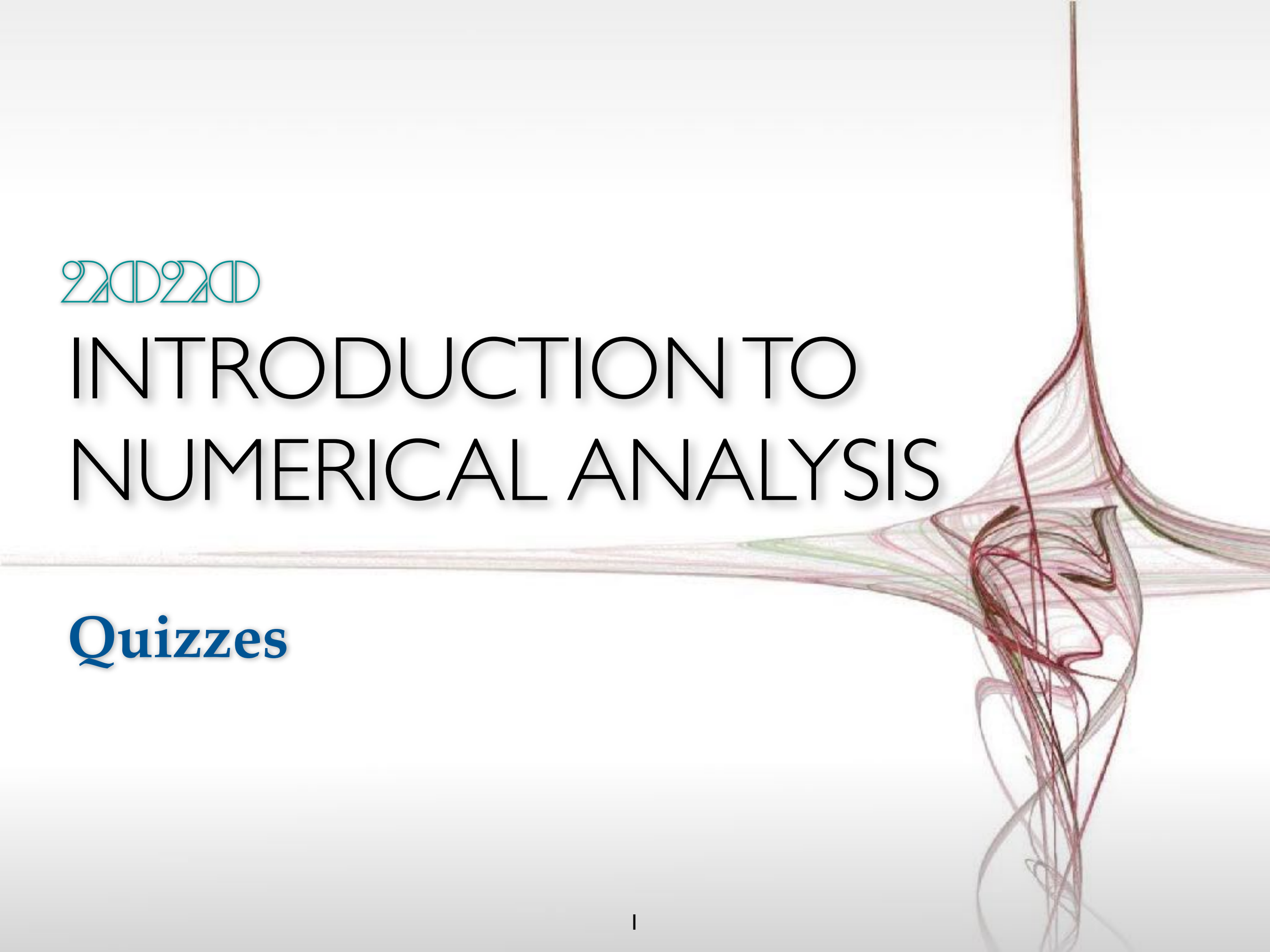
2020

INTRODUCTION TO NUMERICAL ANALYSIS



Quizzes

RULES

- **Please download the template code from CEIBA and implement your answer accordingly!**
- The deadline for answer uploading is **April/27, 17:00**. You have full 2 week time to solve them. **Delay is NOT allowed**.
- There are 5 quizzes in the following slides, you have to at least solve **3 out of 5** correctly (*recommend to solve them all if possible!*). You can google or discuss with other people, but please do NOT just copy the answers!
- You will be able to obtain a coin if
  - You managed to be the first 2 persons who solve any one of the quizzes correctly.
  - Your code has been selected as the "most elegant" solution by TA (1 slot per quiz).

QUIZ I DATA

Taipei Metro: driving time between stations

You can find a python list contain the interval data (source: Taipei gov) in the template code! Please start from there!

```
intervals = [["Tamsui", "Hongshulin", 175],
             ["Hongshulin", "Zhuwei", 136],
             ["Zhuwei", "Guandu", 145],
             ["Guandu", "Zhongyi", 78],
             ["Zhongyi", "Fuxinggang", 109],
             ["Fuxinggang", "Beitou", 145],
             ["Beitou", "Qiyan", 91],
. . . . . .
             ["Yongan Market", "Jingan", 88],
             ["Jingan", "Nanshijiao", 103]]
```

driving time in sec

QUIZ I

Find the shortest total driving time

- As you may know Taipei metro have 121 stations so far. Implement a function, which takes two strings (station names) as the argument, **find out the shortest total driving time between them**, and return the value.
- A couple of test results:

```
from Beitou to Guting: 1192 sec
from Fu Jen University to Tamsui: 2408 sec
from Xindian District Office to Taipei Main Station: 832 sec
from Daan to Nanshijiao: 805 sec
from Technology Building to Zhuwei: 1795 sec
```

Surely this should be shorter than the real traveling time since the waiting time at stations and transfer time are not included…

