Random variables

Note

- there is no chapter in the textbook that corresponds to this topic

Computing random numbers

- the ability to quickly generate random numbers has many very useful applications
  - Monte Carlo methods
  - Monte Carlo simulations
  - statistical sampling
  - cryptography
  - games of chance
- there are two broad classes of methods for generating random numbers on a computer
  - hardware random number generator
  - pseudo random number generator

Hardware random number generators

- often called true random number generators
- generate random numbers by measuring some sort of physical process that is statistically random
  - thermal noise is probably the most common source in commodity hardware
- called true random number generators because the stream of random numbers is more or less impossible to predict or reproduce

Pseudo random numbers

- often called deterministic random number generators
- a computer algorithm that generates a sequence of numbers that is approximately random
  - the period of the sequence is very long
  - numbers are uniformly distributed
  - difficult to predict the next number in the sequence
- called deterministic because if you know the seed value used to initialize the generator then you can reproduce the sequence of random numbers
  - this is useful for double checking your results

Uniformly distributed random numbers

- in a *discrete uniform distribution* all numbers in the set of values occur with equal probability, e.g.,
  - fair coin: heads, tails
  - fair die: 1, 2, 3, 4, 5, 6
  - fairly shuffled deck of playing cards

MATLAB Uniform RNGs

- MATLAB provides three uniform RNG functions
  - **`rand`**
    - floating-point random numbers strictly between **`0`** and **`1`**
  - **`randi`**
    - integer random numbers from **`1`** to some user-specified value
  - **`randperm`**
    - random permutation of the integers from **`1`** to some user-specified value

- try the functions to see what they do
  - draw a histogram for **`rand`** and **`randi`**

Simulating two dice

```
function [total, v1, v2] = roll( )
%ROLL Rolls two six-sided dice
%  [TOTAL, V1, V2] = ROLL() simulates the rolling of two six-sided
%  dice. The sum of the two dice are returned in TOTAL, the value
%  of the first die is returned in V1, and the value of second
%  die is returned in V2.

v1 = randi(6, [1 1]);
v2 = randi(6, [1 1]);
total = v1 + v2;

end
```

Simulating two dice

- what is the most likely total value?
  - roll the dice many times and draw a histogram of the result
- what are the odds of rolling a total of 2? 3? 4? …
  - roll the dice many times and count the number of 2s, 3s, 4s, …

2D random walk

- a 2D random walk is similar to a 1D random walk except it occurs in 2D
- the walk begins at a point (usually the origin)
- each step of the walk is randomly one step:
  - left,
  - right,
  - up, or
  - down

```
function [p, seq] = walk2(n)
%WALK2 2d random walk
%   [P, SEQ] = WALK2(N) performs an N-step 2D random walk starting from
%   the origin. The final position of the walk is returned in P.
%   The sequence of positions along the walk are returned in SEQ.

seq = zeros(2, n);
for idx = 2:n
    direction = randi(4, [1 1]);
    if direction == 1
        seq(:, idx) = seq(:, idx - 1) + [0; 1];   % up
    elseif direction == 2
        seq(:, idx) = seq(:, idx - 1) + [0; -1];  % down
    elseif direction == 3
        seq(:, idx) = seq(:, idx - 1) + [-1; 0];  % left
    else
seq(:, idx) = seq(:, idx - 1) + [1; 0];           % right
    end
end
p = seq(:, end);

end
```

2D random walk

- what is the most likely final position of the random walker?
  - run the random walk many times and histogram the results

```
P = zeros(2, 1000000);
for idx = 1:10000
  I = 100*(idx - 1) + 1;
  [p, P(:, I:(I+99))] = walk2(100);
end
hist3(P', {-20:20, -20:20})
```

Additive Gaussian noise model

- the most common simple noise model is the additive Gaussian noise model
  - measured value = true value + gaussian noise

- the Gaussian distribution is another name for the normal distribution (bell curve)
- if we want to simulate performing a measurement with additive Guassian noise then we need a way to draw a random number from a normal distribution

MATLAB normally distributed random numbers

- MATLAB provides one RNG for one-dimensional normally distributed values
  - **`randn`**
    - floating-point random numbers drawn from the standard normal distribution (mean = **`0`**, standard deviation = **`1`**)

- if you want a standard deviation of *s* then multiply the result by *s*
- try the function to see what it does
  - draw a histogram