COMS 6100 - Homework 11

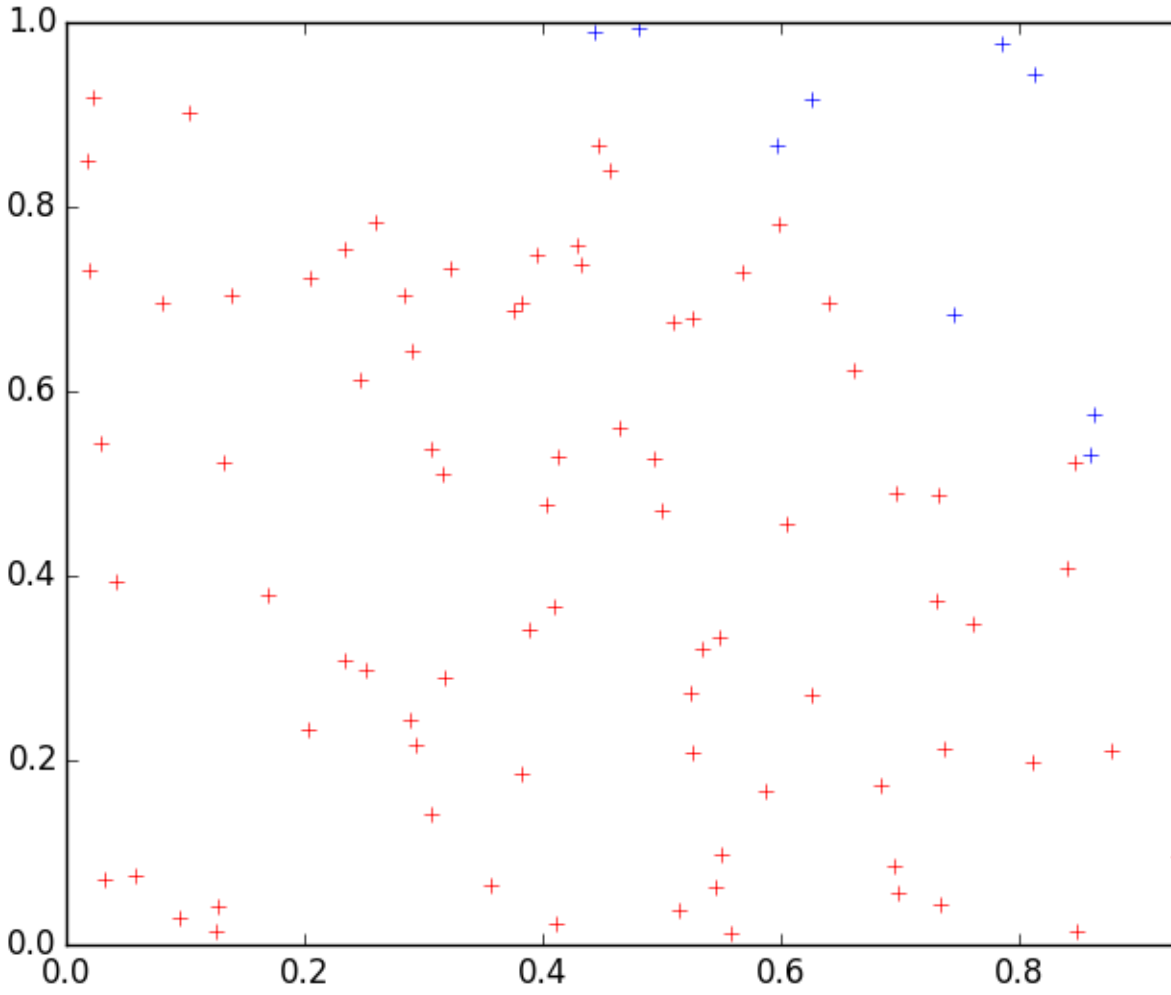
Thomas Goff
Middle Tennessee State University

