EECS 10: Assignment 5

Prof. Rainer Doemer

October 24, 2008

Due Monday 11/03/2008 12:00pm

1 Exercise 4.26 from the textbook, Page 149 [10 points]

Additional instructions: Your program should ask the user for the number of terms (say n) to be used in the series and should print the table in the following format, starting with number of terms to be 1 in the infinite series to n :

```
Please input the number of terms you want to calculate: 15
No Terms  |  approx. PI  |  actual PI  |  difference  | accuracy
   1      |  4.00000000  |  3.14159265 |  0.85840735  | 27.3240%
   2      |  2.66666667  |  3.14159265 |  0.47492598  | 15.1174%
   3      |  3.46666667  |  3.14159265 |  0.32507402  | 10.3474%
   .      |  ...         |  ...        |  ...         | ...
   .      |  ...         |  ...        |  ...         | ...
   .      |  ...         |  ...        |  ...         | ...
   15     |  ...         |  ...        |  ...         | ...
```

For “actual PI”, you may take the constant value shown above. For “accuracy”, simply compute the difference as a percentage of the actual PI value. Please use ’double’ floating-point values to implement this assignment.

The files that you should submit for this part of the assignment are:

- **pi.c**: the source code file.
- **pi.txt**: the brief text file to explain what the program does and why you chose your method of implementation.
- **pi.script**: the typescript file to show that your program works by displaying a table with 20 lines (n = 20).

2 Monte Carlo Calculation of Pi [30 points]

Monte Carlo (MC) methods are stochastic techniques, meaning they are based on the use of random numbers and probability statistics to investigate problems. In this part of the homework, you are asked to write a program to implement a simple geometric MC experiment which calculates the value of Pi based on a “hit and miss” integration.

The figure below shows a unit circle circumscribed by a square. The radius of the circle r equals to 1/2 of the side of the square. Furthermore, the center of the circle and the center of the square are identical.

Imagine we can throw points randomly at the above figure. If we can throw infinite random points, it should be apparent that of the total number of points that hit the circle divded by the the number of points that hit within the square is proportional to the area of that part.
In other words:

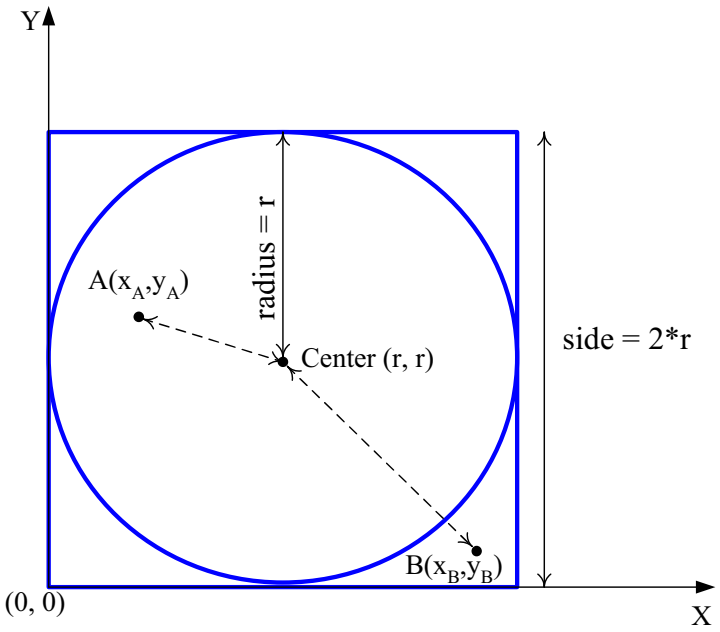

Figure 1: A Circle Circumscribed by a Square

$$\frac{number\ of\ points\ hitting\ circle\ area}{number\ of\ points\ hitting\ square\ area} = \frac{area\ of\ circle}{area\ of\ square} = \frac{\pi \times r \times r}{(2 \times r)^2} = \frac{\pi \times r \times r}{4 \times r \times r} = \frac{\pi}{4}$$

Therefore, we get the formula to calculate π using Monte Carlo method:

$$\pi = 4 \times \frac{number\ of\ points\ hitting\ circle\ area}{number\ of\ points\ hitting\ square\ area}$$

In the real world, we can only throw a finite number of random points, therefore, the π calculated using the above formula is an approximation of the exact value of π.

We can have our computer generate random numbers to simulate the throwing of points. For each point, we can have computer to generate two random floating point numbers to be the x and y coordinates of the point, where $0 \leq x \leq 2r$ and $0 \leq y \leq 2r$ so that (x,y) must fall within the square area. However, this randomly generated point could fall within the circle area or fall out of the circle area.

To decide if the randomly generated point (x,y) is within the area of the circle, we can compare the distance of the point to the center with the radius r. For example, the point A in the above figure is in the circle area since its distance to center is less than r. However, the point B in the above figure is not in the circle area since its distance to center is greater than r.

