Paul Fischer

PHYS 522 Statistical Physics

Dr. Peterson

5/6/2021

Homework 5 Bonus Problem

Bonus Problem: A fun example of a Monte Carlo program is one that estimates the value of π. Imagine a unit circle placed inside a unit square such that the center of each coincide–assume they both exist in the x-y plane. The ratio of the area of the circle to the area of the square is $\pi/4$. A simple Monte Carlo algorithm to estimate π is to choose N (where N is a large number) of random points in the x-y plane such that $x \in [-1, 1]$ and $y \in [-1, 1]$. Add up all the random points M that lie within the unit circle and divide by the total number of points N. The ratio M/N will approach $\pi/4$ as $N \to \infty$.

a) For $N = 100$ random numbers color the number of points in the unit circle red and the points outside the unit circle green. Plot your results.

Let us begin by defining a function mOverN that calculates M/N for N random numbers and colors the number of points in the unit circle red and the points outside the unit circle green and plots the results.

In [148]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import gamma, factorial
```

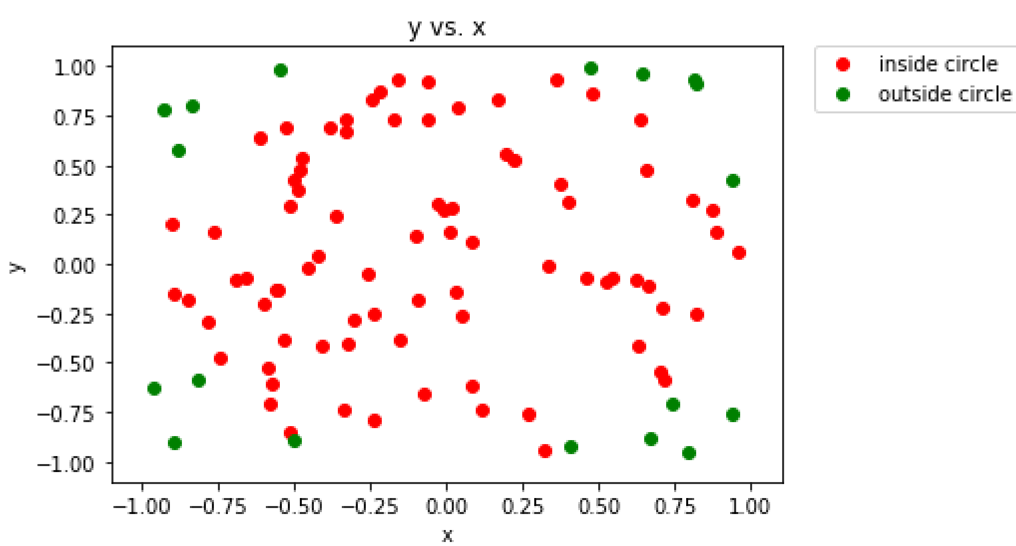
In [149]:
```
def mOverN(n, r, nDim, plot=False):

    """Return the fraction of n random tuples confined to a length 2*r s
ided nDim-cube that lie
    inside an nDim-ball of radius r."""

    # create nDim row list (~x[0], x[1], ..., x[nDim-1]) with n random p
oints in each row range (-r, r]
    x = []
    for i in range(nDim):
        x.append(r * 2*np.random.random_sample(n)-1)

    # initialize nDim arrays (~xIn[0], xIn[1], ..., xIn[nDim-1]) for ran
dom points inside (and outside) nDim-sphere
    # of radius r
    xIn = []
    for i in range(nDim):
        xIn.append([])

    xOut = []
    for i in range(nDim):
        xOut.append([])

    # create arrays of points inside and outside n-sphere of radius r
    for j in range(n):

        # calculate rSq = x[i][0]**2 + x[i][1]**2 + ... * x[i][nDim-1]**
2
        rSq = 0
        for i in range(nDim):
            rSq = rSq + x[i][j]**2

        # determine if rSq is inside or outside n-sphere of radius r
        if rSq <= r:
            for i in range(nDim):
                xIn[i].extend([x[i][j]])
        else:
            for i in range(nDim):
                xOut[i].extend([x[i][j]])

    # plot
    if plot == True and nDim == 2 and r == 1:
        plt.plot(xIn[0], xIn[1], 'ro', label="inside circle")
        plt.plot(xOut[0], xOut[1], 'go', label="outside circle")
        plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxe
spad=0.)
        plt.xlabel('x')
        plt.ylabel('y')
        plt.title('y vs. x')
        plt.axis([-1.1, 1.1, -1.1, 1.1])
        plt.show()

    # calculate M/N
    else:
        res = len(xIn[0])/n
        return res
```

Let us define a function piError that calculates π via this Monte Carlo method and calculates the % error for N points and nTrials trials.

In [146]:
```
def piError(n, nTrials):

