Mini Project 3: Monte Carlo Estimation of Pi (3 pages)

Rule 1. Submit an M-file named `mp3_ABC_WXYZ.m`, where `ABC` is your section number and `WXYZ` are the last 4 digits of your NJIT ID. Observe capitalization. Use an underscore _ instead of a dash - or MATLAB will get confused. **SUPPRESS output for all of your MATLAB commands.**
Rule 2. Send an email to the instructor `alexg+cs101@njit.edu`, to the grader `js87+cs101@njit.edu`, and if you want to make sure that everything went fine with the transmission of your mail, to yourself as well. Include the three words `mp2 ABC WXYZ` in the subject line, separated with space(s) if your id ends with `WXYZ` and you are enrolled in CS101-ABC.
Rule 3. It is imperative that you fully obey Rules 1 and 2. Testing will be done by running a MATLAB program (that will also be given to you afterwards). If you deviate from these rules you will be getting errors (and points deducted). Observe variable names and capitalization.
Due Date: Before noon time of Friday April XX, 2014.

1 Part A: Preliminary

In Figure 1 we have a circle C with center coordinates $(1/2, 1/2)$ of radius $1/2$ inscribed into a square S of side 1. The area of the square A_s is $A_s = 1 \times 1 = 1$ and the area A_c of the circle is $A_c = \pi \times (1/2)^2 = \pi/4$. The ratio of the area of the circle over the area of the square is $A_c/A_s = (\pi/4)/1 = \pi/4$; then $4A_c/A_s = \pi$. From now on we call this fraction $4A_c/A_s$ the `magic number` M i.e.

$$M = \frac{4A_c}{A_s} = \pi.$$

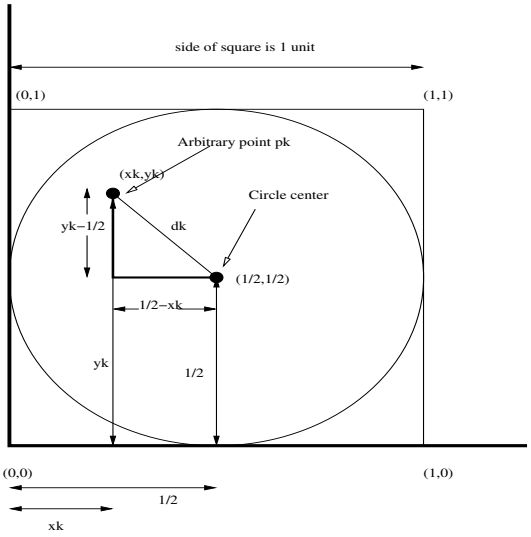

Figure 1: Inscribed circle (of radius 1/2) of a 1×1 square

A **Monte-Carlo** computation of π involves generating $N_s = N$ random points within square S and then counting how many of them fall within the inscribed circle C. The argument is that the number of points N_c within the circle, and the number of points N inside the square will be proportional to the corresponding areas A_c and A_s. Thus $N_c/N_s = N_c/N \approx A_c/A_s = \pi/4$ and thus $4N_c/N \approx \pi$. Thus

estimating π is reduced to the problem of (generating) points within square S and then counting points within circle C. We can describe this process in terms of the following steps.

Step 1. We generate N points p_k, $1 \leq k \leq N$, randomly so that they lie within S. Each point $p_k = (x_k, y_k)$ will have coordinates x_k, y_k such that $0 \leq x_k, y_k \leq 1$ since the square is of side 1 and its end points in the coordinate system are $(0,0), (0,1), (1,1), (1,0)$.

Step 2. For every such random square point p_k that will be generated, we then check whether it also lies within circle C. If the point lies on or is inside the circle we account for it separately using a counter C that counts how many points lie on or inside the circle, among the points of the square. Thus the value of C is the value of N_c and $C = N_c \leq N$.

Step 3. After all N points are generated and inspected we obtain the ratio $4 * C/N$. This becomes an estimate of the magic number M i.e. of π.

