Previous Lecture:
- Functions and expressions
- 1-d array—vector

Today, Lecture 11:
- Probability and random numbers
- Examples of vectors and simulation
- Loop patterns for processing a vector (watch video)

Announcements:
- Exercise 6 (Matlab Grader) due Mon, March 2
- Project 3 due Wed, March 4, at 11pm
- Social lunch Friday 12:20pm Okenshields (sign up on website)
function [xNew,yNew] = Centralize(x,y)

% Translate polygon defined by vectors
% x,y such that the centroid is on the origin. New polygon defined by vectors % xNew,yNew.

n= length(x);
xNew= zeros(n,1); yNew= zeros(n,1);
xBar= sum(x)/n;  yBar= sum(y)/n;

for k = 1:n
    xNew(k)= x(k) - xBar;
    yNew(k)= y(k) - yBar;
end
Read *Insight 6.3* for the rest of the story.
For-loop pattern for working with a vector

% Given a vector v

for k = 1:length(v)
    % Work with v(k)
    % E.g., disp(v(k))
end

% Count odd values in vector v
count = 0;
for k = 1:length(v)
    if rem(v(k),2) == 1
        count = count + 1;
    end
end

% Pair sums of vector v
s = zeros(1,length(v) - 1)
for k = 1:length(v) - 1
    s(k) = v(k) + v(k + 1);
end

v = [5, .4, .9, -4]
s = [5.4, 1.3, -3.1]
Simulation

- Imitates real system
- Requires judicious use of random numbers
- Requires many trials, or multiple points in time
  - → opportunity to practice working with vectors!
Example: rolling dice

Outcomes from 1200 rolls of a fair die

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>180</td>
</tr>
<tr>
<td>3</td>
<td>220</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
</tr>
<tr>
<td>5</td>
<td>150</td>
</tr>
<tr>
<td>6</td>
<td>180</td>
</tr>
</tbody>
</table>

Roll 2 fair dice
Random numbers

- **Pseudorandom** numbers in programming
  - Sequence is reproducible if seeded (e.g., `rng(42)` at start of script)
- Function `rand()` generates random real numbers in the interval $(0,1)$. All numbers in the interval $(0,1)$ are equally likely to occur—**uniform** probability distribution.
- Examples:
  - `rand()` one random # in $(0,1)$
  - `6*rand()` one random # in $(0,6)$
  - `6*rand()+1` one random # in $(1,7)$
Uniform probability distribution in (0,1)  
rand()  

“Normal” distribution with zero mean and unit standard deviation  
randn()  

Distribution of randn(1000000,1)
Step 1: Simulate a fair 6-sided die

Which expression(s) below will give a random integer in [1..6] with equal likelihood?

A. \text{round}(\text{rand}() \times 6)
B. \text{ceil}(\text{rand}() \times 6)
C. \text{Both expressions above}
\texttt{round(rand() \times 6)}

\texttt{ceil(rand() \times 6)}
Step 2: Keep track of results

Possible outcomes from rolling a fair 6-sided die
Simulation
Simulation result

1  2  3  4  5  6
51  60  59  55  59  54
Simulation result

<table>
<thead>
<tr>
<th>Bin numbers</th>
<th>Data in bins</th>
<th>bar(1:6, counts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>54</td>
<td></td>
</tr>
</tbody>
</table>

counts

<table>
<thead>
<tr>
<th>counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>59</td>
</tr>
<tr>
<td>55</td>
</tr>
<tr>
<td>59</td>
</tr>
<tr>
<td>54</td>
</tr>
</tbody>
</table>

1 2 3 4 5 6
0 10 20 30 40 50 60
Keep tally on repeated rolls of a fair die

Repeat the following:

% roll the die

% increment correct "bin"
function counts = rollDie(rolls)

FACES = 6; % #faces on die
counts = zeros(1,FACES);