QUIZ II DATA

Simple Mazes

- A NumPy array of shape 50×151×151 can be loaded from the file "`maze_data.npy`". It contains 50 mazes of dimension 151×151 and it looks like this:

```
>>> import numpy as np
>>> data = np.load('maze_data.npy')
>>> data[0]
array([[2, 2, 2, ..., 2, 2, 2],
       [0, 0, 2, ..., 0, 0, 2],
       [2, 0, 2, ..., 2, 0, 2],
       ...,
       [2, 0, 2, ..., 2, 0, 2],
       [2, 0, 0, ..., 0, 0, 0],
       [2, 2, 2, ..., 2, 2, 2]], dtype=uint8)
```

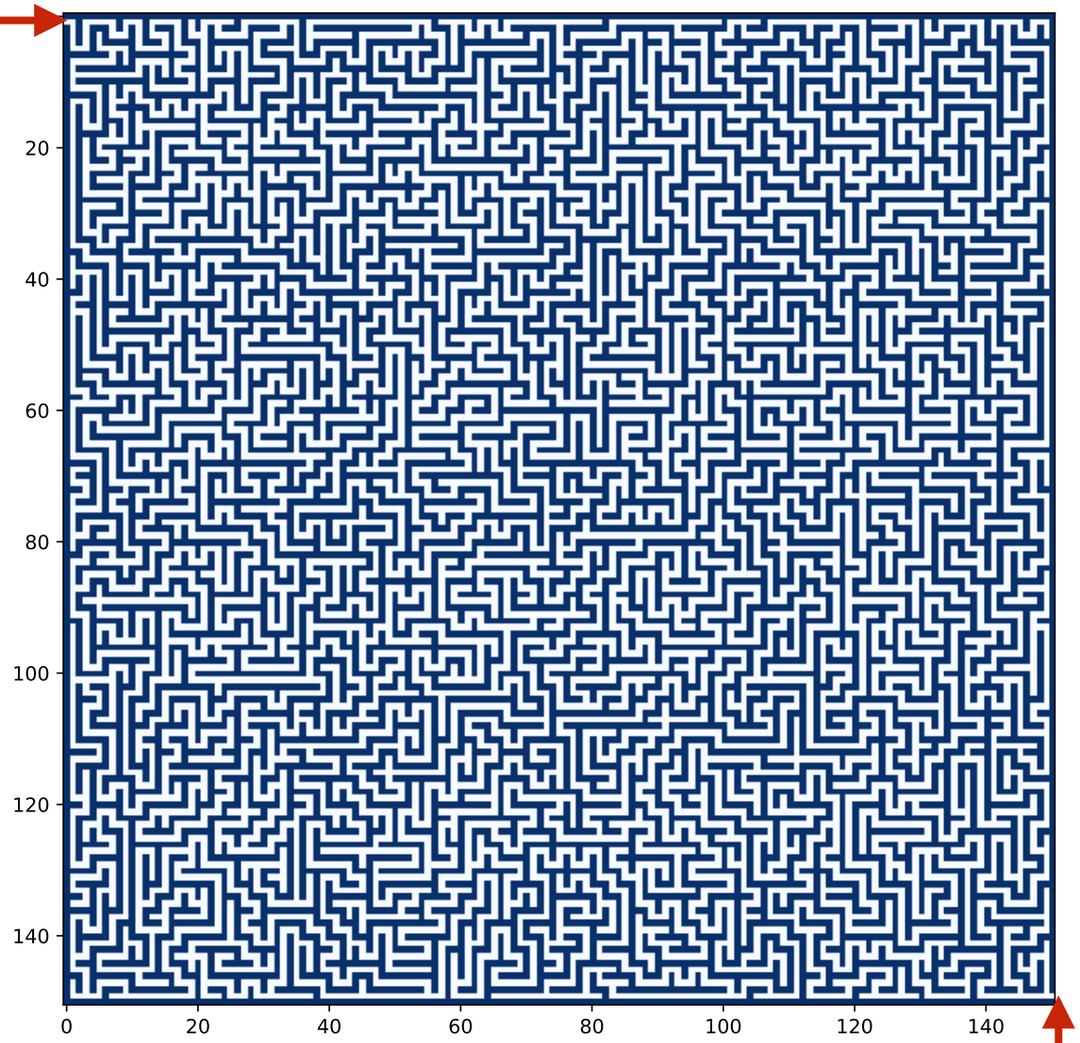
QUIZ II DATA

- In the array, twos indicate the walls, while zeros are pathway:

The snapshot of the first array

```
array([[2, 2, 2, ..., 2, 2, 2],
       [0, 0, 2, ..., 0, 0, 2],
       [2, 0, 2, ..., 2, 0, 2],
       ...,
       [2, 0, 2, ..., 2, 0, 2],
       [2, 0, 0, ..., 0, 0, 0],
       [2, 2, 2, ..., 2, 2, 2]])
```

=



QUIZ II

Solving the Maze

- Construct a function which takes a NumPy array of shape 101×101 as the input argument. The array represents a maze. Examine the maze data, **find the path from entry point [1,0] toward the exit point [149,150]**. Fill the path with **value 1** and return it back.
- Remark:
  - Any of the given mazes has only one unique path (unique solution!)
  - You should not fill the path with a dead end, leave them as 0.
  - Your code should work for all of the given mazes at least!

