Announcements

- Prelim 1 conflicts...better news!
  - Prelim 1 review material: see Prelims on-line
  - Topics? Everything up to and including last lecture, A2, E3
- Exercise due date vote: Thu or Fri?
- Labs: software, reminders:
  - attendance by CMS submission
  - missing MATLAB? not for long!
- TA evals on-line
- What's the deal with the lame green pen? (someone ask ... if you dare)

Summary/Motivation

- Problems → Solutions
- Programming for automating
- English sentences → Prog. Lang statements
  - Data and Actions
  - Nothing, Assign, Express, I/O, Select, Repeat, Call, Break, ...
- Want more power: we've been focused on Action
  - So, focus on more variety in data!
  - Loops good for repeating action...
  - very useful if large amounts of data...
- Arrays!

Array Types in MATLAB

- Classifications:
  - Scalars
  - Arrays
  - Matrices
  - Logical Arrays
  - Strings
  - Structures
  - Cells
  - Vectorized Code
  - I/O
- Our coverage?
Applications

- Databases:
  - students
  - grades
  - CD collection
- Analysis:
  - finite elements
  - boundary elements
  - optimization

Array Basics

- Array: Rectilinear collection of data
  - rows, columns
  - elements have unique coordinates in form (r, c)
  - r, c indices are integers ≥ 1
- Examples)
  \[
  \begin{bmatrix}
  1 & 2 & 3 \\
  \end{bmatrix} \quad \% \quad 1-D!
  \]
  \[
  \begin{bmatrix}
  1 & 2 \\
  3 & 4 \\
  \end{bmatrix} \quad \% \quad 2-D!
  \]
  \[
  1:4
  \]
  \[
  \text{zeros}(2) \quad \% \quad \text{see ones, rand}
  \]
- “Everything” in MATLAB is an array!

Matrix Basics

- Matrix definition:
  - same structure and look as an array
  - usually means rectangular or square, but could have more dimensions
  - a matrix is an array that represents a linear transformation
- Notation for linear transformation: \( Ax = b \)
  - \( A \): coefficient matrix
  - \( x \): solution vector
  - \( b \): source vector

Matrices

- Example)
  \[
  x - y = 0 \quad \rightarrow \\
  x + y = 2 \quad \rightarrow \\
  \]
- Matrix \( A \) transforms vector \( x \) into \( b \)
  - To find \( x \), you need to solve the set of \textit{linearly independent simultaneous equations}
  - see Solving Systems of Equations in Mathematics Resources and help solve ... try this:
    \[
    A=[1 -1 ; 1 1]; \\
    b=[0 2]'; \\
    x=A\backslash b
    \]
- Matrix operations?
Matrix Operations

- add, sub: +, -
- dot product, matrix product: * (see also cross, dot)
- power: ^
- divide: \, / (help slash)
- help ops

Empty Arrays

- Empty array:
  - nothing inside, contains no values
  - handy way to initialize a variable!
  - eg) x = []
- Handy function: isempty
- Be careful:
  [] == []

Scalars

- Scalar:
  - single value
  - “0-dimensional” array: (r,c) = (1,1)
- Examples:
  1 [1] [1]
  - MATLAB collapses redundant [] s
- Operations:
  - see help ops
  - refer to unary operations

1-D Arrays

- 1-D array, sometimes called vector:
  - one row or one column
  - MATLAB stores information as rows, usually
- Notation:
  - to separate individual elements, use spaces or commas:
    [1 2 3] % row vector
    [1,2,3] % row vector
    1, 2, 3 % not a vector
  - to separate individual rows, use semicolons
    [1 ; 2 ; 3] % col vector
More 1-D Arrays

- Transpose:
  - handy way to convert from row to col vector
  - use '
- Example:
  \[ x = [1 2 3]' \]
- Math? \( A_{ij} = A_{ji}' \)

1-D Array Shortcuts

- colon:
  - \( \texttt{start:end} \)
  - \( \texttt{start:incr:end} \)
- \( \texttt{linspace} \):
  - \( \texttt{linspace(x1,x2)} \) generates a row vector of 100 equally spaced points between \( x_1 \) and \( x_2 \)
  - \( \texttt{linspace(x1,x2,N)} \) generates \( N \) points between \( x_1 \) and \( x_2 \)
- colon v.s. \( \texttt{linspace} \)?
  - colon if you know the increment
  - \( \texttt{linspace} \) if you don't know increment
- Operations?