% 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
% Count outcomes of rolling a FAIR die

counts = zeros(1, 6);
for k = 1:100
    face = ceil(rand() * 6);
    if face == 1
        counts(1) = counts(1) + 1;
    elseif face == 2
        counts(2) = counts(2) + 1;
    end
    : 
    elseif face == 5
        counts(5) = counts(5) + 1;
    else
        counts(6) = counts(6) + 1;
    end
end
function counts = rollDie(rolls)

FACES = 6; % #faces on die
counts = zeros(1,FACES);

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

% Show histogram of outcome
% Simulate the rolling of 2 fair dice

totalOutcome = ???

A  ceil(rand() * 12)
B  ceil(rand() * 11) + 1
C  floor(rand() * 11) + 2
D  2 of the above
E  None of the above

Ceil(rand() * 6) + Ceil(rand() * 6)  \{ Two rand() calls summed \}  \Rightarrow \text{Distribution not obvious}  \Rightarrow \text{Sim!}
2-dimensional random walk

Start in the middle tile, (0,0).

For each step, randomly choose between N,E,S,W and then walk one tile. Each tile is $1 \times 1$.

Walk until you reach the boundary.
function [x, y] = RandomWalk2D(N)
% 2D random walk in 2N-1 by 2N-1 grid.
% Walk randomly from (0,0) to an edge.
% Vectors x, y represent the path.

By the end of the function ...

x
y

N = 3
function [x, y] = RandomWalk2D(N)

k=0;  xc=0;  yc=0;

while current position not past an edge
  % Choose random dir, update xc,yc

  % Record new location in x, y

end
function [x, y] = RandomWalk2D(N)

k=0;  xc=0;  yc=0;

while abs(xc)<N && abs(yc)<N
    % Choose random dir, update xc,yc
    % Record new location in x, y
    k=k+1;  x(k)=xc;  y(k)=yc;
end
Making a random choice

- Likelihood of `rand()` being between two numbers is proportional to their difference – *width*
Standing at (xc, yc)
% Randomly select a step

\[
\begin{align*}
& r = \text{rand}(); \\
& \text{if } r < 0.25 \\
& \quad \text{yc} = \text{yc} + 1; \quad \% \text{ north} \\
& \text{elseif } r < 0.5 \\
& \quad \text{xc} = \text{xc} + 1; \quad \% \text{ east} \\
& \text{elseif } r < 0.75 \\
& \quad \text{yc} = \text{yc} - 1; \quad \% \text{ south} \\
& \text{else} \\
& \quad \text{xc} = \text{xc} - 1; \quad \% \text{ west} \\
& \end{align*}
\]

See RandomWalk2D.m
Custom likelihoods

- Suppose you want outcomes with the following likelihoods: **17%, 26%, 12%, 55%**
  What should the thresholds be? Do these even add up to 100%?
- Trick: keep a running sum of the widths
Exercise

- 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 $v$.

$1, 3, 5, 0 \quad v$
$1, 4, 9, 9$ cumulative sums of $v$
\[ C_{\text{sum}}(1) = V(1); \]
\[ \text{for } k = 2 : \text{length}(V) \]
\[ C_{\text{sum}}(k) = C_{\text{sum}}(k-1) + V(k); \]
\[ \text{end} \]

\[ C_{\text{sum}}(k) = C_{\text{sum}}(k-1) + V(k) \]

\[ C_{\text{sum}}(3) = V(1) + V(2) + V(3) \]
\[ C_{\text{sum}}(4) = V(1) + V(2) + V(3) + V(4) \]
Demo: Random walk with biased probabilities
Loop patterns for processing a vector

% Given a vector v
for k=1:length(v)
    % Work with v(k)
    % E.g., disp(v(k))
end

% Given a vector v
k = 1;
while k<=length(v)
    % and possibly other
    % continuation conditions
    % Work with v(k)
    % E.g., disp(v(k))
    k = k+1;
end