Matlab Overview

Daniel Zwillinger, PhD

13 June 2001

Sudbury 1-1-565   telephone 978-440–1660

Daniel_I_Zwillinger@raytheon.com

http://www.az-tec.com/zwillinger/talks/20010613/

Abstract

This course will expose users to the Matlab software language. The language will be described at a high level. Matlab's capabilities (data types, programming constructs, functions, toolboxes, and graphics) and how users tend to use them will be discussed. A discussion of when to use Matlab will be given. A demonstration will be given.

Outline

- Classes of computer languages
- Matlab: A linear algebra language
- Programming
- Functions
- Graphics
- Local facilities
- Conclusion

Classes of computer languages

- Database: mySQL, Oracle, . . .
- Numerical: Basic, FORTRAN, C, Matlab, . . .
- Symbolic: Lisp, Maple, Mathematica, . . .
- Web: HTML, PHP, XML, . . .
- Low level: assembler, machine language, . . .
- Text processing: sed, awk, perl, python, . . .
- Typesetting: Troff, TeX, LaTeX, . . .

Matlab: MATrix LABoratory

- First introduced at Stanford University in 1979
- Initially an interactive shell to FORTRAN routines
- MathWorks was formed to market Matlab
- Webb & Wilson, Dr. Dobb's Journal, Jan 1999

  "Like every other scripting language, Matlab began as a simple way to do powerful things, and it has become a not-so-simple way to do very powerful things."

Matlab: MATrix LABoratory

- Powerful engineering environment and language, useful for problem solving, data analysis, modeling and visualization
- Runs on nearly every operating system
- More than 400 books in 17 languages
- Diverse and powerful built-in functions
  - linear algebra
  - polynomials
  - Fourier analysis
  - differential equations
  - GUI builder
  - Movies & sound

Matlab: A linear algebra language

- Underlying data structure is a multi-dimensional array (e.g., scalar, vector, or matrix)

$$2 \qquad \begin{bmatrix} 4 & 5 & 6 \end{bmatrix} \qquad \begin{bmatrix} 8 \\ 9 \end{bmatrix} \qquad \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \end{bmatrix}$$

Programming I

1. Use interactively or as programming language (interpreted or compiled)
2. Can link to other languages (e.g., compiled C code)
3. Comprehensive help facility
4. Large number of included examples
5. Conditionals, looping, functions, globals, etc
6. Sophisticated debugger, profiler
7. Integrated programming environment
8. GUI development tools

Programming II

1. Object oriented capabilities
2. Variable number of input and output arguments
3. Handles sparse matrices, multi-dimensional arrays
4. Case sensitive variables
5. Operator precedence
   (a) arithmetic ($+$, $-$, $*$, $/$, etc.)
   (b) relational ($==$, $<$, $>$, etc)
   (c) logical (`AND`, `OR`, `NOT`, etc)
6. Memory partitioned into "workspaces"
7. "Toolboxes" contain collections of functions
8. (Optional) Space allocation for data structures

Many tools for matrix manipulation I

```
>> A=zeros(2,2)

A =

     0     0
     0     0

>> B=ones(3,2)

B =

     1     1
     1     1
     1     1
```

```
>> C=eye(3)

C =

     1     0     0
     0     1     0
     0     0     1

>> D=[1 2; 3 4]

D =

     1     2
     3     4
```

Many tools for matrix manipulation II

```
>> s= 1:4
s =
     1     2     3     4
>> s2= 1:2:7
s2 =
     1     3     5     7
>> s3= 1:.5:2
s3 =
   1.0000   1.5000   2.0000
```

```
>> r=rand(2,3)
r =
   0.47   0.85   0.20
   0.42   0.53   0.67
>> r(:,1)
ans =
   0.47
   0.42
>> r(:,1)'
ans =
   0.47   0.42
```

Many tools for matrix manipulation III

```
>> A=zeros(2,2);
>> B=ones(3,2);
>> C=[ [A;B], [B+5;A-7] ]
C =
     0     0     6     6
     0     0     6     6
     1     1     6     6
     1     1    -7    -7
     1     1    -7    -7
```

```
>> C(:,[1 4])
ans =
     0     6
     0     6
     1     6
     1    -7
     1    -7
```

Functions extend naturally to higher dimensional objects

```
>> log( 1 )
ans =
     0
>> log( [1 2] )
ans =
         0    0.6931
>> log( [1 2; 0 NaN] )
Warning: Log of zero.
ans =
         0    0.6931
      -Inf       NaN
```

Function examples

Say `u=[1 2 3]`, then

Input	Output
`u<3`	`[1 1 0]`
`all(u<3)`	`0`
`any(u<3)`	`1`
`find(u<3)`	`[1 2]`

Easy high level manipulations

- Example: find change in eigenvalues when the identity matrix is slightly perturbed

```
>> a = eye(4) + 0.01*rand(4,4)
a =
    1.0095    0.0089    0.0082    0.0092
    0.0023    1.0076    0.0044    0.0074
    0.0061    0.0046    1.0062    0.0018
    0.0049    0.0002    0.0079    1.0041
>> eig(a)
ans =
   1.0232
   1.0009 + 0.0046i
   1.0009 - 0.0046i
   1.0023
```

