ICMA393 Discrete Simulation: HW2

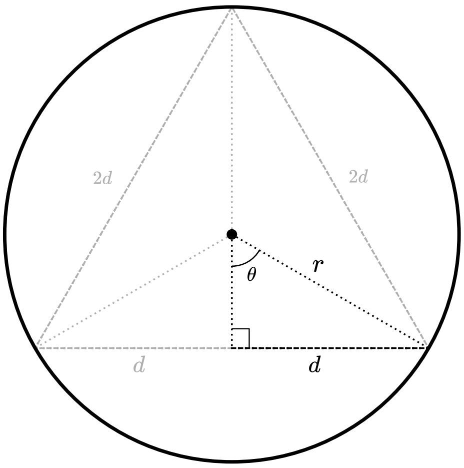
Mahidol University International College
Austin Jetrin Maddison

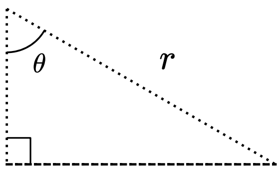
October 9, 2024

1. Archimedes Method of N-Gon

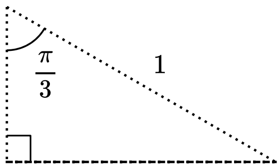
a)

Calculate p_3





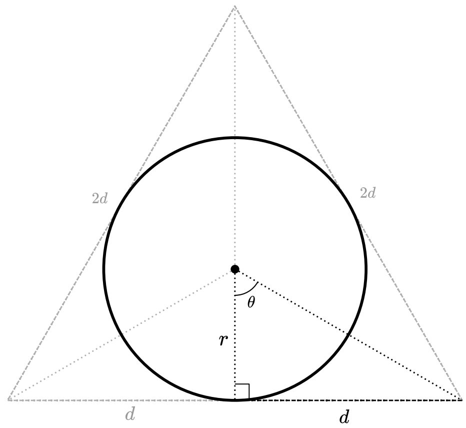
$$d = sin(\theta) \cdot r$$

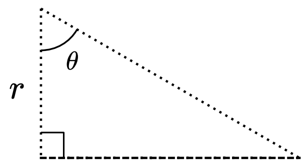


$$d = \sqrt{3}/2$$

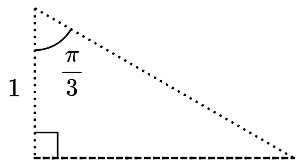
$$\begin{aligned} p_3 &= 2d + 2d + 2d \\ &= 6d \\ &= 6 \cdot \frac{\sqrt{3}}{2} \\ &= 3\sqrt{3} \end{aligned}$$

Calculate P_3





$$d = tan(\theta) \cdot r$$



$$d = \sqrt{3}$$

$$\begin{aligned} P_3 &= 2d + 2d + 2d \\ &= 6d \\ &= 6 \cdot \sqrt{3} \\ &= 6\sqrt{3} \end{aligned}$$

2. Approximate $\sqrt{2}$

a)
Summed all the n-terms of the taylor expansion of $\sqrt{2}$.

```
def f(n=150):
    x = 2
    sum = 0
    an = 1
    bn = 1/2

    for i in range(n):
        term = (an * (x - 1) ** bn) * (x - 1) **
             i / math.factorial(i)
        sum += term

        an *= bn
        bn -= 1

    return sum
```

$f()$ = 1.4142909169379279

b) Approximate $\sqrt{2}$ by uniformly sampling random numbers and checking whether their squares are less than 2.

```
def f(n = 1000000, x=2):
    xs = np.random.ranf(n) * x
    ys = xs**2
    count = ys <= x
    res = np.mean(count) * x
    std = np.std(count) * x

    calc_ci = lambda x, std, n, z = 1.96 : [x -
        z * (std / np.sqrt(n)), x + z * (std /
        np.sqrt(n))]
    return res, calc_ci(res, std, n)
```

$f()$ = 1.4142909169379279
CI = [1.4135070094100637, 1.4170729905899362]

3. Generalized Monty Hall

a)

```
def f(N, max_doors=3, max_cars=1, switch=True):
    max_cars = min(max_cars, max_doors)
    correct = np.zeros(N, dtype=bool)
    doors = np.arange(max_doors)

    for i in range(N):
    cars = np.random.choice(doors, size=max_cars
        , replace=False)
    my_choice = [np.random.randint(0, max_doors)
        ]
    available_doors = np.setdiff1d(doors, np.
        union1d(cars, my_choice))

    host_open = np.random.choice(available_doors
        , size=1, replace=False)

    if switch:
        remaining_doors = np.setdiff1d(doors,
            host_open )
        remaining_doors = np.setdiff1d(
            remaining_doors, my_choice)
        my_choice = np.random.choice(
            remaining_doors, size=1, replace=
            False)

    correct[i] = my_choice in cars

    return np.mean(correct)
```

f(`500000, max_doors=5, max_cars=2`)
= 0.5333

Source Code

https://github.com/AustinMaddison/discrete-simulation/tree/main/hw2/source