Today’s Lecture:
- Vectorized computation
- Introduction to graphics
- Matrix (2-d array)

Announcements:
- **Assignment 1a**: due Tues at 11:59pm, at which time submission on CMS will close. Will re-open for re-submission later.
- **Assignment 1b**: due Tues Sept 13 at 11:59pm
Initialize vectors/matrices if dimensions are known
...instead of “building” the array one component at a time

% Initialize y
x=linspace(a,b,n);
y=zeros(1,n);
for k=1:n
    y(k)=myF(x(k));
end

% Build y on the fly
x=linspace(a,b,n);
for k=1:n
    y(k)=myF(x(k));
end

Much faster for large n!
Monte Carlo Approximation of $\pi$

Throw $N$ darts

Sq. area = $N = L \times L$

Circle area = $N_{in}$

$= \pi L^2 / 4$

$\pi = 4 N_{in} / N$

1. Make a plot
2. Use vectors to store all values
3. Use vectorized arithmetic
Vectorized addition

\[
\begin{array}{c}
\text{x} & \begin{bmatrix}
2 & 1 & .5 & 8
\end{bmatrix} \\
+ \quad \text{y} & \begin{bmatrix}
1 & 2 & 0 & 1
\end{bmatrix} \\
= \quad \text{z} & \begin{bmatrix}
3 & 3 & .5 & 9
\end{bmatrix}
\end{array}
\]

Matlab code: \[ z = x + y \]
Vectorized subtraction

\[
\begin{array}{cccc}
\text{x} & 2 & 1 & .5 & 8 \\
\text{y} & 1 & 2 & 0 & 1 \\
\text{z} & 1 & -1 & .5 & 7 \\
\end{array}
\]

Matlab code: \( z = x - y \)
Vectorized code — a Matlab-specific feature

- Code that performs element-by-element arithmetic/relational/logical operations on array operands in one step

- Scalar operation: \( x + y \)
  where \( x, y \) are scalar variables

- Vectorized code: \( x + y \)
  where \( x \) and/or \( y \) are vectors. If \( x \) and \( y \) are both vectors, they must be of the same shape and length

See Sec 4.1 for list of vectorized arithmetic operations
Vectorized multiplication

\[
\begin{array}{c}
a \times b = c \\
\end{array}
\]

Matlab code:  \texttt{c} = \texttt{a} .* \texttt{b}
Vectorized 
element-by-element arithmetic operations 
on arrays

A dot (.) is necessary in front of these math operators
Shift

Matlab code: \[ z = x + y \]
Reciprocate

\[
\begin{align*}
x & \quad 1 \\
/ \\
y & \quad \begin{bmatrix} 2 & 1 & .5 & 8 \end{bmatrix} \\
\hline
\end{align*}
\]

\[
\begin{align*}
= \\
z & \quad \begin{bmatrix} .5 & 1 & 2 & .125 \end{bmatrix}
\end{align*}
\]

Matlab code: \( z= x \ ./ \ y \)
Vectorized

element-by-element arithmetic operations between an array and a scalar

A dot (.) is necessary in front of these math operators

The dot in .*, .* and ./ is not necessary but OK
Generating tables and plots

\[ \begin{array}{cc}
  x & \sin(x) \\
 0.000 & 0.000 \\
0.784 & 0.707 \\
1.571 & 1.000 \\
2.357 & 0.707 \\
3.142 & 0.000 \\
3.927 & -0.707 \\
4.712 & -1.000 \\
5.498 & -0.707 \\
6.283 & 0.000 \\
\end{array} \]

\( x, y \) are vectors. A vector is a 1-dimensional list of values

\( x = \text{linspace}(0, 2\pi, 9); \)
\( y = \sin(x); \)
\( \text{plot}(x, y) \)

Note: \( x, y \) are shown in columns due to space limitation; they should be rows.
Does this assign to $y$ the values $\sin(0^\circ)$, $\sin(1^\circ)$, $\sin(2^\circ)$, …, $\sin(90^\circ)$?

$x = \text{linspace}(0, \pi/2, 90);$

$y = \sin(x);$
Plot this!

\[ f(x) = \frac{\sin(5x) \exp(-x/2)}{1 + x^2} \quad \text{for} \quad -2 \leq x \leq 3 \]

```matlab
x = linspace(-2,3,200);
y = sin(5*x).*exp(-x/2)./(1 + x.^2);
plot(x,y)
```

Element-by-element arithmetic operations on arrays

See `plotComparison.m`
Element-by-element arithmetic operations on arrays…
Also called “vectorized code”

\[
\begin{align*}
x &= \text{linspace}(-2,3,200); \\
y &= \sin(5x) \times \exp(-x/2)/(1 + x^2);
\end{align*}
\]

Contrast with scalar operations that we’ve used previously…

\[
\begin{align*}
a &= 2.1; \\
b &= \sin(5a);
\end{align*}
\]

The operators are (mostly) the same; the operands may be scalars or vectors.

