Computational Methods
CMSC/AMSC/MAPL 460

Ramani Duraiswami,
Dept. of Computer Science

Course Goals

- Introduction to the use of scientific computing techniques to solve problems in various domains
- Understand principles behind algorithms
- Intelligent choice and use of available software
- Understand how to
  - Convert a model into a discrete system on the computer
  - How to deal with data
  - perform simulations for applications
  - Display and evaluate simulation results
  - Appreciate which computations are feasible

"New Paradigm"

- Scientific Discovery through Computing
- Paradigm?
  - A set of assumptions, concepts, values, and practices that constitutes a way of viewing reality for the community that shares them, especially in an intellectual discipline.
- Engineering (aeronautics, fluid dynamics, circuit design, radar, antennas, signal processing, …)
- Physics (stellar dynamics, materials, …)
- Economics/Sociology (modeling and analyzing data, computational statistics, stock picking, …)
- Biology (biostatistics, computational biology, genomics and proteomics, …)
- Computer Science (modeling systems/network performance, information retrieval, …)
- Your field …

Another “paradigm”: Data driven science

- Grab data and process it
- Audio, video, text, MRI, X-Ray, weather, strain-gage, flow, gene-chip, seismograph, …
- Moore’s law drives both processing power, memory, sensor cost and capability
  - Moore’s law: Processor speed doubles every 18 months
  - More generally: Technology X capability will double in Y months
- Need algorithms to process larger and larger data sets, and extract information from them
  - Fit data, Extract model parameters, Learn relationships
  - In general compute with the data

The Course

- Two lectures a week
- Homework every week or other week
- 40% homework, 25% exam 1, 35 % final
  - Attendance/participation will be a factor
- Class web site: http://www.umiacs.umd.edu/~ramani/cmsc460/index.html
- Required Book

Numerical Computing with MATLAB by **Cleve Moler**

- The good news
- The complete book is online!
- Book is also not as expensive as some others (~$40)

Course

- Course comes with Matlab software that is downloadable from the book web site
- Another excellent book/resource:

Numerical Recipes by William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery

Also available online!

Go to www.nr.com

- Practical “in the trenches” book

Homework

- Homework will involve programming in MATLAB
- mainly problems from the text
- Style/Clarity/Cleanliness of output will count
- Work/Results must be easily understood to be interpreted
  - Visualization (graphs)
  - Commented code

Syllabus

- **Introduction, Computer Arithmetic and Errors** (Chapter 1) (approx. 3 lectures)
  - course survey
  - introduction to Matlab
  - machine arithmetic and error analysis
  - stability and conditioning
- **Solving Linear Systems of Equations** (Chapter 2) (approx. 4 lectures)
  - Gaussian elimination
  - well-conditioning vs. ill-conditioning, matrix and vector norms
  - Notions of algorithm complexity
  - sparse systems: direct and iterative methods

Syllabus

- **Interpolation** (Chapters 3) (approx. 4 lectures)
  - polynomial interpolation
  - Other basis functions and polynomials
  - piecewise polynomial interpolation
  - spline interpolation
- **Zeros and Roots** (Chapter 4) (approx. 3 lectures)
  - Linear and Nonlinear systems of equations
  - Bisection, Secant and Newton method
  - Introduction to optimization
- **Solving Linear Least Squares Problems** (Chapter 5) (approx. 3 lectures)
  - data-fitting and least squares
  - QR factorization

Syllabus

- **Integration/Quadrature** (Chapter 6)
  - elementary integration formulas (midpoint, trapezoid, etc.)
  - compound and adaptive integration formulas
  - Gaussian quadrature
- Fourier Analysis (Chapter 8)
- **Ordinary Differential Equations** (Chapter 9) (approx. 4 lectures)
  - ordinary differential equations and Euler's method
  - adaptive methods for ordinary differential equations
  - methods for stiff systems

MATLAB Overview

- History of MATLAB
- Strengths of MATLAB
- Weaknesses of MATLAB

What is MATLAB?