    """Print the statistical analysis of the estimation of pi via the Monte Carlo method."""

    piEst = []
    for i in range(nTrials):
        mN = mOverN(n, 1, 2, plot=False)
        piEst.append(4 * mN)

    avg = sum(piEst)/len(piEst)
    stDevMean = np.std(piEst) / np.sqrt(nTrials)

    exact = np.pi

    print("pi estimate for", n, "points after", nTrials, "trials:")
    print(avg, "+/-", stDevMean)
    print(np.pi)
    print("% error:", abs(avg - exact)/exact * 100)
```

Let us plot the results for $N = 100$.

In [150]:
```
n = 100
r = 1
nDim = 2

mOverN(n, r, nDim, plot=True)
```

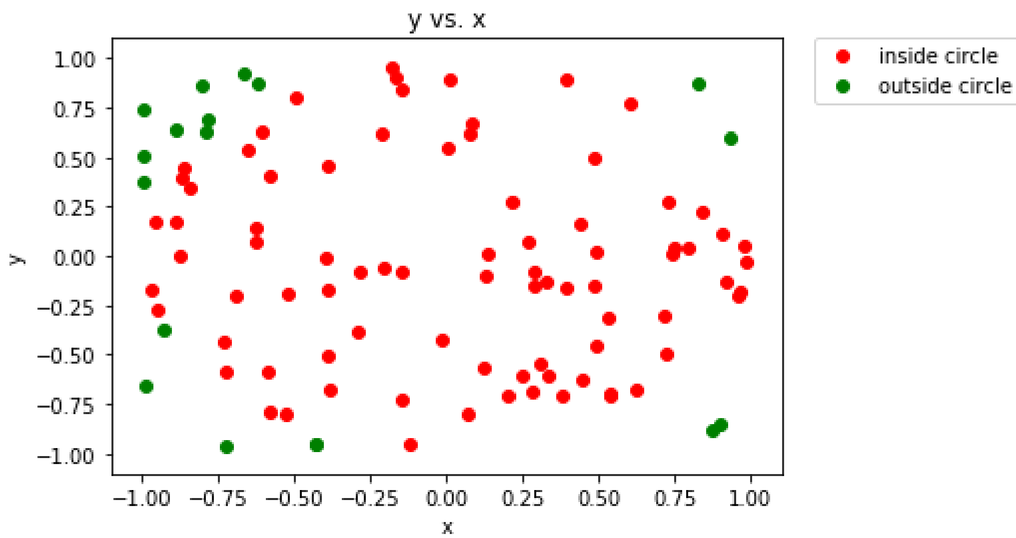


b) Calculate M/N for $N = 100$, 200, 1000, and 1×10^6.

Let us calculate M/N for each N value, plot the results, and calculate the error.

In [157]:
```
n = 100
r = 1
nDim = 2
nTrials = 10

print("For N = ", n, ", M/N = ", mOverN(n, r, nDim, plot=False), ".", se
p="")
mOverN(n, r, nDim, plot=True)
piError(n, nTrials)
```

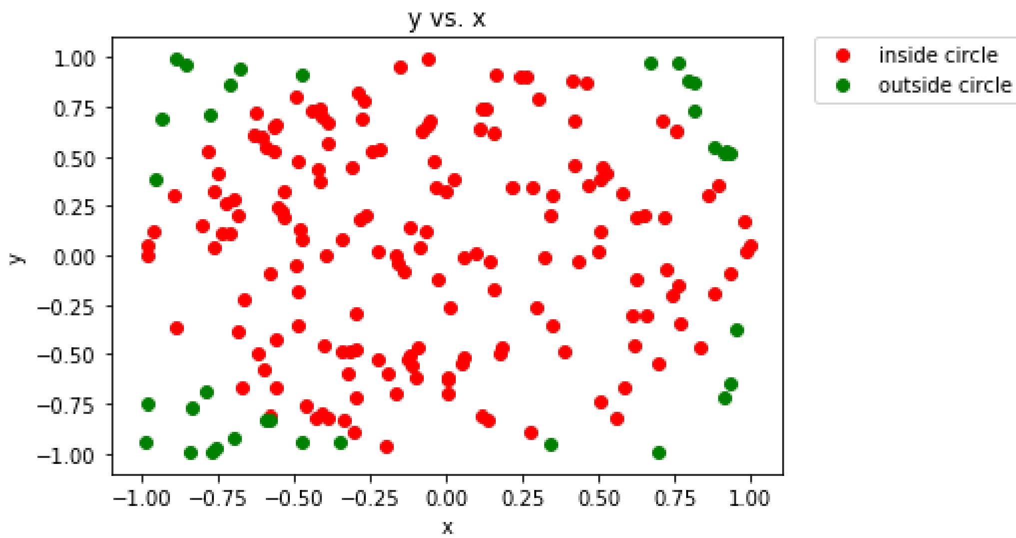
```
For N = 100, M/N = 0.79.
```




