Day 2

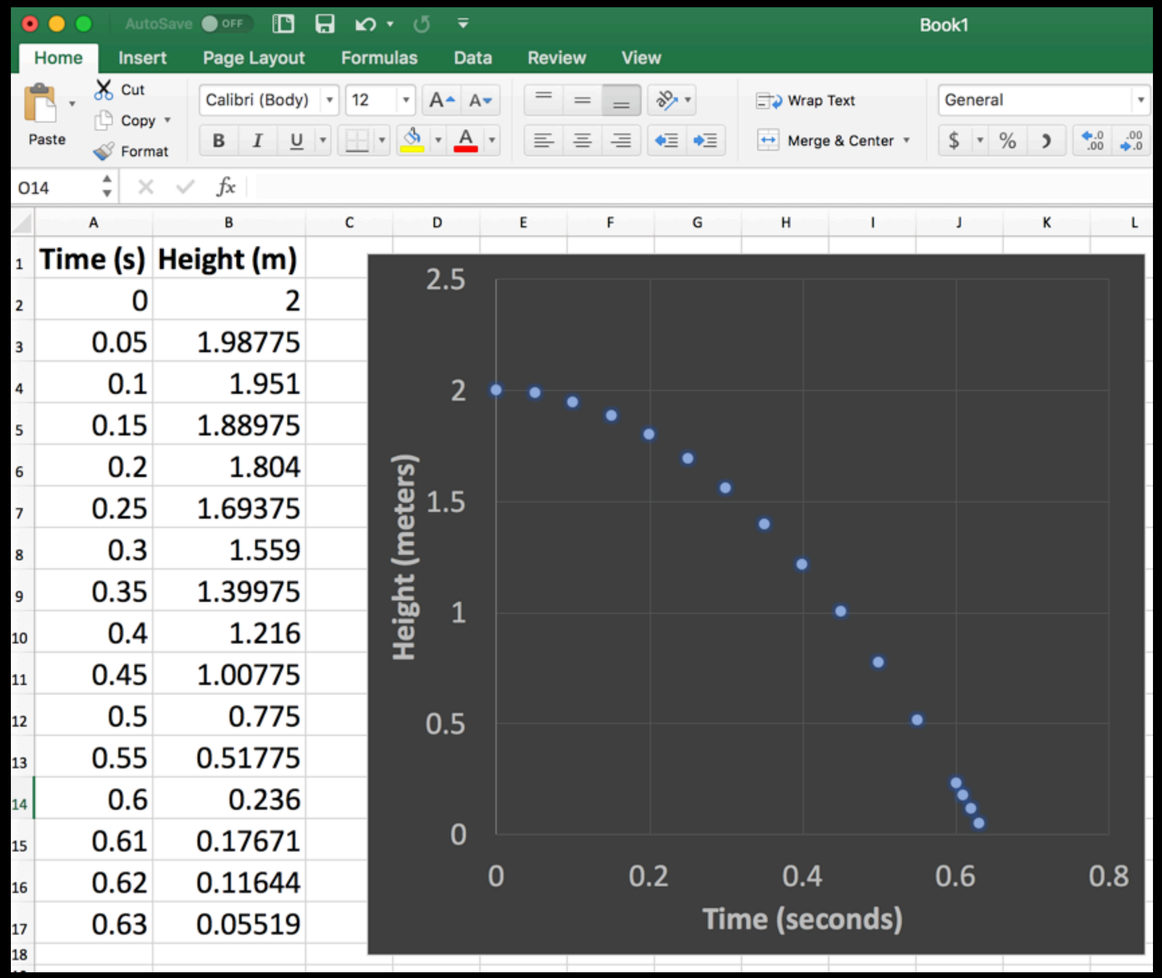
- Plotting results from the pi dartboard
- Unix activity
- Parallelize pi dartboard

Plotting your results

- Scatter plots
- Lists and numpy arrays
- Pyplot plotting

Scatter plots

- Data: result or output given some input
- Example: dropped marker height vs. time
- Tools to make scatter plots
  - Excel
  - Python
    - Lists of numbers
    - Computations on lists of numbers: numpy arrays
    - pyplot: makes scatter plots



Lists

- **Values -** ordered sets of objects, all the same type (like floats or ints)

```
x = [-2.0, -1.0, 0.0, 1.0, 2.0]
y = ["Hello", "world"]
z = [1, 4, 9, 16]
```

- **Operators - [ ], .append()**

```
z.append(25) == [1, 4, 9, 16, 25]
```

- **Easily add on elements in loops** with .append()

```
x[0] == -2.0
x[1] == -1.0
x[4] == 2.0
y[0] == "Hello"
y[-1] == "world"
z[-1] == 16
z[0] == 1
```

Loop over lists

```
for i in [0,1,2,3]:
  print(i*i)
```

```
0
1
4
9
```

```
import numpy as np
print(np.arange(1,5,1))
```

```
[1,2,3,4]
```

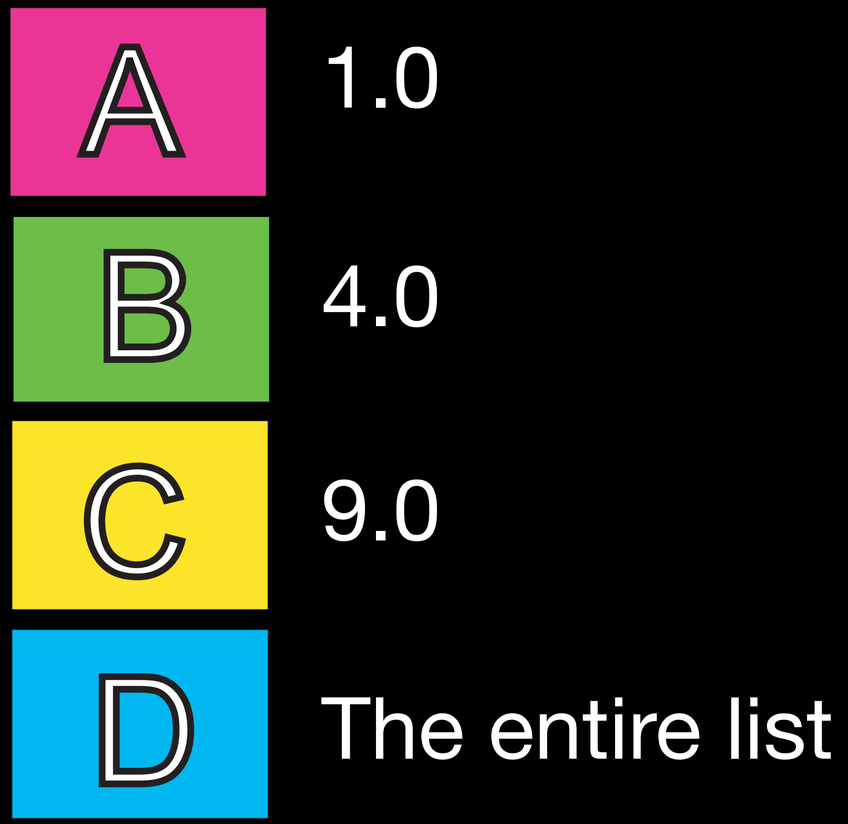
```
import numpy as np
myCountArray = np.arange(1,5)
myList = []
for i in myCountArray:
    myList.append(i*i)
print(myCountArray)
print(myList)
```

```
[1,2,3,4]
[1,4,9,16]
```

Clicker question #2.6

- What value does the program print?

```
x = [1.0, 4.0, 9.0]
print(x[1])
```



Numpy arrays

- **Values -** ordered sets of objects, all the same type (like floats or ints)

- **Operators - [ ], +, -, *, /, np.sqrt(), np.sin(), np.cos(), ...**

- **Easily do math on whole lists at once** (like formulas in Excel)

```
x = np.array([-2.0, -1.0, 0.0, 1.0, 2.0])
y = np.array(["Hello", "world"])
q = np.array([1, 2, 3, 4])
r = q * 2
s = q + r
z = q * q
```

```
r == np.array([2, 4, 6, 8])
s == np.array([3, 6, 9, 12])
z == np.array([1, 4, 9, 16])
```

```
x[0] == -2.0
x[1] == -1.0
x[4] == 2.0
y[0] == "Hello"
y[-1] == "world"
z[-1] == 16
z[0] == 1
```

Making sample data

- Annoying to type [1,2,3,4,…] all the time
- Instead: np.arange(start, stop, step)
- What do all these numbers mean??
- Make a plot to visualize them

