AlexWard MPAGS Stats

Alex Ward - Warwick

15th March 2021

Git Link = $https://github.com/j-alex-ward/AlexWard_MPAGS_Stats$
With regards to the formatting, a lot of the figures didn't fit in the right places, they have captions but i apologise for them not being in the correct positions in the text.

List of Figures

1 Q1

When i fix the code in Figure 1, then running the whole notebook produces Figure 2. The sigma here is improved to now show the true spread we expect from an unbiased fit. However the mean is now not equal to 0. To improve the fit further for the pull we can do a couple of things, we can increase the number of events and decrease the number of bins. The result of which can be seen in Figure 3, the mean is closer to 0 and the sigma is closer to 1 than in the original fitting.

edit the function below to calculate the chi-squared

In [9]:
```
def chi2(h, functionClass):
    ## Note that the theoretical function is normalised to one.
    ## To get the expected number of events, we
    ## will need the total number of events, and the binwidth.
    N=fsum(h[0])
    chi2sum=0
    for Nbin, bin_low, bin_up in zip(h[0], h[1], h[1][1:]):
        binCentre=0.5*(bin_low + bin_up)
        binWidth= bin_up - bin_low
        theory=N*binWidth*functionClass(binCentre) # <<<< uncomment this line and make sure you understand it
        #print(theory)
        measured=Nbin
        error_squared=theory # <<<<<<<<<<<<<<<<<<<<<<< uncomment and complete this line
        chi2sum +=  (( theory-measured) **2) / (error_squared)   # <<<<<<<<<<<<<<<<<<<<<<< uncomment and complete th
    #return np.random.rand()*functionClass(binCentre) + 1 # <<<<<<<<<<<<<<<<<<<<<<< remove this line!!!!
    return chi2sum      # <<<<<<<<<<<<<<<<<<<<<<< uncomment this line
```

Now we have all the ingredients to calculate the chi2. Let's do it:

In [10]:
```
chi2(timeHisto,timeFct)
```

Out[10]:
```
17.880259082803537
```

Figure 1: Q1-1

In [16]:
```
pullStudy(timeFct, numEvents=1000, numExperiments=100
          , trueTau=1.5, scanrange=(0.5, 3), numTimeBins=100)
# why might you get a bias in this fit?
```

```
mean pull (should be 0) 1.1380287996974408  +-  0.10302832626967648
sigma pull (should be 1) 1.0302832626967648  +-  0.07285202815958836
```

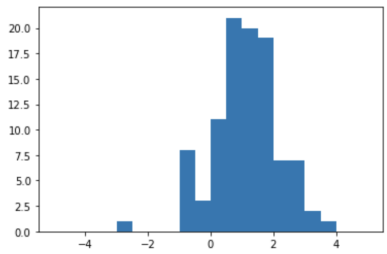


Figure 2: Q1-2

1.1 Why a bias in the fit?

We can see from the plot in cell [14] that the data ranges from 0-4 on the x-axis. The scan is being performed between 0.5 and 3 which could produce a selection

```
In [16]: pullStudy(timeFct, numEvents=100000, numExperiments=100
                   , trueTau=1.5, scanrange=(0.5, 3), numTimeBins=25)
         # why might you get a bias in this fit?
```

```
mean pull (should be 0) 0.008857142854305566  +-  0.09971926626029956
sigma pull (should be 1) 0.9971926626029957  +-  0.07051216938760471
```

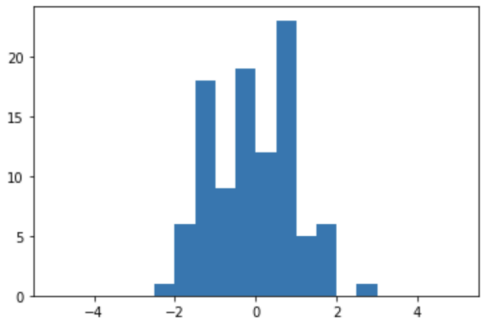


Figure 3: Q1-3

bias as you cut out a significant portion of the results.