```
pi estimate for 100 points after 10 trials:
3.088 +/- 0.044828562323590095
3.141592653589793
% error: 1.705907146445434
```

In [158]:
```
n = 200
r = 1
nDim = 2
nTrials = 10

print("For N = ", n, ", M/N = ", mOverN(n, r, nDim, plot=False), ".", se
p="")
mOverN(n, r, nDim, plot=True)
piError(n, nTrials)
```

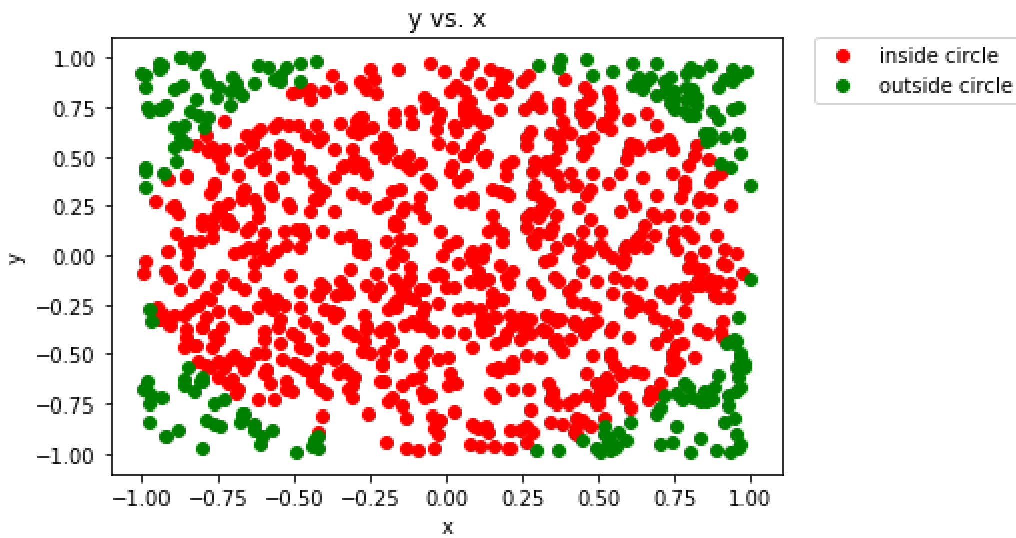
```
For N = 200, M/N = 0.755.
```




```
pi estimate for 200 points after 10 trials:
3.108 +/- 0.012712198865656573
3.141592653589793
% error: 1.0692873740778523
```

In [159]:
```
n = 1000
r = 1
nDim = 2
nTrials = 10

print("For N = ", n, ", M/N = ", mOverN(n, r, nDim, plot=False), ".", se
p="")
mOverN(n, r, nDim, plot=True)
piError(n, nTrials)
```

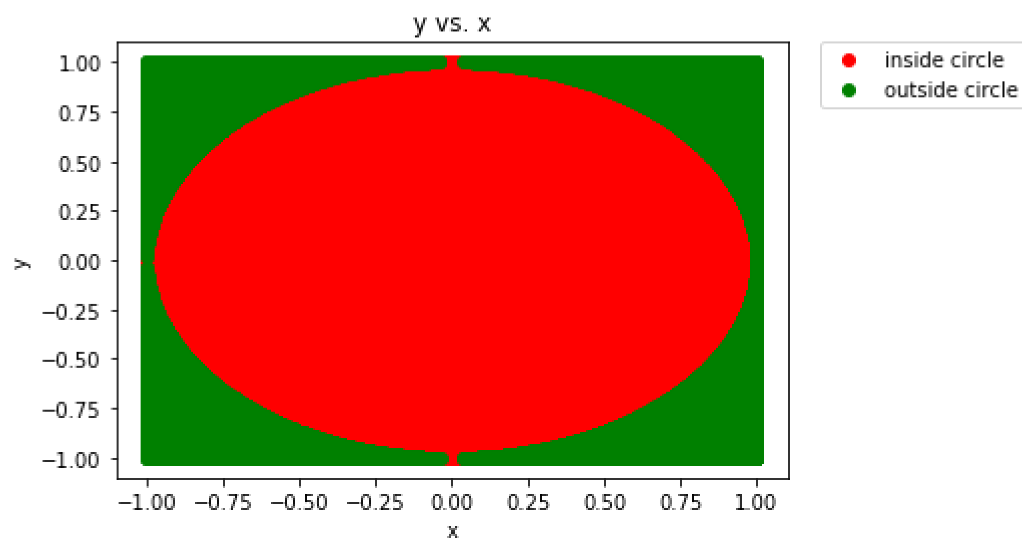
```
For N = 1000, M/N = 0.786.
```