Linear algebra operations

- To solve $Ax = b$ (when A is invertible), write

```
x=inv(A) * b
```

or

```
[L,U,P]=lu(A)
x=inv(U) * inv(L) * P * b
```

or

```
x=A\b
```

or

```
...
```

Solving systems of equations ($Ax = b$)

- Write `x=A\b` (even if A is **not** invertible!)

```
>> A=rand(3,2)
A =
    0.7095    0.1897
    0.4289    0.1934
    0.3046    0.6822
```

```
>> b=rand(3,1)
b =
    0.3028
    0.5417
    0.1509
>> soln=A\b
soln =
    0.6387
   -0.0121
```

Various matrix extensions

- Sparse matrices

```
>> A=speye(100000,100000);
>> A2=2*A;
>> A2(4,5)=5;
>> nnz(A2)
ans =
       100001
```

- Multidimensional arrays

```
>> r=rand(2,2,3)
r(:,:,1) =
    0.1389    0.1987
    0.2028    0.6038
r(:,:,2) =
    0.2722    0.0153
    0.1988    0.7468
r(:,:,3) =
    0.4451    0.4660
    0.9318    0.4186
```

Vectorized operations are fast

- Need a vector of $\sin(t)$ for $0 \leq t \leq 10$

```
>> tic % start timer
>> i=0;
>> for t=0:0.001:10
      i=i+1;
      y(i)=sin(t);
   end
>> time1=toc
time1 =
    0.1936
```

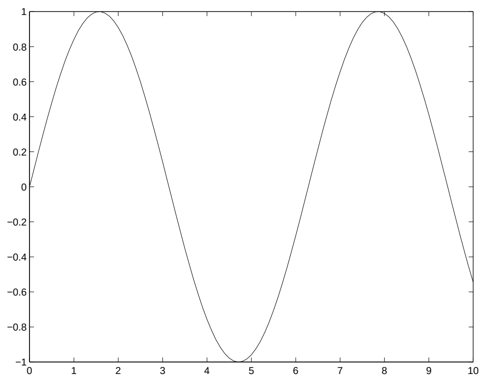
```
>> tic % start timer
>> t=0:0.001:10;
>> y=sin(t);
>> time2=toc
time2 =
    0.0111
>> time1/time2
ans =
   17.4677
```

Special variables

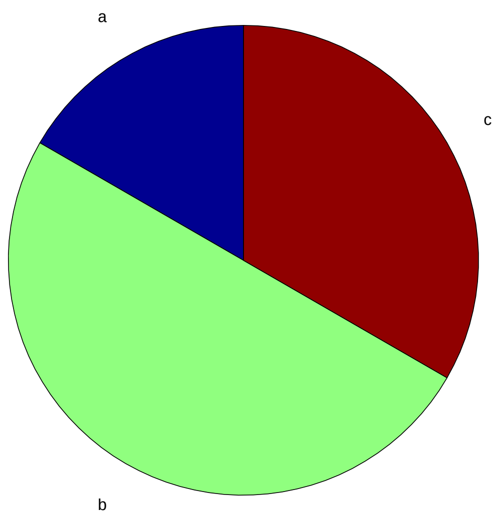
`ans`	most recent result
`eps`	machine epsilon
`flops`	total floating point ops during session
`i,j`	$\sqrt{-1}$
`inf`	∞
`NaN`	not-a-number
`pi`	π
`realmax`	largest positive floating point number
`realmin`	smallest positive floating point number

Graphics I (two-dimensional graphics)

```
>> x=0:.1:10;
>> y=sin(x);
>> plot(x,y)
```

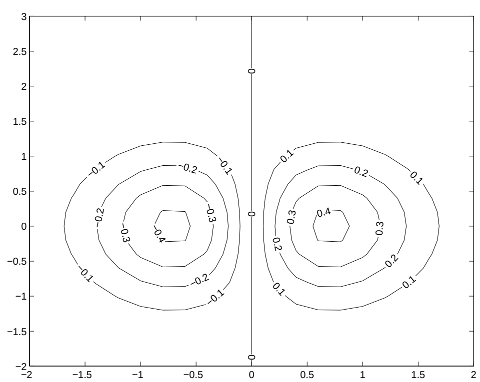



```
>> data=[2 6 4];
>> text={'a','b','c'};
>> pie(data,text)
```

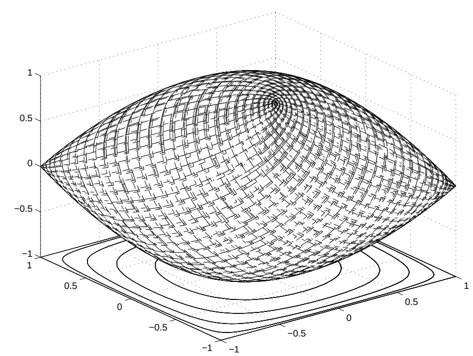


Graphics II (three-dimensional graphics)

```
>> [X,Y]=...
    meshgrid(-2:.2:2,-2:.2:3);
>> Z = X.*exp(-X.^2-Y.^2);
>> [C,h] = contour(X,Y,Z);
>> clabel(C,h)
```