- MATLAB
  - MATrix LABoratory
  - Interactive system
  - Programming language
  - Extendable

What is MATLAB ?: 2

- Considering MATLAB at home
  - Standard edition
    - Available for roughly 2 thousand dollars
  - Student edition
    - Available for roughly 1 hundred dollars.
    - Some limitations
    - Shorter license period
- On campus
  - Site license

History of MATLAB

- Ancestral software to MATLAB
  - Fortran subroutines for solving linear (LINPACK) and eigenvalue (EISPACK) problems

History of MATLAB, con't: 2

- One of the developers of these packages, Cleve Moler wanted his students to be able to use LINPACK and EISPACK without requiring knowledge of Fortran
- MATLAB developed as an interactive system to access LINPACK and EISPACK

History of MATLAB, con't: 3

- MATLAB gained popularity primarily through word of mouth because it was not officially distributed
- In the 1980's, MATLAB was rewritten in C with more functionality (such as plotting routines)

History of MATLAB, con't: 4

- The Mathworks, Inc. was created in 1984
- The Mathworks is now responsible for development, sale, and support for MATLAB

Strengths of MATLAB

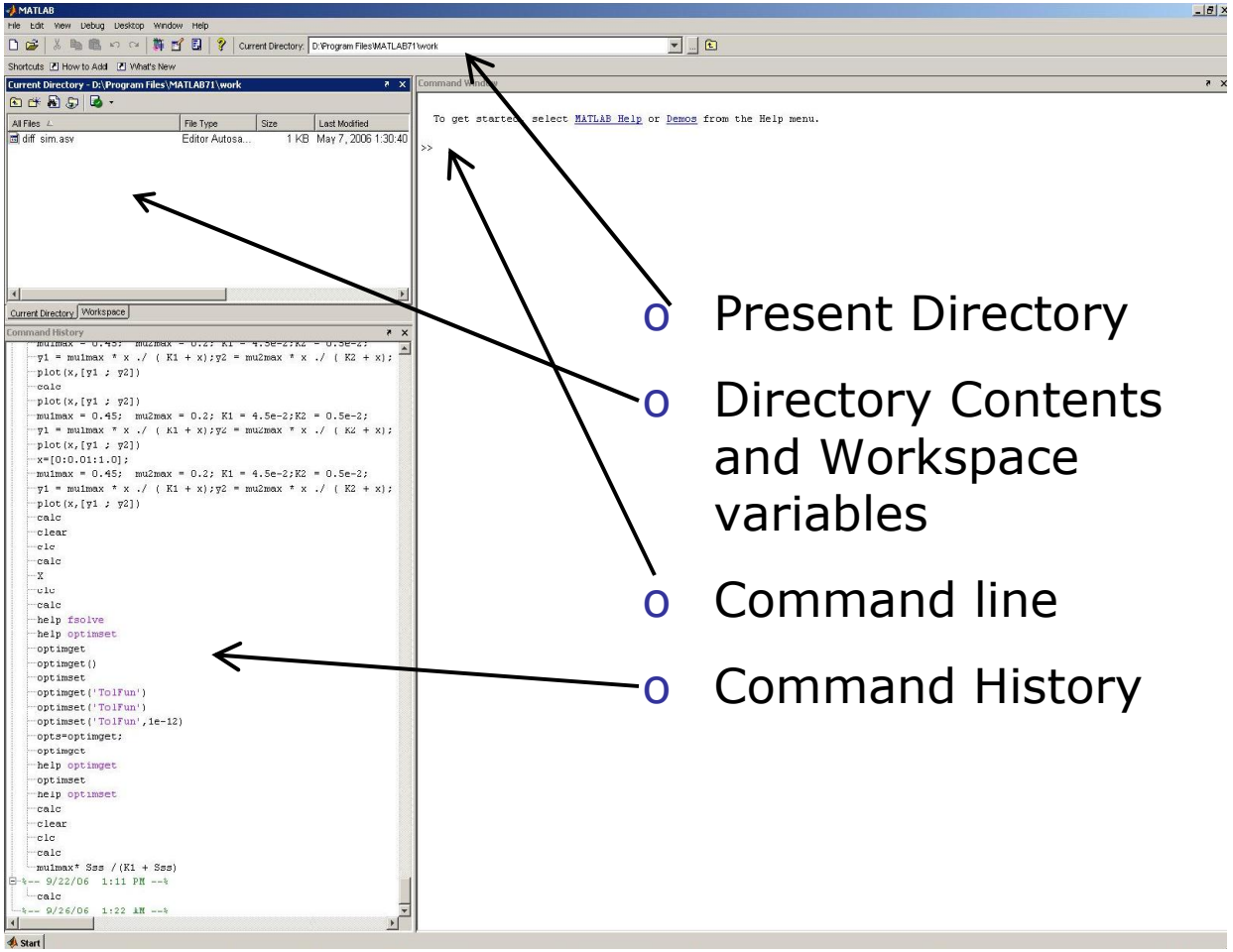
- MATLAB is relatively easy to learn
- MATLAB code is optimized to be relatively quick when performing matrix operations
- MATLAB may behave like a calculator or as a programming language
- MATLAB is interpreted, errors are easier to fix
- Although primarily procedural, MATLAB does have some object-oriented elements