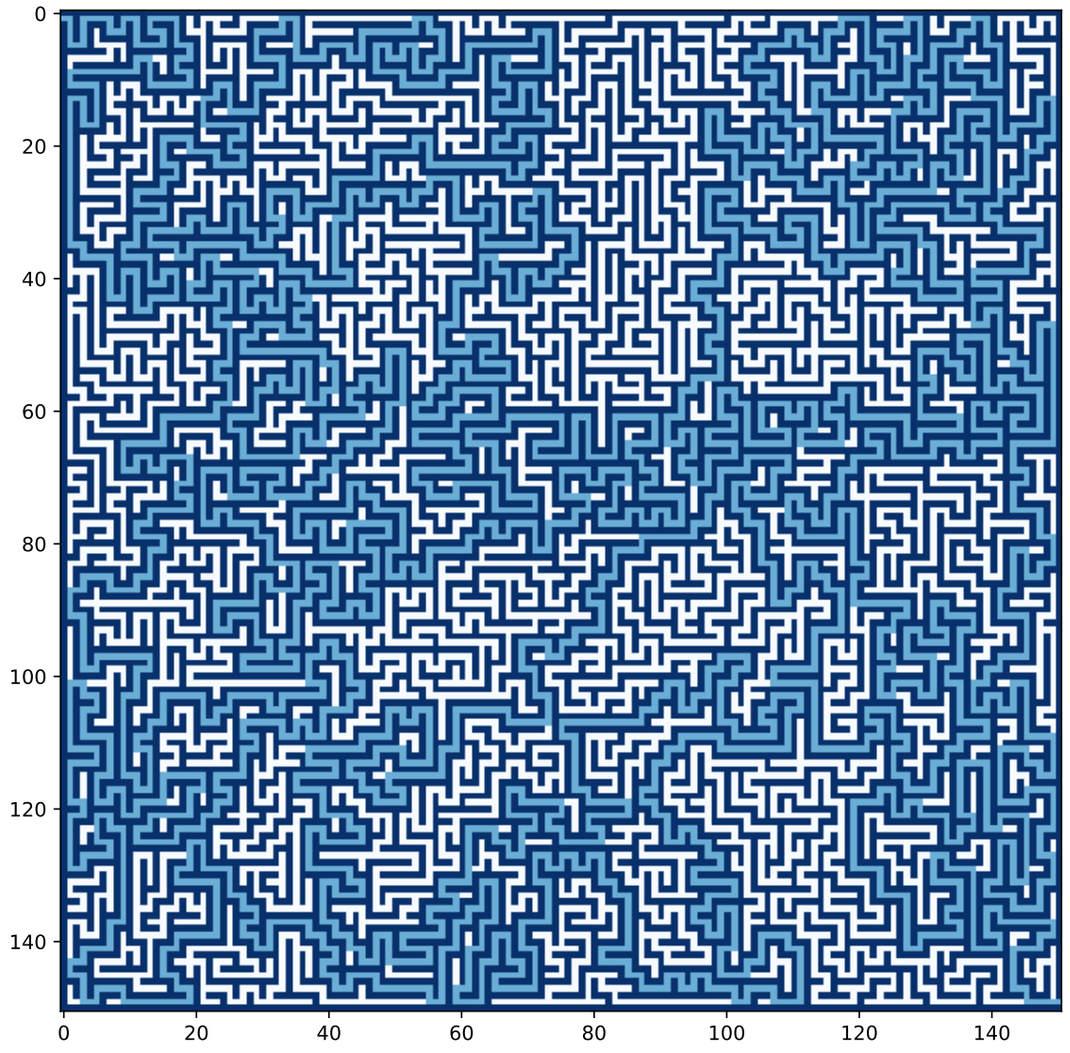
QUIZ II

Test solutions

The snapshot of the first solved maze

```
array([[2, 2, 2, ..., 2, 2, 2],
       [1, 1, 2, ..., 0, 0, 2],
       [2, 1, 2, ..., 2, 0, 2],
       ...,
       [2, 0, 2, ..., 2, 0, 2],
       [2, 0, 0, ..., 1, 1, 1],
       [2, 2, 2, ..., 2, 2, 2]])
```

=



The solutions to the first 5 mazes are stored in **“maze_solve.npy”**. You may compare your solutions with them.

QUIZ III DATA

Jigsaw Puzzles

- A NumPy array of shape 10×360×360×3 can be loaded from the file "`jigsaw_data.npy`". It contains 10 images of dimension 360×360×(R,G,B) looked like this:

```
>>> import numpy as np
>>> data = np.load('jigsaw_data.npy')
>>> data[0]
array([[[ 50,  34,   9],
        [ 52,  37,  10],
        [ 56,  41,  13],
        ...,
        [216, 202, 163],
        [216, 202, 163],
        [215, 201, 162]],
        ...,
        [  2,   2,   2],
        [  3,   3,   1],
        [  4,   3,   2]]], dtype=uint8)
```