I. How to check whether p_k lies in/on circle $(x = 1/2, y = 1/2, r = 1/2)$.

Say we have generated a point within the square (with the help of MATLAB) and let that point be $p_k = (x_k, y_k)$. We then check whether that point lies within the circle. We can check whether this is so by considering the distance between p_k and the center of the circle that has coordinates $(x = 1/2, y = 1/2)$. If this distance is $1/2$, then the point lies on the circle. If the distance is less than $1/2$ it lies inside, and if it is greater it lies outside the circle. Note that p_k by default (we started the paragraph with a **Say**) is a square point and thus it lies within S in all three cases. The distance check is easy to perform. Let d_k be the distance of p_k from the center of the circle. Then d_k is given by

$$d_k = \sqrt{(x_k - 1/2)^2 + (y_k - 1/2)^2}$$

Thus depending on whether d_k is less than or equal to $1/2$, we increment C as needed.

II. Estimating π from C and N is easy: $4 * C/N$ is an estimate of M and π!

III. MATLAB logistics: random numbers and other things.

Function `rand` generates random numbers in MATLAB. In fact a single call to `rand` or `rand()` returns a single "random" real number in the interval $(0, 1)$. Thus calling rand twice, once for x_k and once for y_k, one could potentially generate a random point of the unit-square S. Not much else is needed other than perhaps using the `sqrt` function. It is imperative that you are careful with your implementation of the otherwise simple code. One reason for that is that the benchmarking that will be done by you and is described in the next section might be affected by your implementation choices. If you do not implement things properly you might risk of running out of memory, or output might get delayed (i.e. you wait in vain without getting answers)! For example you might generate all N points at once and then perform the check of Step 2. Or you might generate the points one at a time and perform the check after every point.

2 Part B: MATLAB implementation requirements

Part 1. Function `mp3_ABC_WXYZ`**.**

You will implement a **MATLAB function** named `mp3_ABC_WXYZ` (per Rule 1) stored in an M-file of an appropriate name. Function `mp3_ABC_WXYZ` will have one parameter denoted by `N` and will thus look like `mp3_ABC_WXYZ (N)`. Parameter `N` is the number of points that will be generated in the call to `mp3_ABC_WXYZ (N)`. You then implement the steps mentioned earlier collecting information into a counter variable C (note capitalization). **You must use a loop structure in your code.** (There is a way to implement this part using array operations only, but you must use an iterative MATLAB statement.) Then when you compute C you will print with an `fprintf` statement the values of N, C and $4 * C/N$ i.e. an estimate of π. Finally, function `mp3_ABC_WXYZ` will return to the calling program three values in the following explicitly stated order: C, N, and pi_es, where pi_es is the estimate of π as computed in this N point Monte-Carlo simulation. It is up to you to properly structure function `mp3_ABC_WXYZ` to return C, N, pi_es. The output that this function will print is as follows (note the order).

```
mp3_ABC_WXYZ: N= 123456789 C=  96944443 estimate= 3.1412
```

Things to note for the output. One space precedes `N= C= estimate=`, and the `=` follows the preceding letter immediately and without any space inbetween. (Note that there is an `estimate` here not `pi_estimate`.) The output for N should accommodate exactly 10 digits. And so will C. The estimate for π should print 4 decimal digits, the decimal point and exactly one integer digit but will allow for an extra space between the `=` of `estimate=` and the left-most integer digit of the approximation. Note that in the example above C is an 8-digit number. Two spaces appear after the equal sign of `C=` to accommodate the 10-digit requirement. Note that for `pi_estimate` a value of `3.1412` is printed and a space appears before the `3` and after the preceding `=` character. Make sure you also print correctly your section number `ABC` and the last four digits `WXYZ` of your NJIT ID. Make sure that you use matrices and arrays wisely. A test run on $N = 1,000,000$ or more points might be used in grading. A minimal submission submits a MATLAB M-file appropriately named `mp3_ABC_WXYZ.m`.

First line. The first line of the M-file contains in the form of a MATLAB comment that this is miniproject `MP3` of section `CS 101-ABC`. You then write your first name in lower-case (all characters) and your last name in upper-case (all characters) and then the last four digits of your NJIT ID. Thus if i was to write this first line myself I would have written something akin to

```
% MP3 CS 101-104 : Alex GERBESSIOTIS  WXYZ
```

Second line. A empty line that contains nothing.

3 Grading

Of the 60 points,

- the proper formulation of the first 2 lines will earn you up to 10 points,
- the correct implementation of `mp3_ABC_WXYZ` will earn you up to 29 points (9 points is the appropriate ordering of return values),
- the proper naming, formatting of the output generated by `mp3_ABC_WXYZ` will earn you up to 10 points,
- the proper and correct use of iterative statement(s) and code efficiency will earn you the remaining points.