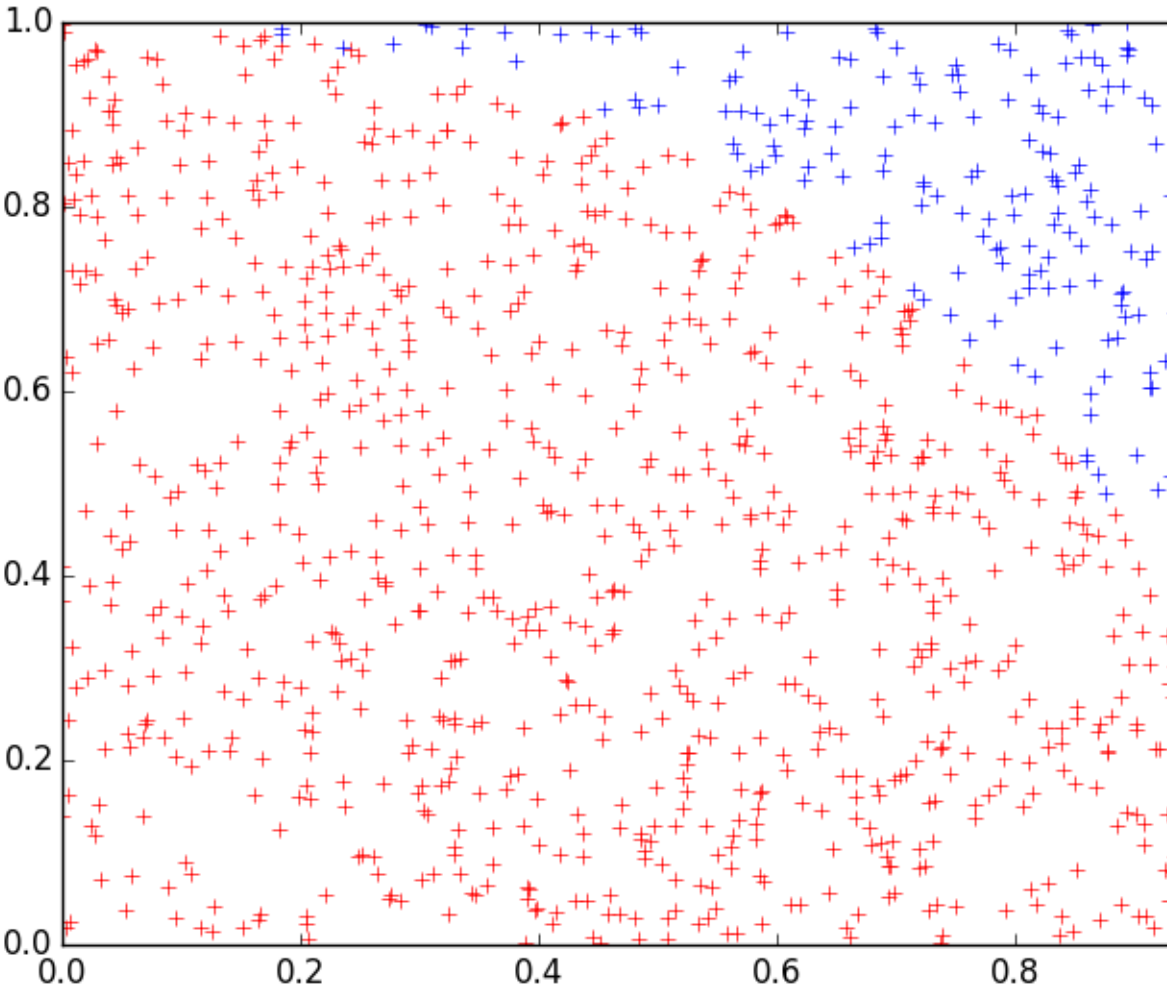
October 18, 2017

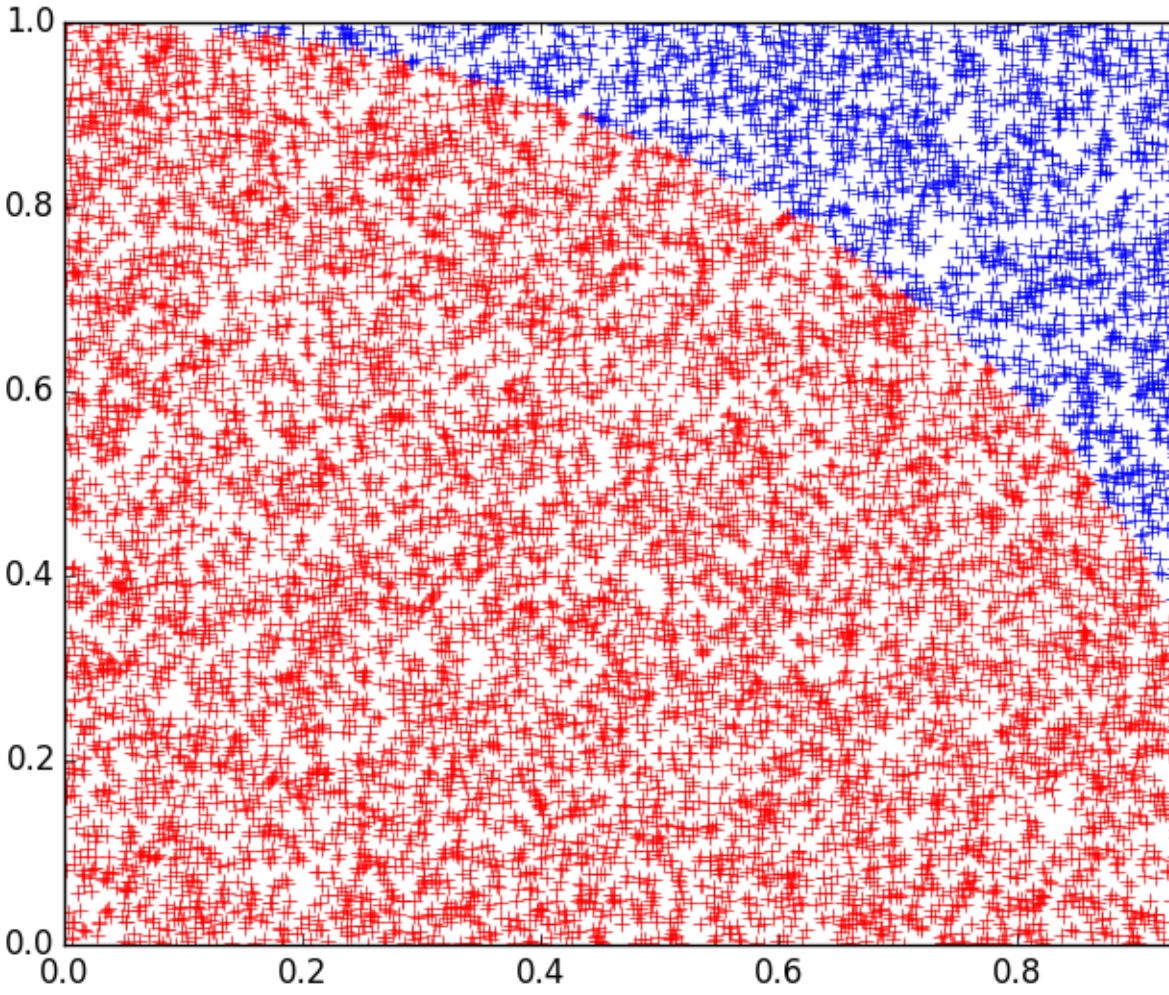
1 Python Implementation

```
import numpy as np
import matplotlib.pyplot as plt
import math

def approxpi(n):

piapprox = 0.0
x = np.array(np.random.random(n)) # array between 0-1 of size n
y = np.array(np.random.random(n))