2-D Arrays

- 2D arrays:
  - multiple rows and columns
  - real-life examples include tabular data and spreadsheets
  - always rectangular! no ragged arrays in MATLAB (unless you use something called a \textit{cell array})
- More than 2D? Yes – called \textit{multidimensional}
- Row major (MATLAB's default):
  - build 1 row at a time
  - extra \( [] \)s are condensed as with scalars
  - rectangularity must be preserved

Row-Major 2-D Arrays

- Construction:
  - rows: spaces/commas separate elements
  - cols: semicolons separate rows
  \[ [1 2; 3 4] \] % rows [1 2] and [3 4]
  \[ [1:3 ; 4:6] \] % short cut!
  \[ [1 2 ; 3 4 ; 5 6] \] % dimensions?
  \[ [1 2 ; 3 ] \] % legal?
Appending Arrays

- Think of arrays in \([a1, a2, ...]\) as *elements*
- Remove all but the outer brackets, as in
  \([[[a1,a2,...],[a3,a4,...]]] \rightarrow [a1,a2,...,a3,a4,...]\)
- Example:
  \[
  \text{A} = \begin{bmatrix}
  1 & 2 \\
  3 & 4
  \end{bmatrix};
  \text{[A,A]} \% \text{2 rows, 4 cols}
  \]

Column-Major 2-D Arrays

- 2 tricks:
  - Transpose:
    - swaps rows with columns with transpose operator ' 
    - means that \(A_{ij}\) becomes \(A_{ji}\) for all combinations of \(i\) and \(j\)
    \[
    \text{A} = \begin{bmatrix}
    1 & 2 \\
    3 & 4
    \end{bmatrix};
    \text{A'}
    \]
    \[
    \text{r1} = [1 \ 3]; \text{r2} = [2 \ 4];
    \text{[r1' r2']}
    \]
  - Single indexed 2-D array:
    \[
    \text{A} = \begin{bmatrix}
    1 & 2 \\
    3 & 4
    \end{bmatrix}; \text{A}(3)
    \]

Array Operations

- Operations performed *per* element!
  - add, sub: same as matrices
  - mul, div, power: use . in front of op
- Examples:
  \[
  \text{A} = \begin{bmatrix}
  1 & 2 \\
  3 & 4
  \end{bmatrix};
  \text{A} + \text{A}
  \text{A} * \text{A}
  \]

Indexing

- Index:
  - numerical location of an element in an array
  - MATLAB indices are integers \(\geq 1\)
- How many indices?
  - O-D: none, scalar
  - 1-D: one, row or col depending on type of array
  - 2-D: two, row and col in order (row,col)
- Math notation: \(A_{ijk}...\)
  - 1st index \((i)\) is 1st dimension (row)
  - 2nd index \((j)\) is 2nd dimension (col),
  - 3rd index \((k)\) is 3rd dimension (page)
MATLAB Indexing

- MATLAB syntax:
  - single element: \[ A(i,j) \]
  - subarray (range of elements): \[ A(i1:i2, j1:j2) \]
  - general subarray: \[ A([a1,a2,...],[b1,b2,...]) \]

- Basic examples:
  \[ x=[5 9 2]; x(1) \]
  \[ x=[1:3;4:6]; x(2,3) \]

- see `help paren` for more information

Subarrays

- MATLAB extracts elements in the order you write indices

- Examples:
  \[ x=[1:4]; x((1 3)) \]
  \[ A=[1 2 3; 4 5 6]; A((1 2),2) A(1, [3 2 1]) A([1 2],[1 3]) \]
  \[ A(:,1) \]
  \[ A(1,:) \]

Indexing Shortcuts

- Using the colon for subarrays
  - if you know that 4 is the end of 1:4, you could use \[ 1:end \]
  - when you see a colon as an index, it means for all of the...

- Examples:
  \[ A=[1 2 ; 3 4]; A(:,:,1) \]
  \[ A(:,1) \]
  \[ A(1,:) \]
More Examples

B = rand(5,4);
B([1 2 3],:)
B([3 2 1],:)
B(:,[4 1])
B([1 2 4],[3 1])

Inserting

- You can insert as well as extract elems in array
  \[ \textbf{array1(indices)} = \textbf{array2} \]
  \textbf{array2} is inserted into \textbf{array1} at \textbf{indices}
- Easy way to rem:
  \[ \textbf{var} = \textbf{expr} \]
  evaluate \textbf{expr} first
- Brief example:
  \[ \textbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \]
  \[ \textbf{A}(1,2) = 5 \]

Useful Functions

- \textbf{ones}, \textbf{zeros}, \textbf{rand}, \textbf{eye}, \textbf{diag}
- \textbf{size} of array:
  \[ [\text{rows cols}] = \text{size}([1:3; 4:6]) \]
- \textbf{length} of array (max of \textbf{size} results):
  \[ \text{length}(1:5) \]
  \[ \text{length}([1:3; 4:6]) \]

