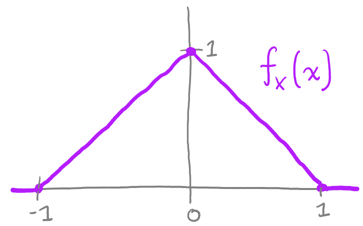
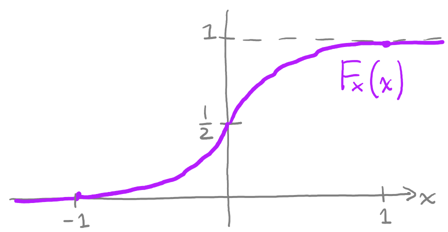
Worksheet Solutions
Math 262 • 6 November 2023

1. Let X have density given by $f_X(x) = \begin{cases} x+1 & \text{if } -1 \le x \le 0, \\ 1-x & \text{if } 0 < x \le 1, \\ 0 & \text{otherwise.} \end{cases}$

(a) Sketch a graph of the pdf $f_X(x)$.



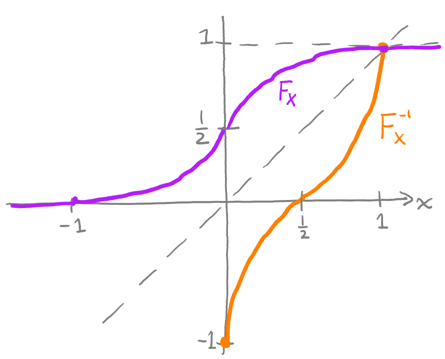
(b) Sketch a graph of the cdf $F_X(x)$. Then find a formula for $F_X(x)$.



$$F_X(x) = \int_{-1}^{x} f_X(t)\,dt = \begin{cases} \int_{-1}^{x} (t+1)\,dt = \left[\frac{1}{2}t^2 + t\right]_{-1}^{x} = \frac{x^2}{2} + x + \frac{1}{2} & \text{if } -1 \le x \le 0 \\ \frac{1}{2} + \int_0^x (1-t)\,dt = \frac{1}{2} + \left[t - \frac{1}{2}t^2\right]_0^x = \frac{1}{2} + x - \frac{x^2}{2} & \text{if } 0 \le x \le 1 \end{cases}$$

(c) Sketch the inverse of $F_X(x)$. Then find a formula for the inverse of $F_X(x)$.

To sketch the inverse, flip F_X across the line $y=x$.



To find a formula for F_X^{-1}, we must consider each piece of F_X separately. Let $u = F_X(x)$ for $-1 \le x \le 1$.

If $-1 \le x \le 0$, then $0 \le u \le \frac{1}{2}$.

In this case: $u = \frac{x^2}{2} + x + \frac{1}{2}$

$2u = x^2 + 2x + 1$

If $0 < x \le 1$, then $\frac{1}{2} < u \le 1$.

In this case: $u = \frac{1}{2} + x - \frac{x^2}{2}$

$-2u = -1 - 2x + x^2$

$$2u = x^2 + 2x + 1$$
$$0 = x^2 + 2x + 1 - 2u$$
so: $x = \dfrac{-2 \pm \sqrt{4 - 4(1-2u)}}{2} = -1 + \sqrt{2u}$

$$-2u = -1 - 2x + x^2$$
$$0 = x^2 - 2x - 1 + 2u$$
so: $x = \dfrac{2 \pm \sqrt{4 - 4(2u-1)}}{2} = 1 - \sqrt{2 - 2u}$

Thus, the inverse of $u = F_X(x)$ is:

$$F_X^{-1}(u) = \begin{cases} -1 + \sqrt{2u} & \text{if } 0 \le u \le \frac{1}{2} \\ 1 - \sqrt{2 - 2u} & \text{if } \frac{1}{2} < u \le 1 \end{cases}$$

(d) Write code to simulate values of X. Simulate thousands of values and make a histogram of the results.

