PHYS 243 Homework 3

Carlos Olivas ID# 861045506

1)

Probability needle will fully land in a single shade when l > t:

$$P = 1 - \frac{2}{\pi}\left[\frac{l}{t} - \sqrt{\left(\frac{l}{t}\right)^2 - 1} + sec^{-1}\left(\frac{l}{t}\right)\right]$$

In [34]:

```python
import numpy as np
import random
from math import pi


# method to calculate arcsecant
def arcsec(input_val):
    return 0.5 * (pi - 2 * np.arcsin((1 / input_val)))

class BuffonMonteCarlo():

    # constructor
    def __init__(self, needle_len, strip_wid):
        self.needle_length = needle_len
        self.strip_width = strip_wid

     # change the values for the needle length and the strip width
    def sim_redefine(self, needle_len, strip_wid):
        self.needle_length = needle_len
        self.strip_width = strip_wid

    # return the analytical probability for the needle length and strip width for this simulation iteration
    def analytic_prob(self):

        return 1 - (2 * self.needle_length) / (self.strip_width * pi) if self.needle_length < self.strip_width \
            else 1 - (2 / pi) * ((self.needle_length / self.strip_width)  - np.sqrt(((self.needle_length / self.strip_width) **
2) - 1) + arcsec((self.needle_length / self.strip_width)))

    def drop_simulation(self):

        # number of drops for the simulation
        drops = 10000

        # number of drops where needle fully lands inside strip
        good_drops = 0

        for _ in range(drops):
            x = random.uniform(0, (self.strip_width / 2 ))
            theta = random.uniform(0, (pi / 2))

            # if needle does not intersect strip borders
            if x >= (self.needle_length / 2) * np.sin(theta):
                good_drops += 1

        # return average times needle fell fully inside strip
        return good_drops /  drops

    def drop_simulation_for_plot(self):

        # number of drops for the simulation
        drops = 100000

        # number of drops where needle fully lands inside strip
        good_drops = 0

        for _ in range(drops):
            x = random.uniform(0, (self.strip_width / 2))
            theta = random.uniform(0, (pi / 2))

            # if needle does not intersect strip borders
            if x > (self.needle_length / 2) * np.sin(theta):
                good_drops += 1

        # return the needle length/strip width ratio and average times needle fell fully inside strip
        return [(self.needle_length / self.strip_width), (good_drops /  drops)]
```

2)

Find the probability using Monte Carlo simulation for l < t:

In [38]:

```python
needle_len = 2
strip_wid = 3
buffon_sim = BuffonMonteCarlo(needle_len, strip_wid)
montecarlo_prob = buffon_sim.drop_simulation()
print("Probability of needle of length %d falling within strip of width %d is: %f"% (needle_len, strip_wid, montecarlo_prob))
```

```
Probability of needle of length 2 falling within strip of width 3 is: 0.579300
```

Find the value of π using special case:

In [39]:

```python
new_pi = (2 * needle_len) / ((1 - montecarlo_prob) * strip_wid)
print("New value for pi: %f"%new_pi)
```

```
New value for pi: 3.169321
```

3)

Find the probability using Monte Carlo simulation for l > t:

In [40]:

```python
needle_len = 4
strip_wid = 2
buffon_sim.sim_redefine(needle_len, strip_wid)
montecarlo_prob = buffon_sim.drop_simulation()
print("Probability of needle of length %d falling within strip of width %d is: %f"% (needle_len, strip_wid, montecarlo_prob))
```

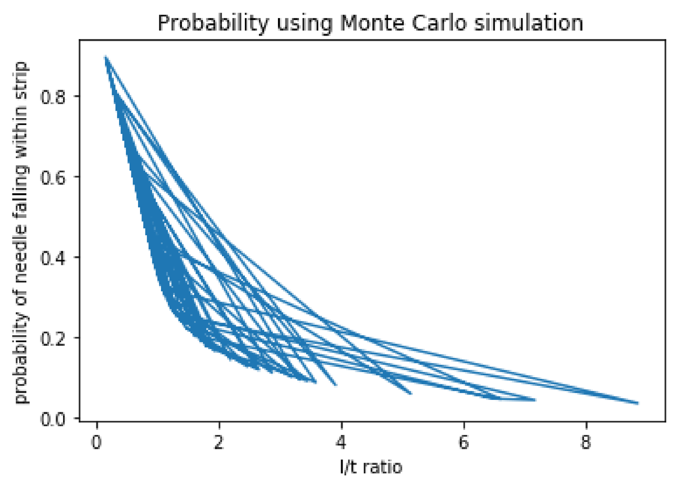
```
Probability of needle of length 4 falling within strip of width 2 is: 0.158600
```

4)

Plot the probability vs l/t ratio:

In [43]:

```python
import matplotlib.pyplot as plt
%matplotlib inline


buffon_sim = BuffonMonteCarlo(0,0)
pairs = []
vals_list = []

for i in range(100):
    needle_len = random.uniform(1, 10)
    strip_wid = random.uniform(1, 10)

    pairs.append([needle_len,strip_wid])

    buffon_sim.sim_redefine(needle_len, strip_wid)
    vals_list.append(buffon_sim.drop_simulation_for_plot())

plot_list = [[vals_list[i][0] for i in range(len(vals_list))], [vals_list[j][1] for j in range(len(vals_list))]]

plt.plot(plot_list[0], plot_list[1])
# plt.xlim(left= 0, right = 1.5)
plt.title('Probability vs l/t ratio')
plt.xlabel('l/t ratio')
plt.ylabel('probability of needle falling within strip')
plt.title('Probability using Monte Carlo simulation')
plt.show()
```



5)

Plot the analytic formula for P(l/t) along side your previous result

In [45]:

```python
analytic_vals = []
for pair in pairs:
    buffon_sim.sim_redefine(pair[0], pair[1])
    analytic_vals.append(buffon_sim.analytic_prob())

plt.plot(plot_list[0], analytic_vals)
plt.title('Probability vs l/t ratio')
plt.xlabel('l/t ratio')
plt.ylabel('probability of needle falling within strip')
plt.title('Probability using analytic formula')
plt.show()
```