Mini Project #1

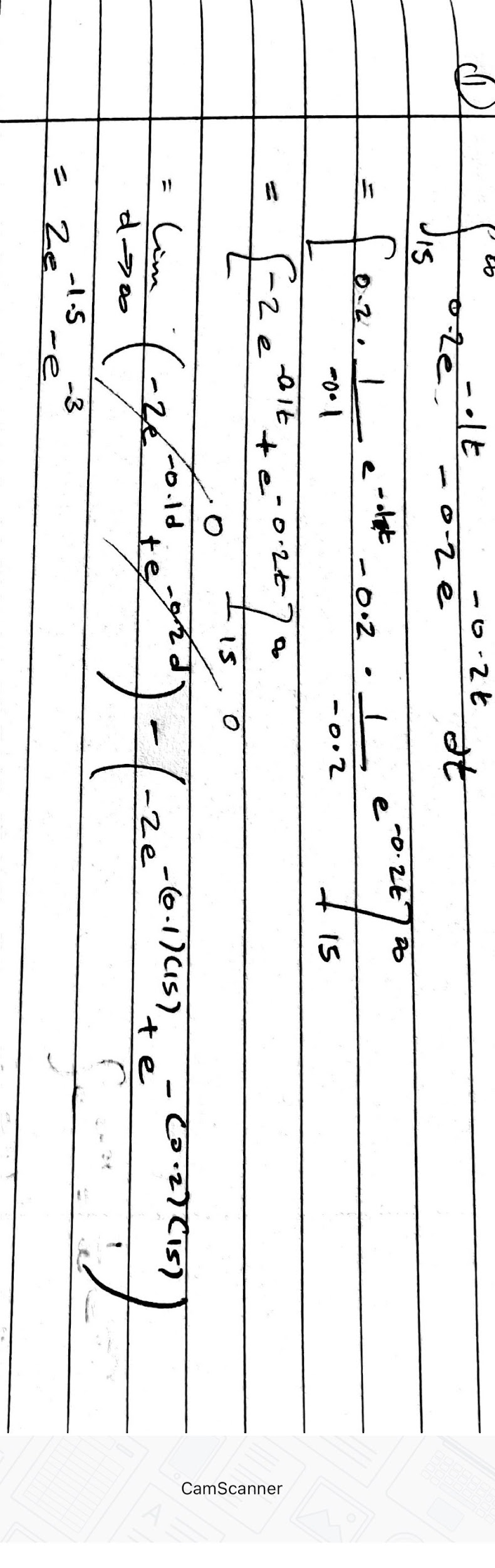
Sudarshana Jagadeeshi

Contribution of each group member: I completed the project in full

Section 1. Answers to the specific questions asked

1

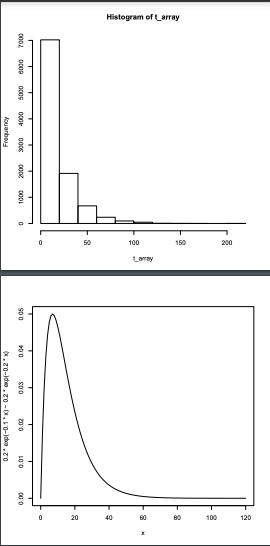
(a). This is equivalent to integrating over 15 to infinity.



Which is around 0.396

b)i-ii. see code attached

iii. I had trouble overlaying the graphs.



We see that the histogram is the area under the pdf and has the same shape. It has its tallest chunk under 0-20, and then gradually tails off as time increases. This makes sense in the context of the problem, as a long lifetime is progressively less likely.

iv. We take the mean of the array. The result is 15.755 years, which is about 5% off compared to the given 15 years.

v. We just see how many values are > 15, and divide by n. The result is: 0.3775. This is off by less than 0.02 from the actual, 0.396.

vi.

Trial 1

[1] 16.20145

[1] 0.3844

Trial 2

[1] 16.02284

[1] 0.3856

Trial 3

[1] 15.87061

[1] 0.3779

Trial 4

[1] 15.87236

[1] 0.3756

Notes: The Expectation continues to be a bit high, while the lifetime probability continues to be a bit low. It is curious that none of the draws broke this rule. They seem to be centered around 0.38 with a deviation of +/- 0.005.

c)

|  | 1,000 | 100,000 |
| --- | --- | --- |
| E(T) | [1] 15.77447  [1] 15.61624  [1] 15.2656  [1] 15.78696  [1] 16.02325 | [1] 16.06664  [1] 16.00788  [1] 16.022  [1] 15.83557  [1] 16.04263 |
| P(E(T) > 15) | [1] 0.357  [1] 0.372  [1] 0.382  [1] 0.378  [1] 0.381 | [1] 0.38027  [1] 0.37774  [1] 0.37878  [1] 0.37599  [1] 0.3783 |

Notes: the values for E(T) continue to be consistently high, even increasing slightly in the 100,000 case, instead of getting closer to 15 as expected. The reason for this may be that we are getting a few extremely large lifetimes every draw. Even so, it should balance, so there may be an error with my code or methodology.

The probability values don’t seem to improve either, which makes me believe that the distribution was already in a good shape after 1000 draws.

2. Estimating PI

You can see the code to see the methodology. The results seemed to be between 3.1 and 3.2 very consistently even after only 400 trials.

Section 2: R code.

(b).

The code should compile and run without issues.

The code for part 1:

n=100000

t\_array <- rep(NA, n)

for(i in 0:n){

block\_a <- 2\* rexp(1, 0.1) #format: (n, lambda) we do 2x because 0.2exp(0.1 \* lambda)

block\_b <- rexp(1, 0.2)

t\_value= block\_a - block\_b

if(t\_value < 0){

t\_value <- 0; #this is in the case that rexp drew from the the negatives

}

t\_array[i] <- t\_value

}

#print(t\_array)

hist(t\_array)

curve(0.2\*exp(-0.1\*x)-0.2\*exp(-0.2\*x),NULL, from=0, to=120)

part4= mean(t\_array) #this is the definition of expectation

print(part4)

part5= ( length(t\_array[t\_array > 15]) )/n #the number of entries with >15 lifetime, divided by the length of the array

print(part5)

And the code for part 2:

hits= 0 #number of times the point was within the circle

n=400 #number of trials

rand\_x=0.0 # this is the x-coordinate

rand\_y=0.0 # y-coordinate

for (i in 0:n){

rand\_x <- runif(1)

rand\_y <- runif(1)

distance= sqrt ( (0.5- rand\_x) \*\* 2 + (0.5- rand\_y) \*\* 2) #Euclidean distance from (0.5,0.5) center

if(distance < 0.5){

hits = hits + 1

}

}

my\_pi= 4\* hits/n #P is area of circle/area of square, or (pi/4)/1. So we need to multiply by 4

print(hits)

print(my\_pi)