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 Nov 1 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 = $N_{in}$

$= \pi L^2 / 4$

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

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

Matlab code: \( z = x + y \)
Vectorized subtraction

\[
\begin{align*}
\mathbf{x} & = \begin{bmatrix} 2 & 1 & .5 & 8 \end{bmatrix} \\
\mathbf{y} & = \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix} \\
\mathbf{z} & = \mathbf{x} - \mathbf{y} = \begin{bmatrix} 1 & -1 & .5 & 7 \end{bmatrix}
\end{align*}
\]

Matlab code: \( \mathbf{z} = \mathbf{x} - \mathbf{y} \)
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
Vectorized multiplication

Matlab code: \( c = a \times b \)
Vectorized element-by-element arithmetic operations on arrays

$\begin{align*}
\text{Vectorized} \\
element-by-element \text{ arithmetic operations} \\
on \text{arrays}
\end{align*}$

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

\[
\begin{array}{c}
\text{x} & 3 \\
+ & \begin{bmatrix} 2 & 1 & .5 & 8 \end{bmatrix} \\
\hline
\text{z} & \begin{bmatrix} 5 & 4 & 3.5 & 11 \end{bmatrix}
\end{array}
\]

Matlab code: \( \text{z} = \text{x} + \text{y} \)
Reciprocate

Matlab code: \( z = \frac{x}{y} \)
Vectorized

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

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

\text{\#.} + \text{\#.}
\text{\#.} - \text{\#.}
\text{\#.} \ast \text{\#.}
\text{\#.} / \text{\#.}

Not necessary but OK to use \text{\#.} for these:
\text{\#.} \ast \text{\#.}
\text{\#.} \ast \text{\#.}
\text{\#.} / \text{\#.}
Generating tables and plots

<table>
<thead>
<tr>
<th>x</th>
<th>\sin(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.784</td>
<td>0.707</td>
</tr>
<tr>
<td>1.571</td>
<td>1.000</td>
</tr>
<tr>
<td>2.357</td>
<td>0.707</td>
</tr>
<tr>
<td>3.142</td>
<td>0.000</td>
</tr>
<tr>
<td>3.927</td>
<td>-0.707</td>
</tr>
<tr>
<td>4.712</td>
<td>-1.000</td>
</tr>
<tr>
<td>5.498</td>
<td>-0.707</td>
</tr>
<tr>
<td>6.283</td>
<td>0.000</td>
</tr>
</tbody>
</table>

\(x, y\) 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 \textit{columns} due to space limitation; they should be \textit{rows}. 

![Graph of \sin(x) with data points](image)
Does this assign to \( y \) the values \( \sin(0^\circ), \sin(1^\circ), \sin(2^\circ), \ldots, \sin(90^\circ) \)?

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

\[
y = \sin(x);
\]

A: yes  B: no
Plot this!

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

for 

\[-2 \leq x \leq 3\]

\[
x = \text{linspace}(-2,3,200);
y = \sin(5*x).*\exp(-x/2)./(1 + x.^2);
\text{plot}(x,y)
\]

Element-by-element arithmetic operations on arrays

See \text{plotComparison.m}
Element-by-element arithmetic operations on arrays… Also called “vectorized code”

\[
x = \text{linspace}(-2, 3, 200);
y = \sin(5 \times x) \times \exp(-x/2) \div (1 + x \times x);
\]

Contrast with scalar operations that we’ve used previously…

\[
a = 2.1;
b = \sin(5 \times a);
\]

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
```

17
Start with drawing a single line segment

\[
\begin{align*}
  a &= 0; \quad \% \text{ x-coord of pt 1} \\
  b &= 1; \quad \% \text{ y-coord of pt 1} \\
  c &= 5; \quad \% \text{ x-coord of pt 2} \\
  d &= 3; \quad \% \text{ y-coord of pt 2} \\
  \text{plot}([a \ c], [b \ d], \text{‘-*’})
\end{align*}
\]
Making an x-y plot

\[ a = \begin{bmatrix} 0 & 4 & 3 & 8 \end{bmatrix}; \quad \% \quad \text{x-coords} \]
\[ b = \begin{bmatrix} 1 & 2 & 5 & 3 \end{bmatrix}; \quad \% \quad \text{y-coords} \]
\[ \text{plot}(a, b, \quad \text{'-*'}); \]

x-values (a vector)  
y-values (a vector)  

Line/marker format

![Graph showing an x-y plot with points marked by stars and connected by line segments.](image-url)
Making an x-y plot with multiple graphs (lines)

\[ a = [0 \ 4 \ 5 \ 8] ; \]
\[ b = [1 \ 2 \ 5 \ 3] ; \]
\[ f = [0 \ 4 \ 6 \ 8 \ 10] ; \]
\[ g = [2 \ 2 \ 6 \ 4 \ 3] ; \]
\[ \text{plot}(a,b,\'-*\',f,g,\'c\') \]
\[ \text{legend(}\text{\'}graph 1 name\text{\'); }} \]
\[ \text{legend(}\text{\'}graph 2 name\text{\'); }} \]
\[ \text{xlabel(}\text{\'}x values\text{\'); }} \]
\[ \text{ylabel(}\text{\'}y values\text{\'); }} \]
\[ \text{title(}\text{\'}My graphs\text{\', \'}Fontsize\text{\',}14) \]

See also plotComparison.m
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 **size** returns the dimensions of a matrix

- `[nr, nc] = size(M)` % `nr` is # of rows,
  % `nc` is # of columns

- `nr = size(M, 1)` % # of rows
- `nc = size(M, 2)` % # of columns
Example: minimum value in a matrix

function val = minInMatrix(M)
% val is the smallest value in matrix M
Pattern for traversing a matrix $M$

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

\begin{verbatim}
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
\end{verbatim}
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