Try in cocalc!

```
import numpy as np
x = np.arange(-4.0, 4.0, 0.01)
y = np.sin(x)**3
print(x)
print(y)
```

```
-9.99825171e-01 -9.99351433e-01 -9.98578166e-01 -9.97505912e-01
-9.96135421e-01 -9.94467651e-01 -9.92503769e-01 -9.90245148e-01
-9.87693366e-01 -9.84850205e-01 -9.81717651e-01 -9.78297888e-01
-9.74593301e-01 -9.70606471e-01 -9.66340175e-01 -9.61797379e-01
-9.56981241e-01 -9.51895105e-01 -9.46542499e-01 -9.40927131e-01
-9.35052889e-01 -9.28923832e-01 -9.22544191e-01 -9.15918365e-01
-9.09050915e-01 -9.01946561e-01 -8.94610179e-01 -8.87046794e-01
-8.79261581e-01 -8.71259853e-01 -8.63047062e-01 -8.54628794e-01
-8.46010761e-01 -8.37198799e-01 -8.28198860e-01 -8.19017011e-01
-8.09659425e-01 -8.00132377e-01 -7.90442239e-01 -7.80595473e-01
-7.70598629e-01 -7.60458333e-01 -7.50181290e-01 -7.39774268e-01
-7.29244102e-01 -7.18597680e-01 -7.07841944e-01 -6.96983877e-01
-6.86030504e-01 -6.74988880e-01 -6.63866088e-01 -6.52669231e-01
-6.41405427e-01 -6.30081800e-01 -6.18705479e-01 -6.07283586e-01
-5.95823237e-01 -5.84331527e-01 -5.72815532e-01 -5.61282298e-01
-5.49738839e-01 -5.38192126e-01 -5.26649084e-01 -5.15116589e-01
-5.03601455e-01 -4.92110435e-01 -4.80650212e-01 -4.69227393e-01
-4.57848505e-01 -4.46519990e-01 -4.35248195e-01 -4.24039375e-01
-4.12899678e-01 -4.01835147e-01 -3.90851715e-01 -3.79955193e-01
-3.69151273e-01 -3.58445520e-01 -3.47843366e-01 -3.37350109e-01
-3.26970907e-01 -3.16710771e-01 -3.06574566e-01 -2.96567003e-01
-2.86692639e-01 -2.76955868e-01 -2.67360924e-01 -2.57911871e-01
```

Plotting sample data

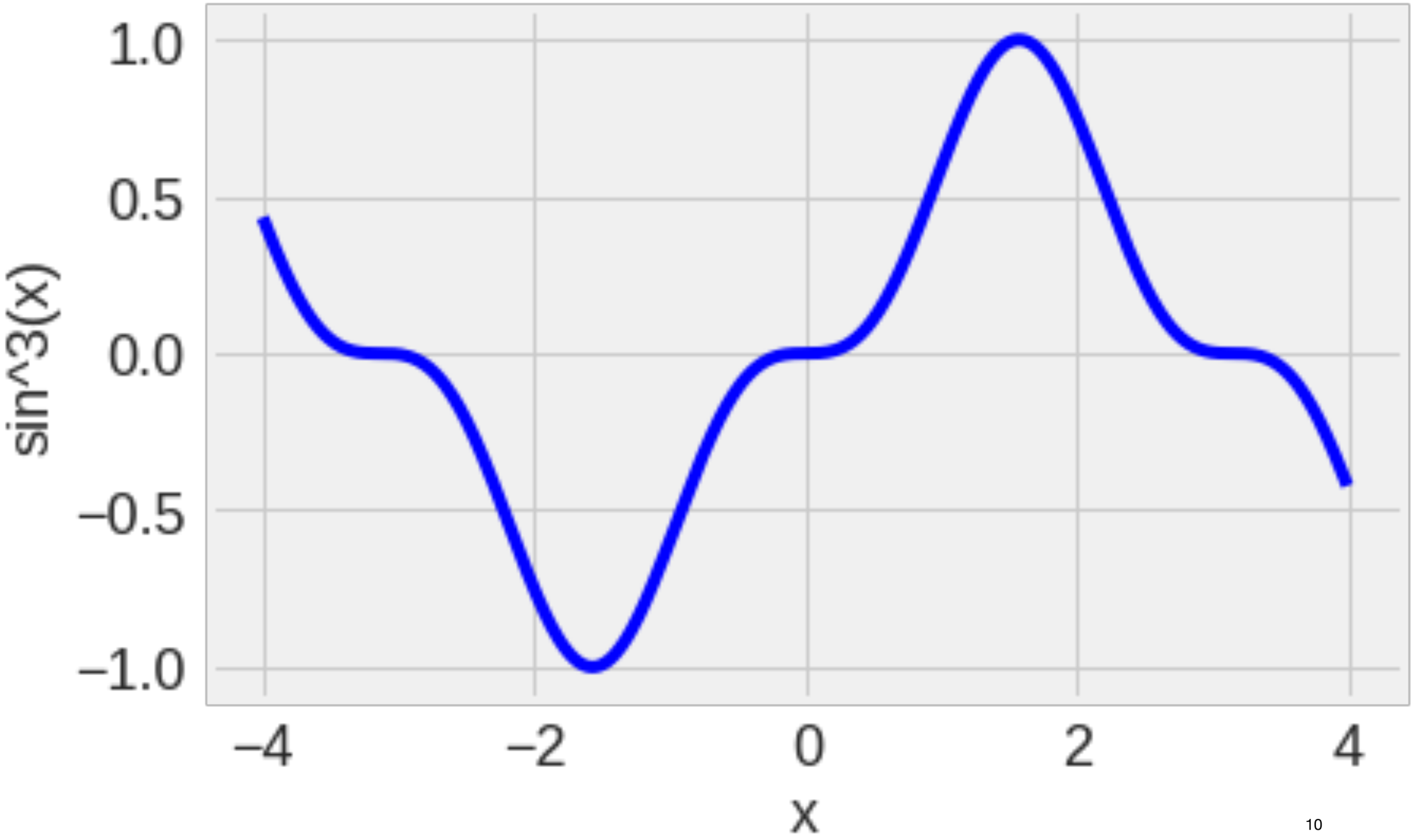
- Make plots with pyplot

Try in cocalc!

```
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
matplotlib.rc('axes', labelsize=18)
matplotlib.rc('xtick', labelsize = 18)
matplotlib.rc('ytick', labelsize = 18)
```

```
x = np.arange(-4.0, 4.0, 0.01)
y = np.sin(x)**3
```

```
plt.clf() #clear figure
plt.plot(x,y, color='b')
plt.xlabel('x')
plt.ylabel('sin^3(x)')
plt.show()
```


1.0
0.5
0.0
−0.5
−1.0
sin^3(x)
−4
−2
0
2
4
x

Functions

- Input(s) ("arguments")
- Returns output
- Functions can call other functions

Try in cocalc!

```
def square(x):
  return x*x

square(4)
16
```

Plotting π 1

- **Activity: edit your code**
  - Make everything but the imports and the last line inside a function that takes one input, n
  - Instead of setting throws = 10, set throws=n
  - Return the pi estimate

```
import math
import random

hits = 0
throws = 10

i = 0
while i < throws:
    x = random.random()
    y = random.random()

    rSquared = x*x + y*y
    if rSquared < 1:
        hits = hits + 1
    i = i + 1

pi = 4.0 * float(hits) / float(throws)
print(pi)
```

Solution 1

- **Activity: edit your code**
  - Make everything but the last two lines inside a function that takes one input, n
  - Instead of setting throws = 10, set throws=n
  - Return the pi estimate

```
import math
import random

def estimatePi(throws):
  hits = 0
  i = 0
  while i < throws:
      x = random.random()
      y = random.random()

      rSquared = x*x + y*y
      if rSquared < 1:
          hits = hits + 1
      i = i + 1

  pi = 4.0 * float(hits) / float(throws)
  return pi

print(estimatePi(1e4))
```

Plotting π 2

```
import math
import random

def estimatePi(throws):
  # (same definition of estimatePi function here)
  return pi

piEstimates = [estimatePi(x) for x in [10, 100, 1000, 10000]]
print(piEstiamtes)
```

Plotting π 2

```
import math
import random

def estimatePi(throws):
  # (same definition of estimatePi function here)
  return pi

trials = [10**j for j in np.arange(0,6)]
piEstimates = [estimatePi(x) for x in trials]
print(trials, piEstimates)
```

Plotting π 2

```
import math
import random
import matplotlib
from matplotlib import pyplot as plt

def estimatePi(throws):
  # (same definition of estimatePi function here)
  return pi

trials = [10**j for j in np.arange(0,6)]
piEstimates = [estimatePi(x) for x in trials]
print(trials, piEstimates)

plt.clf()
plt.plot(trials,piEstimates)
plt.xlabel("Number of darts")
plt.ylabel("Pi est.")
plt.show()
```

Plotting π 2

```
import math
import random
import matplotlib
from matplotlib import pyplot as plt

def estimatePi(throws):
  # (same definition of estimatePi function here)
  return pi

trials = [10**j for j in np.arange(0,6)]
piEstimates = [estimatePi(x) for x in trials]
print(trials, piEstimates)

plt.clf()
plt.plot(trials,piEstimates)
plt.xlabel("Darts thrown")
plt.ylabel("pi Estimate")
plt.xscale('log')
plt.show()
```

