Math 270AB. Numerical Mathematics

Introduction to Matlab

Instructor: Philip E. Gill

September 20, 2006

Matlab is an interactive matrix manipulation program which allows the user to perform standard matrix operations such as multiplication, inversion, solution of linear equations, LU and QR factorizations, and eigenvalue and singular value decompositions. It includes its own programming language, and two and three-dimensional graphics facilities.

This note provides just a brief introduction to Matlab to "get you going". For further details see the User's Guide [2], and Chapter 4 of [1], in both of which Matlab is fully documented. Matlab has an excellent built-in help facility, as described below, so you might be able to get by without reference to the texts: just ask the system for help on any command you're not sure about!

How to quit

`EXIT` or `QUIT` gets you out of Matlab and back to the operating system.

How to get help

`HELP` displays several screens full of Matlab commands. For help on a specific command, such as `NORM`, say `HELP NORM`. The command `DEMO` runs a set of demonstrations.

To get hard copy

`DIARY filespec` copies all subsequent (non-graphics) output to the named file. `filespec` must be a valid file specification. `DIARY` again turns off the echoing of output. To obtain hard copy print the diary file in the recommended way for your system. The file is in standard ASCII format so may be edited with a text editor prior to printing.

Matrices

Matlab's only data type is a complex matrix that does not require dimensioning. A vector is a matrix with one row (row vector) or one column (column vector); Matlab distinguishes between row and column vectors. When entering matrices, a semi-colon denotes the end of a row (alternatively type a carriage return and start the next row on the next line), and a right apostrophe `'` outside the square brackets denotes conjugate transpose. Here are some examples of how to enter and manipulate matrices:

`A = [2 4 3; -5 0 12]`	2×3 matrix named `A`
`v = [9; 8; -3]`	column vector v
`B = [2*3 pi log(6); 1/3 0 sqrt(-1)]'`	3×2 complex matrix
`C = A' + B`	matrix addition
`x = A*v`	matrix-vector product
`D = inv(A*B)`	inverse of the 2×2 matrix product
`m = 4; n = 7;`	scalars, i.e., 1×1 matrices
`A = zeros(m,n)`	$m \times n$ matrix of zeros
`A = ones(m,n)`	$m \times n$ matrix of 1s
`A = rand(m,n)`	$m \times n$ matrix of random numbers in $[0, 1]$
`E = eye(m,n)`	$m \times n$ identity matrix
`zeros(n)`	same as `zeros(n,n)`
`ones(A)`	matrix of 1's with the dimension of `A`

Accessing parts of a matrix:

`A(i,j)`	matrix element
`A(i,:)`	i-th row
`A(:,j)`	j-th column
`A(p:q,r:s)`	submatrix comprising intersection of rows `p`–`q` and columns `r`–`s`
`A(:)`	the columns of `A` strung out into one long vector

Bits and bobs

When you enter a matrix expression, MATLAB prints the result out unless a trailing semicolon is appended to the expression. The permanent variable `ANS` always contains the result of the most recently evaluated expression, which is useful if you forget to assign the result to a variable and want to recall it.

Other permanent variables include `eps`, the unit roundoff ($2^{-52} \approx 2.2 \times 10^{-16}$), and `pi` $= \pi$.

The format of printed output is controlled by the `FORMAT` statement. The default printing is `FORMAT SHORT`, which is 5 digit scaled fixed point. Also useful are `FORMAT SHORT E`, 5 digit floating point, and the above with `SHORT` replaced by `LONG`, which gives 15 digits.

`i:j:k` is a row vector comprising the numbers from `i` to `k` in steps of `j`, e.g. `0:.5:10` is [0, .5, 1.0, 1.5, ..., 10], and `6:-2:-8` is [6, 4, 2, 0, ..., −8]. `i:k` is the same as `i:1:k`. If `A` has `n` columns then `A(:,1:n) = A(:,n:-1:1)` reverses the columns of `A`.

MATLAB commands must be typed in lower case (though are often shown here as upper case for clarity). Variables can be of either case, and MATLAB is initially case sensitive; case sensitivity can be toggled using `CASESEN`.

