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

Announcements:
- Exercise 5 (Matlab Grader) due Sun, March 21
- Project 3 due Wed, March 24, at 11pm
- Prelim instructions, study materials to be posted by tomorrow
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
    xBar
    yBar
    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

% Example
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>200</td>
</tr>
<tr>
<td>3</td>
<td>225</td>
</tr>
<tr>
<td>4</td>
<td>175</td>
</tr>
<tr>
<td>5</td>
<td>210</td>
</tr>
<tr>
<td>6</td>
<td>200</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) $\text{rand}()$

“Normal” distribution with zero mean and unit standard deviation $\text{randn}()$

Distribution of $\text{randn}(1000000,1)$
Step 1: Simulate a fair 6-sided die

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

A  \texttt{round(rand() \times 6)}

B  \texttt{ceil(rand() \times 6)}

C  \textit{Both expressions above}
```
round(rand() * 6)
```
round(rand() * 6)

ceil(rand() * 6)
Step 2: Keep track of results

Possible outcomes from rolling a fair 6-sided die
Simulation

1 2 3 4 5 6
Simulation result

51
59
60
55
59
54

1
2
3
4
5
6
Simulation result

Data in bins

bar(1:6, counts)

Bin numbers

counts

<table>
<thead>
<tr>
<th></th>
<th>51</th>
<th>60</th>
<th>59</th>
<th>55</th>
<th>59</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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); % 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 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;
    elseif face==3
        counts(3)= counts(3) + 1;
    elseif face==4
        counts(4)= counts(4) + 1;
    elseif face==5
        counts(5)= counts(5) + 1;
    else
        counts(6)= counts(6) + 1;
    end
end

counts
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
rollDie.m
% Simulate the rolling of 2 fair dice

totalOutcome=  

\[ \text{ceil}(\text{rand}() \times 12) \]
\[ \text{ceil}(\text{rand}() \times 11) + 1 \]
\[ \text{floor}(\text{rand}() \times 11) + 2 \]

2 of the above

None of the above

\( \text{ceil}(\text{rand}() \times 6) + \text{ceil}(\text{rand}() \times 6) \)
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\times1.

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 ...

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

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 \( \text{rand}() \) being between two numbers is proportional to their difference – *width*
% Standing at (xc,yc)
% Randomly select a step

```matlab
r = rand();
if r < 0.25
    yc = yc + 1;  % north
elseif r < 0.5
    xc = xc + 1;  % east
elseif r < 0.75
    yc = yc - 1;  % south
else
    xc = xc - 1;  % west
end
```

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 $\mathbf{v}$.
- The cumulative sums should be stored in a vector of the same length as $\mathbf{v}$.

$1, 3, 5, 0 \quad \mathbf{v}$

$1, 4, 9, 9$ cumulative sums of $\mathbf{v}$
\[ V = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} \]
\[ \text{csum} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \]

\[ \text{csum}(k) = \text{csum}(k-1) + v(k) \]

\[ \text{csum}(3) = v(1) + v(2) + v(3) \]
\[ \text{csum}(4) = v(1) + v(2) + v(3) + v(4) \]

\[ \text{csum}(k) = \text{csum}(k-1) + v(k); \]
for \( k = 2 : \text{length}(v) \)
\[ \text{csum}(k) = \text{csum}(k-1) + v(k); \]
end