Plotting π 2

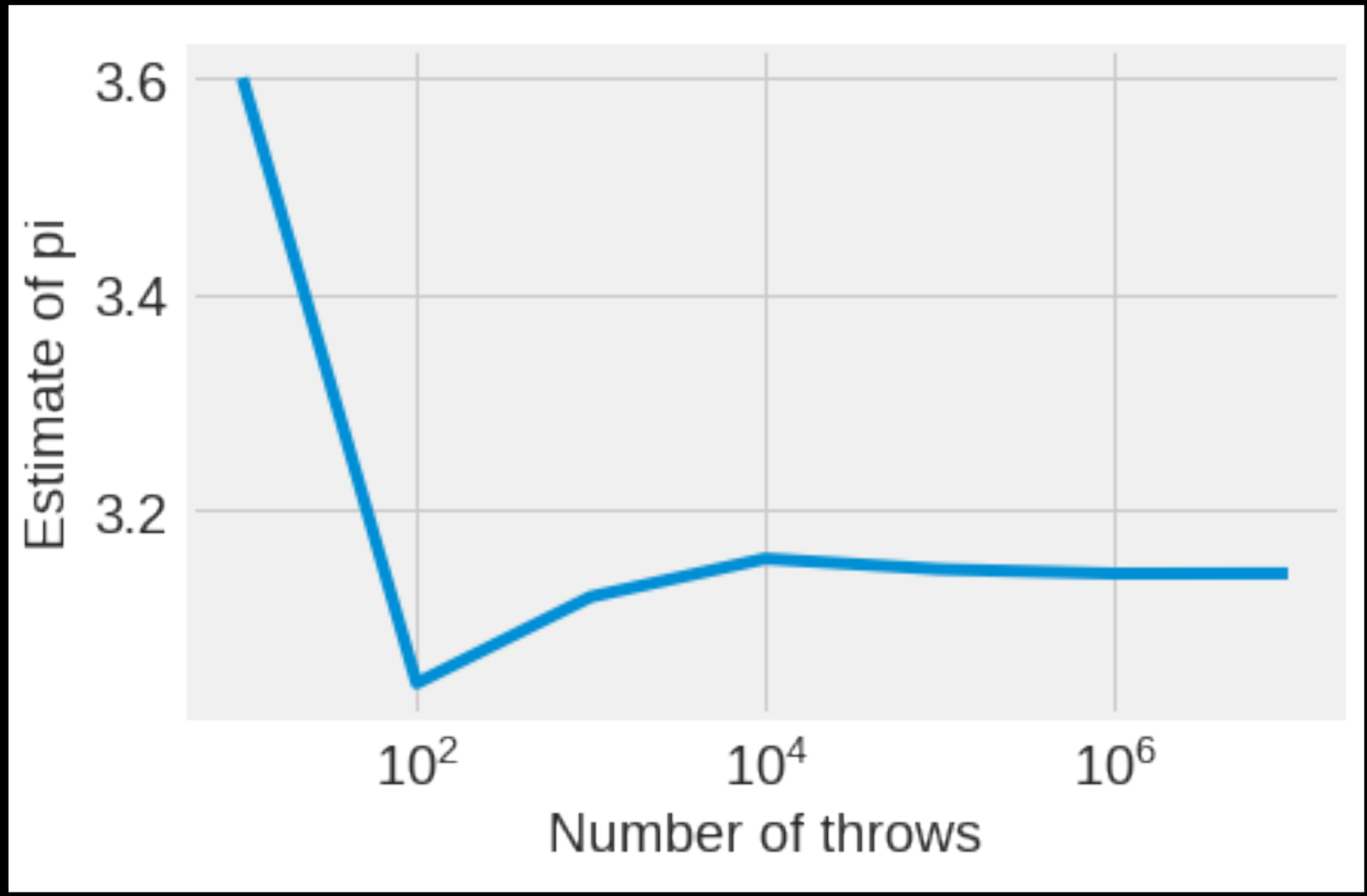
```
import math
import random
import matplotlib
from matplotlib import pyplot as plt

def estimatePi(throws):
  # (same definition of estimatePi function here)
  return pi

trials = [10**j for j in np.arange(0,6)]
piEstimates = [estimatePi(x) for x in trials]
print(trials, piEstimates)

plt.clf()
plt.plot(trials,np.abs(np.array(piEstimates)-np.pi))
plt.xlabel("Darts thrown")
plt.ylabel("|pi Estimate - pi|")
plt.xscale('log')
plt.show()
```

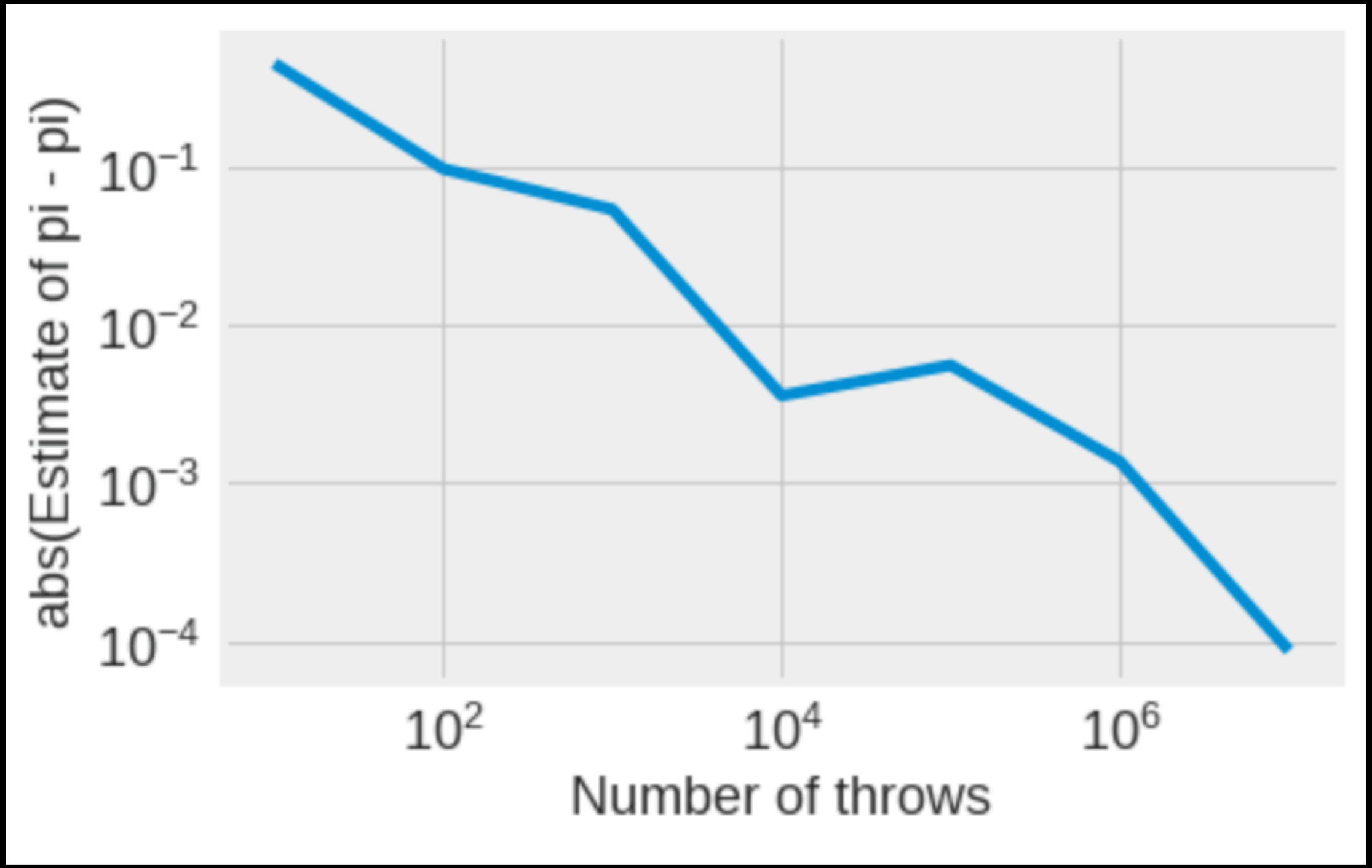
Plotting π 2

```
import math
import random
import matplotlib
from matplotlib import pyplot as plt

def estimatePi(throws):
  # (same definition of estimatePi function here)
  return pi

trials = [10**j for j in np.arange(0,6)]
piEstimates = [estimatePi(x) for x in trials]
print(trials, piEstimates)

plt.clf()
plt.plot(trials,np.abs(np.array(piEstimates)-np.pi))
plt.xlabel("Darts thrown")
plt.ylabel("pi Estimate")
plt.xscale('log')
plt.yscale('log')
plt.show()
```

Accuracy of the π dart board

- As throws goes up, answer gets closer to pi
- But it's hard to see how close it is later on
- So instead, plot the difference between the estimate and the real answer




abs(Estimate of pi - pi)
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{2}
10^{4}
10^{6}
Number of throws

