Randomized Algorithms: Assignment 1

Jane Doe [*1]

1. Throwing Darts

In this first exercise we compute an approximation to the value of π by taking advantage of the fact that the area of a circle inscribed in a unit square is $\frac{\pi}{4}$. We can generate random points in the unit square and check if they fall inside the inscribed circle. The proportion of points that do should be approximately equal to $\frac{\pi}{4}$.

Any point $P = (P_x, P_y)$, has its distance to the center of the circle $C = (C_x, C_y)$ with radius r given by

$$d(P, C) = \sqrt{(P_x - C_x)^2 + (P_y - C_y)^2}$$

Such a point is inside the circle iff $d(P, C) < r$. This can be simplified by the fact that $d(P, C)^2 < r^2$, and so we can define

$$d'(P, C) = d(P, C)^2 = (P_x - C_x)^2 + (P_y - C_y)^2$$

1.1. Implementation

For this specific scenario, the circle $C_0 = (0.5, 0.5)$ inscribed in a unit square has the center coordinates $(0.5, 0.5)$ and radius 0.5. Thus, if we take the random variables $X, Y \sim U[0, 1]$, and the point $P = (X, Y)$ we see that

equation omitted

We can further simplify the computation of the distance by considering $X' = X - 0.5$ and $Y' = Y - 0.5$, and so $X', Y' \sim U[-0.5, 0.5]$. We find that

equation omitted

The high-level description of our algorihm to run the simulation is given in Algorithm 1, but our implementation uses numpy vectors instead of for explicit **for** loops, so it can take advantage of low level SIMD optimizations. A $(2, N)$ vector is initialized with random values following a uniform distribution [additional explanation omitted here] as in the algorithm above. More explanations here ... The full Python listing can be found in Appendix A at the end of this report.

Algorithm 1 Dart Throwing

```
function THROW_DARTS(N)
    in_count ← 0
    for i ← 1, N do
        x ← ...; y ← ...; dist ← ...
        if ... then
            in_count ← in_count + 1
        end if
    end for
    return 4 * in_count
end function
```

The value of the approximation P_N is plotted in Figure 1, together with the true value of π (dashed red line). In fact, for the experiment we fix a very large of N ($N = \ldots$) and collect the current value of P_i at designated intermediate values, e.g., when i is a multiple of 100.

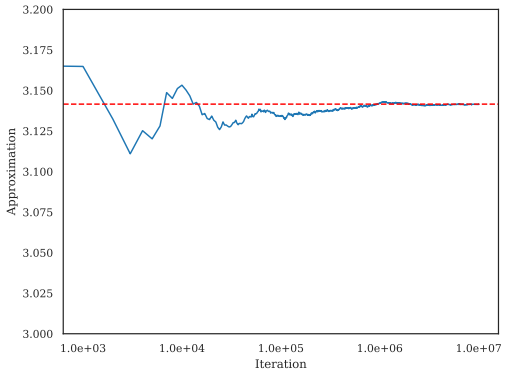

Figure 1. Convergence for the approximation P_N. X axis is in log scale. The red dashed line is the true value of π.

Next we plot, again in logscale, the evolution of the relative error

$$\epsilon_N = \left| \frac{\pi - P_N}{\pi} \right|$$

between π and its approximation P_N.

Table 2 in Appendix B collects some representative values of P_N as well as the relative errors ϵ_N.

Notice that once $N \geq 100$ the relative error is never above ...

[1]A student at Universitat Politcnica de Catalunya, Barcelona, Spain. Correspondence to: Jane Doe <jane.doe@estudiantat.upc.edu>.