`whos`	lists current variables, their sizes, and amount of free memory
`[m,n]=size(A)`	sets `m` and `n` to the number of rows and columns of A
`clear`	clears the workspace: all variables are lost
`save filespec`	saves all variables to the named file
`load filespec`	converse of save: loads all variables back from the named file

Element-by-element operations

Each arithmetic operator `*`, `^`, `/` and `\` has a "dot" counterpart which carries out the operation componentwise. Thus if `C = A.*B`, `D = A.^3`, then `c(i,j) = a(i,j)*b(i,j)` and `d(i,j) = a(i,j)^3` for all `i`, `j`.

Logical and Relational operators

The logical operators are `~` (not), `&` (and), `|` (or). The relational operators are `<`, `<=`, `>`, `>=`, `==`, and `~=`. Note the use of a double `==` for an equality test; a single `=` is used only for assignment. All these operators do element-by-element comparisons between two matrices: they return a matrix of the same size, with elements set to one where the relation is true, and zero where it is not true. To reduce a matrix of ones and zeros to a scalar use `ANY` or `ALL`, which return 1 if, respectively, *any* or *all* the elements of the matrix argument are nonzero, and 0 otherwise.

M-files

Disk files called M-files—ones having a `.m` extension—are used to store sequences of MATLAB statements. A disk file may be defined as a FUNCTION, in which case arguments may be passed in and out and variables are local to the function. M-files behave just like built-in commands and are used in the same way; in fact, many of the commands provided with MATLAB are actually M-files stored in a special directory.

Functions can return multiple output arguments, e.g. `[V,D] = eig(A,B)`. On a particular call not all input or output arguments need be specified, e.g. `D = eig(X)`. The permanent variables `nargin` and `nargout` are used by the function to sense how many input arguments have been passed to it and how many output arguments have been requested.

A `%` denotes that the rest of a line is to be treated as a comment. `TYPE file` lists `file.m` to the screen, assuming `file.m` exists somewhere that MATLAB can find it. `HELP file` lists the leading comment lines of `file.m`. `WHAT` shows a directory listing of the `.m` files (and the `.mat` files created by `SAVE`) in the current directory.

Here is an example of an M-file (stored as `ge.m`). Try to follow this style: good leading comment lines so that `HELP function` really does help; nice indenting and spacing; and small size—aim to keep each M-file no longer than a single page (60 lines), by making full use of MATLAB syntax and breaking long algorithms into separate functions.

```
function [L, U] = ge(A)
%GE     Gaussian elimination without pivoting.
%       [L, U] = GE(A) computes the factorization A = LU,
%       where L is unit lower triangular and U is upper triangular.
%       By itself, GE(A) returns the final reduced matrix from the
%       elimination containing both L and U.

[n, n] = size(A);

for k = 1:n-1

    if A(k,k) == 0
      error('Elimination breaks down with zero pivot.  Quitting...')
    end

    A(k+1:n,k) = A(k+1:n,k)/A(k,k);          % Multipliers

    % Elimination
    i = k+1:n;
    A(i,i) = A(i,i) - A(i,k) * A(k,i); % A rank-1 update to the active submatrix!
```

```
end

if nargout <= 1
   L = A;
   return
end

L = tril(A,-1) + eye(n);
U = triu(A);
```

Finally, some information on the functions you will need most often.

`\` Backslash or matrix left division. `A\B` is roughly the same as `INV(A)*B` , except it is computed in a different way. If `A` is an `N`-by-`N` matrix and `B` is a column vector with `N` components, or a matrix with several such columns, then `X = A\B` is the solution to the equation `A*X = B` computed by Gaussian elimination. Thus `A\EYE(A)` produces the inverse of `A`, for example.

If `A` is an `M`-by-`N` matrix with `M ~= N` and `B` is a column vector with `M` components, or a matrix with several such columns, then `X = A\B` is the solution in the least squares sense to the under- or overdetermined system of equations `A*X = B`. The effective rank, `K`, of `A` is determined from the QR decomposition with pivoting. A solution `X` is computed which has at most `K` nonzero components per column.