Concepts in numerical programming

- Resolution
- Accuracy
- Precision

Resolution

Low resolution

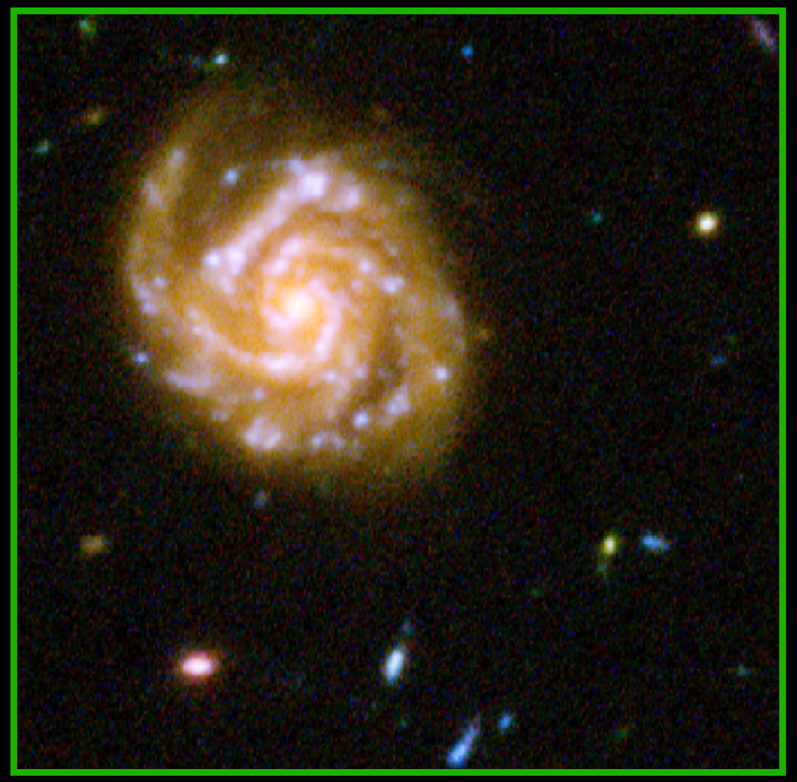
Entire image: 227KB

Large galaxies
1 billion light years away

Small galaxies up to
13 billion light years away

Image courtesy NASA

Resolution



High resolution

Entire image: 110MB

Large galaxies
1 billion light years away

Small galaxies up to
13 billion light years away

Image courtesy NASA

Resolution

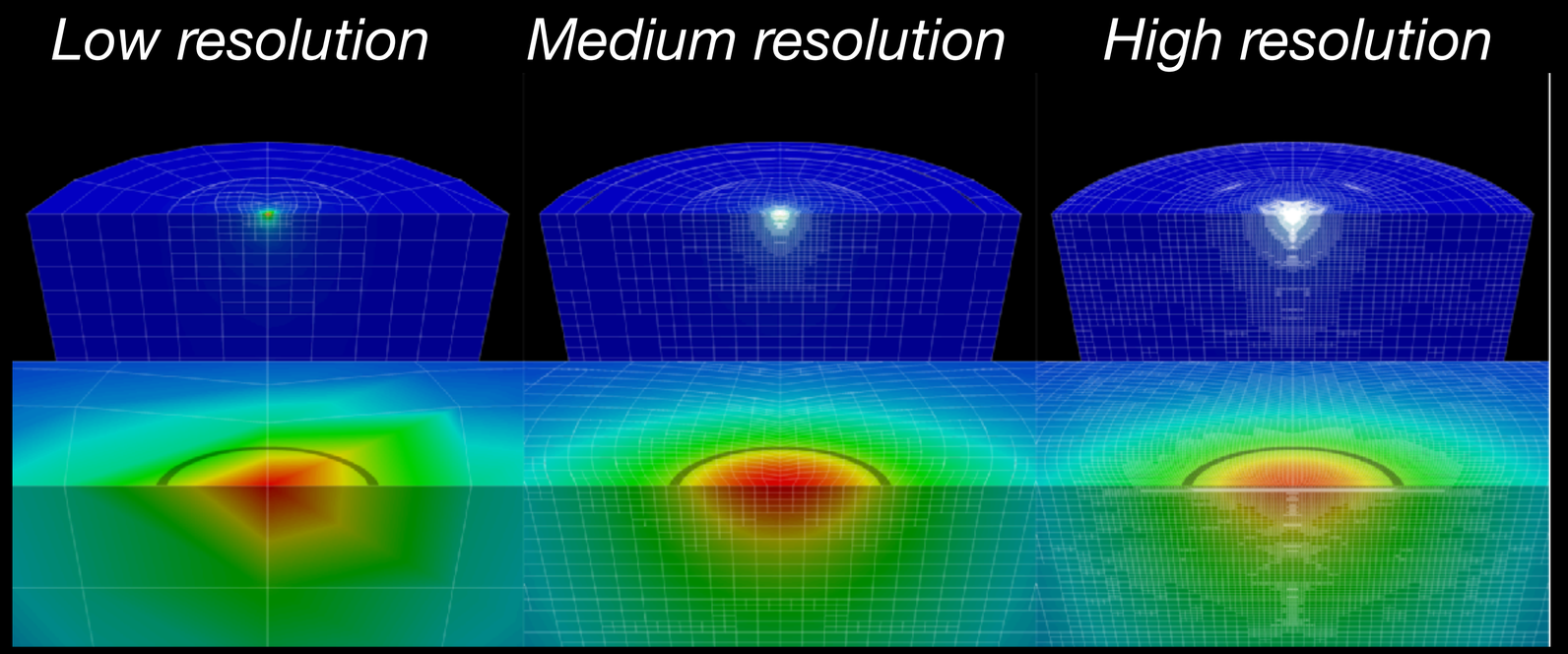
- **Low resolution**
  - Smaller data
  - Faster computation
  - Less precise
- **High resolution**
  - Bigger data
  - Slower computation
  - More precise

Precision & accuracy

- **Precision**
  - How much result changes when you add more resolution
  - "How many digits"
- **Accuracy**
  - How close result is to the correct result

Example: thermal noise

- Thermal noise of a mirror in LIGO depends on how much potential energy it gets when you push on the face

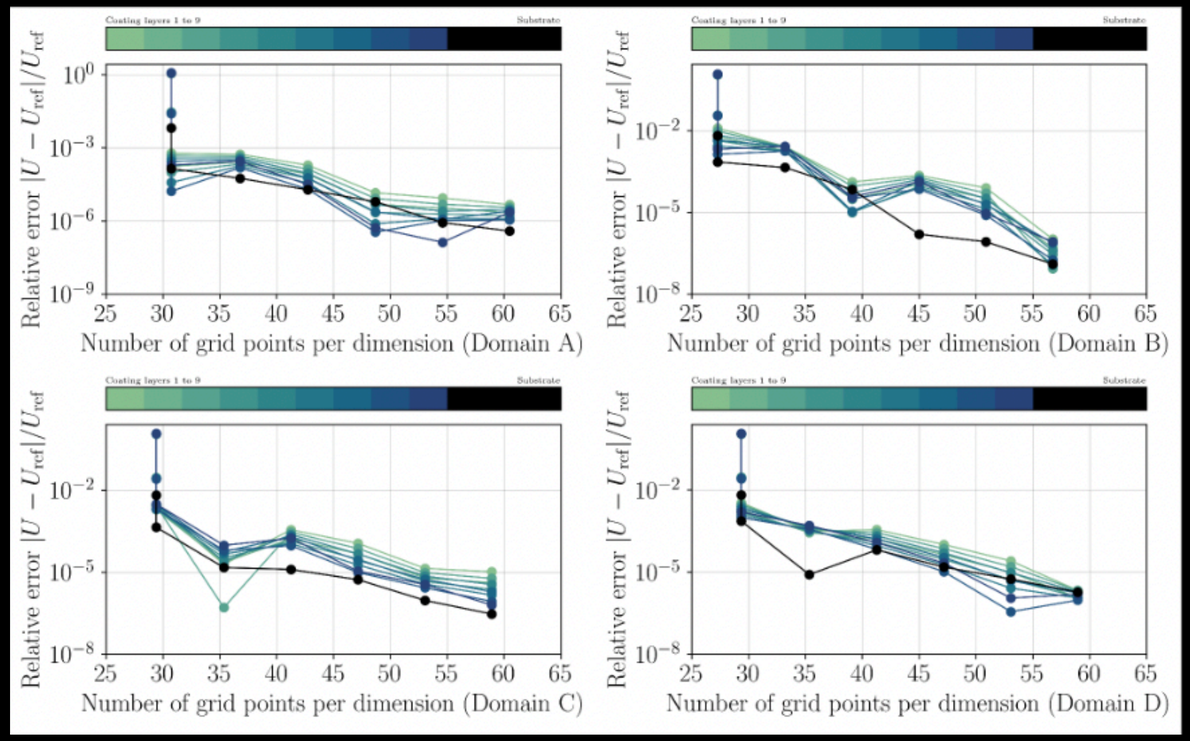


Color = how much mirror deforms

Lovelace, Demos, Khan (2018)

Example: thermal noise

- Potential energy *E* in deformed mirror
  - Precision of energy as resolution increases
    - Label resolution by integer *N*

