent, APL. All are interactive terminal languages that ordinarily accept single-line commands or statements, process them immediately, and print the results. All have arrays or matrices as principal data types. But for MATLAB, the matrix is the only data type (although scalars, vectors and text are special cases), the underlying system is portable and requires fewer resources, and the supporting subroutines are more powerful and, in some cases, have better numerical properties.

Together, LINPACK and EISPACK represent the state of the art in software for matrix computation. EISPACK is a package of over 70 Fortran subroutines for various matrix eigenvalue computations that are based for the most part on Algol procedures published by Wilkinson, Reinsch and their colleagues [5]. LINPACK is a package of 40 Fortran subroutines (in each of four data types) for solving and analyzing simultaneous linear equations and related matrix problems.

Since MATLAB is not primarily concerned with either execution time efficiency or storage savings, it ignores most of the special matrix properties that LINPACK and EISPACK subroutines use to advantage. Consequently, only 8 subroutines from LINPACK and 5 from EISPACK are actually involved.

This paper gives a brief description of MATLAB from the user's point of view and presents a formal description of the MATLAB language. The system was designed and programmed using techniques described by Wirth [6], implemented in nonrecursive, portable Fortran.

1. Elementary operations

MATLAB works with essentially only one kind of object, a rectangular matrix with complex elements. If the imaginary parts of the elements are all zero, they are not printed, but they still occupy storage. In some situations, special meaning is attached to 1 by 1 matrices, that is scalars, and to 1 by n and m by 1 matrices, that is row and column vectors.

Matrices can be introduced into MATLAB in four different ways:

- Explicit list of elements,
- Use of FOR and WHILE statements,
- Read from an external file,
- Execute an external Fortran program.

The explicit list is surrounded by angle brackets, '{' and '}', and uses the semicolon ';' to indicate the ends of the rows. For example, the input line

\[ A = (1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9) \]
will result in the output

\[ A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \]

The matrix \( A \) will be saved for later use. The individual elements are separated by commas or blanks and can be any MATLAB expressions, for example

\[ x = \begin{bmatrix} -1.3, 4/5, 4*\text{atan}(1) \end{bmatrix} \]

results in

\[ X = \begin{bmatrix} -1.3000 & 0.8000 & 3.1416 \end{bmatrix} \]

The elementary functions available include \( \text{sqrt} \), \( \log \), \( \exp \), \( \sin \), \( \cos \), \( \text{atan} \), \( \text{abs} \), \( \text{round} \), \( \text{real} \), \( \text{imag} \), and \( \text{conjg} \).

The FOR statement allows the generation of matrices whose elements are given by simple formulas. The above matrix \( A \) could also have been produced by

for \( i = 1:3 \), for \( j = 1:3 \), \( A(i,j) = 3*(i-1) + j; \)

The semicolon at the end of the line suppresses the printing, which in this case would have been nine versions of \( A \) with changing elements. Several statements may be given on a line, separated by semicolons or commas.

Names of variables are formed by a letter, followed by any number of letters and digits, but only the first four characters are remembered.

The special character prime (\( ' \)) is used to denote the transpose of a matrix, so

\[ x = x' \]

changes the row vector above into the column vector

\[ X = \begin{bmatrix} -1.3000 \\ 0.8000 \\ 3.1416 \end{bmatrix} \]

Addition, subtraction and multiplication of matrices are denoted by \(+\), \(-\), and \(*\). The operations are performed whenever the matrices have the proper dimensions. For example, with the above \( A \) and \( x \), the expressions \( A + x \) and \( x*A \) are incorrect because \( A \) is 3 by 3 and \( x \) is now 3 by 1.

However,

\[ b = A * x \]

is correct and results in the output

\[ B = \begin{bmatrix} 9.7248 \\ 17.6496 \\ 25.5743 \end{bmatrix} \]

Note that both upper and lower case letters are allowed for input (on those systems which have both), but that lower case is converted to upper case.

