Course Business:

  - Contains syllabus, lecture notes, examples, homework
- Office Hours
  - Tuesday & Wednesday, 1-2 in 3134 Snee (or by appointment)
- Registration:
  - get my signature or CS Undergrad office (303 Upson)
  - S/U only, 1 credit
  - Last day to add/drop: Monday, September 9!

Outline

- Homework I.
- Getting started: the Desktop & Workspace
- Matlab as calculator
- Variables
- Arrays
- Array Operations
Homework I.

- Download from web page
- Use a text editor (e-mailer?) to insert answers
- Paste (DON’T ATTACH) into e-mail and send to me by 5PM Wed. next week.
- Please turn off HTML formatting on your e-mail program. It is much easier for me to manage if you send as text.

Starting Up

- On Windows:
  - Launch from START, or find matlab.exe & double click
- On UNIX/Linux/MacOS X
  - Open a terminal, type “matlab &”
  - Problems:
    - “Command not found”--check your path
    - Splash window of 6.X hangs--try “matlab -nojvm”

Windows, windows, and more windows

- As of 6.0, Matlab has lots of windows inside a “Desktop”
- The Workspace is the center of the Matlab universe
  - Holds your data
  - Waits for your commands
  - (other windows are fluff)
- 5.X only has command window
Basic Math

- Matlab is a command-line calculator
  - Simple arithmetic operators
    • + - * / ^
  - Basic functions
    • sin(), log(), log10(), exp(), rem()
  - Constants
    • pi

Big deal, a calculator’s $20

- Matlab is a fully-functional programming language
- This means we get variables
  - name = value
    • Name can be anything made of letters, numbers, and a few symbols (.,_). Must start with a letter
  - End line with ‘;’ to avoid output
    • Can also use ‘;’ to put multiple commands on a line
  - List with who
  - Delete with clear
  - More info with whos

1D Arrays--aka Vectors

- An array is anything you access with a subscript
- 1D arrays are also known as “vectors”
- Everything (nearly) in Matlab is a “double array”
- Create arrays with brackets [ ]
- Separate elements with commas or spaces
- Access with ()’s
Regular arrays

- We can create regularly spaced arrays using "::"
  - A=st:en produces [st, st+1, st+2, ... en]
  - A=1:5 is [1 2 3 4 5]
  - A=-3.5:2 is [-3.5 -2.5 -1.5 0 1.5]—note, stops before 2!
  - What happens if en < st?
  - Can also insert a "step" size: A=st:step:en
    - A=0:2:6 is [0 2 4 6]
    - A=5:-2.5:0 is [5 -2.5 0];

Accessing vectors

- Matlab arrays start at 1
- In most languages (C, Java, F77) can only access arrays one element at a time:
  - a(1)=1; a(2)=2.5; a(3)=-3; etc.
- In Matlab, can access several elements at a time using an array of integers (aka an index)
  - a(1:5) is [a(1),a(2),a(3),a(4),a(5)]
  - a(5:-2:1) is [a(5), a(3), a(1)]

Accessing vectors

- Index vectors can be variables:
  - A=10:10:100; I=[1:2:9]; A(I) gives
    [10,30,50,70,90]
  - J=[2:2:10]; A(J) gives [20,40,60,80,100];
  - What does A(I)=A(J) do?
**Column vectors**

- "row vectors" are 1-by-n
- "column vectors" are n-by-1
- Row/column distinction doesn’t exist in most languages, but VERY IMPORTANT in MATLAB
- Create column vectors with semi-colons
  - \( A=[1; 2; 3] \)
- Can force to column vector with (:) 
  - \( A=1:3 \) is \([1 2 3]\)
  - \( A(:) \) is \([3 1 2]\)

**Column vectors**

- Convert column-to-row and back with transpose (‘)
- Can access the same way as row vectors

**2D arrays--matrices**

- From using commas/spaces and semi-colons 
  - \( A=[1 2 3 ; 4 5 6 ; 7 8 9] \)
  - \( A(j,k)= j'th \) row, \( k'th \) column
  - \( A(2:3,1:2)= \) rows 2 through 3 and columns 1 through 2
  - \( A([1,3,4], :)= all \) of rows 1, 3 and 4
  - \( A(:, 1)= first \) column
Size matters

• "A is m-by-n" means A has m rows and n columns
• [m,n]=size(A) gets size of A
• length(a) gets length of vectors (max of m and n).
• A(1:3,2)=v, v better have length 3
• A(1:2:5,2:3)=B, B better be 3-by-2

Array Arithmetic

• C=A+B
  – if A and B are the same size, C(j,k)=A(j,k)+B(j,k)
  – If A is a scalar, C(j,k)=A+B(j,k)
• Same for -

Array Multiplication

• Multiplication is weird in Matlab
  – Inherited from linear algebra
  – To multiply by a scalar, use *
  – To get C(j,k)=A(j,k)*B(j,k) use ".*"
  • Also applies to ".^" and "/"