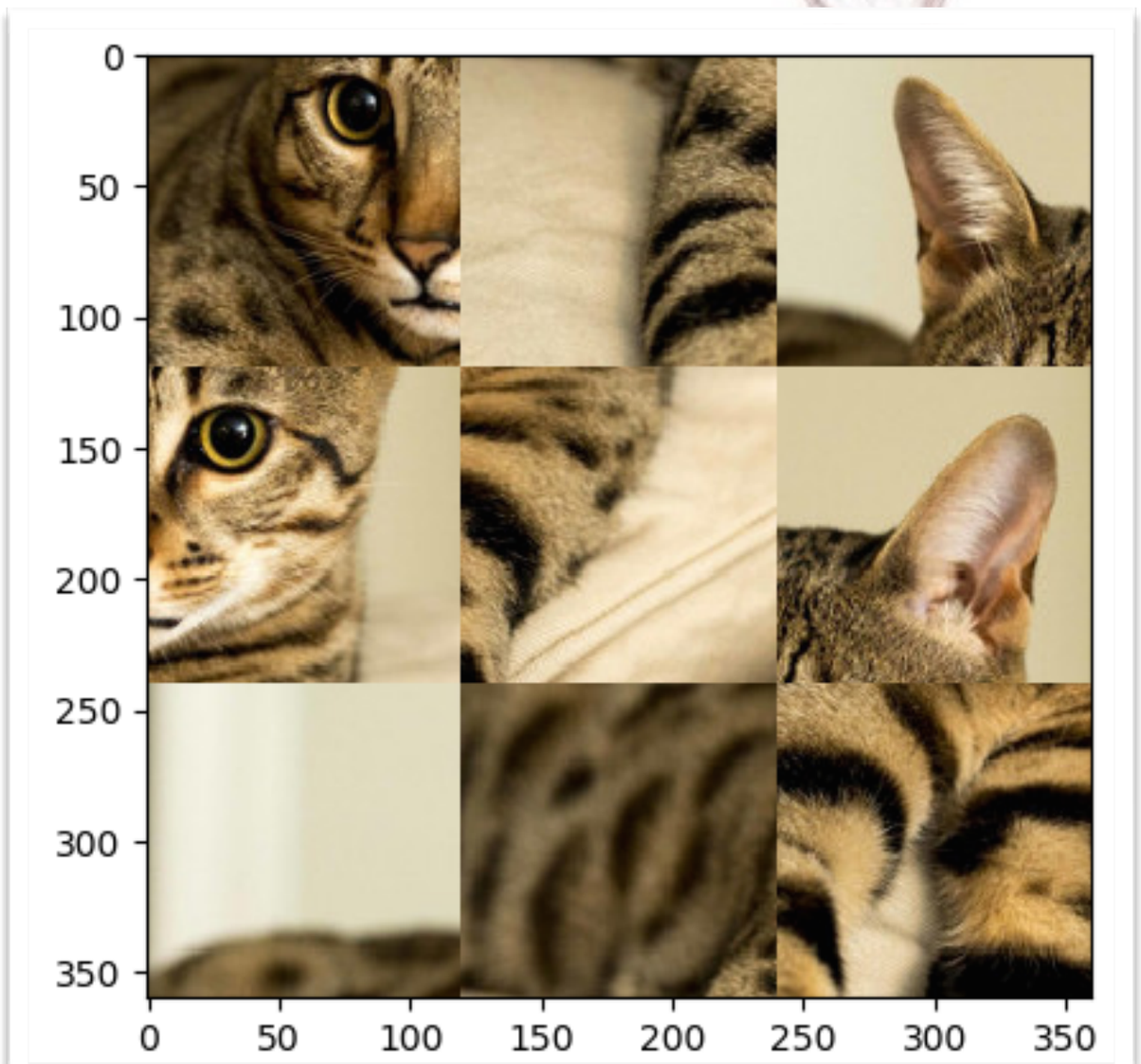
QUIZ III DATA

- It's nothing more than a **jigsaw puzzle** of 9 pieces!

```
import numpy as np
import matplotlib.pyplot as plt

data = np.load('jigsaw_data.npy')

plt.imshow(data[0])
plt.show()
```

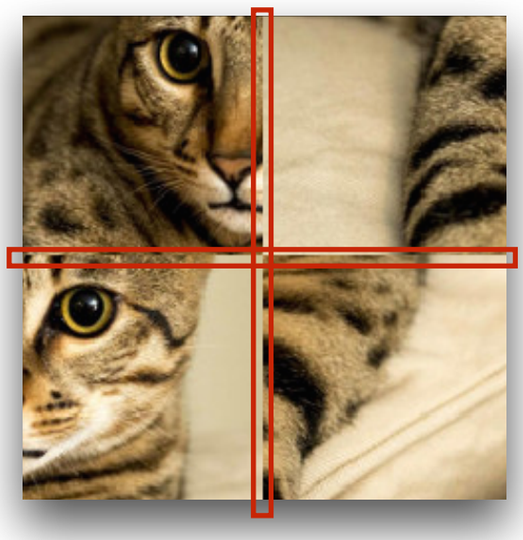


QUIZ III

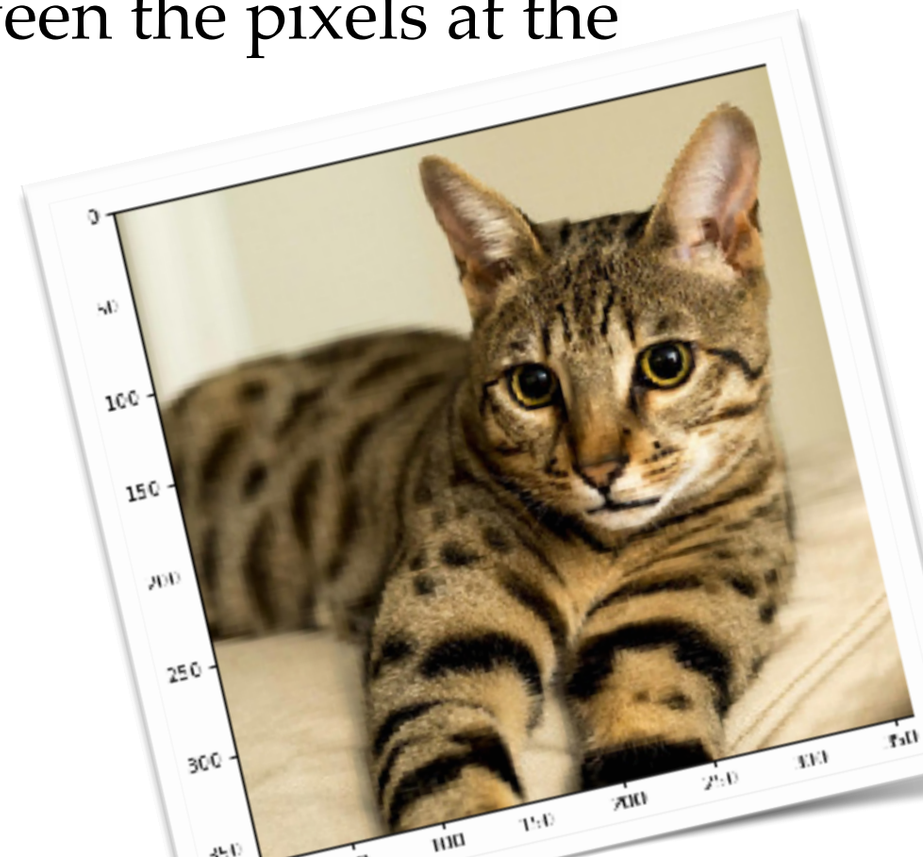
Your code should work for all of the given puzzles at least!

Solving the Jigsaw Puzzle

- Construct a function which takes a NumPy array of shape 360×360×3 as the input argument. The array represents an image of jigsaw puzzle. **Please solve the puzzle and return the solution as a NumPy array in the same dimension as the input**.
- The trick is minimizing the difference between the pixels at the edge of the adjacent pieces, e.g.



The original picture should be more-or-less 'continuous'!



