Lec 04: FOR-Loops

Approximating π

Suppose the circle $x^2 + y^2 = n^2$, $n \in \mathbb{N}$, is drawn on graph paper.

- The area of the circle can be approximated by counting the number uncut grids, N_{in}.

  $$\pi n^2 \approx N_{\text{in}},$$

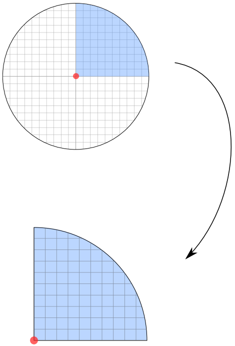
  and so

  $$\pi \approx \frac{N_{\text{in}}}{n^2}.$$

- Using symmetry, may only count the grids in the first quadrant and modify the formula accordingly:

  $$\pi \approx \frac{4N_{\text{in},1}}{n^2},$$

  where $N_{\text{in},1}$ is the number of inscribed grids in the first quadrant.



Approximating π

Problem Statement

Write a script that inputs an integer n and displays the approximation of π by

$$\rho_n = \frac{4N_{\text{in},1}}{n^2},$$

along with the (absolute) error $|\rho_n - \pi|$.

Note. The approximation gets enhanced and approaches the true value of π as $n \to \infty$.

Strategy: Iterate

The key to this problem is to count the number of uncut grids in the first quadrant programmatically.

Set $N_{\text{in},1} = 0$.

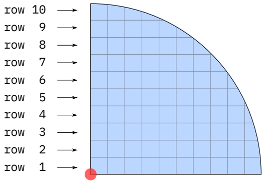
> Count the number of uncut grids in `row 1`. Add that to $N_{\text{in},1}$.

> Count the number of uncut grids in `row 2`. Add that to $N_{\text{in},1}$.

$\vdots$

> Count the number of uncut grids in `row 10`. Add that to $N_{\text{in},1}$.

Set $\rho_{10} = 4N_{\text{in},1}/10^2$.



MATLAB Way

The repeated counting can be delegated to MATLAB using `for`-loop. The procedure outlined above turns into

Assume `n` is initialized and set $N_{\text{in},1}$ to zero.

`for k = 1:n`

> Count the number of uncut grids in `row k`. Add that to $N_{\text{in},1}$.

`end`

Set $\rho_{10} = 4N_{\text{in},1}/10^2$.

Counting Uncut Tiles

The problem is reduced to counting the number of uncut grids in each row.

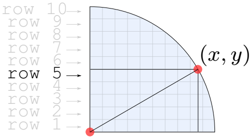
- The x-coordinate of the intersection of the top edge of the kth row and the circle $x^2 + y^2 = n^2$ is

$$x = \sqrt{n^2 - k^2}.$$

- The number of uncut grids in the kth row is the largest integer less than or equal to this value, *i.e.*,

$$\lfloor \sqrt{n^2 - k^2} \rfloor. \qquad \text{(floor function)}$$

- MATLAB provides `floor`.



For $n = 10$ and $k = 5$:

$$\begin{aligned} x &= \sqrt{n^2 - k^2} \\ &= \sqrt{10^2 - 5^2} = 8.6602\ldots \end{aligned}$$

Main Fragment Using FOR-Loop

```
N1 = 0;
for k = 1:n
    m = floor(sqrt(n^2 - k^2));
    N1 = N1 + m;
end
rho_n = 4*N1/n^2;
```

Exercise. Complete the program.

Exercise 1: Overestimation

Question

Note that ρ_n is always less than π. If N_1 denotes the total number of grids, both cut and uncut, within the quarter disk, then $\mu_n = 4N_1/n^2$ is always larger than π. Modify the previous (complete) script so that it prints ρ_n, μ_n, and $\mu_n - \rho_n$.

- `ceil`, an analogue of `floor`, is useful.

Notes on FOR-Loop

- The construct is used when a code fragment needs to be repeatedly run. The number of repetition is known in advance.

```
for <loop variable> = 1:<arithmetic expression>
    <code fragment>
end
```

- Examples:

```
for k = 1:3
    fprintf('k = %d\n', k)
end
```

```
nIter = 100;
for k = 1:nIter
    fprintf('k = %d\n', k)
end
```

Caveats

Run the following script and observe the displayed result.

```
for k = 1:3
    disp(k)
    k = 17;
    disp(k)
end
```

- The loop header `k = 1:3` guarantees that `k` takes on the values 1, 2, and 3, one at a time even if `k` is modified within the loop body.

- However, it is a recommended practice that the value of the loop variable is *never* modified in the loop body.

Simulation Using `rand`

`rand` is a built-in function which generate a (uniform) “random” number between 0 and 1. Try:

```
for k = 1:10
    x = rand();
    fprintf('%10.6f\n', x);
end
```

Let’s use this function to solve:

Question

A stick with length 1 is split into two parts at a random breakpoint. *On average*, how long is the shorter piece?

Program Development – Single Instance

Consider breaking *one* stick.

- Random breakage can be simulated with `rand`; denote by $x \in (0, 1)$.
- The length of the shorter piece can be determined using `if`-construct; denote by $s \in (0, 1/2)$.

```
x = rand();          % x: the location of breakage
if x <= 0.5          % if x ≤ 0.5
    s = x;           % shorter part has length x
else                 % otherwise
    s = 1-x          % shorter part has length 1 − x
end
```

Program Development – Multiple Instances

- Repeat the previous multiple times using a `for`-loop. Pseudocode: if 1000 breaks are to be simulated:

```
nBreaks = 1000;
for k = 1:nBreaks
    <code from previous page>
end
```

- But how are calculating the *average* length of the shorter pieces?

Calculating Average Using Loop

Recall how the total number of uncut grids were calculated using iterations.

Assume `n` is initialized and set $N_{\text{in},1}$ to zero.

```
for k = 1:n
```

> Count the number of uncut grids in `row k`. Add that to $N_{\text{in},1}$.

```
end
```

The value of $N_{,1}$ is the total numbers of uncut grids.

Similarly, we can compute an average by:

Assume `n` is initialized and set s to zero.

```
for k = 1:n
```

> Simulate a break and find the length of the shorter piece. Add that to s.

```
end
```

Set $s_{\text{avg}} = s/\mathtt{n}$.

Complete Solution

```
nBreaks = 1000;
s = 0;
for k = 1:nBreaks
    x = rand();
    if x <= 0.5
        s = s + x;
    else
        s = s + (1-x);
    end
end
s_avg = s/nBreaks;
```

Exercise 2: Game of 3-Stick

Game: 3-Stick

Pick three sticks each having a random length between 0 and 1. You win if you can form a triangle using the sticks; otherwise, you lose.

Question

Estimate the probability of winning a game of 3-Stick by simulating one million games and counting the number of wins[a].

[a] Of course, divide it by 1,000,000 to calculate the probability