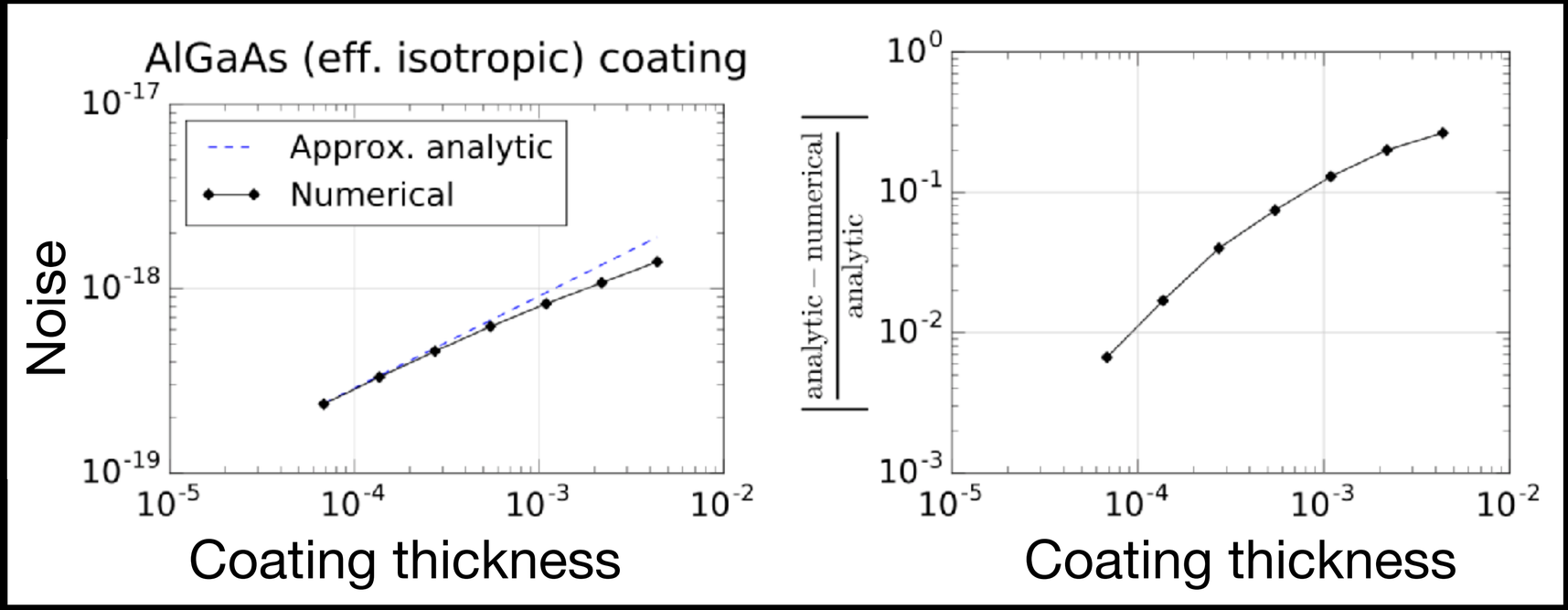


Higher resolution

Image courtesy Kyle Pannone

Example: thermal noise

- Thermal noise of thin coating
- Accuracy: compare code to known "analytic" solution

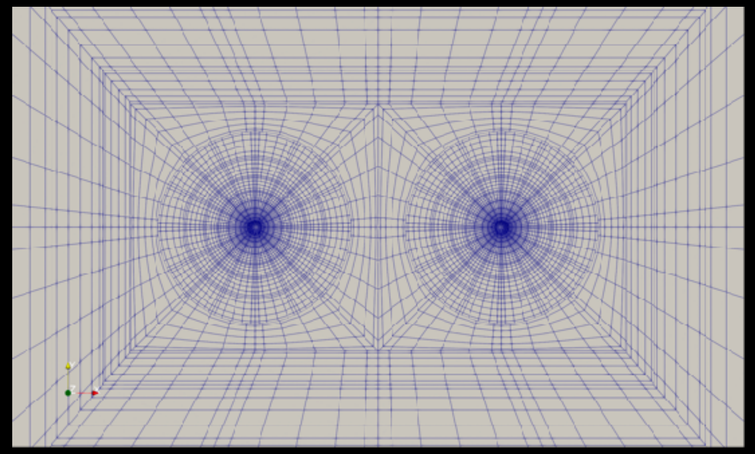


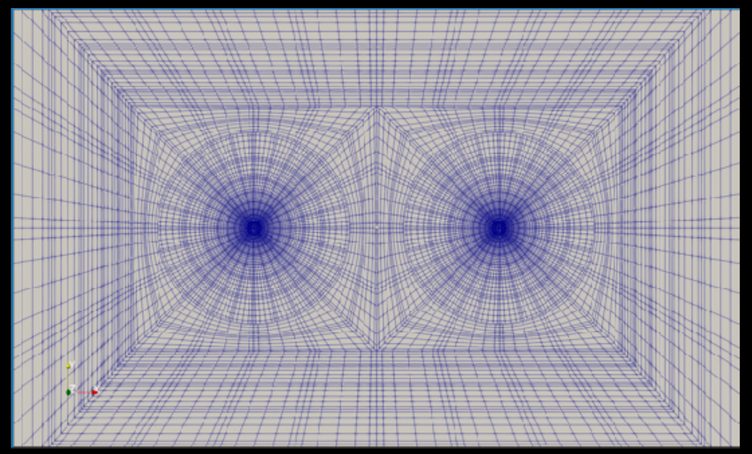
Lovelace, Demos, Khan (2018)

