Gabe©
GabeL@virginia.edu
https://gabriel-vzh2vs.github.io/SYS3062-Website/
UVA's Systems and Information Engineering

January 20th, 2026

SYS 3062: Lab 1

Darts on a Board: Monte Carlo

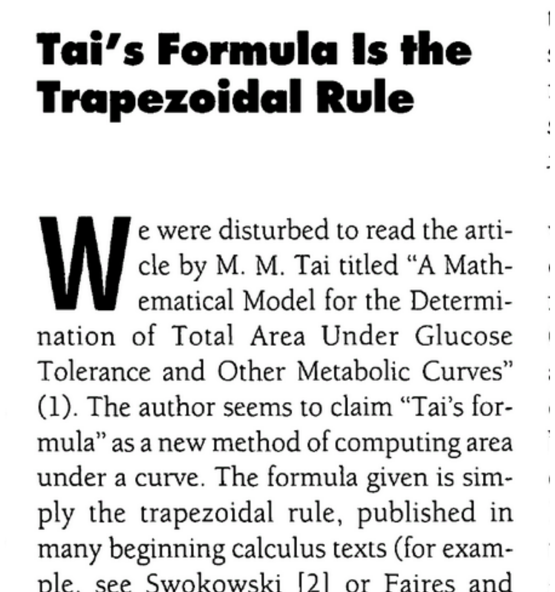
Why are we Learning?

One of the reasons is so that this does not happen to you!

> *"We have developed a novel mathematical model using trapezoids for estimating the area under curves from various metabolic studies", 1993: A Mathematical Model for the Determination of Total Area Under Glucose Tolerance and Other Metabolic Curves.*
>
> Mary M. Tai, MS, EDD

Hmmm, this might sound familiar to some of the audience.

The Punchline



Tai's Formula Is the Trapezoidal Rule

We were disturbed to read the article by M. M. Tai titled "A Mathematical Model for the Determination of Total Area Under Glucose Tolerance and Other Metabolic Curves" (1). The author seems to claim "Tai's formula" as a new method of computing area under a curve. The formula given is simply the trapezoidal rule, published in many beginning calculus texts (for example, see Swokowski [2] or Faires and

Thought Exercise

Imagine you're interning at a poorly-ran major automotive parts manufacturer. You're tasked with modelling and optimizing a new manufacturing line that produces electric vehicle battery components.

However, you wonder how are you going to model this system in the first place, as it has a ton of moving parts:

- Assembly robot cycle times (varying between 45-65 seconds)
- Quality inspection pass rates (92-98%)
- Component supply delivery times (20-40 minutes)
- Worker shift changeover impacts (efficiency drops 5-15% per switch over)

Okay, but what is Monte Carlo?

Definition

A **Monte Carlo Method** is a set of numerical techniques used to solve complex systems (deterministic or stochastic) by evaluating them by sampling from random variables.

Applicable Systems:

- Integrals and Differential Equations
- Complex Queuing Networks

Key Mechanics

- **Randomness:** Uses pseudo-random number generation to sample from probability distributions.
- **Convergence:** Relies on the *Law of Large Numbers*, as we increase the number of samples, the result converges to the parameter's value.

History: Origins & Insight

1777: The Early Precursor

- **Georges-Louis Leclerc, Comte de Buffon** proposed the famous "Needle Experiment."
- **The Experiment:** Estimating π by dropping needles on a lined surface.
- **Significance:** It was the first demonstration that *random sampling* could be used to solve deterministic mathematical problems.

1946: The Modern Insight

- **Stanislaw Ulam** (mathematician at Los Alamos) was recovering from illness and playing solitaire.
- **The Question:** What is the probability of winning a game?
- **The Realization:** Instead of calculating complex combinatorics, it was easier to simply *play 100 games* and count the wins.

History: The Manhattan Project Era

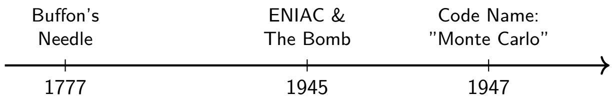
Catalysts for Development (1945-1947)

Monte Carlo flourished due to two events:

1. The need to model neutron diffusion for the atomic bomb.
2. The invention of the ENIAC (the first electronic computer), which made thousands of simulations possible.

Why "Monte Carlo"?

- The name was coined by physicist Nicholas Metropolis.
- It was a code name inspired by Ulam's uncle, who frequently borrowed money to gamble at the Monte Carlo Casino in Monaco.



Boom

Lab Assignment Overview: Computing π

Objective

To estimate the value of the mathematical constant π using a stochastic (random) approach rather than traditional deterministic formulas.

Key Concepts:

- **Monte Carlo Method:** Using random sampling to solve numerical problems.
- **Geometric Probability:** Relating area to the likelihood of a random point.
- **Law of Large Numbers:** As we increase the number of trials (N), our estimate converges to the true value.

Today, we will simulate throwing thousands of "darts" to find π.