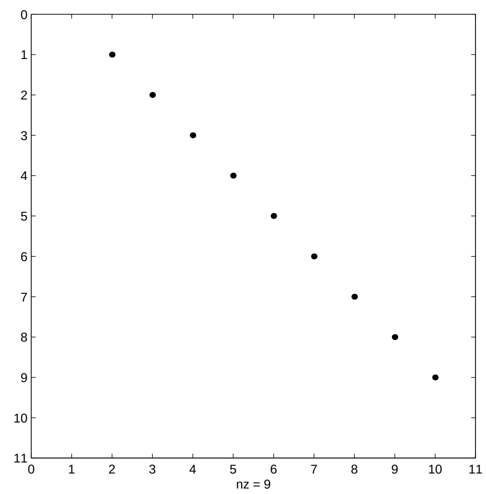
```
>> t=0:.1:10;
>> x=sin(t);
>> y=cos(t);
>> z=x'*y;
>> meshc(x,y,z);
```



Graphics III (matrix visualization)

```
>> format +
>> A=random(5,15)-1/2
A =
++--+++-+++++-+
+-+---+-++-+++-
-++--+-+--+-+--
+-+--+---+++---
-+-++-----+--+-
```

```
>> B=zeros(10,10)
>> for i=1:9
>>    B(i,i+1)=2;
>> end
>> spy(B)
```



Graphics IV

- Dozens of graphic styles
- Lighting schemes
  - ambient light
  - diffuse reflection
  - specular reflection
  - specular exponent
  - specular color reflectance
- Movies: store frames as columns of a matrix
- Can create fly-bys and other animation features
- Easy GUI creation

Functions

- Usually, each function in its own file:

file *run.m*

```
function a=run(b)
% comments
.
.
.
d=2*b
c=foo(d)
.
.
.
a=log(c)
```

file *foo.m*

```
function e=foo(f,g)
.
.
.
[h,i]=bar(f)
.
.
.
e=sqrt(h)
```

file *bar.m*

```
function [j,k,l]=bar(m)
.
.
.
j=m
k=2*m
l=m*m
```

- Have variables **nargin** and **nargout**
- Frequently have many short files

Simulink

- Simulink is a companion to MATLAB
- Useful for modeling dynamic systems
- Provides GUI for building/using block diagrams
- Models are hierarchical
- Similar to National Instrument's *LabView*, except building blocks are Matlab functions

Toolboxes

- Available from the MathWorks and other sources
- `http://www.mathtools.net/MATLAB/toolboxes.html` lists more than 200

1. Communications Toolbox
2. Control System Toolbox
3. Data Acquisition Toolbox
4. Filter Design Toolbox
5. Financial Derivatives Toolbox
6. Financial Toolbox
7. Frequency Domain System Id
8. Fuzzy Logic Toolbox
9. Higher-Order Spectral Analysis
10. Image Processing Toolbox
11. Instrument Control Toolbox
12. Instrument Control Toolbox
13. LMI Control Toolbox
14. Mapping Toolbox
15. Model Predictive Control
16. Mu-Analysis and Synthesis
17. Neural Network Toolbox
18. Optimization Toolbox
19. Partial Differential Equation
20. Robust Control Toolbox
21. Signal Processing Toolbox
22. Spline Toolbox
23. Stateflow Coder Toolbox
24. Statistics Toolbox
25. Symbolic Math Toolbox
26. System Identification Toolbox
27. Wavelet Toolbox

Local facilities

- Matlab available for UNIX and Microsoft platforms
- `http://nesystemsengineering.rsc.ray.com/Tools/Matlab/sysmatlabtools.htm` contains
  - Binary conversion tools
  - Calculations-data summary tools
  - Clustering algorithm tools
  - Clutter tools
  - Coordinate transformations
  - Data filtering tools
  - Dave Shnidman detection models
  - Detection models tools
  - File & matrix processing
  - Filter design tools
  - General signal processing
  - General tools
  - Label plots
  - Missile & radar tools
  - Other general plotting tools
  - Probability routines
  - Scale factors rise time
  - Smith chart tools
  - Swerling detection models
  - Target jammer noise samples
  - Thresholds statistics image processing
  - Transfer function conversions

Octave (free Matlab look-alike)

`http://www.octave.org` (UNIX and Microsoft)

GNU Octave is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with Matlab. It may also be used as a batch-oriented language.

Octave has extensive tools for solving common numerical linear algebra problems ... It is easily extensible and customizable via user-defined functions written in Octave's own language, or using dynamically loaded modules written in C++, C, Fortran, or other languages.

GNU Octave is also freely redistributable software. You may redistribute it and/or modify it under the terms of the GNU General Public License (GPL) as published by the Free Software Foundation.

Conclusion

- Use Matlab for problems involving linear algebra

1. Algorithmic design
2. Data analysis & visualisation
3. Detailed design
4. End-to-end performance
5. Fast prototyping
6. Modeling & simulation
7. Sensitivity analysis
8. Trade studies
9. Web/GUI interaction
10. Typically not for real-time operation