CS 1112 Introduction to Computing Using MATLAB

Instructor: Dominic Diaz

Website:
https://www.cs.cornell.edu/courses/cs1112/2022fa/

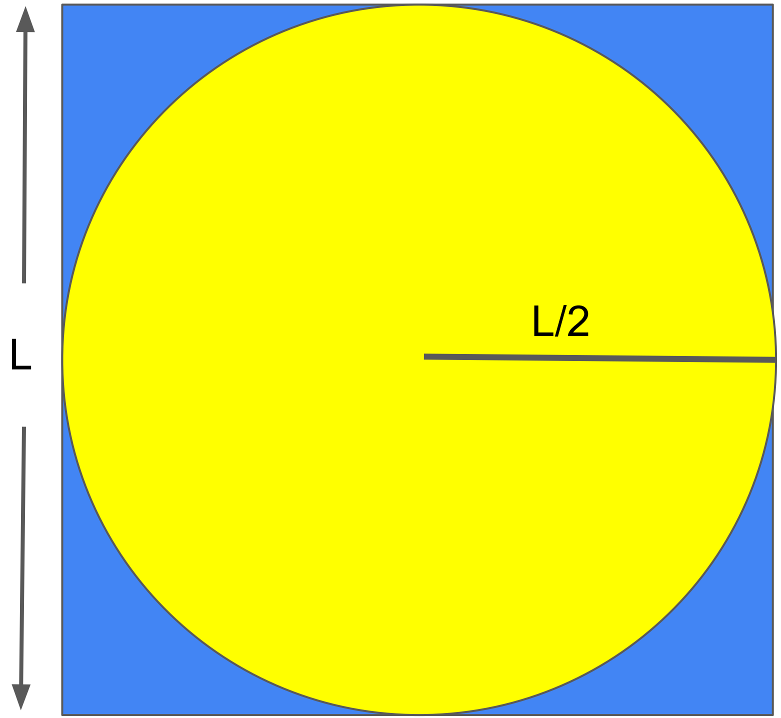
Today: More for loops and while loops

Agenda and announcements

- Last time
  - `for` loops
- This time
  - More `for` loops and `while` loops
- Announcements
  - Project 2 will be released Friday or Saturday (due 9/19)
    - Partner matching survey for P2 is posted (*only submit if you need a partner*)
  - We do not use `break` or `continue` in this course
  - Come to office/consulting hours to get help! Or sign up for tutoring via CMS (Sunday-Tuesday). You want to have a firm foundation now in order to build on it.

Example: Monte Carlo approximation of π

Throw N darts uniformly on square dart board with an inscribed circle.
What is the probability of landing in circle, P_{in}?



Monte Carlo method: Approximate a quantity by relating it to a probability that can be estimated using simulations

Probability 101

What is the probability that Dominic walks in to lecture listening to Bad Bunny?

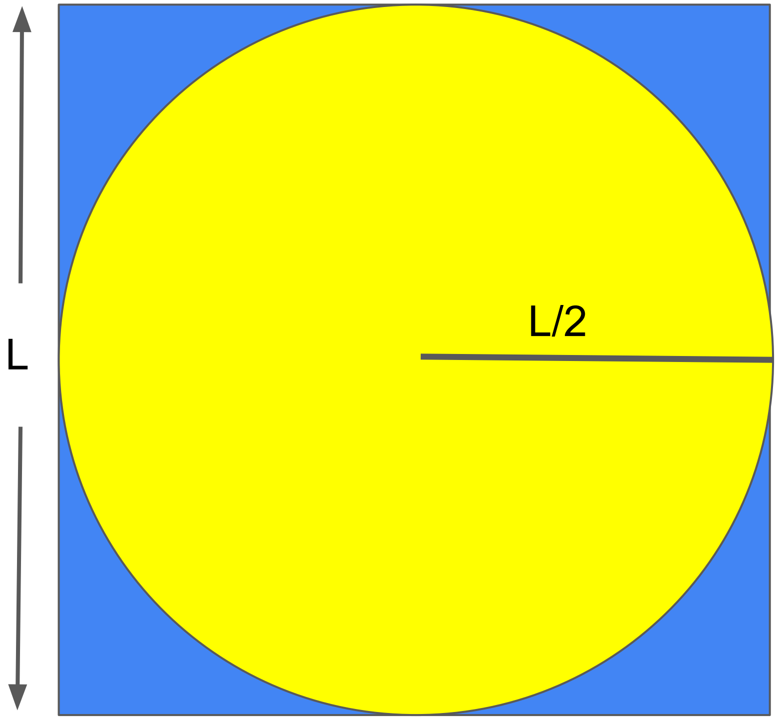
- Dominic walked into lecture 6 different days
- Was listening to Bad Bunny while walking in 2 different days
- I estimate $P_{\text{listenBadBunny}} = 2/6 = 1/3$

$$\text{Probability of an event E} = \frac{\text{Number of times E happened in the past}}{\text{Number of trials}}$$

Estimating π

Throw N darts uniformly on square dart board with inscribed circle.