`CHOL` Cholesky factorization. `CHOL(X)` uses only the diagonal and upper triangle of `X`. The lower triangle is assumed to be the (complex conjugate) transpose of the upper. If `X` is positive definite, then `R = CHOL(X)` produces an upper-triangular `R` so that `R'*R = X`. If `X` is not positive definite an error message is printed.

`LU` Factors from Gaussian elimination. `[L,U] = LU(X)` stores a upper-triangular matrix in `U` and a "psychologically lower triangular matrix", i.e., a product of lower triangular and permutation matrices, in `L`, so that `X = L*U`. By itself, `LU(X)` returns the output from LINPACK'S DGEFA routine.

`QR` QR decomposition. `[Q,R] = QR(X)` produces an upper-triangular matrix `R` of the same dimension as `X` and a unitary matrix `Q` so that `X = Q*R`. `[Q,R,E] = QR(X)` produces a permutation matrix `E`, an upper-triangular `R` with decreasing diagonal elements and a unitary `Q` so that `X*E = Q*R`. By itself, `QR(X)` returns the output of LINPACK'S DQRDC routine. `TRIU(QR(X))` is `R`.

`SVD` Singular value decomposition. `[U,S,V] = SVD(X)` produces a diagonal matrix `S`, of the same dimension as `X` and with nonnegative diagonal elements in decreasing order, and unitary matrices `U` and `V` so that `X = U*S*V'`. By itself, `SVD(X)` returns a vector containing the singular values. `[U,S,V] = SVD(X,0)` produces the "economy size" decomposition. If `X` is `m-by-n` with `m > n`, then only the first `n` columns of `U` are computed and `S` is `n-by-n`.

`EIG` Eigenvalues and eigenvectors. `EIG(X)` is a vector containing the eigenvalues of a square matrix X. `[V,D] = EIG(X)` produces a diagonal matrix `D` of eigenvalues and a full matrix `V` whose columns are the corresponding eigenvectors so that `X*V = V*D`.

`DIAG` If `V` is a row or column vector with `N` components, `DIAG(V,K)` is a square matrix of order `N+ABS(K)` with the elements of `V` on the `K`-th diagonal. `K = 0` is the main diagonal, `K > 0` is above the main diagonal and `K < 0` is below the main diagonal. `DIAG(V)` simply puts `V` on the main diagonal. For example,

```
DIAG(-M:M) + DIAG(ONES(2*M,1),1) + DIAG(ONES(2*M,1),-1)
```

produces a tridiagonal matrix of order `2*M+1`. If `X` is a matrix, `DIAG(X,K)` is a column vector formed from the elements of the `K`-th diagonal of `X`. `DIAG(X)` is the main diagonal of `X`. `DIAG(DIAG(X))` is a diagonal matrix.

`TRIU` Upper triangle. `TRIU(X)` is the upper-triangular part of `X`. `TRIU(X,K)` is the elements on and above the `K`-th diagonal of `X`. `K = 0` is the main diagonal, `K > 0` is above the main diagonal and `K < 0` is below the main diagonal.

`TRIL` Lower triangle. `TRIL(X)` is the lower-triangular part of `X`. `TRIL(X,K)` is the elements on and below the `K`-th diagonal of `X`. `K = 0` is the main diagonal, `K > 0` is above the main diagonal and `K < 0` is below the main diagonal.

`NORM` *NB: The 1 and ∞ matrix norms used by* MATLAB *differ from the "genuine norms" when the matrix is complex.* For matrices:

`NORM(X)`	is the largest singular value of `X`
`NORM(X,1)`	is the 1-norm of `X`, the largest column sum, `MAX(SUM(ABS(REAL(X))+ABS(IMAG(X)))`
`NORM(X,2)`	is the same as `NORM(X)`
`NORM(X,inf)`	is the infinity norm of `X`, the largest row sum, `MAX(SUM(ABS(REAL(X'))+ABS(IMAG(X')))`
`NORM(X,'fro')`	is the Frobenius-norm, `SQRT(SUM(DIAG(X'*X)))`

For vectors:

`NORM(V,P)`	=	`SUM(ABS(V)^P)^(1/P)`
`NORM(V)`	=	`NORM(V,2)`
`NORM(V,inf)`	=	`MAX(ABS(V))`
`NORM(V,-inf)`	=	`MIN(ABS(V))`

`INV` Matrix inverse. `INV(X)` is the inverse of the square matrix `X`. A warning message is printed if `X` is badly scaled or nearly singular.