xinside = []  # r < 1
xoutside = [] # r >= 1
yinside = []
youtside = []
# find out inside or outside
for i in range (1, n):
r = np.sqrt(np.square(x[i]) + np.square(y[i]))
if r < 1:
xinside.append(x[i])
yinside.append(y[i])
else:
xoutside.append(x[i])
youtside.append(y[i])

# plot the stuff
fname = "plot" + str(n) + ".png"
plt.plot(xinside,yinside, '+r')
```

```
plt.plot(xoutside,youtside, '+b')
plt.savefig(fname)
piapprox = 4 * ((1.0 * len(xinside)) / n)

return piapprox
def main():
pi1 = approxpi(100)
pi2 = approxpi(1000)
pi3 = approxpi(10000)
print "At 100 points, pi is approximately: " + str(pi1)
print "At 1000 points, pi is approximately: " + str(pi2)
print "At 10000 points, pi is approximately: " + str(pi3)

main()
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8



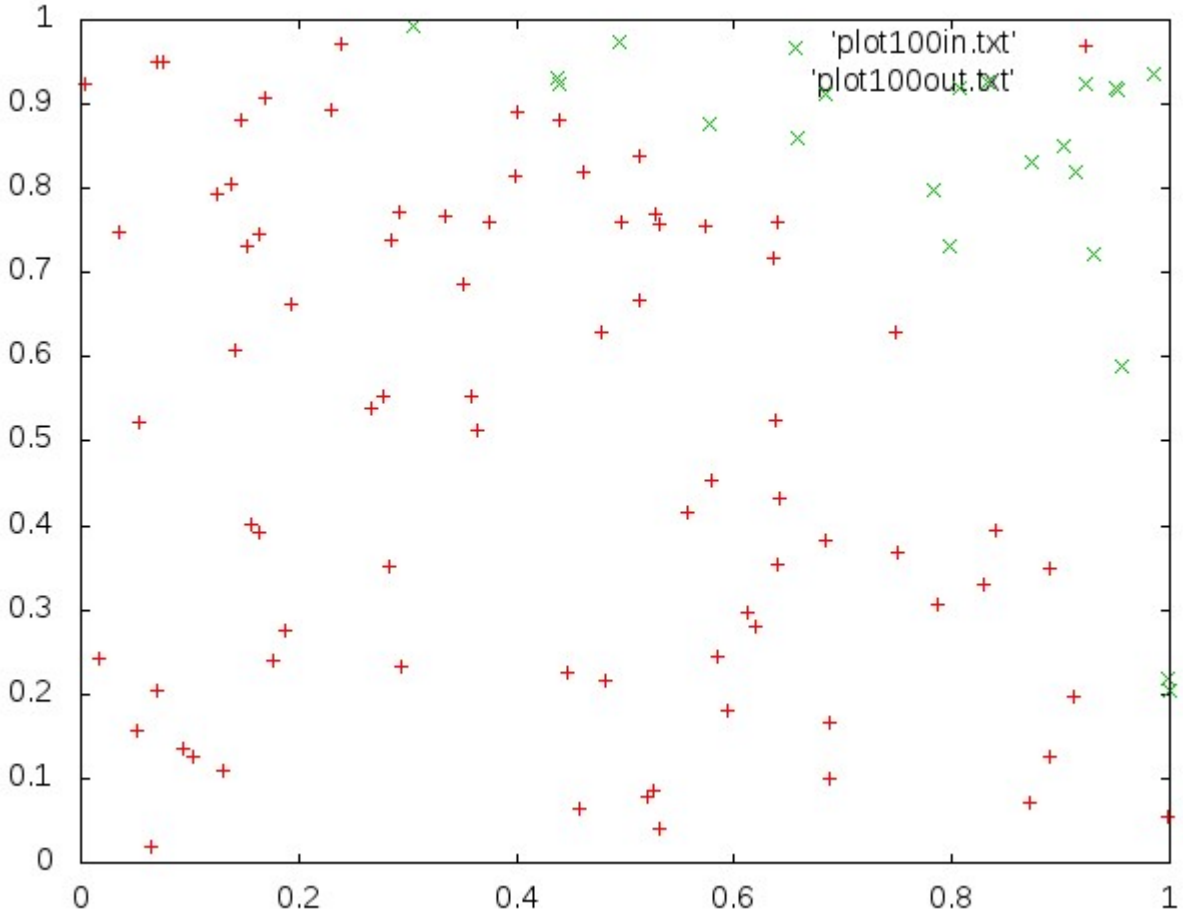
2 C++ Implementation

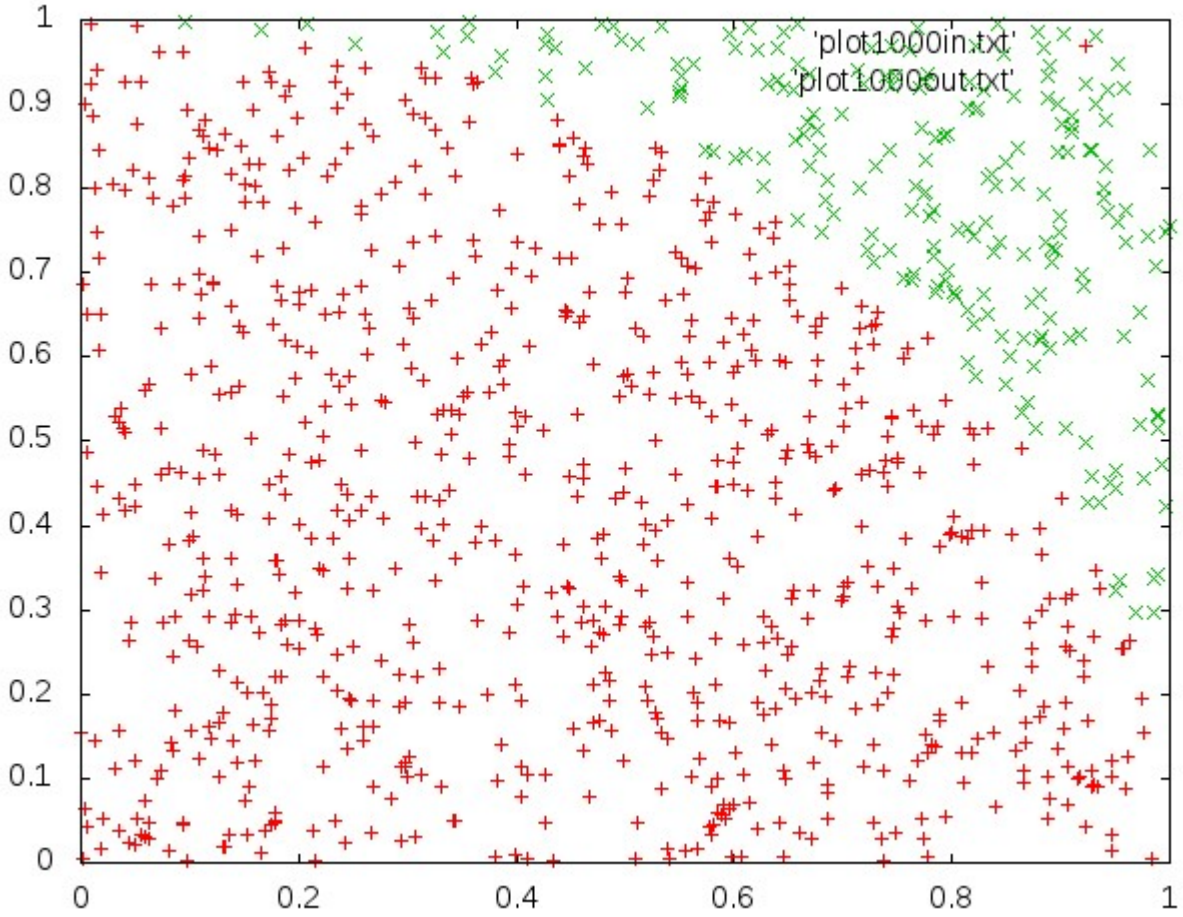
```
/* Thomas Goff COMS 6100 Dr John Wallin HW 10 10 October 2017 */