When an operand is a vector, you have “vectorized code.”
Some format commands to use with `plot`

```matlab
xlabel('text for labeling x-axis')
ylabel('text for labeling y-axis')
title('text for plot title at top center')
hold on  % hold subsequent plot commands to current axes
hold off % subsequent plot command refreshes axes--
          % erase previous items
close all % close all graphics windows
axis equal % same scaling for x, y axes
axis off  % hide axes
axis on   % show axes
```
Start with drawing a single line segment

\[ a = 0; \quad \% \text{x-coord of pt 1} \]
\[ b = 1; \quad \% \text{y-coord of pt 1} \]
\[ c = 5; \quad \% \text{x-coord of pt 2} \]
\[ d = 3; \quad \% \text{y-coord of pt 2} \]

\[ \text{plot}([a \ c], [b \ d], \ '-*') \]
Making an x-y plot

```matlab
a = [0 4 3 8];  % x-coords
b = [1 2 5 3];  % y-coords
plot(a, b, '-*')
```

- **x-values** (a vector)
- **y-values** (a vector)
- Line/marker format

![Graph](image-url)
Making an x-y plot with multiple graphs (lines)

\[
\begin{align*}
    a &= [0 \ 4 \ 5 \ 8]; \\
    b &= [1 \ 2 \ 5 \ 3]; \\
    f &= [0 \ 4 \ 6 \ 8 \ 10]; \\
    g &= [2 \ 2 \ 6 \ 4 \ 3]; \\
    \text{plot}(a,b, '-*', f,g, 'c') \\
    \text{legend('graph 1 name', 'graph 2 name')}
\end{align*}
\]

See also plotComparison.m
Drawing a polygon (multiple line segments)

% Draw a rectangle with the lower-left corner at (a,b), width w, height h.
x= [ ]; % x data
y= [ ]; % y data
plot(x, y)

Fill in the missing vector values!
Drawing a polygon (multiple line segments)

% Draw a rectangle with the lower-left
% corner at (a,b), width w, height h.
x = [a a+w a+w a a a]; % x data
y = [b b b+h b+h b b ]; % y data
plot(x, y)
2-d array: matrix

- An array is a named collection of like data organized into rows and columns
- A 2-d array is a table, called a matrix
- Two indices identify the position of a value in a matrix, e.g.,
  \[ \text{mat}(r,c) \]
  refers to component in row \( r \), column \( c \) of matrix \( \text{mat} \)
- Array index starts at 1
- Rectangular: all rows have the same #of columns
Creating a matrix

- Built-in functions: `ones`, `zeros`, `rand(1)`
  - E.g., `zeros(2,3)` gives a 2-by-3 matrix of 0s
- “Build” a matrix using square brackets, `[ ]`, but the dimension must match up:
  - `[x y]` puts `y` to the right of `x`
  - `[x; y]` puts `y` below `x`
  - `[4 0 3; 5 1 9]` creates the matrix
  - `[4 0 3; ones(1,3)]` gives
  - `[4 0 3; ones(3,1)]` doesn’t work
Function \texttt{size} returns the dimensions of a matrix

\begin{itemize}
  \item \texttt{[nr, nc]= size(M)} \hspace{1em} \% nr is \# of rows, \%
                   \hspace{1em} \% nc is \# of columns
  \item \texttt{nr= size(M, 1)} \hspace{1em} \% \# of rows
  \item \texttt{nc= size(M, 2)} \hspace{1em} \% \# of columns
\end{itemize}
% What will M be?
M = [ones(1,3); 1:4]

A
1 1 1 1 0
1 2 3 4

B
1 1 1
1 2 3

C
Error – M not created
What will $A$ be?

$$A = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} A' & \text{ones}(2,1) \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0; & A & A \end{bmatrix}$$
Example: minimum value in a matrix

function val = minInMatrix(M)

% val is the smallest value in matrix M
Pattern for traversing a matrix $M$

$$[nr, nc] = \text{size}(M)$$

for $r = 1:nr$

% At row $r$

for $c = 1:nc$

% At column $c$ (in row $r$)

% Do something with $M(r,c)$ …

end

end
Matrix example: Random Web

- N web pages can be represented by an N-by-N Link Array $A$.
- $A(i,j)$ is 1 if there is a link on webpage $j$ to webpage $i$.
- Generate a random link array and display the connectivity:
  - There is no link from a page to itself.
  - If $i \neq j$ then $A(i,j) = 1$ with probability $\frac{1}{1+|i-j|}$.
  - There is more likely to be a link if $i$ is close to $j$. 

function A = RandomLinks(n)
% A is n-by-n matrix of 1s and 0s
% representing n webpages

A = zeros(n,n);
for i=1:n
    for j=1:n
        r = rand(1);
        if i~=j && r<= 1/(1 + abs(i-j))
            A(i,j) = 1;
        end
    end
end
Random web

$N = 20$

```
01110000010010000000
10001000111000000100
01010000000000000000
00101000000000000000
00010000001100000000
0000000000000010100000
01111100010110000000
0000001000010000000011
01000000010010001000
0000000110100000000001
0000001000001100000000
0000001001000000000001
0010000110101011000000
0000001000000001100000
0000001000000001100000
00000010000000010010001
00000010000000010001010
01000000100010101010
00000000000000011001
0000001000000000000000
00000000000000001010
```
Represent the web pages graphically…

100 Web pages arranged in a circle. Next display the links…. 
Represent the web pages graphically…

Bidirectional links are blue. Unidirectional link is black as it leaves page j, red when it arrives at page i.
for i = 1:n
    for j = 1:n
        if A(i,j) == 1 && A(j,i) == 1
            % Blue
        elseif A(i,j) == 1
            % Black-Red
            j → mid mid → i
    end
end

Somewhat inefficient: each blue line gets drawn twice.
See ShowRandomLinks.m
Transposes—like switching row and column indices