QUIZ IV DATA

Histogram Data

- A NumPy array of shape 50×100 can be loaded from the file "`hist_data.npy`". It contains 50 histograms of 100 bins looked like this:

```
>>> import numpy as np
>>> data = np.load('hist_data.npy')
>>> data[0]
array([135, 146, 149, 121, 118, 125, 120, 147, 141, 142, 120, 124, 118,
       141, 119, 111, 133, 130, 117, 104, 110, 105, 107, 114, 106,  90,
       112, 124, 116,  95, 107, 118, 117,  81, 129,  98, 106, 119, 120,
       119,  99, 105,  99, 115, 134, 161, 138, 180, 189, 198, 227, 195,
       188, 155, 157, 110, 102,  88,  59,  91,  82,  62,  72,  73,  75,
        70,  51,  62,  69,  53,  66,  60,  59,  66,  59,  57,  65,  48,
        61,  64,  50,  66,  67,  54,  48,  51,  42,  61,  73,  54,  64,
        47,  44,  66,  61,  56,  58,  56,  52,  43])
```

QUIZ IV DATA

Histogram Data

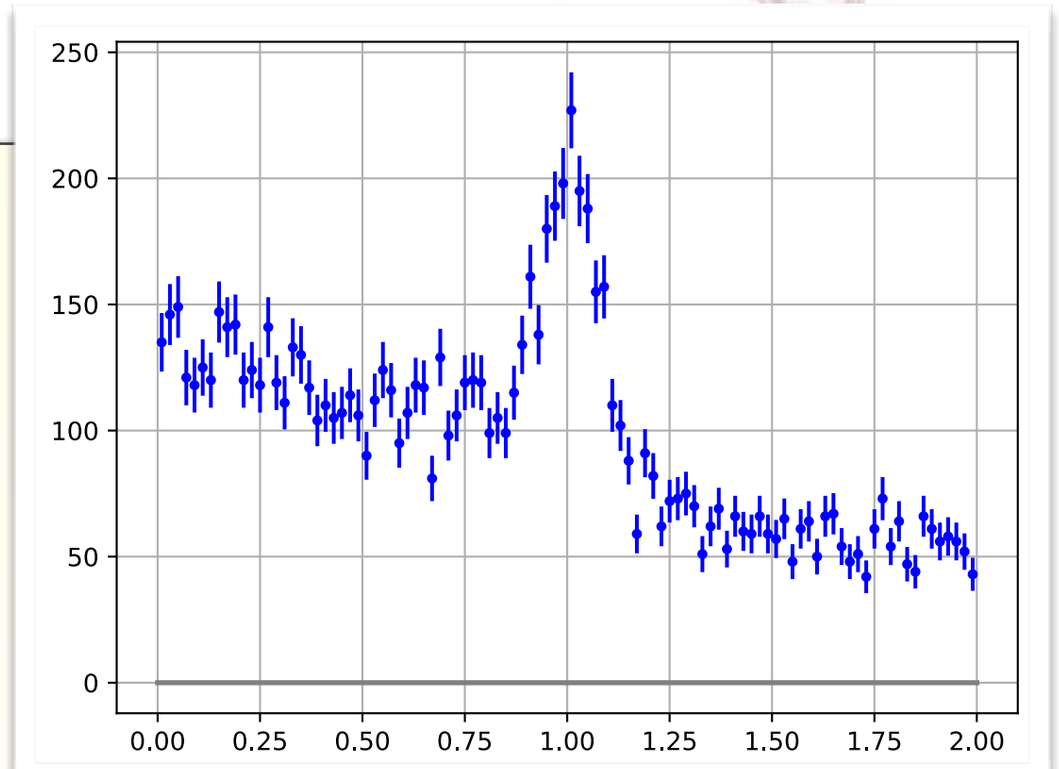
- The histogram data can be displayed with a simple code like:

```
import numpy as np
import matplotlib.pyplot as plt

data = np.load('hist_data.npy')

xmin, xmax, xbinwidth = 0., 2., 0.02
vx = np.linspace(xmin+xbinwidth/2,\
xmax-xbinwidth/2,100)
vy = data[0]
vyerr = vy**0.5

plt.errorbar(vx, vy, vyerr, c='blue', fmt = '.')
plt.plot([xmin, xmax],[0.,0.],c='gray',lw=2)
plt.grid()
plt.show()
```