Example of resolution

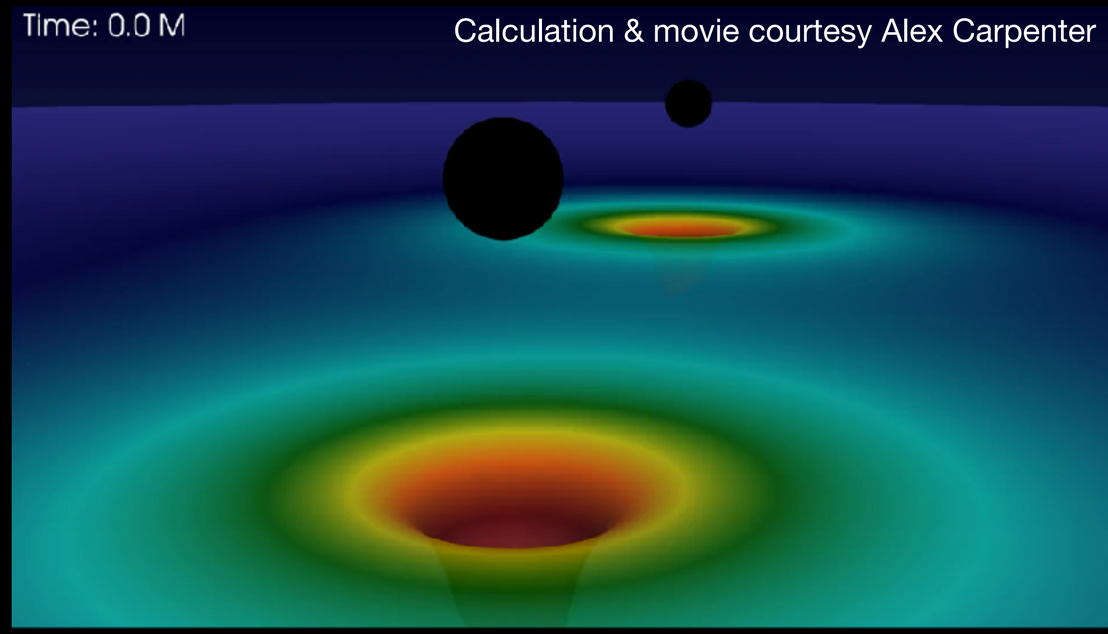
3-index constraint

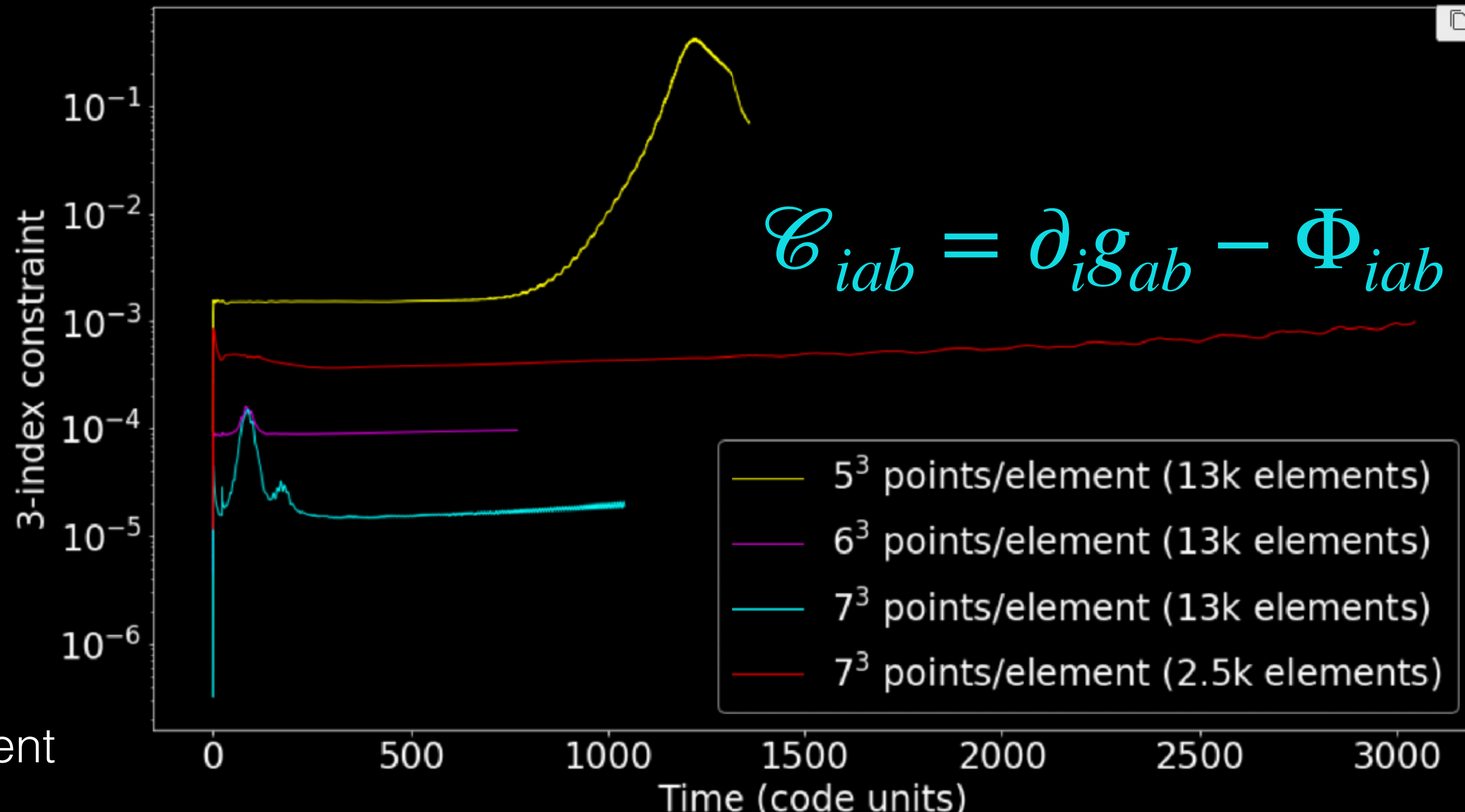
2.5k elements
7^3 points/element





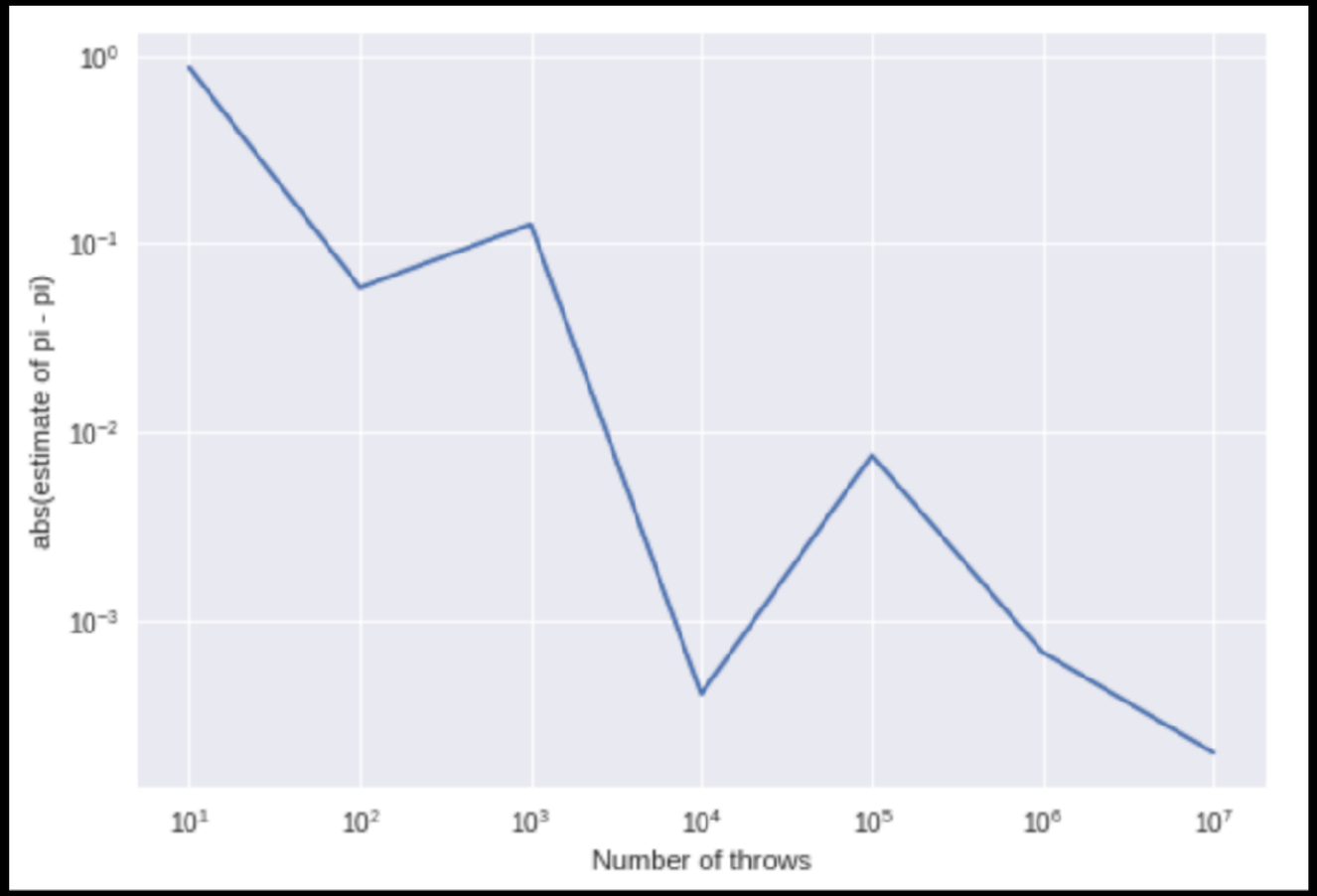
13k elements
7^3 points/element





Clicker question #2.7a

- Your graph plots abs(estimate of pi - pi) vs number of throws. The **horizontal axis** shows



A Precision

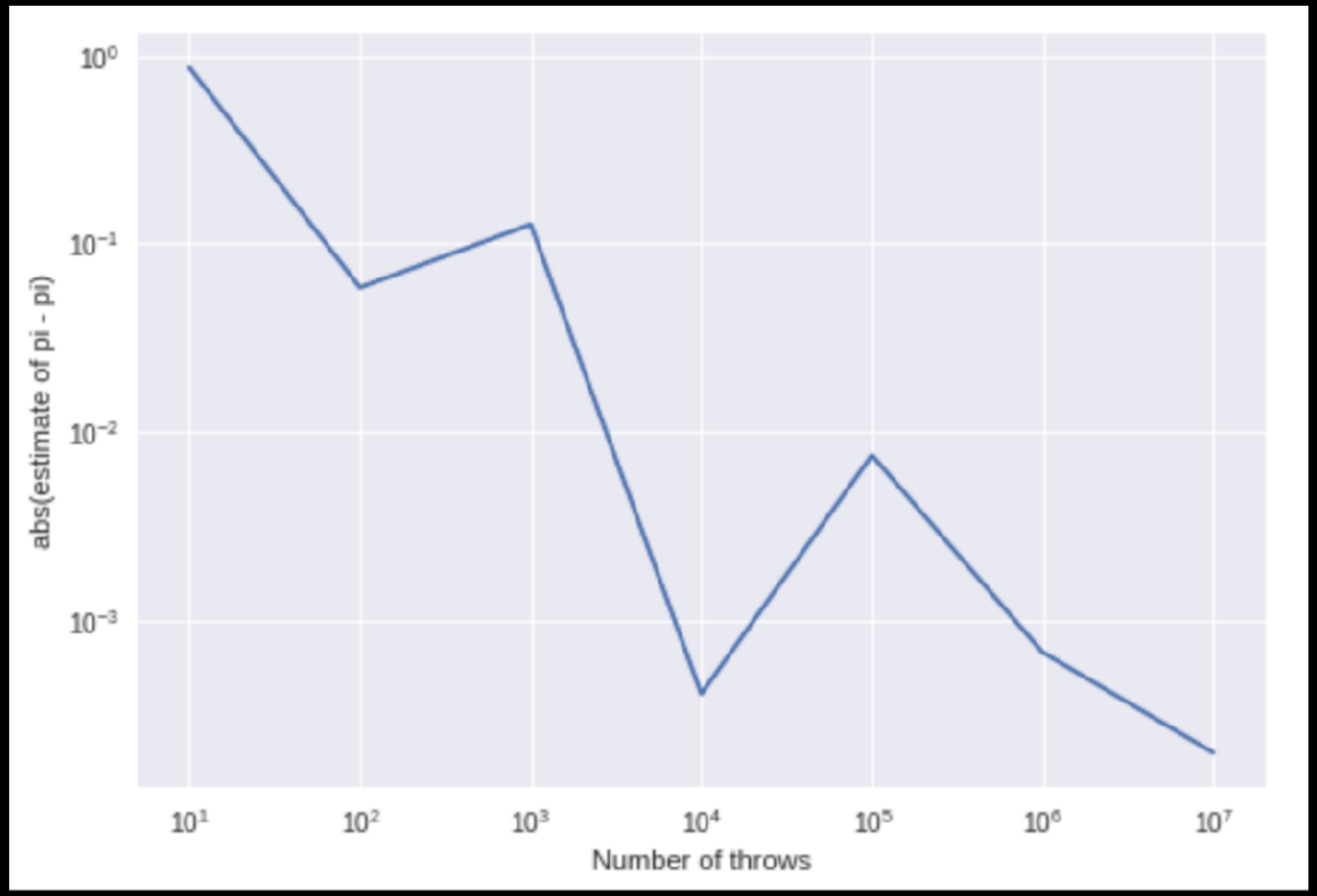
B Accuracy

C Resolution

D None of ABC

Clicker question #2.7b

- Your graph plots abs(estimate of pi - pi) vs number of throws. The **vertical axis** shows