```
pi estimate for 1000 points after 10 trials:
3.1435999999999997 +/- 0.00834170246412566
3.141592653589793
% error: 0.06389582073643074
```

In [160]:
```
n = 1 * 10**6
r = 1
nDim = 2
nTrials = 10

print("For N = ", n, ", M/N = ", mOverN(n, r, nDim, plot=False), ".", se
p="")
mOverN(n, r, nDim, plot=True)
piError(n, nTrials)
```

```
For N = 1000000, M/N = 0.78582.
```




```
pi estimate for 1000000 points after 10 trials:
3.1407008000000003 +/- 0.0005812455040686328
3.141592653589793
% error: 0.028388581465955885
```

c) Use this method to approximate the volume of a hyper-sphere of unit radius in 5-dimensions.

The volume of an n-ball is given by the relation $V_n(R) = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} R^n$ [1]. Let us define a function nBallVolume to calculate this volume to benchmark our Monte Carlo result.

In [161]:
```
def nBallVolume(n, r):

    """Return the volume of an n-ball of radius R."""

    res = np.pi**(n/2) / gamma(n/2 + 1) * r**n

    return res
```

Let us benchmark this function by calculating the volume of a 3-ball of radius 1.

In [162]:
```
print(nBallVolume(3, 1), 4/3 * np.pi, sep="\n")
```

```
4.188790204786391
4.1887902047863905
```

Let us define a function nSphVolErr that calculates the n-ball of radius R volume via the Monte Carlo method and calculates the % error for N points and nTrials trials.

In [170]:
```
def nSphVolErr(n, r, nDim, nTrials):

    """Print the statistical analysis of the estimation of the n-ball vo
lume via the Monte Carlo method."""

    nCubeVol = (2*r)**nDim

    nSphVolEst = []
    for i in range(nTrials):
        mN = mOverN(n, r, nDim, plot=False)
        nSphVolEst.append(mN * nCubeVol)

    avg = sum(nSphVolEst) / len(nSphVolEst)
    stDevMean = np.std(nSphVolEst) / np.sqrt(n)

    exact = nBallVolume(nDim, r)

    print(nDim, "-sphere of radius ", r, " volume estimate for ", n, " p
oints after ", nTrials, " trials:", sep="")
    print(avg, "+/-", stDevMean)
    print(exact)
    print("% error:", abs(avg - exact)/exact * 100)
```

Let us benchmark the result of this function against the volume of the 3-ball of radius 1 calculated above.

In [171]:
```
n = 100
r = 1
nDim = 3
nTrials = 10

nSphVolErr(n, r, nDim, nTrials)
```

```
3-sphere of radius 1 volume estimate for 100 points after 10 trials:
4.296 +/- 0.036494383129462536
4.188790204786391
% error: 2.55944532841735
```

Let us now use this method to approximate the volme of a hyper-sphere of unit radius in 5-dimensions for the N values given above.

In [172]:
```
n = 100
r = 1
nDim = 5
nTrials = 10

nSphVolErr(n, r, nDim, nTrials)
```

```
5-sphere of radius 1 volume estimate for 100 points after 10 trials:
5.4079999999999995 +/- 0.10164723311531898
5.263789013914324
% error: 2.7396802133305025
```

In [173]:
```
n = 200
r = 1
nDim = 5
nTrials = 10

nSphVolErr(n, r, nDim, nTrials)
```

```
5-sphere of radius 1 volume estimate for 200 points after 10 trials:
5.408 +/- 0.05674927312309824
5.263789013914324
% error: 2.7396802133305194
```

In [174]:
```
n = 1000
r = 1
nDim = 5
nTrials = 10

nSphVolErr(n, r, nDim, nTrials)
```

```
5-sphere of radius 1 volume estimate for 1000 points after 10 trials:
5.4208 +/- 0.014008545963089817
5.263789013914324
% error: 2.9828510540721203
```

In [175]:
```
n = 1 * 10**6
r = 1
nDim = 5
nTrials = 10

nSphVolErr(n, r, nDim, nTrials)
```

```
5-sphere of radius 1 volume estimate for 1000000 points after 10 trial
s:
5.2697952 +/- 9.844080584798092e-06
5.263789013914324
% error: 0.11410385313315476
```

References

[1] Wikipedia contributors, "Volume of an n-ball," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Volume_of_an_n-ball&oldid=1013837368 (https://en.wikipedia.org/w/index.php?title=Volume_of_an_n-ball&oldid=1013837368) (accessed May 6, 2021).