Today’s topics
- Loops
- Conditionals
- More on user-defined function
- 1-d array

Announcement/Reminder:
- Assignment 1a is due Oct 27\textsuperscript{th} at 11:59pm
- My email ygao@cs.cornell.edu
Monte Carlo Approximation of π

Throw $N$ darts

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

Circle area = $N_{in} = \pi L^2 / 4$
Monte Carlo Approximation of $\pi$

Throw $N$ darts

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

Circle area = $N_{in} = \pi L^2 / 4$

$\pi = 4 \frac{N_{in}}{N}$
Monte Carlo Approximation of $\pi$

For each of $N$ trials

Throw a dart

If it lands in circle

add 1 to total # of hits

$\pi$ is $4 \times \text{hits}/N$
Monte Carlo $\pi$ with N darts on L-by-L board

\begin{verbatim}
N= ___;
for k = 1:N
  
end

myPi = 4*hits/N;
\end{verbatim}
Monte Carlo π with N darts on L-by-L board

N = ___;
for k = 1:N
    % Throw kth dart

    % Count it if it is in the circle

end
myPi = 4*hits/N;
Monte Carlo $\pi$ with N darts on L-by-L board

N= ___;  L= ___;

for k = 1:N
    % Throw kth dart
    x = rand(1)*L - L/2;
    y = rand(1)*L - L/2;
    % Count it if it is in the circle
end

myPi = 4*hits/N;
Monte Carlo $\pi$ with $N$ darts on $L$-by-$L$ board

N= ___;  L= ___;  hits= 0;
for k = 1:N
    % Throw kth dart
    x = rand(1)*L - L/2;
    y = rand(1)*L - L/2;
    % Count it if it is in the circle
    if sqrt(x^2+y^2) <= L/2
        hits = hits + 1;
    end
end
myPi = 4*hits/N;
Syntax of the `for` loop

```
for <var> = <start value>:<incr>:<end bound>
    statements to be executed repeatedly
end
```

Loop body
Loop header specifies all the values that the index variable will take on, one for each pass of the loop.

E.g, \( \mathbf{k} = 3 : 1 : 7 \) means \( \mathbf{k} \) will take on the values 3, 4, 5, 6, 7, one at a time.
for loop examples

for k= 2:0.5:3
   k takes on the values 2,2.5,3
   disp(k)
end

Non-integer increment is OK

for k= 1:4
   k takes on the values 1,2,3,4
   disp(k)
end

Default increment is 1

for k= 0:-2:-6
   k takes on the values 0,-2,-4,-6
   disp(k)
end

“Increment” may be negative

for k= 0:-2:-7
   k takes on the values 0,-2,-4,-6
   disp(k)
end

Colon expression specifies a bound

for k= 5:2:1
   disp(k)
end
for loop examples

for k= 2:0.5:3
    disp(k)
end

k takes on the values __________
Non-integer increment is OK

for k= 1:4
    disp(k)
end

k takes on the values __________
Default increment is 1

for k= 0:-2:-6
    disp(k)
end

"Increment" may be negative

for k= 0:-2:-7
    disp(k)
end

k takes on the values __________
Colon expression specifies a bound

for k= 5:2:1
    disp(k)
end
## for loop examples

<table>
<thead>
<tr>
<th>Loop Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>for k= 2:0.5:3</code></td>
<td><strong>k</strong> takes on the values 2, 2.5, 3. Non-integer increment is OK.</td>
</tr>
<tr>
<td><code>for k= 1:4</code></td>
<td><strong>k</strong> takes on the values 1, 2, 3, 4. Default increment is 1.</td>
</tr>
<tr>
<td><code>for k= 0:-2:-6</code></td>
<td><strong>k</strong> takes on the values 0, -2, -4, -6. &quot;Increment&quot; may be negative.</td>
</tr>
<tr>
<td><code>for k= 0:-2:-7</code></td>
<td><strong>k</strong> takes on the values 0, -2, -4, -6. Colon expression specifies a <em>bound</em>.</td>
</tr>
<tr>
<td><code>for k= 5:2:1</code></td>
<td>The set of values for <strong>k</strong> is the empty set: the loop body won’t execute.</td>
</tr>
</tbody>
</table>
The **if** construct

```plaintext
if boolean expression1
    statements to execute if expression1 is true
elseif boolean expression2
    statements to execute if expression1 is false but expression2 is true
:
else
    statements to execute if all previous conditions are false
end
```

Can have any number of elseif branches but at most one else branch
Monte Carlo $\pi$ with N darts on L-by-L board

N= ___;  L= ___;  hits= 0;

for k = 1:N
    % Throw kth dart
    x = rand(1)*L - L/2;
    y = rand(1)*L - L/2;
    % Count it if it is in the circle
    if sqrt(x^2+y^2) <= L/2
        hits = hits + 1;
    end
end

myPi = 4*hits/N;
Using a `while`-loop

N = ___;  L = ___;  hits = 0;  k = 1;
while k <= N
    % Throw kth dart
    x = rand(1)*L - L/2;
    y = rand(1)*L - L/2;
    % Count it if it is in the circle
    if sqrt(x^2 + y^2) <= L/2
        hits = hits + 1;
    end
    k = k+1;
end
myPi = 4*hits/N;
Common loop patterns