Note: If the distance of point to the center equals to radius r, then that point is considered within the circle area.

Assume the radius of the circle is r and the coordinates of the randomly generated point P is (x,y), then the distance of P to the center is:
$Distance(P,Center) = \sqrt{(x-r) \times (x-r) + (y-r) \times (y-r)}$

To avoid the square root calculation, you can compare $Distance(P,Center) \times Distance(P,Center)$ with the radius squared $r \times r$ in order to decide if the randomly generated point is within the circle area.

At the beginning, your program should ask for the input of radius *r* and the number of random points *N* the computer needs to generate. The output should like this:

```
Enter the radius of circle:  5
Enter the number of points:  10
```

During the generation, whenever a random point is generated, your program should print out the coordinates of the point and whether the point is *In* or *Out* the circle area like this:

```
Point No.1(x=0.000000,y=6.551714):  OUT
Point No.2(x=3.048189,y=6.749779):  IN
Point No.3(x=1.067537,y=5.165868):  IN
Point No.4(x=4.896695,y=6.024659):  IN
Point No.5(x=3.699454,y=2.566607):  IN
Point No.6(x=3.741874,y=8.255867):  IN
Point No.7(x=1.727042,y=2.977996):  IN
Point No.8(x=6.435438,y=7.896664):  IN
Point No.9(x=9.878231,y=8.005921):  OUT
Point No.10(x=4.642476,y=5.389874):  IN
```

At the end, your program should output the number of points within and out of the circle, together with the approximation value of π like this:

```
/******In Summary******/
Points within circle area:  8
Points out of circle area:  2
Pi= 3.200000
```

To show that your program works correctly, run it once with the radius = 10 and the number of points = 20. Submit the output as your script file (**mc.script**). Please compile your C code using **-ansi -Wall** options as below to specify ANSI code with all warnings:

```
gcc -o mc -ansi -Wall mc.c
```

The files that you should submit for this part of the assignment are:

- **mc.c**: the source code file.
- **mc.txt**: the brief text file to explain what the program does and why you chose your method of implementation.
- **mc.script**: the typescript file to show that your program works with the radius = 10 and the number of points = 20.

HINT In assignment 4, you have learned how to generate random integer numbers within the range of $[0,n)$ (n is exclusive). However, in this homework, you need to generate floating point numbers with the range of $[0,n]$ (n is inclusive). Therefore, there are some modification of the code described in homework4.

You need to replace the following line in assignment 4
int randomNumber = rand() % n; with
*double randomNumber = ((double)rand())/((double)RAND_MAX)*s;* /* s is the side of the square */

Furthermore, in assignment 5, we want you to use the same seed for the random numbers generation in order to generate the same series of random numbers. Therefore, take out the following line:
#include ⟨time.h⟩

and replace the following line in homework4

srand(time(NULL)); with
srand(0);

3 Bonus Problem [5 points]

Could we use the Monte Carlo method to calculate Pi given the figure2 below? It's a quadrant of a circle circumscribed by a square. The radius of circle equals to the side of square and their centers are overlapped.

If yes, explain your method and formula in the same text file of part 2 (**mc.txt**) concisely within 10 sentences.

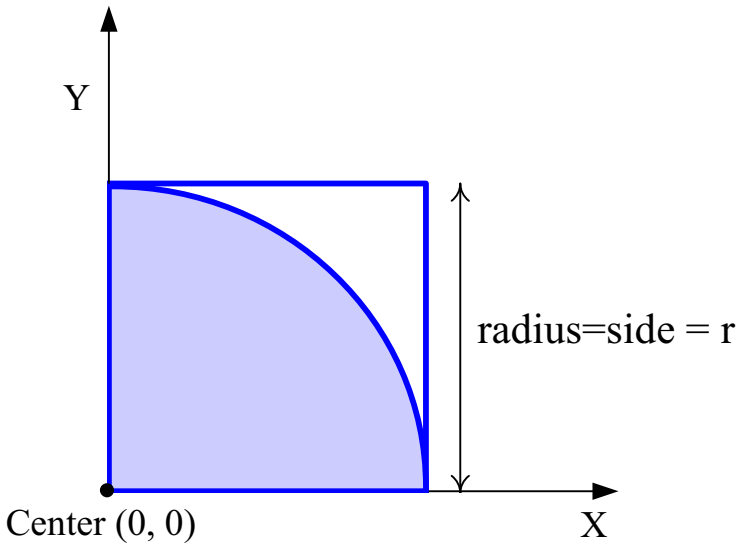

Figure 2: A Quadrant of a Circle Circumscribed by a Square

4 Submission

Submission for these files will be similar to last week's assignment. The only difference is that you need to create a directory called **`hw5/`**. Put all the files for assignment 5 in that directory and run the **`/ecelib/bin/turnin`** command to submit your homework.

Note: We do require the *exact* file names. If you use different file names, we will not see your files for grading. Also, please pay attention to any announcements on the course noteboard.