Step 1: The Geometry

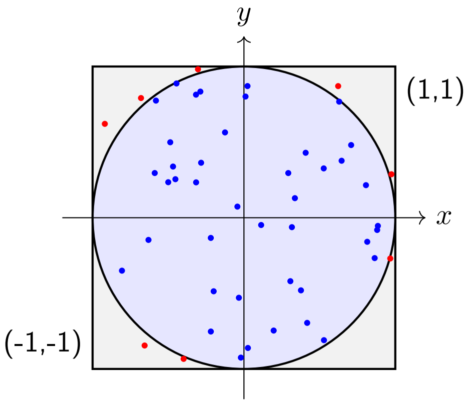
The Setup:

- Imagine a square with side length $a = 2$.
- Inscribe a circle with radius $r = 1$.
- Area of Square $= 2 \times 2 = 4$.
- Area of Circle $= \pi r^2 = \pi(1)^2 = \pi$.

The Logic:

$$\frac{\text{Area}_{\text{circle}}}{\text{Area}_{\text{square}}} = \frac{\pi}{4}$$

$$\therefore \pi = 4 \times \left(\frac{\text{Hits in Circle}}{\text{Total Throws}} \right)$$



Step 2: Algorithm Design

Before coding, we should define the logic flow used to approximate the ratio $\pi/4$.

Algorithm 1 Monte Carlo π Estimation

```
1:  $N \leftarrow$ Total number of darts
2:  $count\_inside \leftarrow 0$
3:  for $i = 1$ to $N$ do
4:      $x \leftarrow$ random number between $-1$ and $1$
5:      $y \leftarrow$ random number between $-1$ and $1$
6:      $distance \leftarrow \sqrt{x^2 + y^2}$
7:      if $distance < 1$ then
8:          $count\_inside \leftarrow count\_inside + 1$
9:      end if
10: end for
11: $Ratio \leftarrow count\_inside/N$
12: $\pi_{estimate} \leftarrow 4 \times Ratio$
```

Step 3a: Throwing the Dart

First, we need a function to generate random coordinates (x, y) within our square boundaries $[-1, 1]$.

Function: `throw_dart()`

```python
import random

def throw_dart():
    # Scale random(0,1) to range (-1,1)
    # Formula: -1 + 2 * random.random()

    x = -1 + 2 * random.random()
    y = -1 + 2 * random.random()

    return (x, y)
```

Step 3b: Checking the Landing

Next, we use the distance formula (Pythagorean theorem) to check if the dart landed inside the unit circle.

Function: `is_within_circle()`

```python
from math import sqrt

def is_within_circle(x, y):
    # Calculate distance from origin (0,0)
    dist = sqrt(x**2 + y**2)

    if dist < 1:
        return True
    else:
        return False
```

Step 4: The Simulation Loop

We run the experiment N times. As N increases, the error decreases (Law of Large Numbers).

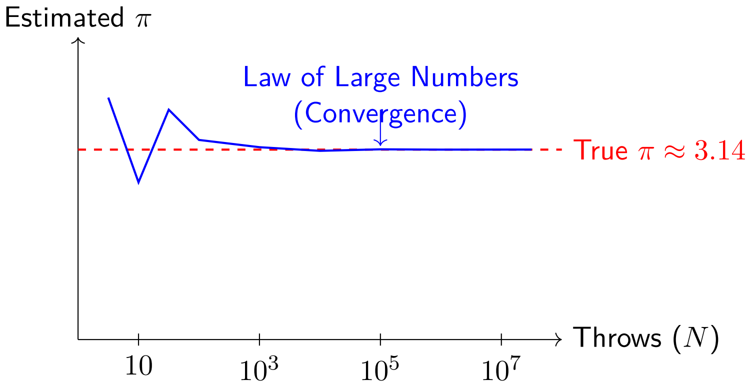
```python
def compute_pi(n_throws):
    count_inside = 0
    for i in range(n_throws):
        r1, r2 = throw_dart()
        if is_within_circle(r1, r2):
            count_inside += 1

    return 4 * (count_inside / n_throws)
```

Example Results:

- $N = 10 \to \pi \approx 4.0$ (High Error)
- $N = 1,000 \to \pi \approx 3.148$
- $N = 1,000,000 \to \pi \approx 3.1415$ (Converged)

Step 5: Visualizing Convergence

The accuracy improves as we increase the number of "darts" (N).



Your Turn

Now, it's your turn to implement Monte Carlo in Python or using XLRisk! Good Luck!

GabeL@virginia.edu

Feedback Form

Throughout the course, I may ask for your feedback on:

- **Course Materials**
- **Lecture & Lab**
- **Assignments**

Action Items

First, I will thank you all again for coming to lab at 5 pm on a Tuesday!

And now, you need to do the following tasks:

- **Submit** your Python file to Gradescope by next week's lab!
- **Read** Prelab 2 for next week!

It's going to be an interesting semester!