What is the probability of landing in circle, P_{in}?



$$P_{in} = N_{inCircle} \,/\, N_{total}$$

$$P_{in} = Area_{circle} \,/\, Area_{square}$$

$$P_{in} = (\pi(L/2)^2) \,/\, L^2 = \pi/4$$

$$\pi \cong 4\, N_{inCircle} \,/\, N_{total}$$

Pseudocode

```
For N trials
    Throw a dart
    If it lands in the circle
        Add 1 to total # of hits

Pi = 4*hits/N
```

Now we need to convert this into MATLAB code! Let's do this using top-down design!

Monte carlo approximation of π using a dart board

```
N = 10000;      L = 1;     hits = 0;
for k = 1:N






    end
end
piCalc = 4*hits/N;
```

Pseudocode:

For N trials
    Throw a dart
    If it lands in the circle
        Add 1 to total # of hits

Pi = 4*hits/N

Monte carlo approximation of π using a dart board

```
N = 10000;     L = 1;    hits = 0;
for k = 1:N
   % throw kth dart



   % count if it is in the circle



   end
end
piCalc = 4*hits/N;
```

Pseudocode:

For N trials
  Throw a dart
  If it lands in the circle
    Add 1 to total # of hits

Pi = 4*hits/N

Monte carlo approximation of π using a dart board

```
N = 10000;     L = 1;     hits = 0;
for k = 1:N
    % throw kth dart
    x = rand()*L - L/2;
    y = rand()*L - L/2;
    % count if it is in the circle
    if sqrt(x^2 + y^2) <= L/2
        ________________________
    end
end
piCalc = 4*hits/N;
```

Pseudocode:

For N trials
  Throw a dart
  If it lands in the circle
    Add 1 to total # of hits

Pi = 4*hits/N

Monte carlo approximation of π using a dart board

```
N = 10000;     L = 1;    hits = 0;
for k = 1:N
    % throw kth dart
    x = rand()*L - L/2;
    y = rand()*L - L/2;
    % count if it is in the circle
    if sqrt(x^2 + y^2) <= L/2
        hits = hits + 1;
    end
end
piCalc = 4*hits/N;
```

Pseudocode:

```
For N trials
    Throw a dart
    If it lands in the circle
        Add 1 to total # of hits

Pi = 4*hits/N
```

Another way to approximate π

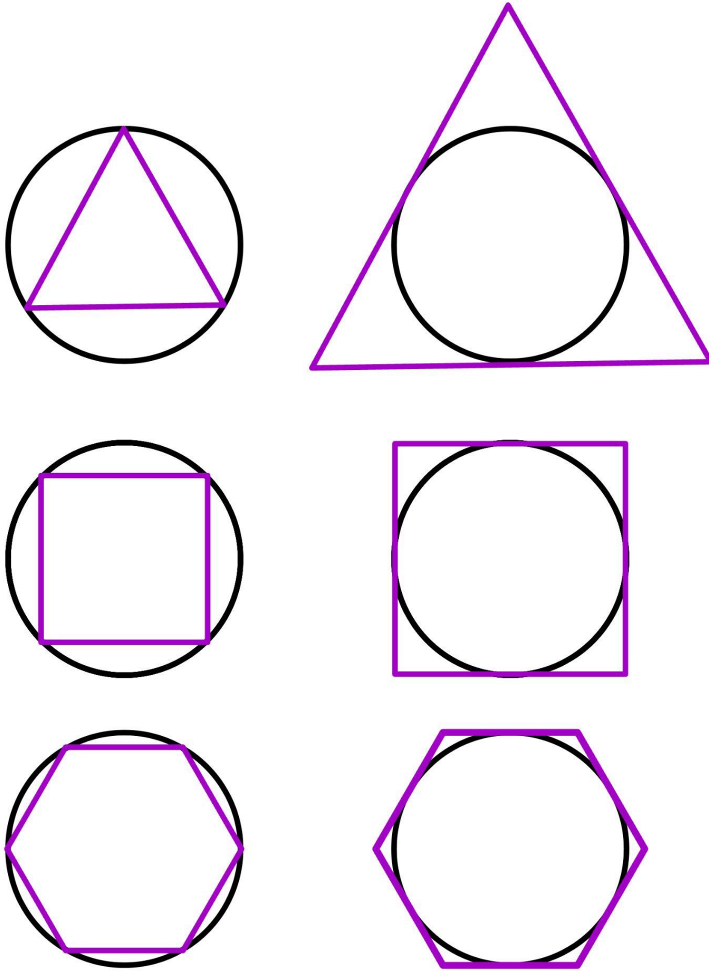
Consider n-sided regular inscribed and circumscribed polygons (about the unit circle).

What do you notice as the number of sides, n, increases?