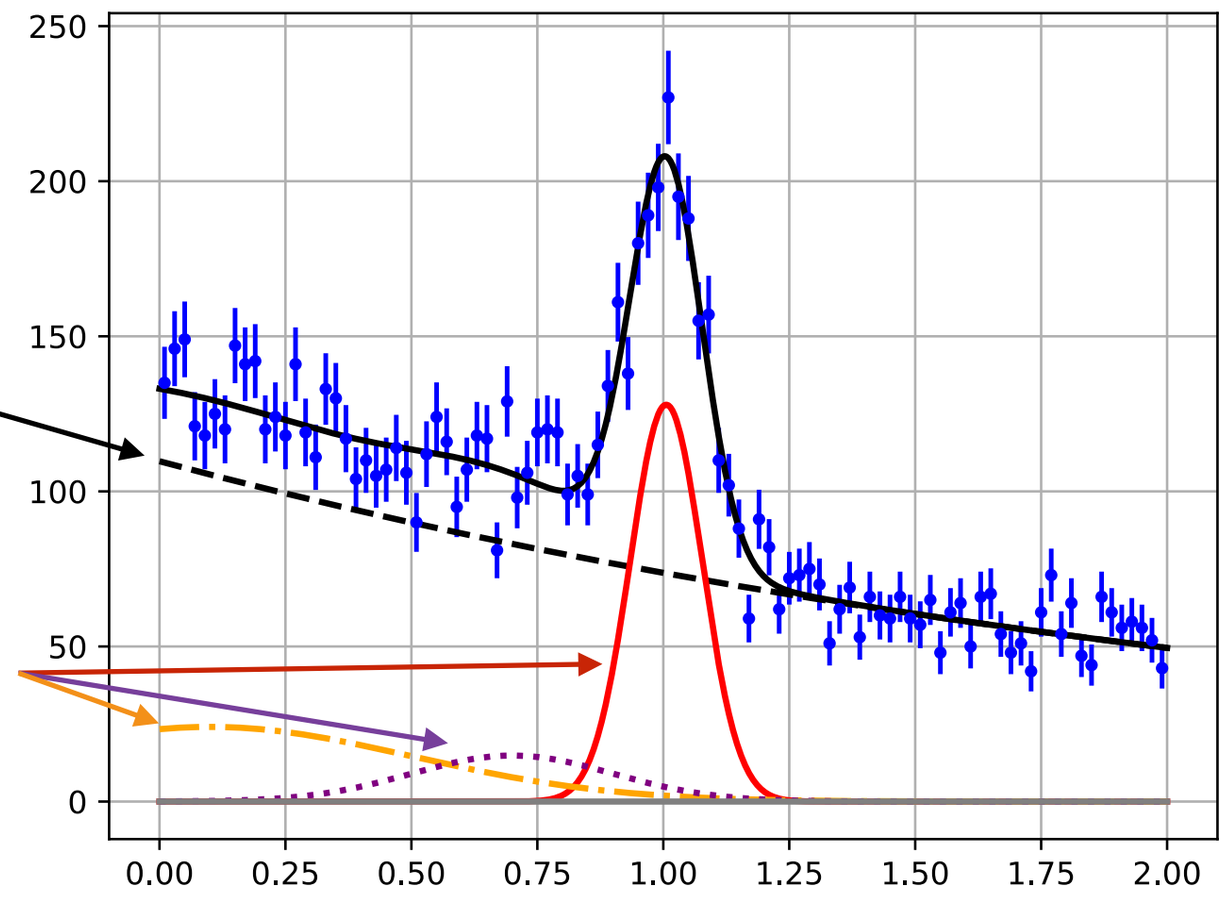
QUIZ IV

Modeling Histogram Data

- Suppose the given histogram data can be modeled by 4 components: 3 Gaussian functions + 1 exponential function.
- The parameters can be extracted with a χ^2 fit.

$$f(x;\tau) = \exp\left(\frac{-x}{\tau}\right)$$

$$g(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$



QUIZ IV

Modeling Histogram Data (cont.)

- Construct a function which takes a NumPy array of shape (100,) as the input argument, as the array represents the histogram data. Please perform a **χ^2 fit to the histogram data** with the model:

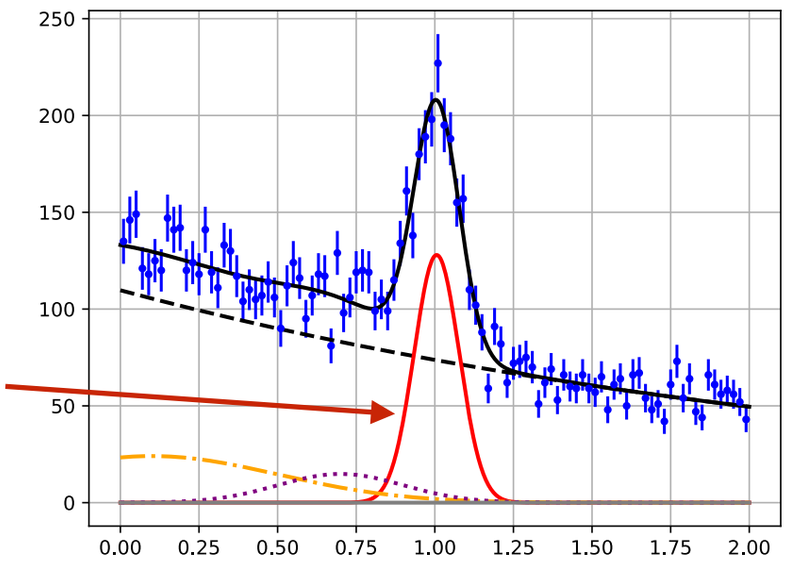
$$m(x) = N_1 \Delta x \cdot g(x; \mu_1, \sigma_1) + N_2 \Delta x \cdot g(x; \mu_2, \sigma_2) + N_3 \Delta x \cdot g(x; \mu_3, \sigma_3) + N_4 \cdot f(x; \tau)$$

Some given parameters:
(*other parameters are floated in the fit*)

Δx : bin width (=0.02)
$\mu_2 = 0.1$, $\sigma_2 = 0.4$
$\mu_3 = 0.7$, $\sigma_3 = 0.2$

and **return the parameters introduced for the main signal Gaussian [N_1, μ_1, σ_1]** as a NumPy array.

