1. Language Usage and Matlab Interface

American grammar is unusual in that punctuation, even if it is not part of the quote, is “absorbed” into the quote; for example,

He said “oh, damn,” paused, then added “assessment fails me again.”

In this document I will use proper English grammar so that punctuation not intrinsic to the quote will be used as a delimiter, and appear outside the quote; for example,

He said “oh, damn”, paused, then added “assessment fails me again”.

Commands to be entered into Matlab on the command line will be set in TeX fixed, non-proportional teletype font; for example, asking Matlab to compute the expression

\[ 4.56 \cos(2.5\pi/4 + \pi/8) + \exp(-\pi) \]
can be done by typing
\[ 4.56 \times \cos(2.5 \times \pi/4 + \pi/8) + \exp(-\pi); \]
\[ \text{ans} \]
at the Matlab command prompt.

All of the following refers to usage under Matlab version 7.13 (R2011b). Like any software package there are some situations which require additional advice.

1. Sometimes the Desktop gets corrupted with overlapping windows, etc.; do the following to restore it: click Desktop -> Desktop Layout -> Default
2. To tile a graph on the Desktop, click (on the graph) Desktop -> Dock Figure
3. You may need to change the Current Folder (probably workspace) if your m-files are in another directory; otherwise you will get a “file (or function) not found” error. If you start Matlab from a cmd.exe prompt, then this will be the current working directory.
4. If you want to know about, say “matlab_widgit” you can type “doc matlab_widgit” and help pages will come up.

An on-line reference can be found at

2. Matlab Global Syntax and Semantic Structure

Matlab, like any programming language has standard operators for assignment and arithmetic and logical operations and most of these apply to the toolkits as well. One fine point is that some operations (such as multiplication) can be done either entry-wise or operator-wise, so the “dot”-operator is used as a prefix in order to distinguish the two operations.

Table I - General Operators and Constructs

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>assignment operator</td>
</tr>
<tr>
<td>( )</td>
<td>parentheses for expressions</td>
</tr>
<tr>
<td>+ - *</td>
<td>scalar/matrix addition subtraction and multiplication</td>
</tr>
<tr>
<td>^</td>
<td>scalar/matrix exponentiation</td>
</tr>
<tr>
<td>/ \</td>
<td>right and left matrix division</td>
</tr>
<tr>
<td>.+ .- .*</td>
<td>entry-wise matrix addition subtraction and multiplication</td>
</tr>
<tr>
<td>.^</td>
<td>entry-wise matrix exponentiation</td>
</tr>
<tr>
<td>./</td>
<td>entry-wise matrix division</td>
</tr>
<tr>
<td>'</td>
<td>matrix transpose</td>
</tr>
<tr>
<td>;</td>
<td>complete command and suppress output</td>
</tr>
<tr>
<td>...</td>
<td>continue on next line</td>
</tr>
</tbody>
</table>

**remark:** one needs both right and left division for matrices because matrix multiplication is **not** commutative and we can form both \( \text{inv}(A) \times B = (A \backslash B) \) and \( B \times \text{inv}(A) = (B/A) \).

The following constants and special variables are built in.
Table II - Constants and Special Variables

<table>
<thead>
<tr>
<th>i or j</th>
<th>the square root of $-1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi$</td>
<td>$\pi = 3.1415926\ldots$</td>
</tr>
<tr>
<td>ans</td>
<td>variable holding the answer to the last computation</td>
</tr>
<tr>
<td>eps</td>
<td>the machine epsilon, i.e. the smallest floating point number $h$ such that $1 + h &gt; 1$</td>
</tr>
<tr>
<td>inf</td>
<td>infinity</td>
</tr>
</tbody>
</table>

Note that you can do a calculation without assignment and then use **ans** to recall the result; for example

$$(3.45 \times 2.18)/2.07;$$

**ans**

The are some very useful general commands and functions.

Table III - General Commands and Functions

<table>
<thead>
<tr>
<th>clear</th>
<th>clear all non-permanent variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear all</td>
<td>clear variables, functions, etc.</td>
</tr>
<tr>
<td>clf</td>
<td>clear figures</td>
</tr>
<tr>
<td>who</td>
<td>list all variables</td>
</tr>
<tr>
<td>whos</td>
<td>list all variables together with their sizes</td>
</tr>
<tr>
<td>ver</td>
<td>list all installed toolboxes</td>
</tr>
</tbody>
</table>

3. Matlab Numerical Mathematics

The Matlab core contains fundamental numerical operations to handle both real and complex scalars and matrices with double-precision floating point accuracy.

**Data Types.** There is only one data type and that is a matrix with (possibly complex) double-precision entries. Row vectors are $1 \times n$ matrices $v$ where $n = \text{length}(v)$. Column vectors are transposes of row vectors. Scalars are $1 \times 1$ matrices. The **size** function can be used on the command line or in function calls to find out the number of rows and columns; for example

$$z = 3.5;$$

$$[m \ n] = \text{size}(z)$$

will print out “$m = 1$” and “$n = 1$”, whereas

$$z = [3.4 \ -5.6];$$

$$[m \ n] = \text{size}(z)$$

will print out “$m = 1$” and “$n = 2$”.