1.2 How to reduce the bias?

Increase the number of events and reduce the number of bins in the data set. Also change the scan range.

2 Q2

2.1 Fixing Uncertainty

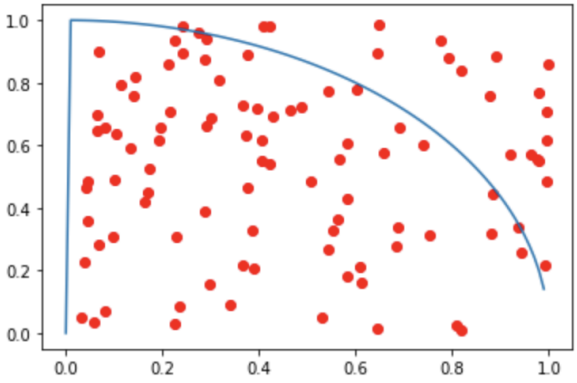
Figures 4 and 5

```
fractionInside = float(counter)/len(Numbers[0])
boxsize        = (xmax-xmin)*ymax
integralValue  = fractionInside * boxsize
#The uncertainty will be Sqrt( n_tot *Frac *(1-Frac) ) * binSize/n_tot
uncertainty    = sqrt(howMany*fractionInside*(1-fractionInside))/howMany * boxsize
if plotopt == "withPlot":
    plotRandomNumbers(Numbers)
    plotFunction(f, xmin, xmax)
    plt.show()
    print("integral value = ", integralValue, " +- ", uncertainty)

return integralValue, uncertainty
```

Figure 4: Q2-1

```
In [20]: result = MCIntegration(quarterCircle, 0, 1, 1, 100)
         print("pi = ", result[0]*4, " +- ", result[1]*4)
         print("Your task: calculate the uncertainty correctly")
```




```
integral value =  0.8  +-  0.039999999999999994
pi =  3.2  +-  0.15999999999999998
Your task: calculate the uncertainty correctly
```

Figure 5: Q2-2

2.2 Pull Study

Figures 6 and 7

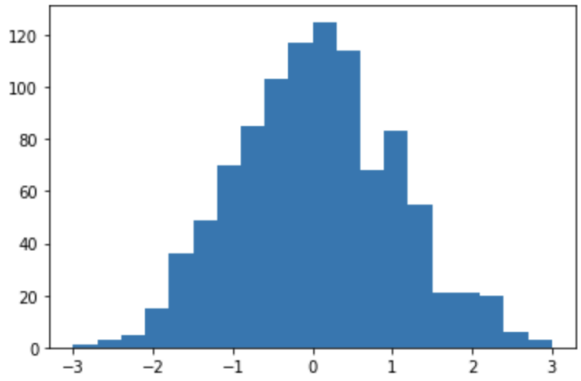
```
In [36]: def PiPullStudy(nExperiments):
             #Pull. = (MeasuredVal - TrueVal)/MeasuredUncertainty
             #The function MCIntegration provides both in an array
             #MeasuredVal = MCIntegration[0]
             #MeasuredUnc = MCIntegration[1]
             #TrueValue (for pi) = np.pi
             #a circle is 4pi so we multiply MeasuredVal*4.
             #Therefore we run n experiments
             #for each experiment we want to measure the pull value of the measured result
             pullList=[]
             for i in range(nExperiments) :
                 Measured = MCIntegration(quarterCircle, 0, 1, 1, 1000, plotopt = "No")
                 pullVal = ( 4 * Measured[0] - np.pi ) / ( 4 * Measured[1] )
                 pullList.append(pullVal)

             pullSum = sum(pullList)
             pullSumSq = 0
             for i in range(len(pullList)):
                 pullSumSq = pullSumSq + pullList[i]**2
             pullMean = pullSum / nExperiments
             pullMeanSq = pullSumSq / nExperiments
             Variance = pullMeanSq - pullMean**2 # <x**2> - <x>**2
             sig = np.sqrt(Variance)
             meanSig = sig / np.sqrt(nExperiments)
             sigSig = sig / np.sqrt( 2 * nExperiments )
             plt.hist(pullList, 20, range=(-3, 3) )
             plt.show()
             print("Pull Mean = {0} ± {1}, Pull Sigma = {2} ± {3}".format( round(pullMean,4), round(meanSig,4), round(sig,4),

         PiPullStudy(1000)
```

