An Introduction to MATLAB for DSP

Brady Laska

SYSC 4405

Carleton University

September 13, 2007

Overview

1. MATLAB background
2. Basic MATLAB
3. DSP functions
4. Coding for speed
5. Demos

Accessing MATLAB

- Labs on campus
- Purchase it
  - ↪ commercial editions $$$, student editions $ + toolboxes $
- Use GNU Octave
  - ↪ Compatible syntax
  - ↪ Free download (Windows, Mac, Linux) from `http://www.octave.org`
  - ↪ Octave-forge add-on contains most functions from signal processing toolbox
  - ↪ All in-class examples will run in both MATLAB and Octave (possibly with modification)

Uses of MATLAB in DSP

- Analyze data
  - ↪ import, export, number-crunch, curve fitting
- Visualize and explore data
  - ↪ interactive, easy to transform data, powerful plotting/graphics
- Implement/prototype/test algorithms
  - ↪ vast library of built-in functions, available add-on toolboxes, integration with Simulink
  - ↪ easy to map algebra of DSP algorithms to MATLAB syntax
- Simulation, modelling

MATLAB is

- A programming language
- An interactive numerical computation environment
- An interactive development environment
- A programming library and API
- A graphics system (for plotting, GUI creation)

MATLAB the programming language

Typical programming constructs

Looping: `for`, `while`, `break`

Branching: `if-elseif-else`, `switch-case`

Datatypes

- Standard datatypes are scalar, vector and matrix of double
  ↪ also: integer, boolean, char, string, structure, cell array
- Arrays (vector/matrix) are 1-based and automatically re-size

MATLAB the programming language

Other language aspects

- Case sensitive
- Dynamically typed
- Interpreted (mostly)
- Whitespace and terminating ';' are optional
- Interfaces with other languages (C, FORTRAN, Java)
- Object Oriented
  - ↪ classes, operator overloading

Navigating

- The MATLAB prompt supports common Linux and Windows shell commands

`pwd`	current directory path
`cd` *newdirectory*	change directory
`ls/dir`	lists files in current directory
`!`*command*	executes *command* in the system shell
	example: `>>!grep fft *.m`

Housekeeping

- When you're lost

`who,whos`	list variables and sizes
`help` *commandname*	prints usage and documentation
`lookfor` *key*	scans documentation for *key* and prints matches
Tab completion, history	

- Cleaning up

`clear` *x*	clear variable *x*, or use `clear all`
`clc,clf`	clears the console and current figure respectively

Mathamathical operations

Functions

All functions that are frequently used in DSP are included and named as you'd expect:
`sin,cos,tan,exp,sinc,log,log10,log2,sqrt,pow,...`

Arithmetic operators

Matrix operators perform the linear-algebra-defined matrix operation (matrix multiplication, exponential).
Array operators work element-by-element and are indicated by adding a period before the operator.

Complex numbers

- Built-in complex number suppport
- Keywords `i,j` both equal $\sqrt{-1}$ (watch when using index variables and complex numbers in the same function)
  - ↪ example: creating a complex number `>>x = 1 + 2j`
- Functions for manipulating complex numbers:
  `real,imag,conj,abs,angle,cart2pol,pol2cart`

Vectors

Unlike other programming languages, MATLAB has two distinct types of 1-dimensional arrays (vectors).

Row vectors: `>>x = [1,2,3];`
Default for range operations such as `x = 1:10`.

Column vectors: `>>y = [1;2;3];`
Default for signals. Functions such as `plot`, `fft`, `sum`, `mean`, etc. that take a vector input will evaluate each column of a matrix as a separate signal.

Common matrix creation commands

`ones(M,N)`	matrix of ones
`zeros(M,N)`	matrix of zeros
`eye(N)`	$N \times N$ identity matrix
`randn(M,N)`	matrix of zero-mean unit variance Gaussian random numbers, aka white noise
`rand(M,N)`	matrix of uniform random numbers on $[0, 1]$
`diag(x)`	matrix with `x` along the diagonal

Note that `ones(1E6)` will attempt to create a $10^6 \times 10^6$ matrix, not a $10^6 \times 1$ vector.

Matrix Indexing

- Most functions can operate on either scalar, vector, or matrix; clever indexing allows functions to be applied to a select subset of your data.
- Elements in a matrix can be accessed using subscripts or linear indices. Functions `sub2ind` and `ind2sub` are used to convert back and forth.
- Subsets defined by logical matrix or index set.

Indexing methods

Logical matrix

- A matrix of logical ones and zeros (or `boolean` datatype in new versions) same size as vector/matrix.
- Logical matrices can be combined using Boolean algebra and logical operators: `==`,`~=`,`>`,`<`,`&`,`|`,`xor`. Note that `&`,`|` accept and return matrices while `&&`,`||` accept and return scalars and are used for control statements.
- Logical vectors can be collapsed to scalars for control statements using `any` and `all`.

Indexing methods

Index set

- A matrix of linear indices in the range `1:prod(size(A))`.
- Expression is evaluated at the indices in the set.
- Sets created using `find(`*Boolean statement*`)`.
- Index sets can be combined using set operations: `union`, `intersect`, `unique`, `setxor`, `setdiff`.

Matrix reshaping

`size(A)`	returns the size of the matrix
`A(:)`	convert any matrix or vector to a column vector
`A', A.'`	conjugate and non-conjugate transpose. Generally use conjugate transpose in DSP.
`reshape`	reshapes a matrix, traverses column-wise
`repmat`	useful for adding/multiplying a vector to each row/column of a matrix
`flipud,fliplr`	flips the vector/matrix

Working with transfer functions

MATLAB has many functions for analyzing and constructing filters and transfer functions.

`roots`	find the zeros of a polynomial
`poly`	construct a polynomial from a set of roots
`zplane`	plot poles and zeros on the complex plane
`residuez`	z-transform partial fraction expansion
`fdatool`	filter design and analysis tool
`fvtool`	filter visualization tool

Other useful DSP functions

`filter(B,A,x)`	FIR and IIR filtering
`fftfilt(B,x)`	FIR filtering using the FFT
`conv`	discrete convolution (polynomial multiplication)
`buffer`	divide a signal into (possibly overlapping) frames
windows	`hanning`, `hamming`, `blackman`
	`kaiser`, `bartlett`
`xcorr`	auto and cross-correlation

Vectorizing

Definition

Vectorization: replacing loops with calls to vector functions.

- MATLAB used to be entirely interpreted and loops were very slow. MATLAB now has JIT acceleration so code using loops with built-in functions can be as fast as vectorizing.
- Vectoring can still make your code faster, more readable, and more amenable to parallelization. Code says *what* you want to do, not *how* to do it.
- Vectorization makes extensive use of index sets and logical matrices.

Coding for speed

As usual, don't sacrifice readability and clarity for speed.

Pre-allocate
: Use `ones,zeros` to intialize vectors/matrices. Very important, especially for big matrices.

Profile
: Use `profile on, profile report` and `tic, toc` to time code execution.

Mex functions
: If you *really* need speed, write your function in C or FORTRAN with Mex interfaces.

Demos

Example

AM Modulation Create a baseband signal, modulate it using a carrier sinusoid.

Example

Noise removal Identify signal components, remove noise to recover signal.

References

- P. Venkataraman, "Matlab: A Fast Paced Introduction", Online at: `http://www.rit.edu/~pnveme/Matlab_Course/DEFAULT.HTM`.
- S. Roth and A. Balan, "Introduction to Matlab (Demo)", Online at: `http://www.cs.brown.edu/courses/csci1430/MatlabDemo.html`.