More Examples

\[ \textbf{A} = \begin{bmatrix} 1:3 & 4:6 \end{bmatrix}; \]
\[ \textbf{A}([1, 2],:) = \textbf{A}([2, 1],:) \]
% new row 1 of \textbf{A} gets old row 2 of \textbf{A}
% new row 1 of \textbf{A} gets old row 1 of \textbf{A}
% or, "swap rows of \textbf{A}"

\[ \textbf{A} = \begin{bmatrix} 1:3 & 4:6 \end{bmatrix}; \]
\[ \textbf{A}([2, 1],[1, 3]) = \textbf{A}([2, 2],[1, 2]) \]
% do RHS first!

\[ \textbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}; \]
\[ \textbf{A}(6,6) = 10 \]
% \text{MATLAB} fills in zeros as necessary
Functions and Arrays

- functions take arrays as input
  - some cases: applied to each element
    $$\sqrt{[1 \ 4; \ 9 \ 16]}$$
  - advice: look up help on function before using!
- functions return one or multiple arrays
  - single: $$\sqrt{1:10}$$
  - multiple: $$\text{size}([1, 2, 3])$$
- more on functions next lecture

Logical Arrays

- **logical array**:
  - array with logical values: eg, [1 0 1]
  - numerical array “considered as” logical values
- reminders:
  - values: 0, 1
  - logical operators: ~, |, ||, &, &&, xor
  - relational operators: <, <=, >, >=, ==, ~=
  - functions: coming up...

Making Logical Arrays

- scalar comparison:
  10 == [30 10 20 0]

- array comparison:
  5:-1:1 == 1:5

- shortcuts:
  true(2), false(1,4)

- conversion:
  logical([10 0 20 0])

Handy Logical Functions

- **all**: test to determine if all elements are nonzero
  all(1:5)

- **any**: test for any nonzero elements
  any(0 == floor(rand(4)*2)

- **false, true**: false, true array

- **find**: find indices and values of nonzero elements; note: find(a) regards a as a(:)
  find(10 == [10 20 10 30])

- see also:
  - search MATLAB Help on is*
  - search MATLAB Help for bit-wise operations
Logical Indexing

- **array(logicalindicies):** extract elements from positions that have true values
- example) find even integers from an array
  vals = 0:10;
  indicies = mod(vals,2) == 0
  vals(indicies)

- warning: use **logical**!
  x=[10 20 30];
  x([1 0 1])
  x(logical([1 0 1]))

Strings and Characters

- **MATLAB alphabet:** ASCII, UNICODE
- **Bits:** binary digits, 0 and 1
  q=quantizer([7 0]);
  num2bin(q,1)
  num2bin(q,3)
- Characters as values:
  - bit representations for all characters
  - use decimal representations instead of bits
    double('a')
    char(97)
- more info:
  - [http://www.asciitable.com/](http://www.asciitable.com/)

Character Operations

- Character/Number combinations:
  - characters **promote** to integers
  - operations:
    - 'a' + 1
    - 'a' - 'A' % == 'b'-'B'? why?
  - functions:
    - sqrt('a') % does this work?
  - Handy tricks:
    - shift a character:
      char('a' + 1) %
    - convert to lower/uppercase:
      char('b' - ('a'-'A')) %
      char('B' + ('a'-'A')) %

Strings

- **String:**
  - collection of characters
  - arrays in MATLAB
  - see help strings
- Many functions!
  - see help strfun
- 0-D, scalar:
  - what we have been calling characters
  - empty string: '' (2 single quotes, no space)
  - isempty('')
  - help char
1-D Strings

- **1-D**
  - characters in single line
  - most common model (text processing)

- **Forming:**
  - make 1-D array of chars:
    
    ```
    ['a' 'b' 'c']
    'abc'
    char(97:122)
    ```

  - make 2-D array of chars:
    
    ```
    x = ['abc'; 'def']
    ```

Some String Functions

- **concatenation:**
  - “glue” strings together
    
    ```
    s=strcat('abc','def')
    ```

  - beware of trailing blanks
    
    ```
    s=strcat('abc ','def ') s=='abc def '
    ```

- **comparison:**
  - by operators:
    
    ```
    'ab' == 'ab'
    ```

  - by function:
    
    ```
    strcmp('ab','ab')
    ```

Meta Programming

- **metaprogram:**
  - program that modifies or generates other programs
  - MATLAB?
    
    - commands are supplied as strings, so...
    
    - you can write program that writes and calls another program

- **handy function eval:**
  - `eval(stringarray)`
    
    - MATLAB concatenates the strings in `stringarray` and executes the command in that string
    
    ```
    eval('x = 1 + 1')
    eval(['x '='1+1'])
    ```

  - `eval( input('What do you want to run? ','s') )`

Vectorization

- Ideally, you should eliminate loops when possible
  - more efficient code
  - less overall code

- **examples**
  - reverse an array:
    
    ```
    a=1:10; _____________
    ```

  - find mean gradient between any pair
    
    ```
    h=[10,5,25,10];
    max(abs(h - _____________ ))
    ```

- Why don't we do this more in this class? (are we an intro to MATLAB?)