Figure 6: Q2-3

```
PiPullStudy(1000)
```




```
Pull Mean = 0.0565 ± 0.0316, Pull Sigma = 0.9977 ± 0.0223
```

Figure 7: Q2-4

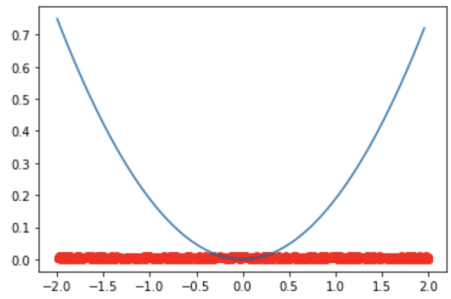
2.3 Wrong y-max

Figures 8

2.4 Other Functions

Figures 9 and 10 and 11

In [51]:
```
#incorrect y-axis value test
MCIntegration(NormalisedParabola, -2, 2, 0.01, 1000)
ParabolaArray=MCGeneration(NormalisedParabola, -2, 2, 0.01, 10000)
```




```
integral value =  0.03728  +-  0.0003184361788490748
```

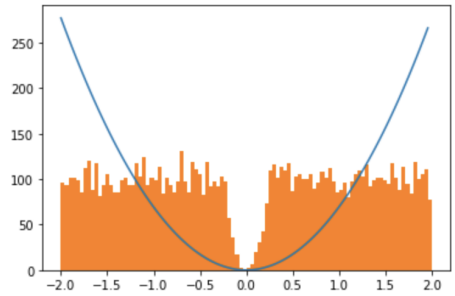
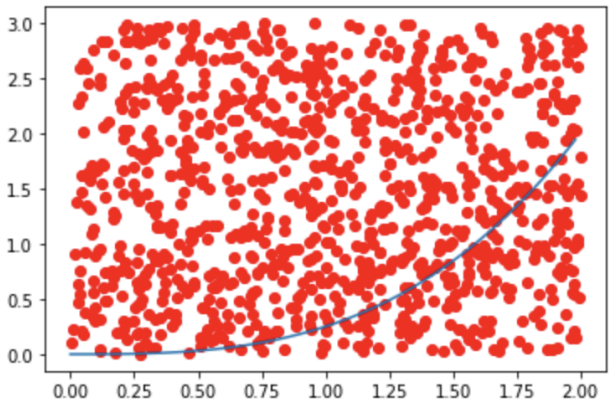


Figure 8: Q2-ymax

```
In [52]: def NormalisedCube(x, xmin, xmax):
             return x**3 * 4/(xmax**4-xmin**4)

         MCIntegration(NormalisedCube, 0, 2, 3, 1000)
         FuncArray=MCGeneration(NormalisedCube, 0, 2, 3, 10000)
```




```
integral value =  1.026  +-  0.07143755314958652
```