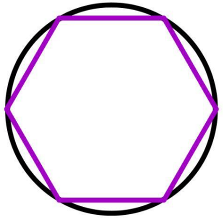
The areas of the polygons approach the area of the circle!

Inscribed polygon area: $$A_n = \frac{n}{2}\sin\left(\frac{2\pi}{n}\right)$$

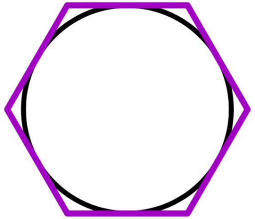
Circumscribed polygon area: $$B_n = n\tan\left(\frac{\pi}{n}\right)$$



Another way to approximate π



$A_n = \frac{n}{2}\sin\left(\frac{2\pi}{n}\right)$



$B_n = n\tan\left(\frac{\pi}{n}\right)$

Goal: Find the smallest n such that A_n and B_n converge.

Calculate triangle case (n = 3 and calculate A_n and B_n)

Repeat until difference between A_n and B_n is small

    Increase n

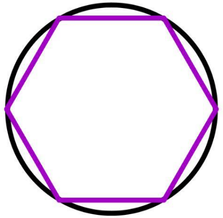
    Calculate A_n and B_n for current n

    diff = A_n-B_n

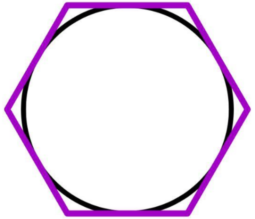
Display the final approximation

This is an example of indefinite iteration - when a set of instructions are repeated until a condition becomes false.

Another way to approximate π



$$A_n = \frac{n}{2}\sin\left(\frac{2\pi}{n}\right)$$



$$B_n = n\tan\left(\frac{\pi}{n}\right)$$

Goal: Find the smallest n such that A_n and B_n converge.

Calculate triangle case (n = 3 and calculate A_n and B_n)

while | A_n-B_n | > tolerance

    Increase n

    Calculate A_n and B_n for current n

    diff = A_n-B_n

Display the final approximation

```
% Approximate pi using indefinite iteration (simplified from Eg2_2.m)

tol = input('Enter the error tolerance: ');

% The triangle case
n = 3;
A_n = (n/2)*sin(2*pi/n);
B_n = n*tan(pi/n);
Error = B_n - A_n;

% Repeat until error less than or equal to tolerance
while _________________________
    n = n + 1;
    A_n = (n/2)*sin(2*pi/n);
    B_n = n*tan(pi/n);
    Error = B_n - A_n;
end

% Display the final approximation
```

```matlab
% Approximate pi using indefinite iteration (simplified from Eg2_2.m)

tol = input('Enter the error tolerance: ');

% The triangle case
n = 3;
A_n = (n/2)*sin(2*pi/n);
B_n = n*tan(pi/n);
Error = B_n - A_n;

% Repeat until error less than or equal to tolerance
while Error > tol
    n = n + 1;
    A_n = (n/2)*sin(2*pi/n);
    B_n = n*tan(pi/n);
    Error = B_n - A_n;
end

% Display the final approximation
```

Iteration caps

While loops run until some condition stops being true. Sometimes this takes makes the code run forever though.

- In some cases, it is undesirable to let the program keep computing something indefinitely.
  - For example, “I need to submit by 11 PM; give me your best answer right now!”
  - Solution: impose a maximum number of iterations (number of times the statements nested inside the loop are executed)

```
% Approximate pi using indefinite iteration (Eg2_2.m)

tol = input('Enter the error tolerance: ');
nMax = input('Enter the iteration bound: ');

% The triangle case
n = 3;
A_n = (n/2)*sin(2*pi/n);
B_n = n*tan(pi/n);
Error = B_n - A_n;

% Repeat until small error or max num of iters
while Error > tol ____________
    n = n + 1;
    A_n = (n/2)*sin(2*pi/n);
    B_n = n*tan(pi/n);
    Error = B_n - A_n;
end

% Display the final approximation
```

```matlab
% Approximate pi using indefinite iteration (Eg2_2.m)

tol = input('Enter the error tolerance: ');
nMax = input('Enter the iteration bound: ');

% The triangle case
n = 3;
A_n = (n/2)*sin(2*pi/n);
B_n = n*tan(pi/n);
Error = B_n - A_n;

% Repeat until error is small or we meet the max num of iters
while Error > tol && n < nMax
    n = n + 1;
    A_n = (n/2)*sin(2*pi/n);
    B_n = n*tan(pi/n);
    Error = B_n - A_n;
end

% Display the final approximation
```

For loops versus while loops

Do something n times (or a fixed number of times)

```
for [var] = [start]:[step]:[end]
    [code executed multiple times]
end
```

Do something until a condition stops being true

```
while [continueCriteria]
    [code executed multiple times]
end
```

```
n = 10;
for k = 1:1:n
    disp(k);
end
```

These two codes do the same thing!

```
k = 1; n = 10;
while k <= n
    disp(k);
    k = k + 1;
end
```