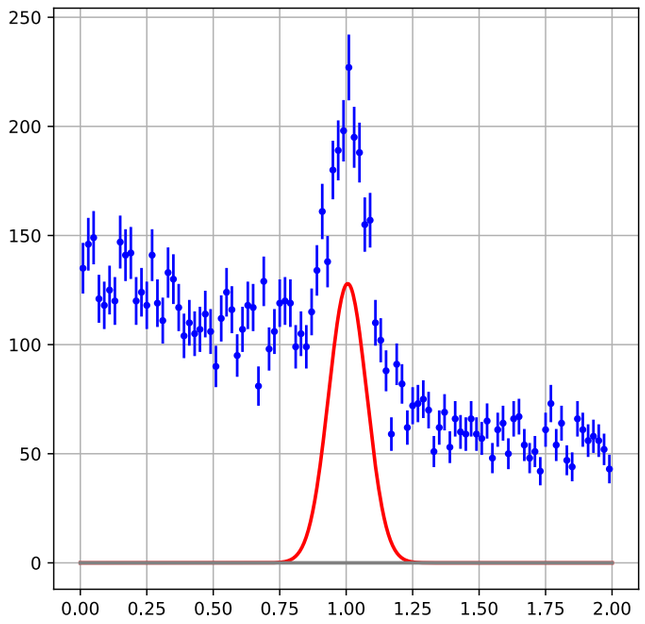
main signal Gaussian



QUIZ IV

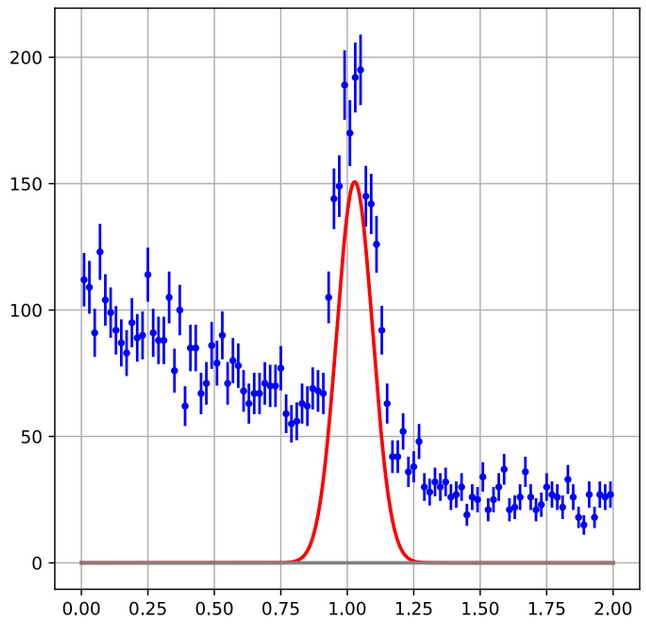
Test Solutions for first 3 histograms

#0: data[0]



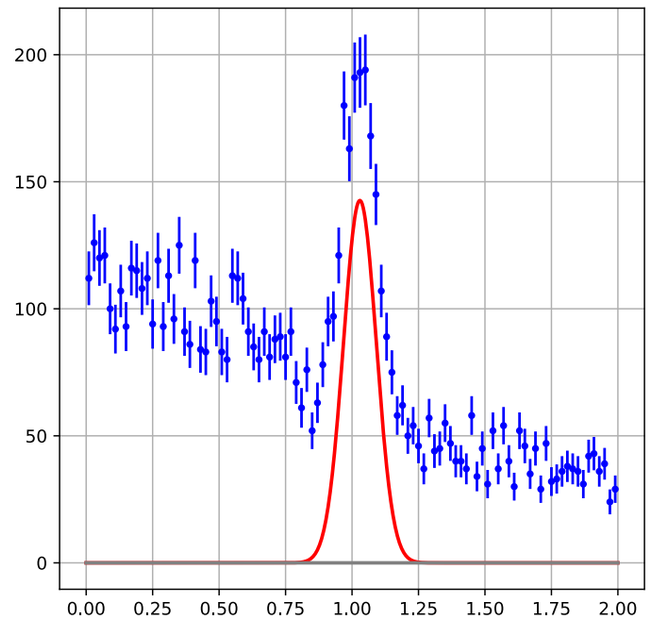

```
Area: 1147.7
Mean: 1.0058
Sigma: 0.0716
```

#1: data[1]




```
Area: 1279.3
Mean: 1.0281
Sigma: 0.0677
```

#2: data[2]




```
Area: 1105.4
Mean: 1.0295
Sigma: 0.0618
```

QUIZ V

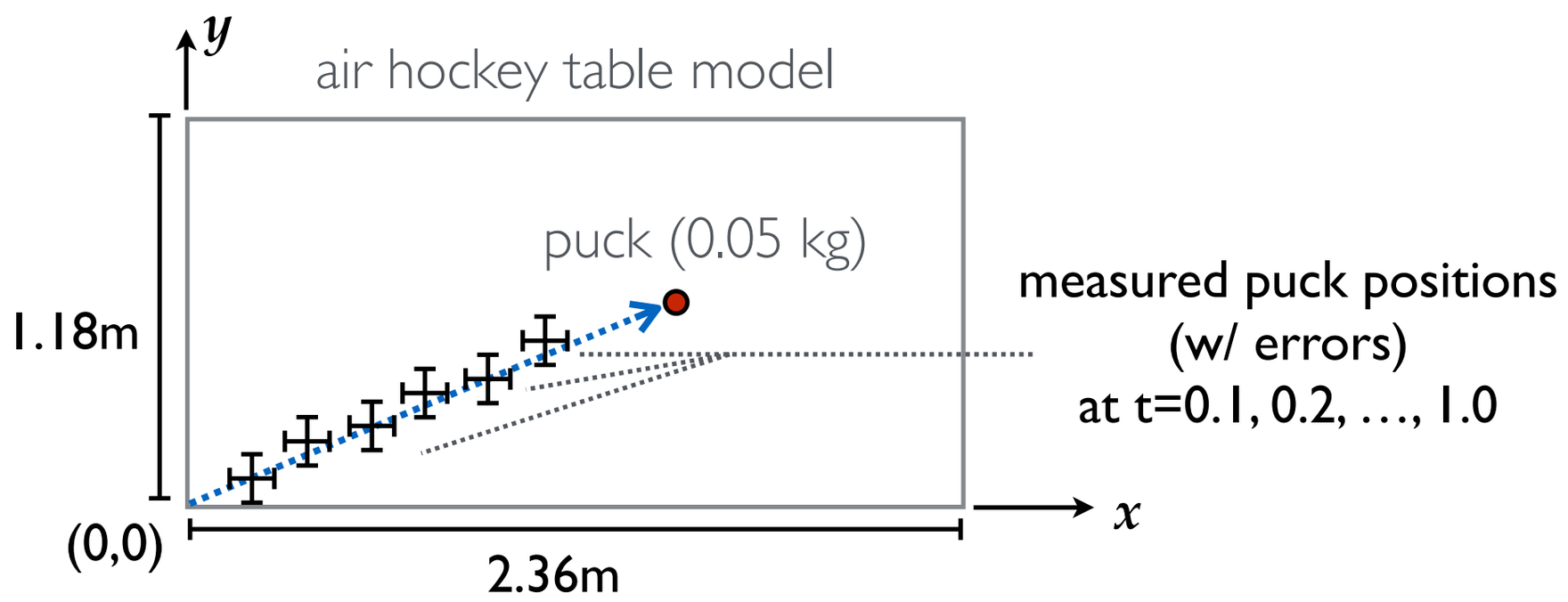
Tracking the air hockey puck

- There is an air hockey table and a puck moving on it. Suppose the puck is starting from one of the corner (set to origin) with unknown initial speed. The puck is moving on the table, hit the wall, and stop in the end.
- If the positions of the puck can be measured at a time interval of 0.1 sec for 10 times (*with some finite uncertainties*), **estimate where will the puck stop in the end?**

QUIZ V

Tracking the air hockey puck: the model

- Suppose the air hockey table and the puck are perfect — no friction, in the shape of perfect rectangle; the wall of the table can perfectly reflect the puck without energy loss. The only way to slow down the puck is due to the **air friction force which is the form of $-bv$**, where v is the speed of the puck, **b=0.01** $\mathrm{N \cdot s/m}$ The puck has a mass of 0.05 kg and can be considered to be point-like particle.