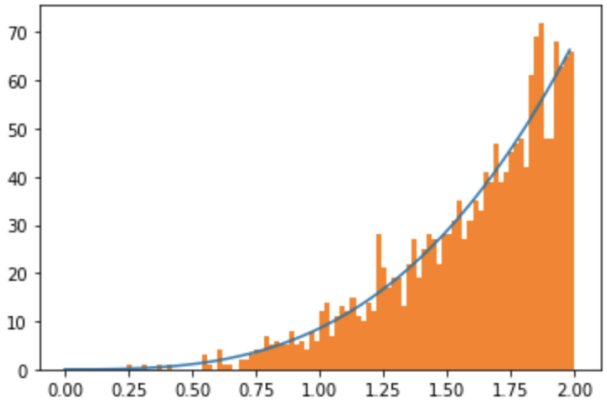
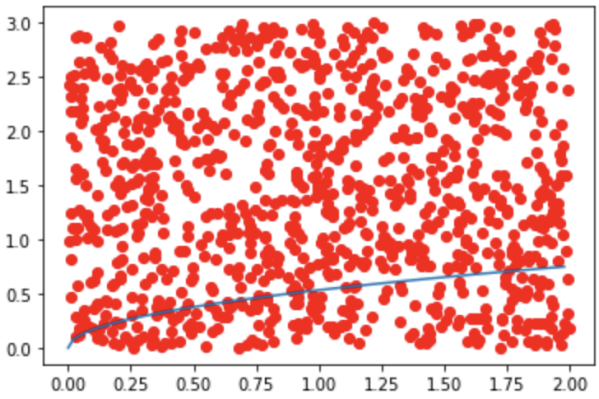


Figure 9: Q2-cube function

```
In [53]: def NormalisedRoot(x, xmin, xmax):
             return x**0.5 * 1.5/(xmax**1.5-xmin**1.5)

         MCIntegration(NormalisedRoot, 0, 2, 3, 1000)
         FuncArray=MCGeneration(NormalisedRoot, 0, 2, 3, 10000)
```




```
integral value =  1.092  +-  0.07320885192379402
```

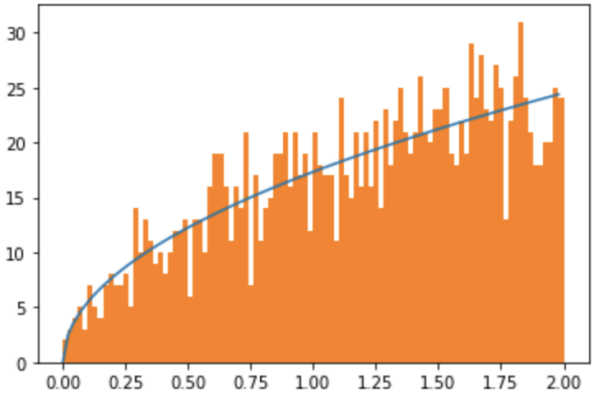
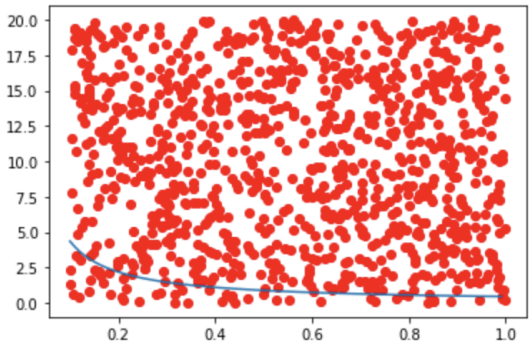


Figure 10: Q2-sqrt function

```
In [67]: def NormalisedRecpirocal(x, xmin, xmax):
             return 1/x/(np.log(xmax)-np.log(xmin))

         MCIntegration(NormalisedRecpirocal, 0.1, 1, 20, 1000)
         FuncArray=MCGeneration(NormalisedRecpirocal, 0.1, 1, 20, 10000)
```




```
integral value =  0.882  +-  0.12287422838007976
```