#include <iostream>
#include <fstream>
```

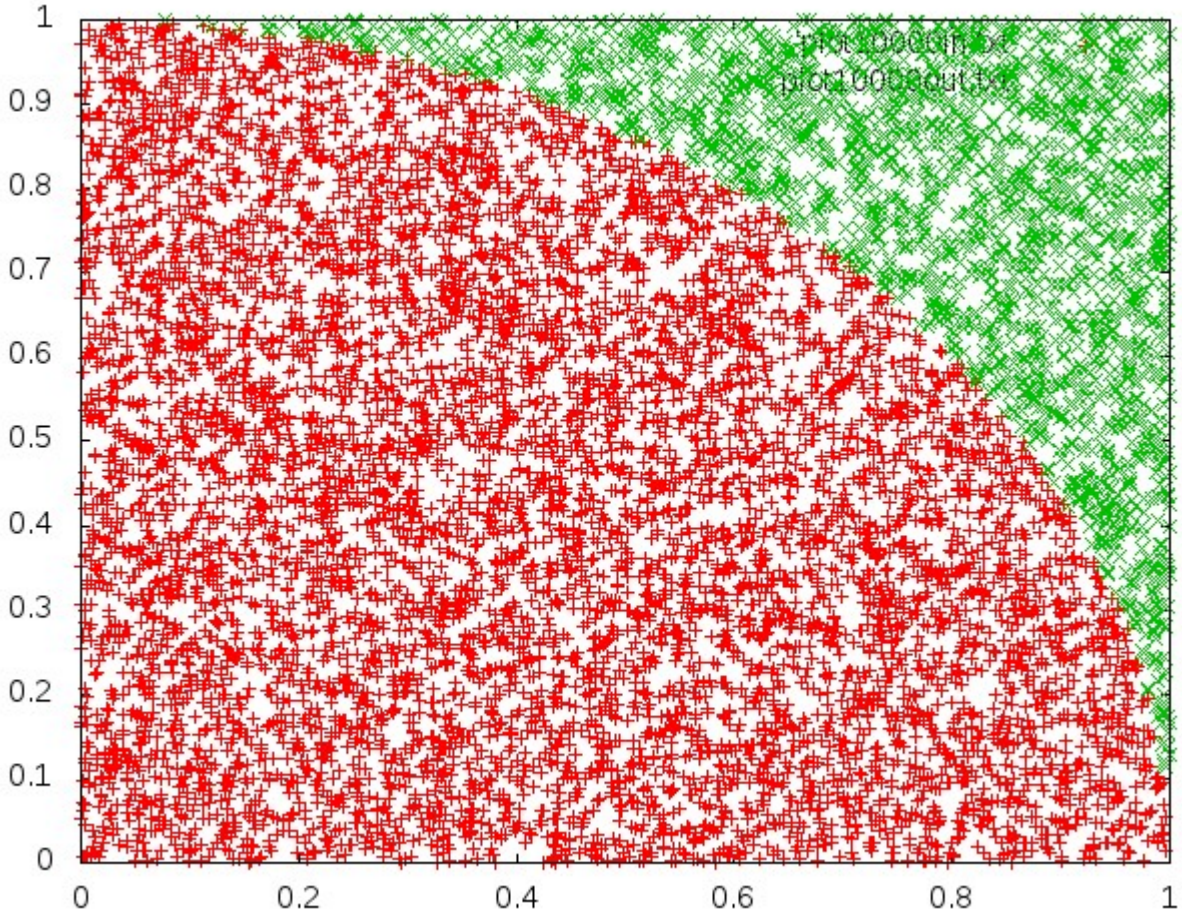
```
#include <cmath>
#include <string>
#include <cstdlib>
#include <stdlib.h>