Mathematica:

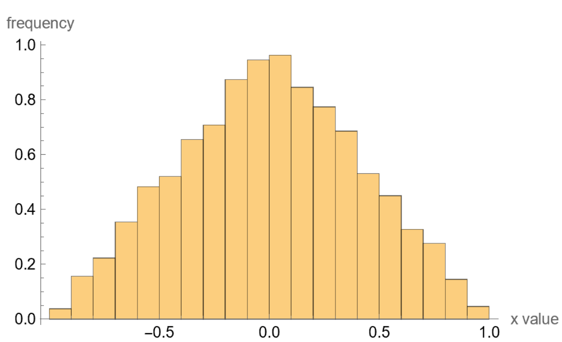
```
simX[] := Module[{},
  u = RandomReal[];
  If[u ≤ 1 / 2, x = -1 + Sqrt[2 u], x = 1 - Sqrt[2 - 2 u]];
  Return[x]
 ]
```

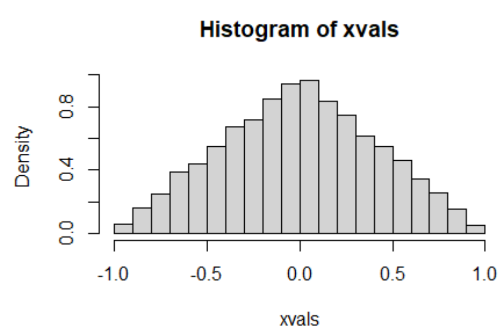
R:

```
simX <- function(){
  u = runif(1)
  if(u <= 0.5){
    return(-1 + sqrt(2*u))
  } #else
  return(1 - sqrt(2 - 2*u))
}

xvals = replicate(10000, simX())
hist(xvals, freq=FALSE)
```

Here are sample histograms from 10,000 simulations each:





2. Brownian motion is the random motion of a particle, such as a gas molecule or a tiny piece of dust floating in air.

   We can simulate 1-dimensional Brownian motion with discrete time steps. Suppose that at time 0, a particle starts at position 0. At each time step, the particle moves according to a random variable with distribution given in problem #1. This distribution implies that the particle could move up to one unit left or right at any time step, but it often moves only a tiny distance per time step.

Specifically, simulate a random variable X_1, which gives the position of the particle at time 1. Simulate another random variable X_2; the position of the particle at time 2 is $X_1 + X_2$. Simulate another random variable X_3; the position of the particle at time 3 is $X_1 + X_2 + X_3$. Continue in this manner to simulate the position of the particle for hundreds of time steps.

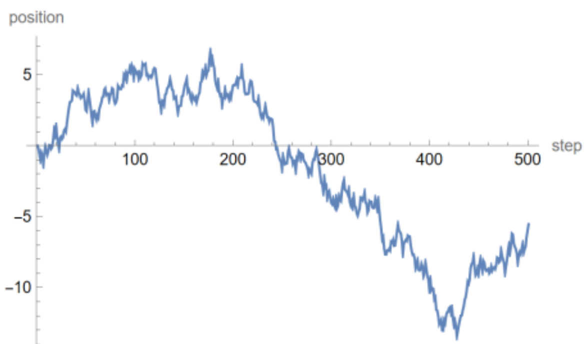
(a) Simulate the Brownian motion described above. Make a plot showing the position of your simulated particle over time.

Mathematica:

```
In[5]:= positions = {0};
        Do[
         AppendTo[positions, simX[] + positions[[-1]]],
         500]

In[8]:= ListLinePlot[positions, AxesLabel → {"step", "position"}]
```

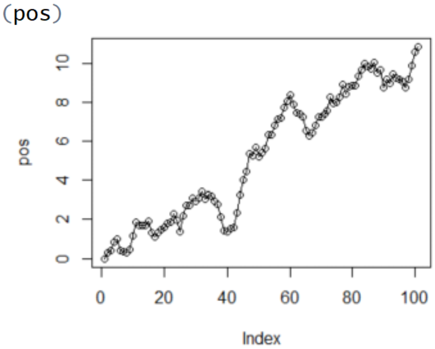
Out[8]=



R:

```
# simulate Brownian motion
pos <- c(0)
for(i in 1:100){
  pos <- append(pos, pos[length(pos)] + simX())
}

plot(pos)
lines(pos)
```



(b) Use simulation to answer the question: What is the average number of time steps until the particle's position is at least ten units from the origin?

Mathematica:

```
In[10]:= timeToTen[] := Module[{pos = 0, time = 0},
           While[Abs[pos] < 10,
            pos += simX[];
            time += 1
           ];
           Return[time]
          ]

In[11]:= timeToTen[]

Out[11]= 156

In[14]:= tenVals = Table[timeToTen[], 1000];

In[15]:= Mean[tenVals] // N

Out[15]= 646.943
```

Your answer will vary.

R:

```
# how much time until distance 10 from origin?
timeToTen <- function(){
  pos = 0
  time = 0
  while(abs(pos) < 10){
    pos = pos + simX()
    time = time + 1
  }
  return(time)
}

times = replicate(1000, timeToTen())
mean(times)
```