A Precision

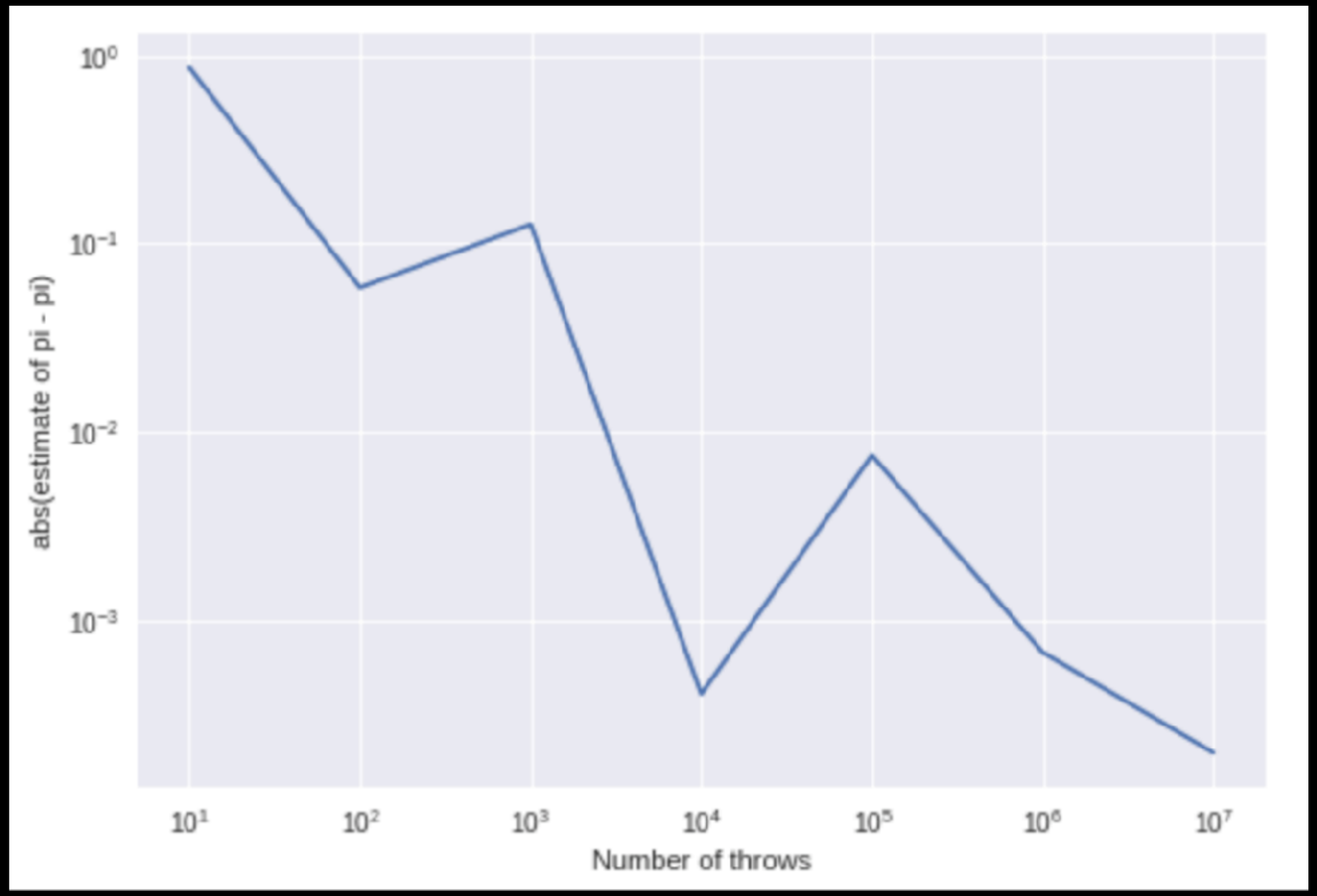
B Accuracy

C Resolution

D None of ABC

Clicker question #2.7c

- As the number of throws increases, the **resolution**



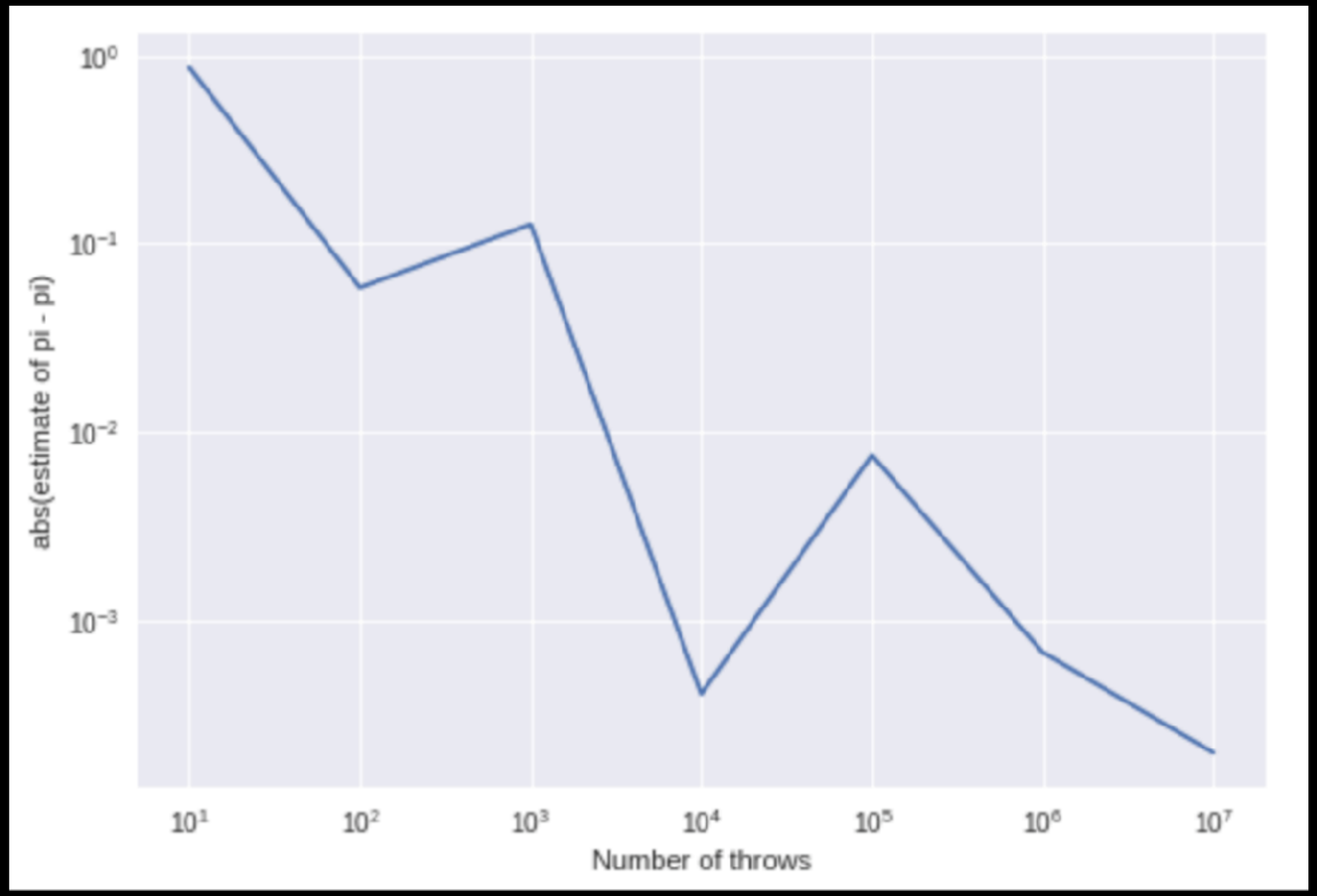
A Increases

B Decreases

C Stays the same

Clicker question #2.7d

- As the number of throws increases, the **accuracy**



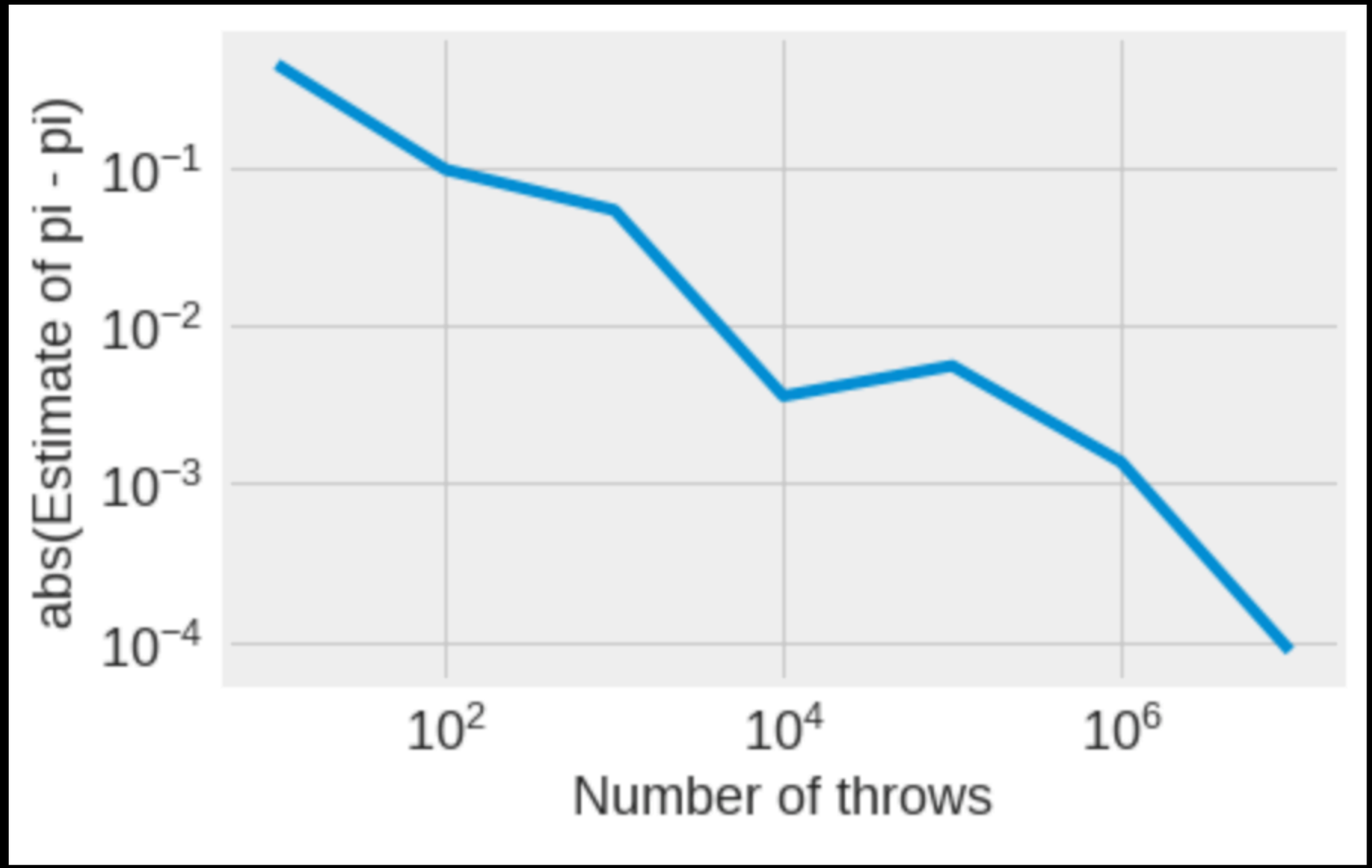
A Increases

B Decreases

C Stays the same

Example: π dart board

- Need roughly 100x more darts to get 10x more accuracy
  - That is, 100x darts gives you 10x more accuracy
  - This gets slow fast!
  - Can we do better?



Unix commands to know

- **Commands to know**
  - ls, pwd, cd, mkdir
  - ./, ../, paths
  - **cp, mv, rm, rmdir**
  - cat, **less**
  - nano
  - **cal, date, …**

- **Play along…**

Clicker question #1.9

- Which command makes a new directory called “TestFolder”?

A ls TestFolder

B cd TestFolder

C mkdir TestFolder

D cp TestFolder

Clicker question #1.8

- I want to list the files in the directory I’m in. Which command would I use?

A ls

B cd

C pwd

D nano

Clicker question #1.9

- Which command edits the file “Hello.txt” in the directory I am currently in?

A nano ./Hello.txt

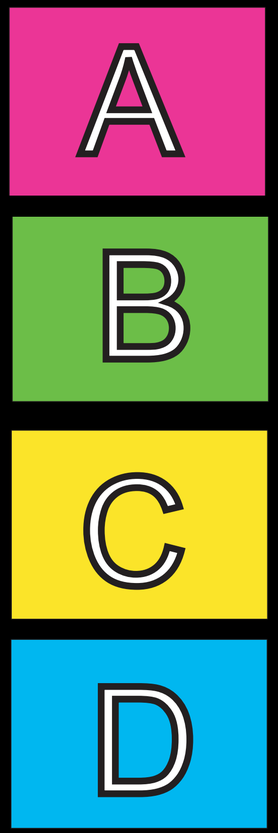
B cat ./Hello.txt

C nano ../Hello.txt

D cat ../Hello.txt

Clicker question #1.9

- Which command removes everything in the current directory, which is not empty?



rmdir ./

rm -r ./

rm -r . /

More than one of these will work

Parallel computing

- Supercomputers have lots of cores
- But each core is not much faster than a PC
- To take full advantage, you have to write code that can run on more than one core at the same time
- That is, code that runs in parallel

Image courtesy Blue Waters

Parallel computing 1

- Log into cocalc
- In a terminal:

```
#Replace YourName with your name
ssh ws2021@ocean.fullerton.edu
cd StudentFolders
cd 2023
mkdir YourName
cd YourName

mkdir PiDart
cd PiDart
```

Parallel computing 2

- # In your terminal, make a file "Hello.py" and put the following Python code into it
- nano Hello.py

```
print(“Hello")
```

- mpirun -np 8 python Hello.py
- **What happens? What happens if you change 8 to another number less than 8?**

What happened?

- mpirun ran many copies of “Hello.py”
- Each copy printed “Hello”