/* Prototypes */

double approxpi(int n);

/* Main */

int main() {
/* Global declarations */
int nArray[3] = {100, 1000, 10000};
double piapprox[3];

/* Call the function and display data */
for (int i = 0; i < 3; i++)
{
piapprox[i] = approxpi(nArray[i]);
std::cout << "pi using " << nArray[i] << " points: " << piapprox[i] << std::endl;
}
return 0;
}

/* Function definitions */

double approxpi(int n) {
double piapprox = 0.0; // double for pi approximation returned  by function
double *xinside = new double[n]; // dynamically declaring arrays needed
double *yinside = new double[n];
double *xoutside = new double[n];
double *youtside = new double[n];
double *x = new double[n];
double *y = new double[n];
double r = 0;
double r2 = 0;
int incount = 0; // keep up with number of points inside r
int outcount = 0; // keep up with outside points
// generate x and y arrays of random numbers
```

```
for (int i = 0; i < n; i++)
{
x[i] = std::rand() / (float)RAND_MAX; // between 0 and 1
y[i] = std::rand() / (float)RAND_MAX;

}

// calculate r and populate insides and outsides
for (int i = 0; i < n; i++)
{
r2 = std::pow(x[i],2) + std::pow(y[i],2); // r squared
r = std::pow(r2,0.5); // radius
if (r < 1)
{
xinside[incount] = x[i];
yinside[incount] = y[i];
incount++; // increase coutn on the out
}
else
{
xoutside[outcount] = x[i];
youtside[outcount] = y[i];
outcount++; // increase count on the items
}
}
piapprox = 4.0 * ((float)incount / ((float)incount + (float)outcount)); // calculate
// plot the data
std::string fname = "plot"; // deal with file naming issues
fname+=std::to_string(n);
fname+="in.txt";

std::ofstream myfile;
myfile.open(fname.c_str());
for (int i = 0; i < incount; i++)
{
myfile << xinside[i] << ", " << yinside[i] << "\n"; // write all data to file in corr
}
myfile.close();
// file name setting for outside radius of circle
std::string fname1 = "plot";
```

```
fname1+=std::to_string(n);
fname1+="out.txt";

std::ofstream myfile1;
myfile1.open(fname1.c_str());
for (int i = 0; i < outcount; i++)
{
myfile1 << xoutside[i] << ", " << youtside[i] << "\n";
}
myfile1.close();

/* form command for gnuplot and plot the data */

std::string gnucmd = "gnuplot -e \"set output 'plot";
gnucmd+=std::to_string(n);
gnucmd+=".jpg'; set term jpeg; set xrange[0:1]; plot 'plot";
gnucmd+=std::to_string(n);
gnucmd+="in.txt' w points, 'plot";
gnucmd+=std::to_string(n);
gnucmd+="out.txt' w points; exit \"";
// pass command off to gnu via the system
system(gnucmd.c_str());

/* cleanup */
delete [] xinside;
delete [] yinside;
delete [] xoutside;
delete [] youtside;
delete [] x;
delete [] y;

return piapprox;
}
```


'plot100in.txt'
'plot100out.txt'
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1


'plot1000in.txt'
'plot1000out.txt'
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1



3 Writeup

This task was very fun and interesting for me. Implementing the python code was very simple, and didn't require that many lines of code, whereas, including blank spaces and comments, the c++ implementation took almost twice as many lines. The main issues I ran into at first was not typecasting in python whenever I should have. The biggest issue that I had while implementing the c++ version of the assignment was making sure that all of the std functions that I wanted to use would be included. In order for my c++ code to compile, the -std=c++11 option must be given to g++, because of a certain function that I used while typecasting. Overall, the task was fun

in both languages, however, the python implemenation was much quicker to code. On the efficiency side though, the c++ implemenation should be able to execute much more quickly and even use larger values of n.