**Identifiers.** Variable are case sensitive and must start with a letter; following that, letters, digits, and underscores can be added (but only the first 19 characters are significant). Do **not use the hyphen** in either variable or function names.
Table IV - Matlab Matrix Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>$[m \ n] = \text{size}(A)$ returns rows and columns of $A$</td>
</tr>
<tr>
<td>length</td>
<td>returns the length of a vector</td>
</tr>
<tr>
<td>det</td>
<td>returns the determinant of a square matrix</td>
</tr>
<tr>
<td>inv</td>
<td>returns the inverse of the matrix</td>
</tr>
<tr>
<td>norm</td>
<td>returns the $\ell^2$-norm of the matrix</td>
</tr>
<tr>
<td>cond</td>
<td>returns the condition number of the matrix $A$, i.e. $\text{norm}(A)*\text{norm}(\text{inv}(A))$</td>
</tr>
<tr>
<td>rank</td>
<td>returns the rank of the matrix, i.e. the maximum number of linearly independent rows</td>
</tr>
<tr>
<td>eig</td>
<td>$[V \ D] = \text{eig}(A)$ returns eigenvalues and eigenvectors of the matrix $A$</td>
</tr>
</tbody>
</table>

Certain functions operate element-wise on matrices; some important ones are:

Table V - Matlab Scalar and Element-wise Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigonometric</td>
<td>cos, sin, tan, acos, asin, atan,</td>
<td>cosine, sine, tangent inverse cosine, sine, tangent</td>
</tr>
<tr>
<td>General</td>
<td>abs, log, exp, sqrt, sign</td>
<td>absolute value (natural) logarithm exponential function square root sign (positive, negative, etc.)</td>
</tr>
<tr>
<td>Complex</td>
<td>real, imag, cong</td>
<td>real and imaginary parts complex conjugate</td>
</tr>
</tbody>
</table>

Some Linear Algebra. Scalars are entered into Matlab the same as in (almost) any programming language with the assignment operator, for example

$$z = 3.5$$

Note that if you do not end an assignment with a semicolon, Matlab will confirm the value(s) of the assignment. You may want this interactively, but in most m-file scripts you would want the assignment to be “silent”, so you would terminate the assignment with a semicolon.

$$z = 3.5;$$

Row vectors ($1 \times n$ matrices) are entered with the values separated by white space between square brackets, for example

$$x = [2.3 \ -3.4 \ 0.3 \ 1.7];$$

You may enter complex vectors as well but be careful with whitespace, for example, try

$$y = [2.3-3.4*j \ 0.3+1.7*j];$$

whos

Note that you can check assignment sizes with the operator “whos”. The transpose operator (single quote ‘) can be used to make a column vector from a row vector (or vice-versa), for example

$$z = x';$$
4. Matlab Symbolic Mathematics

The Matlab Symbolic Math Toolbox contains functions designed to do mathematics symbolically, obtaining closed-form solutions \textbf{whenever possible}. Obviously, solving certain types of, for example, differential equations may not yield closed-form solutions so these operations may fail, and it is the responsibility of the user to understand this.

Table VI - Symbolic Math Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculus</td>
<td>diff</td>
</tr>
<tr>
<td></td>
<td>int</td>
</tr>
<tr>
<td></td>
<td>differentiate</td>
</tr>
<tr>
<td></td>
<td>integrate</td>
</tr>
</tbody>
</table>
References

[1] ABET, Criteria For Accrediting Computing Programs, 2010–11, online see 
http://www.abet.org/forms.shtml#For_Computing_Programs_Only -> 2010-2011 Criteria (pdf) -> 
Proposed Changes to the Accreditation Criteria (pdf)

[2] ABET, Criteria For Accrediting Engineering Programs, 2010–11, online see 
http://www.abet.org/forms.shtml#For_Engineering_Programs_Only -> 2010-2011 Criteria (pdf) -> 
Proposed Changes to the Accreditation Criteria (pdf)


[4] ACM, Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering, 
online see 

Criterion 3, online see 
http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1000&context=aen_fac
Appendix I - Sample m-file Functions

The unit-step function: ustep.m

function y = ustep(s)

% the heaviside (or unit step) function
%
% example (using both ramp() and ustep()):
% t = -5:0.01:5;
% plot(t, 3*ramp(t+3) - 6*ramp(t+1) + 3*ramp(t) - 3*ustep(t-3);
%
% important: matlab seems to "assume" that user-defined functions
% are automatically vector (not scalar) functions. So we first
% have to get the size:
%
[m n] = size(s);
%
% case: scalar
if n == 1 & s < 0
    y = 0;
elseif n == 1 & s == 0
    y = 0.5;
elseif n == 1 & s > 0
    y = 1;
elseif n == 1
    error('invalid real value: s=%f\n', s);
%
% case: vector (important: semicolon before "end;" keeps it silent)
%
else
    for i = 1:n, y(i) = ustep(s(i)); end;
end
The ramp function: ramp.m

function y = ramp(s)
% ramp function resulting from integrating
% the heaviside (or unit step function)
% example (using both ramp() and ustep()):
% t = -5:0.01:5;
% plot(t, 3*ramp(t+3) - 6*ramp(t+1) + 3*ramp(t) - 3*ustep(t - 3);
% important: matlab seems to "assume" that user-defined functions
% are automatically vector (not scalar) functions. So we first
% have to do get the size:
% [m n] = size(s);
% case: scalar
% if n == 1 && s < 0
% y = 0;
elseif n == 1 && s == 0
y = 0;
elseif n == 1 && s > 0
y = s;
elseif n == 1
    error('invalid real value: s=%f\n', s);
else
% case: vector (important: semicolon before "end;" keeps it silent)
for i = 1:n, y(i) = ramp(s(i)); end;
end
Appendix I - Sample m-file Scripts
DEPARTMENT OF COMPUTER AND ELECTRICAL ENGINEERING, AND COMPUTER SCIENCE, CALIFORNIA STATE UNIVERSITY AT BAKERSFIELD, 9001 STOCKDALE HWY, BAKERSFIELD, CA, 93311-1022, USA

E-mail address: marc@cs.csubak.edu