Weaknesses of MATLAB

- MATLAB is NOT a general purpose programming language
- MATLAB is an interpreted language (making it for the most part slower than a compiled language such as C++)
- MATLAB is designed for scientific computation and is not suitable for some things (such as parsing text)

Matlab Windows

- Command line Interface (Main Window)
- Editor Window



Matrices in Matlab

- Entering a Matrix:

```
>> A = [ 0 -0.8 -0.6 ; 0.8 -0.36 0.48 ; 0.6 0.48 -0.64]
A =

         0   -0.8000   -0.6000
    0.8000   -0.3600    0.4800
    0.6000    0.4800   -0.6400
```

- Matrix referencing:

```
>> A(1,2)
ans =

   -0.8000

>> A(2,:)
ans =

    0.8000   -0.3600    0.4800

>> A(:,1)

ans =

         0
    0.8000
    0.6000
```

- Matrix Operations:

```
>> A+A;
>> A.*A;
>> 3*A;
>> A*A

            ans =

              -1.0000         0         0
                   0   -0.2800   -0.9600
                   0   -0.9600    0.2800
```

Built-in functions

- Determinant

```
>> det(A)
ans =

    -1.000
```

- Rank

```
>> rank(A)
ans =

     3
```

- Inverting a Matrix

```
>> inv(A)

ans =

   -0.0000    0.8000    0.6000
   -0.8000   -0.3600    0.4800
   -0.6000    0.4800   -0.6400
```

- Transpose of a Matrix

```
>> A'

ans =

         0    0.8000    0.6000
   -0.8000   -0.3600    0.4800
   -0.6000    0.4800   -0.6400
```

Solving Linear System

- Linear system of algebraic equations:

$$-x_1 + x_2 + 2x_3 = 2$$

$$3x_1 - x_2 + x_3 = 6$$

$$-x_1 + 3x_2 + 4x_3 = 4$$

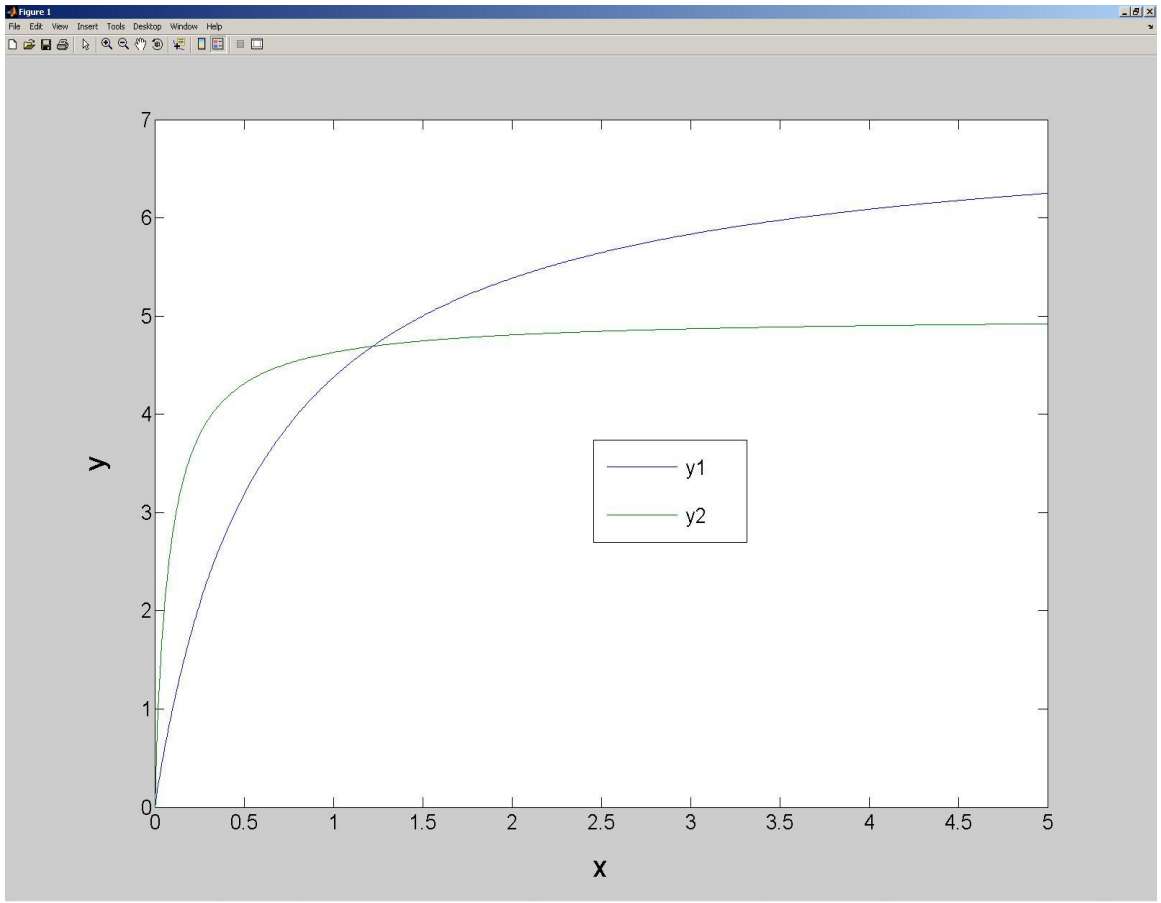
$$Ax = b$$

```
>> A = [ -1 1 2; 3 -1 1 ; -1 3 4]

A =

    -1     1     2
     3    -1     1
    -1     3     4

>> b = [2 6 4 ]'

b =

     2
     6
     4

>> rank(A)

ans =

     3

>> x = b\A
(also could do inv(A)*b, but not recommended)

x =

    1.0000
   -1.0000
    2.0000
```

Plotting a function

$$y_1 = \frac{7x}{0.6 + x} \qquad y_2 = \frac{5x}{0.08 + x}$$

```
>> x = [ 0:0.01:5];
>> y1 = 7 * x ./ ( 0.6 + x );
>> y2 = 5 * x ./ ( 0.08 + x );
>> plot(x,y1,x,y2)
>> legend('y1','y2')
```



Introduction to MATLAB

- Vectors, Matrices, Syntax
- Vector operations, including the \dot commands
  - length, size, linspace, logspace, size, rand, randn, randperm
- Special vectors and matrices: zeros, ones, eye, magic
- Scripts and functions
  - Diary
- Graphing:
  - plot, special fonts, plot3, semilogx, semilogy, title, xlabel, ylabel, axis, grid, legend, subplot,
- Formatted output:
  - Sprintf, ;, disp, input
- Programming:
  - for, if, while, &, |, ~
- General/misc commands
  - ginput set, size, max, sum, close, figure, hist, any, all , floor, fix, round,
- Graphical programming and callbacks