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

Announcement/Reminder:
- Assignment 1a due Tuesday at 11:59pm
- Be sure to do pre-class reading (viewing) before class

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

$N = \_\_\_;$
for $k = 1 : N$
  \text{\% Throw $k$th dart}
  x = \text{rand} \times L - L/2;
  y = \text{rand} \times L - L/2;
  \text{\% Count it if it is in the circle}
  \text{if} \sqrt{x^2+y^2} \leq L/2
    \text{hits} = \text{hits} + 1;
  \end
end
myPi = 4 \times \text{hits} / N;

Syntax of the for loop

\begin{verbatim}
for <var>=<start value>:<incr>:<end bound>
  \text{statements to be executed repeatedly}
end
\end{verbatim}

Loop header specifies all the values that the index variable will take on, one for each pass of the loop.
Eg. $k = 3:1:7$ means $k$ will take on the values 3, 4, 5, 6, 7, one at a time.
for loop examples

for k= 2:0.5:3
    disp(k)
end
k takes on the values 2,2.5,3
Non-integer increment is OK

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

for k= 0:-2:-6
    disp(k)
end
k takes on the values 0,-2,-4,-6
"Increment" may be negative

for k= 0:-2:-7
    disp(k)
end
Colon expression specifies a bound

for k= 5:2:1
    disp(k)
end
The set of values for k is the empty set: the loop body won’t execute

The if construct

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

Arrays

The basic variable in Matlab is a matrix:

- Scalar → 1 x 1 matrix
- 1-d array of length 4 → 1 x 4 matrix or 4 x 1 matrix
- 2-d array → a matrix, naturally

Array index starts at 1

Let k be the index of vector x, then

- k must be a positive integer
- 1 ≤ k ≤ length(x)
- To access the kth element: x(k)

Here are a few different ways to create a vector

count= zeros(1,6)
count = [0; 0; 0; 0; 0; 0]
Similar functions: ones, rand

a= linspace(10,30,5)
a = [10 15 20 25 30]
b= 7:-2:0
b = [7 5 3 1]
c= [3 7 2 1]
c = [3 7 2 1]
d= [3; 7; 2]
d = [3; 7; 2]

Example

- Write a program fragment that calculates the cumulative sums of a given vector v.
- The cumulative sums should be stored in a vector of the same length as v.

1, 3, 5, 0 v
1, 4, 9, 9 cumulative sums of v
General form of a user-defined function

```
function [out1, out2, ...] = functionName(in1, in2, ...)
% 1-line comment to describe the function
% Additional description of function

Executable code that at some point assigns
values to output parameters out1, out2, ...
```

- `in1, in2, ...` are defined when the function begins execution. Variables `in1, in2, ...` are called function parameters and they hold the function arguments used when the function is invoked (called).
- `out1, out2, ...` are not defined until the executable code in the function assigns values to them.

Function header is the "contract" for how the function will be used (called)

You have this function:
```
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:
```
% Convert polar (r1,t1) to Cartesian (x1,y1)
[r1,t1] = [1; 30];
[x1,y1] = polar2xy(r1,t1);
plot(x1,y1,'b*')
```

Given this function:
```
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 function file `polar2xy.m`
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 succinctly.

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.

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:

```matlab
% roll the die
% increment correct "bin"
```

```
function count = rollDie(rolls)
    FACE= 6;               % #faces on die
    count= zeros(1,FACE);  % bins to store counts
    % Count outcomes of rolling a FAIR die
    for k= 1:rolls
        % Roll the die
        face= ceil(rand*FACE);
        % Increment the appropriate bin
        count(face)= count(face) + 1;
    end
    % Show histogram of outcome
end
```

```matlab
function count = rollDie(rolls)
    FACE= 6;               % #faces on die
    count= zeros(1,FACE);  % bins to store counts
    % Count outcomes of rolling a FAIR die
    for k= 1:rolls
        % Roll the die
        face= ceil(rand*FACE);
        % Increment the appropriate bin
        count(face)= count(face) + 1;
    end
    % Show histogram of outcome
end
```