There are two "matrix division" symbols in MATLAB, \( \backslash \) and \( / \). If \( A \) and \( B \) are matrices, then \( A \backslash B \) and \( B / A \) correspond formally to left and right multiplication of \( B \) by the inverse of \( A \), that is \( \text{inv}(A) \backslash B \) and \( B \backslash \text{inv}(A) \), but the result is obtained directly without the computation of the inverse. In the scalar case, \( 3! \) and \( 1/3 \) have the same value, namely one-third. In general, \( A \backslash B \) denotes the solution \( X \) to the equation \( A \backslash X = B \) and \( B / A \) denotes the solution to \( X \cdot A = B \).

Left division, \( A \backslash B \), is defined whenever \( B \) has as many rows as \( A \). If \( A \) is square, it is factored using Gaussian elimination. The factors are used to solve the equations \( A \cdot X(:,j) = B(:,j) \) where \( B(:,j) \) denotes the \( j \)th column of \( B \). The result is a matrix \( X \) with the same dimensions as \( B \). If \( A \) is nearly singular (according to the LINPACK condition estimator, \( \text{RCOND} \)), a warning message is printed. If \( A \) is not square, it is factored using Householder orthogonalization with column pivoting. The factors are used to solve the overdetermined equations in a least squares sense. The result is an \( m \) by \( n \) matrix \( X \) where \( m \) is the number of columns of \( A \) and \( n \) is the number of columns of \( B \). Each column of \( X \) has at most \( k \) nonzero components, where \( k \) is the effective rank of \( A \).

Right division, \( B / A \), can be defined in terms of left division by \( B / A = (A^\backslash B)' \).

The expression \( A^{*p} \) means \( A \) to the \( p \)th power. It is defined if \( A \) is a square matrix and \( p \) is a scalar. If \( p \) is an integer greater than one, the power is computed by repeated multiplication. For other values of \( p \) the calculation involves the eigenvalues and eigenvectors of \( A \).

There are three predefined variables, \( \text{RAND} \), \( \text{EYE} \) and \( \text{FLOP} \). The value of \( \text{RAND} \) is a random variable, with a uniform or a normal distribution. The name \( \text{EYE} \) is used in place of \( I \) to denote identity matrices because \( I \) is often used as a subscript or as \( \sqrt{-1} \). The dimensions of \( \text{EYE} \) are determined by context. For example,

\[ B = A + 3*\text{EYE} \]

adds 3 to the diagonal elements of \( A \) and

\[ X = \text{EYE} / A \]

is one of several ways in MATLAB to invert a matrix.

\( \text{FLOP} \) provides a count of the number of floating point operations, or "flops," required for each calculation.

All computations are done using either single or double precision real arithmetic, whichever is appropriate for the particular computer. There is no mixed-precision arithmetic. The Fortran \( \text{COMPLEX} \) data type is not used because many systems create unnecessary underflows and overflows with complex operations and because some systems do not allow double precision complex arithmetic.

2. MATLAB functions

Much of MATLAB's computational power comes from the various matrix functions available. The current list includes:

\( \text{INV}(A) \) — Inverse.
\( \text{DET}(A) \) — Determinant.
\( \text{COND}(A) \) — Condition number.
\( \text{RCOND}(A) \) — A measure of nearness to singularity.
\( \text{EIG}(A) \) — Eigenvalues and eigenvectors.
\( \text{SCHUR}(A) \) — Schur triangular form.
\( \text{POLY}(A) \) — Characteristic polynomial.
The colon is used in several other ways in MATLAB, but all of the uses are based on the following definition.

\[ j:k \] is the same as \((j, j+1, \ldots, k)\)
\[ j:k \] is empty if \(j>k\)
\[ j:i:k \] is the same as \((j, j+i, j+2i, \ldots, k)\)
\[ j:i:k \] is empty if \(i>0\) and \(j>k\) or if \(i=0\) and \(j<k\).