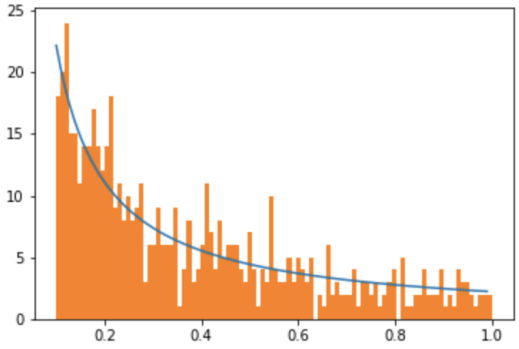


Figure 11: Q2- reciprocal function

3 Q3

At first ('*Q3_Micromorts.py*') i tested an assumed binomial probability, with $N = TotalDist/StepSize$ discrete measurements, and a probability surviving in a step, $p = 1 - micromort$. This attempt can be seen in the following code and output. As i ran this multiple times the order of magnitude on the probability remained the same for all three probabilities.

```
from numpy import random

def numpyBin(MaxMiles):
    stepsize                = 6
    steps                   = int(MaxMiles/stepsize)
    probOfAccidentEachStep = 1.e-6
    nExperiments = 100000
    x = random.binomial(n=steps, p=1-probOfAccidentEachStep, size=nExperiments)
    DeathChance = sum(x != steps) / nExperiments
    Perc_DeathChance = round(DeathChance*100,2)
    print("P(Die in {2} Mile Journey) = {0} = {1}%".format(DeathChance,
    Perc_DeathChance, MaxMiles) )
    return 0

numpyBin(15E3)
numpyBin(6E6)
numpyBin(6E7)
```

```
In [163]: run Q3_Micromorts.py
P(Die in 15000.0 Mile Journey) = 0.0026 = 0.26%
P(Die in 6000000.0 Mile Journey) = 0.62951 = 62.95%
P(Die in 60000000.0 Mile Journey) = 0.99995 ~ 100.0%
```

Therefore for each step (of 6 miles equating to 1 micromort) there is a binomial probability of surviving in that journey, thus analytically,

$$P(\ Surviving\ N\ steps) = \begin{bmatrix} n \\ N \end{bmatrix} p^N (1-p)^{n-N} \tag{1}$$

where n is the total steps, $n = MaxMiles/stepsize$, p is the probability of success, or surviving the step ($p = 1 - micromort$) and N is the total succesful steps.

Therefore we can repeat this experiment any number of times (with n clones), 100000 in the example above, and calculate the number of clone which died and then divide this by the number of experiments, eg:

$$P(\ dying\ in\ N\ steps) = 1 - \begin{bmatrix} n \\ N \end{bmatrix} p^N (1-p)^{n-N} \tag{2}$$

$$P(\ Clone\ dying) = \frac{\sum N_{\ Clones\ Survived}}{N_{Clones}} \tag{3}$$

Where N_{Clones} is the total number of experiments (or clones) and $N_{\ Clones\ Survived}$ is number of clones who survived out of the total number of clones.

4 Q4 - Calculate the probability that a person that tests positive for the virus, does indeed carry the virus.

This question was answered using the following code with the provided answer following.

```
import numpy

P_Infected = 0.002
P_NotInfected = 1 - P_Infected
P_Positive_Giv_Infect = 0.99
P_Positive_Giv_NotInfect = 0.0015

P_PositiveANDInfected = P_Infected * P_Positive_Giv_Infect
P_PositiveANDNotInfected = P_NotInfected * P_Positive_Giv_NotInfect

P_PositiveTest = P_PositiveANDInfected + P_PositiveANDNotInfected

P_Infected_Given_PositiveTest = P_Positive_Giv_Infect
* ( P_Infected / P_PositiveTest )

print("P( Infected | Positive Test ) = {}%".format(100*round(
P_Infected_Given_PositiveTest,4)))
```

The calculated probability = $P(Infected|PositiveTest) = 56.95\%$

5 Q5 - Which distribution best describes the following:

5.1 a) The number of flashes of lightening within on hour of a thunderstorm.

Poisson - The number of flashes of lightening in a time period are not set to a specific number, neither are the number of flashes continuous. Therefore an unspecified, discrete data set is described best by a Poisson distribution.

5.2 b) The number of Higgs events at the LHC in a year of running.