  - But the processors are not working together yet, or even doing anything different
- Next: make different processors do different things

Parallel computing 3

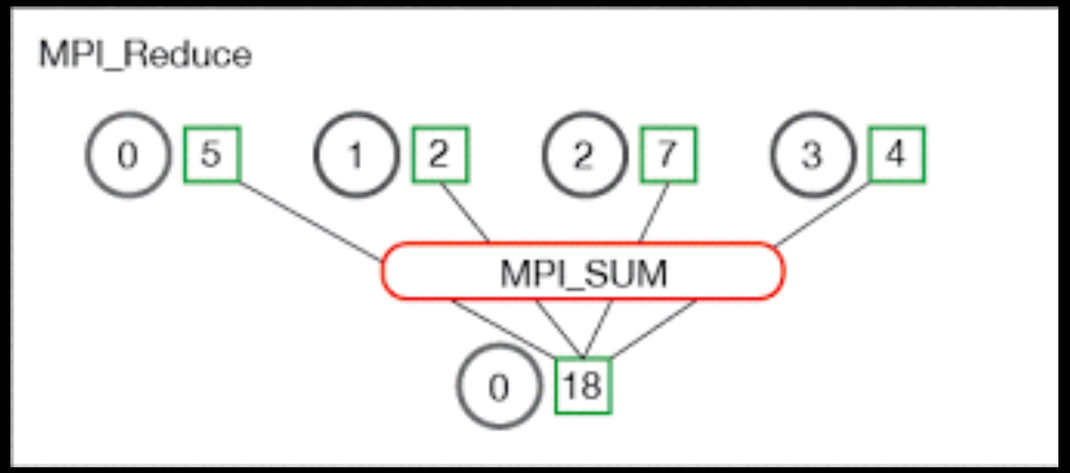
- cp Hello.py MpiHello.py
- nano MpiHello.py

```
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()
print("Hello from processor "+str(rank)+" out of
"+str(size))
```

- mpirun -np 4 python MpiHello.py
- mpirun -np 8 python MpiHello.py

Paralleizing the dartboard

- **What if we combined results from the whole class's π dartboard?**
- **Even better**
  - Run lots of copies of the dartboard on lots of cores
  - At the end, each copy tells the others how many hits it had
  - Each copy adds up the number of hits on all processors and computes pi



Parallelizing the dartboard 2

- `cp /home/ws2021/solutions/PiDart/PiDart.py .`
- `nano PiDart.py`
- `#Add the same mpi4py lines at the top`

```
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()
```

Parallelizing the dartboard 3

- nano PiDart.py
- #At the bottom, instead of getting pi, print the number of hits on each processor

```
print(str(hits)+" hits on processor "+str(rank)+" out of "+str(int(throws))+" throws.")
```

- mpirun -np 12 python piEstimate.py
  - What happens?

```
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()
```

Parallelizing the dartboard 4

- nano PiDart.py
- #Divide the darts to throw
  among the processors, instead
  of each processor
  throwing the total

```
hits = 0
throws = 1e7 // size
# … rest of program
```

- mpirun -np 12 python piEstimate.py
  - What happens?

Parallelizing the dartboard 5

- nano PiDart.py
- #Have on processor add up the totals across all processors

```
print(str(hits)+" hits on processor "+str(rank)+" out of "+str(throws)+" throws.”)


throwsAllProcessors = throws * size
hitsAllProcessors = comm.allreduce(hits, op=MPI.SUM)

if rank == 0:
    print(str(hitsAllProcessors)+" hits on all processors, with "+str(throwsAllProcessors)+" throws.")
```

-

Parallelizing the dartboard 6

- nano PiDart.py
- #Compute pi

```
if rank == 0:
    print(str(hitsAllProcessors)+" hits on all processors, with "+str(throwsAllProcessors)+" throws.”)

    pi = 4.0 * float(hitsAllProcessors) / float(throwsAllProcessors)
    print(pi)
```

-