The colon is usually used with integers, but it is possible to use arbitrary real scalars as well. Thus

\[ 1:4 \] is the same as \((1,2,3,4)\)
\[ 0:0.1:0.5 \] is the same as \((0,0.1,0.2,0.3,0.4,0.5)\)

In general, a subscript can be a vector. If \(X\) and \(V\) are vectors, then \(X(V)\) is \((X(V(1)), X(V(2)), \ldots, X(V(n)))\). This can also be used with matrices. If \(V\) has \(m\) components and \(W\) has \(n\) components, then \(A(V,W)\) is the \(m\) by \(n\) matrix formed from the elements of \(A\) whose subscripts are the elements of \(V\) and \(W\). Combinations of the colon notation and the indirect subscripting allow manipulation of various submatrices. For example,

\[
A((1,5,:))=A(5,1,:) \text{ interchanges rows 1 and 5 of } A.
\]
\[
A(2:k,1:n) \text{ is the submatrix formed from rows 2 through } k \text{ and columns 1 through } n \text{ of } A.
\]

4. FOR, WHILE and IF

The FOR clause allows statements to be repeated a specific number of times. The general form is

```
FOR variable = expr, statement, ..., statement, END
```

The END and the comma before it may be omitted. In general, the expression may be a matrix, in which case the columns are stored one at a time in the variable and the following statements, up to the END or the end of the line, are executed. The expression is often of the form \(j:k\), and its “columns” are simply the scalars from \(j\) to \(k\). Some examples (assume \(n\) has already been assigned a value):

\[
A = \text{EYE}(n); \text{ for } i = 1:n, A(i,j) = 1(i+j-1); \text{ end; generates the Hilbert matrix.}
\]

\[
A(i,j) = 0; \text{ end; } A(j,j) = j; \text{ end; } A \text{ changes all but the “outer edge” of the lower triangle and then prints the final matrix.}
\]

```
\text{for } h = 1.0: -0.1: -1.0, (h,cos(pi*h))
```

prints a table of cosines.

\[
\langle X,D \rangle = \text{EIG}(A); \text{ for } v = X,v,A*v
\]
displays eigenvectors, one at a time.

Some of these functions have different interpretations when the argument is a matrix or a vector and some of them have additional optional arguments.

Several of these functions can be used in a generalized assignment statement with two or three variables on the left hand side. For example

\[
\langle X,D \rangle = \text{EIG}(A)
\]
stores the eigenvectors of \(A\) in the matrix \(X\) and a diagonal matrix containing the eigenvalues in the matrix \(D\). The statement

\[
\text{EIG}(A)
\]
simply computes the eigenvalues and stores them in \(\text{ANS}\).

3. Rows, columns and submatrices

Indirect elements of a matrix can be accessed by giving their subscripts in parentheses, e.g., \(A(1,2), x(i), \text{TAB(index}(k-1)+1)\). An expression used as a subscript is rounded to the nearest integer.

Individual rows and columns can be accessed using a colon ‘:’. For the free subscript. For example, \(A(1,:)\) is the first row of \(A\) and \(A(:,j)\) is the \(j\)th column. Thus

\[
A(i,:) = A(i,:) + c*A(k,:)
\]
adds \(c\) times the \(k\)th row of \(A\) to the \(i\)th row.

The colon is used in several other ways in MATLAB, but all of the uses are based on the following definition.

\[ j:k \] is the same as \((j, j+1, \ldots, k)\)
\[ j:k \] is empty if \(j>k\)
\[ j:i:k \] is the same as \((j, j+i, j+2i, \ldots, k)\)
\[ j:i:k \] is empty if \(i>0\) and \(j>k\) or if \(i=0\) and \(j<k\).

The colon is usually used with integers, but it is possible to use arbitrary real scalars as well. Thus

\[ 1:4 \] is the same as \((1,2,3,4)\)
\[ 0:0.1:0.5 \] is the same as \((0.0,0.1,0.2,0.3,0.4,0.5)\)

In general, a subscript can be a vector. If \(X\) and \(V\) are vectors, then \(X(V)\) is \((X(V(1)), X(V(2)), \ldots, X(V(n)))\). This can also be used with matrices. If \(V\) has \(m\) components and \(W\) has \(n\) components, then \(A(V,W)\) is the \(m\) by \(n\) matrix formed from the elements of \(A\) whose subscripts are the elements of \(V\) and \(W\). Combinations of the colon notation and the indirect subscripting allow manipulation of various submatrices. For example,
The WHILE clause allows statements to be repeated an indefinite number of times. The general form is

```
WHILE expr relop expr, statement, ..., statement, END
```

where relop is =, <, >, <=, =>, or (not equal). The statements are repeatedly executed as long as the indicated comparison between the real parts of the first components of the two expressions is true. Here are two examples. (Exercise for the reader: What do these segments do?)

```
eps = 1;
while 1 + eps > 1, eps = eps / 2;
eps = 2 * eps
```

```
E = A + F + EYE; n = 1;
while norm(E + F - E,1) > 0, E = E + F; F = A * F / n;
n = n + 1;
E
```