Poisson - Discrete trials, nearly continuous but can be described using a Poisson for number of Higgs events.

5.3 c) The number of students per hundred carrying the H1F1 virus.

Binomial - there are two outcomes (infected or not), a fixed number of trials (100) which are independent.

5.4 d) Weight of individual A4 pieces of paper in a notebook

Gaussian - the weight of the paper is continuous.

5.5 e) The number of sand grains in 1kg of sand.

Poisson - discrete value with a non-fixed number of trials, can have expected number of grains per 1kg.

6 Q6

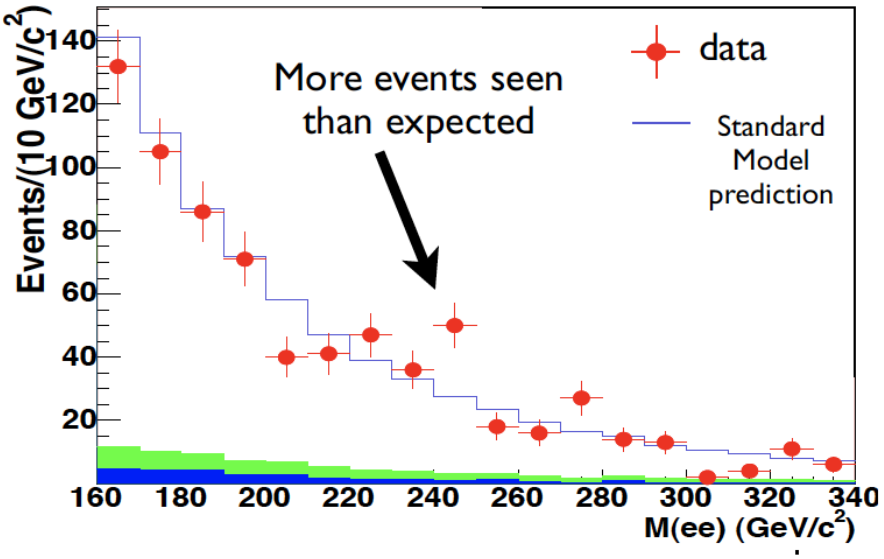


Figure 12: $Z' \to \mu\mu$: The plot shows recent data from the CDF experiment at Fermilab

The code for this question is found in Q6.py, however i have included the output from running the code below:

```
Using a one-tailed Poisson confidence integral for 84 bins (measurements):
Measured Value x = 48 is +3.78 Sigma from Mean = 28
This has a probability of 0.999922 = 99.9922%
P(at least one increase of 20 events) =
1 - P(all within 20 event increase)^84 = 0.65654%
x Value for 3 Sigma = 43.87
x Value for 5 Sigma = 54.46
```

6.1 a) Estimate the significance of this observation, by calculating the probability of observing such an increased number of events as a result of a statistical fluctuation, taking into account that the physicists are looking simultaneously in 84 bins in the mass range from 160 GeV/c2 to 1000 GeV/c2 (for clarity, only some of that range is shown in the plot above).

Using the Poisson distribution we can define parameters:

- $x = 48$ = Measured Number of Events
- $\mu = 28$ = Expected Mean Number of Events
- $\sigma = \sqrt{\mu} = \sqrt{28}$ = Standard Deviation
- $s = \frac{x-\mu}{\sigma} = +3.78$
- $NBins = NMeasurements = 84$ = Number of bins
- Increase in events (excess) = 20

We are looking at an excess of events therefore we need to use a one-sided integral for the probability, Equation 4 defines this as $P(All\ yields\ within\ excess) = 99.9922\%$

$$I = \int_{\infty}^{s} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}s^2} ds \tag{4}$$

We now look to obtain the significance of $P(At\ least\ one\ increase\ of\ N\ events) = 0.65654\%$, defined as Equation 5,

$$P(At\ least\ one\ increase\ of\ N\ events) = 1 - P(All\ yields\ within\ excess)^{NBins} \tag{5}$$

6.2 b) Your answer above should be a fairly small number - less than a percent. So, this is quite an interesting result. Why do you think this has still not been published as a discovery of a new particle (that would have been all over the media)?