`ORTH` Orthogonalization. `Q = ORTH(A)` is an orthonormal basis for the range of `A`. The columns of `Q` span the same space as the columns of `A`, the number of columns of `Q` is the rank of `A` and `Q'*Q = EYE(A)`.

`NULL` Null space. `Q = NULL(A)` is an orthonormal basis for the null space of `A`. `Q'*Q = I` and `A*Q = 0`.

`SCHUR` Schur decomposition. `[U,T] = SCHUR(X)` produces a Schur matrix `T` and a unitary matrix `U` so that `X = U*T*U'` and `U'*U = EYE(U)`. By itself, `SCHUR(X)` returns `T`. If `X`is complex, the Complex Schur Form is returned in matrix `T`. The Complex Schur Form is upper triangular with the eigenvalues of `X` on the diagonal. If `X` is real, the Real Schur Form is returned. The Real Schur Form has the real eigenvalues on the diagonal and the complex eigenvalues in 2-by-2 blocks on the diagonal. See `RSF2CSF` to convert from Real to Complex Schur form.

MATLAB built-in functions: Copyright (c) 1984, The MathWorks, Inc.

intro	<	chol	end	function	ltitr	qr	sort
help	>	clc	eps	global	lu	quit	sprintf

demo	=	clear	error	grid	macro	qz	sqrt
[	&	clg	eval	hess	magic	rand	startup
]	|	clock	exist	hold	max	rcond	string
(	~	conj	exit	home	memory	real	subplot
)	abs	contour	exp	ident	mesh	relop	sum
.	all	cos	expm	if	meta	rem	svd
,	ans	cumprod	eye	imag	min	return	tan
;	any	cumsum	feval	inf	nan	round	text
%	acos	delete	fft	input	nargin	save	title
!	asin	det	filter	inv	norm	schur	type
:	atan	diag	find	isnan	ones	script	what
'	atan2	diary	finite	isstr	pack	semilogx	while
+	axis	dir	fix	keyboard	pause	semilogy	who
-	balance	disp	floor	load	pi	setstr	xlabel
*	break	echo	flops	log	plot	shg	ylabel
\	casesen	eig	for	loglog	polar	sign	zeros
/	ceil	else	format	logop	prod	sin	
^	chdir	elseif	fprintf	ltifr	prtsc	size	

Directory of M-files

acosh	cond	expm3	histogra	matdemo	ode23	rat	table2
angle	conv	feval	humps	matlab	ode45	ratmovie	tanh
asinh	conv2	fft2	idft	mean	odedemo	readme	toeplitz
atanh	corr	fftshift	ieee	median	orth	residue	trace
bar	cosh	fitdemo	ifft	membrane	pinv	roots	translat
bench	ctheorem	fitfun	ifft2	menu	plotdemo	rot90	tril
bessel	deconv	flipx	info	meshdemo	poly	rref	triu
bessela	demo	flipy	int2str	meshdom	polyfit	rrefmovi	unmkpp
besselh	demolist	funm	invhilb	mkpp	polyval	rsf2csf	vdpol
besseln	dft	gallery	isempty	movies	polyvalm	sinh	why
blanks	diff	gamma	kron	nademo	ppval	spline	wow
cdf2rdf	edit	gpp	kronplot	nelder	print	sqrtm	zerodemo
census	eigmovie	hadamard	length	neldstep	quad	square	zeroin
cla	etime	hankel	log10	nnls	quaddemo	startup	
compan	expm1	hilb	logm	null	quadstep	std	
computer	expm2	hist	logspace	num2str	rank	table1	

References

[1] T.F. Coleman and C.F. Van Loan, *A Matrix Computation Handbook*, SIAM Publications, Philadelphia, 1988.

[2] C.B. Moler, J.N. Little and S. Bangert, *PC-Matlab User's Guide*, The MathWorks, Inc., 20 North Main St., Sherborn, Massachusetts 01770, 1987.

N.J. Higham, 31 Jul 1989
LaTeX Version by P.E. Gill, 14 Sep 1989