The IF clause allows conditional execution of statements. The general form is

```
If expr relop expr, statement, ..., statement, ELSE statement, ..., statement
```

The first group of statements is executed if the relation is true and the second group is executed if the relation is false. The ELSE and the statements following it may be omitted. For example,

```
if abs(i - j) = 2, A(i,j) = 0;
```

5. Commands, text and files

MATLAB has several commands which control the output format and the overall execution of the system.

The HELP command allows on-line access to short portions of text describing various operations, functions and special characters.

Results are usually printed in a scaled fixed point format that shows four or five significant figures. The commands SHORT, LONG, SHORT E, and LONG E alter the output format, but do not alter the precision of the computations or the internal storage.

The command CHOP(p) causes p octal or hexadecimal figures to be chopped off after each subsequent floating point operation, thereby simulating a computer with a shorter word length. CHOP(0) restores full accuracy.

The CLEAR command erases all stored variables, except FLOP, RAND and EYE. The statement A = ( ) indicates that a "0 by 0" matrix is to be stored in A. This causes A to be erased so that its storage can be used for other variables.

MATLAB has a limited facility for handling text. Any string of characters delimited by quotes (with two quotes used to allow one quote within the string) is saved as a vector of integer values. For example

```
'A = 2 + 2' is the same as (10 36 46 36 2 36 41 36 2)
```

It is possible, though seldom very meaningful, to use such strings in matrix operations. More frequently, the text is used as a special argument to various functions.

```
NORM(A,'inf') computes the infinity norm of A.
EXEC(T) replaces the remainder of the input line with
the text stored in T.
EXEC('file') obtains subsequent MATLAB input from
an external file.
SAVE('file') stores all the current variables, pointers,
etc. in an external file.
LOAD('file') retrieves everything stored by a previous
SAVE('file')
```

```
PUT('file', 'X') writes X on a file so that it can be retrieved
with GET or accessed by another program.
X = GET('file') reads X from a file where it was placed
by PUT or another program.
```

The operations which access external files cannot be handled in a completely machine-independent manner by portable Fortran code. It is necessary for each particular installation to provide a subroutine which associates external text files with Fortran logical unit numbers.

6. Syntax diagrams

A formal description of the language acceptable to MATLAB, as well as a flow chart of the MATLAB program, is provided by the syntax diagrams or syntax graphs of Wirth [6]. There are ten nonterminal symbols in the language:

```
line, statement, clause, expression, term, factor, number,
integer, name, command.
```

The following syntax diagrams define each of the nonterminal symbols using the others and the terminal symbols:

```
letter—A through Z,
digit—0 through 9,
character—( ) ; : + - * / \ = . , ,< )
text—any sequence of letters, digits, and characters.
```

```
line |
```
```
| | -> statement >-
```
```
| -> clause ------
```
```
| | -> command >---|------>
```
```
| -> expr >-----
```
```
| |----------|-----|
```
```
| | <- ; , <-
```
```
| |---| < ---|---|
```
```
| | <- , <-|
```

From the collection of the Computer History Museum (www.computerhistory.org)
ACKNOWLEDGMENT

Work on MATLAB has been carried out at the University of New Mexico, and during visits to Stanford Linear Accelerator Center, Argonne National Laboratory and Los Alamos Scientific Laboratory. Support has been provided by the National Science Foundation and the Department of Energy.

REFERENCES

Image Processing

NCC '80 has addressed three important areas of image processing through individually organized sessions. These topics are medical imaging, facsimile transmission, and image understanding. In addition, the panel discussion organized by T. Wiener addresses various current topics.

The addressed image processing activities have been strongly influenced by computer technology advances.

Andrew Tescher
Area Director