The 3σ confidence level has a limit on the yield (for this bin) of 43.87 while the 5σ level has a yield of 54.46. The norm in particle physics is to publish a 3σ result as 'Evidence' while the 5σ measurement is considered a 'Discovery'. The (approximate) measured yield for this bin is 48, is 6.46 events lower than the calculated 5σ value, therefore it cannot be published as a discovery (although is above the 3σ limit and therefore worth publishing as evidence and following up with an independent measurement).

7 Q7

Errorbars : Which of these 4 plots look like what you would expect if the theoretical prediction (black line) describes the data (red circles with blue error bars) and the error bars are Gaussian? Which ones don't? Why?

7.1 a

The distribution of this plot looks like the spread of the data describes the theoretical prediction well, however the error bars look to be very small, non-existent on some data points, and therefore the error bars are not Gaussian.

7.2 b

This data looks like it describes the theory well, the error bars look sensible size and there is a reasonable spread of approximately 1/3 of the data points having the theory outside the error bars.

7.3 c

The theory does not fit the data, the data is clearly not linear and therefore this is a poor correlation.

7.4 d

The data describes the theory however with the error bars looking too large for some of the data points and 100% of the data points agree with the theory this is a suspiciously good data-theory match, therefore would need to be tested for bias.

8 Q8 : Probability questions with a twist

8.1 When tossing a coin which has an equal chance of giving you heads or tails, which of the following (ordered) sequences is the most likely? (from most to least likely)

The sequences are already in order, the same run of 5 heads appears in all three sequences and each sequence creates a more unlikely, loner sequence, therefore (from most to least likely):

- (A) HHHHH
- (B) TTHHHHH
- (C) TTHHHHHTT

8.2 In a terrible road accident on a usually safe and accident-free stretch of road in South Yorkshire, 5 people get severely injured. Rank the following statements by the probability that they are true. (from most to least likely)

- (B) The accident was caused by an experienced driver who was drunk.
- (A) The accident was caused by an experienced driver.

9 Q9

$$P(t) = \begin{cases} Ke^{\frac{-t}{\tau}}, & \text{if } > 0 \\ 0, & \text{otherwise} \end{cases}$$

9.1 a - Show that $K = \frac{1}{\tau}$

Normalise and solve for K:

$$\int_0^\infty Ke^{\frac{-t}{\tau}} = 1 = K\tau \tag{6}$$

$$K = \frac{1}{\tau} \tag{7}$$

9.2 b - What is the expectation value and the standard deviation of P(t)?

9.2.1 Expectation Value:

$$< t >= \int_0^\infty tP(t)dt = \int_0^\infty tKe^{-tK}dt = \frac{1}{k} = \tau \tag{8}$$

9.2.2 Variance:

$$< t^2 >= \int_0^\infty t^2P(t)dt = \frac{2}{k^2} \tag{9}$$

$$V_t = \int_0^\infty (t- < t >)^2P(t)dt = \int_0^\infty t^2P(t)dt- < t >^2= \frac{2}{k^2} - \frac{1}{k^2} = \frac{1}{k^2} \tag{10}$$

9.2.3 Standard Deviation:

$$\sigma_t = \sqrt{V_t} = \frac{1}{k} = \tau \tag{11}$$

9.3 c - There are 3 particles at t = 0. What is the probability that at least one of these particles still exists (i.e. has not decayed) at t = 3τ?

Probability of having x = 1 events observed at time t, with a=3 initial events,

$$P(x) = \frac{e^{-a}a^x}{x!} = 0.149 \tag{12}$$

9.4 d - Given the PDF for the variable t given above, find the PDF for the variable $x(t) = 1 - e^{\frac{-t}{\tau}}$.