Do something $n$ times

```
for k = 1:1:n
    % Do something
end
```

Do something an indefinite number of times

```
% Initialize loop variables

while ( not stopping signal )
    % Do something
    % Update loop variables
end
```
Patterns to do something n times

for k = 1:1:n
    % Do something
end

% Initialize loop variables
k = 1;
while ( k <= n )
    % Do something
    % Update loop variables
    k = k + 1;
end
General form of a user-defined function

\[
\text{function } [out 1, out 2, \ldots] = \text{functionName} (in 1, in 2, \ldots) \\
\%	ext{ 1-line comment to describe the function} \\
\%	ext{ Additional description of function}
\]

Executable code that at some point assigns values to output parameters \( out 1, out 2, \ldots \)

- \( in 1, in 2, \ldots \) are defined when the function begins execution. Variables \( in 1, in 2, \ldots \) are called function \textit{parameters} and they hold the function \textit{arguments} used when the function is invoked (called).
- \( out 1, out 2, \ldots \) are not defined until the executable code in the function assigns values to them.
function myPi = mcPiFun(N)

% myPi is Monte Carlo estimate of pi by
% throwing N darts

N= ___;  L= ___;  hits= 0;
for k = 1:N
    % Throw kth dart
    x = rand(1)*L – L/2;
    y = rand(1)*L – L/2;
    % Count it if it is in the circle
    if sqrt(x^2+y^2) <= L/2
        hits = hits + 1;
    end
end
myPi = 4*hits/N;
function [x, y] = polar2xy(r,theta)
% Convert polar coordinates (r,theta) to
% Cartesian coordinates (x,y).
% theta is in degrees.

rads= theta*pi/180;  % radian
x= r*cos(rads);
y= r*sin(rads);
Function header is the “contract” for how the function will be used (called)

You have this function:

```matlab
function [x, y] = polar2xy(r, theta)
% Convert polar coordinates (r, theta) to
% Cartesian coordinates (x,y). Theta in degrees.
...
```

Code to call the above function:

```matlab
% Convert polar (r1,t1) to Cartesian (x1,y1)
r1 = 1;  t1 = 30;
[x1, y1] = polar2xy(r1, t1);
plot(x1, y1, 'b*')
...
```
function m = convertLength(ft,in)
% Convert length from feet (ft) and inches (in)
% to meters (m).
.

How many proper calls to convertLength are shown below?

% Given f and n
d= convertLength(f,n);
d= convertLength(f*12+n);
d= convertLength(f+n/12);
x= min(convertLength(f,n), 1);
y= convertLength(pi*(f+n/12)^2);

A: 1  B: 2  C: 3  D: 4  E: 5 or 0
Comments in functions

- Block of comments after the function header is printed whenever a user types `help <functionName>` at the Command Window.
- 1st line of this comment block is searched whenever a user types `lookfor <someWord>` at the Command Window.
- Every function should have a comment block after the function header that says what the function does concisely.
Accessing a function

- A function is accessible if it is in the **current directory** or if it is on the **search path**
- Easy: put all related m-files in the same directory
- Better: the **path** function gives greater flexibility
Subfunction

- There can be more than one function in an M-file
- **top** function is the main function and has the name of the file
- remaining functions are **subfunctions**, accessible only by the functions in the same m-file
- Each (sub)function in the file begins with a **function header**
- Keyword **end** is not necessary at the end of a (sub)function
Arrays

The basic variable in Matlab is a matrix:

- Scalar $\rightarrow$ 1 x 1 matrix

- 1-d array of length 4 $\rightarrow$
  
  1 x 4 matrix  or  4 x 1 matrix

- 2-d array $\rightarrow$ a matrix, naturally
Array index starts at 1

Let $k$ be the index of vector $x$, then

- $k$ must be a positive integer
- $1 \leq k \leq \text{length}(x)$
- To access the $k^{th}$ element: $x(k)$

<table>
<thead>
<tr>
<th>$x$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>.4</td>
<td>.91</td>
<td>-4</td>
<td>-1</td>
<td>7</td>
</tr>
</tbody>
</table>
Here are a few different ways to create a vector

\text{count} = \text{zeros}(1, 6)

\begin{align*}
\text{Similar functions: ones, rand} \\
\text{a} &= \text{linspace}(10, 30, 5) \\
\text{b} &= 7:-2:0 \\
\text{c} &= [3 \ 7 \ 2 \ 1] \\
\text{d} &= [3; 7; 2]
\end{align*}
Possible outcomes from rolling a fair 6-sided die

1 2 3 4 5 6
Keep tally on repeated rolls of a fair die

Repeat the following:

% roll the die

% increment correct “bin”
function count = rollDie(rolls)

FACES = 6; % #faces on die
count = zeros(1,FACES); % bins to store counts

% Count outcomes of rolling a FAIR die
for k = 1:rolls
    % Roll the die
    % Increment the appropriate bin
end

% Show histogram of outcome
function count = rollDie(rolls)

FACES= 6; % #faces on die
count= zeros(1,FACES); % bins to store counts

% Count outcomes of rolling a FAIR die
for k = 1:rolls
    % Roll the die
    face= ceil(rand*FACES);
    % Increment the appropriate bin

end

% Show histogram of outcome
function count = rollDie(rolls)

FACES = 6; % #faces on die
count = zeros(1,FACES); % bins to store counts

% Count outcomes of rolling a FAIR die
for k = 1:rolls
  % Roll the die
  face = ceil(rand*FACES);
  % Increment the appropriate bin
  count(face) = count(face) + 1;
end

% Show histogram of outcome
.Initialize vectors/matrices if dimensions are known

...instead of “building” the array one component at a time

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

```matlab
% 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!