QUIZ V

Estimate the ending position

- Construct a function which takes a NumPy array of shape **10×5** as the input argument. The array represents the time, position of the puck and their uncertainties **(t, x, Δx, y, Δy)**.
- **Examine the input data, find the best-matching initial velocity of the puck, extrapolate the trajectory of the puck until it stops. Return the ending position of the puck.**
- Remark:
  - The puck starts from the origin **$(x,y) = (0,0)$**, and the ending position is defined by when the **speed of the puck is below 10^{-5}**.
  - The wall of the table is assumed to be perfect, so it just reflects the puck perfectly.
  - The "best-matching" indicates the smallest χ^2 value, given in the next slide.

QUIZ V

All the formulations

- The exact equations for the puck moving as well as the definition of best matching χ^2:

$$v_x = \frac{dx}{dt}$$

$$v_y = \frac{dy}{dt}$$

$$a_x = \frac{d^2x}{dt^2} = -b\frac{v}{m}\left(\frac{v_x}{v}\right)$$

$$a_y = \frac{d^2y}{dt^2} = -b\frac{v}{m}\left(\frac{v_y}{v}\right)$$

$$m = 0.05 \text{ kg}$$

$$b = 0.01 \text{ N} \cdot \text{s/m}$$

$$\chi^2 = \sum_i \frac{(x_f(t_i) - x_i)^2}{\Delta x_i^2} + \sum_i \frac{(y_f(t_i) - y_i)^2}{\Delta y_i^2}$$

The x_f and y_f are the expected positions at time t_i, given by the puck trajectory.

QUIZ V DATA

Test solutions

- The first set of data is looked like this:

```
>>> import numpy as np
>>> data = np.load('puck_data.npy')
>>> data[0]
array([[0.1       , 0.08758765, 0.02085707, 0.09453228, 0.02393222],
       [0.2       , 0.10626263, 0.02356729, 0.19662216, 0.03774669],
       [0.3       , 0.13792858, 0.03979937, 0.28798386, 0.03761879],
       [0.4       , 0.23550102, 0.0257037 , 0.31334829, 0.03567852],
       [0.5       , 0.29160824, 0.02565471, 0.34575759, 0.03645099],
       [0.6       , 0.26821037, 0.03046127, 0.51570562, 0.0327782 ],
       [0.7       , 0.37801904, 0.02335146, 0.57017269, 0.03777084],
       [0.8       , 0.40881692, 0.03962285, 0.62615169, 0.02225559],
       [0.9       , 0.4493256 , 0.03436981, 0.71439707, 0.02727523],
       [1.        , 0.53290056, 0.03382465, 0.79044566, 0.0290102 ]])
```

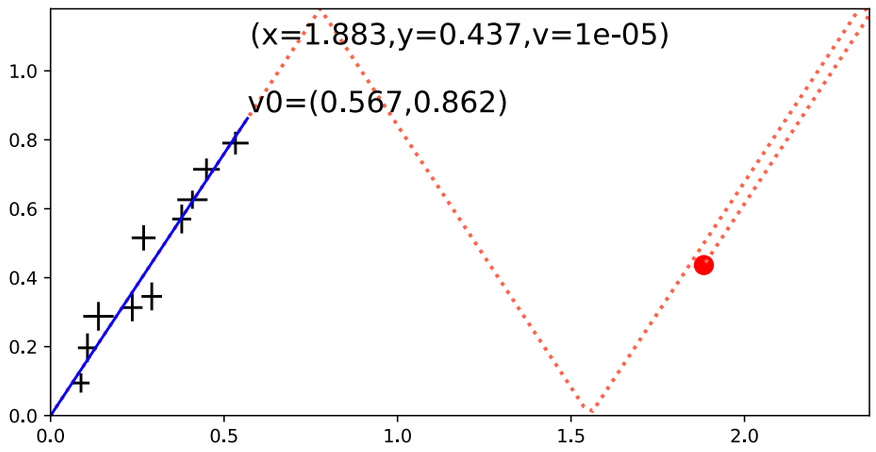
(t, x, Δx, y, Δy)

QUIZ V DATA

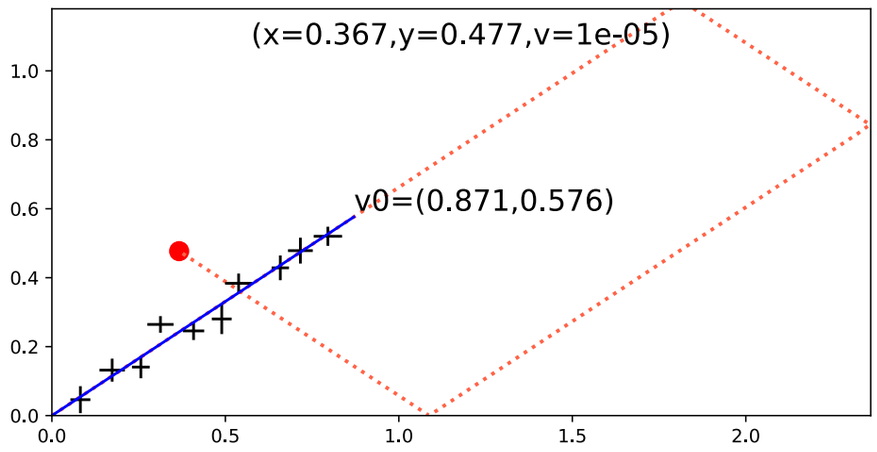
Test solutions

- First 4 ending positions (solutions) are given:

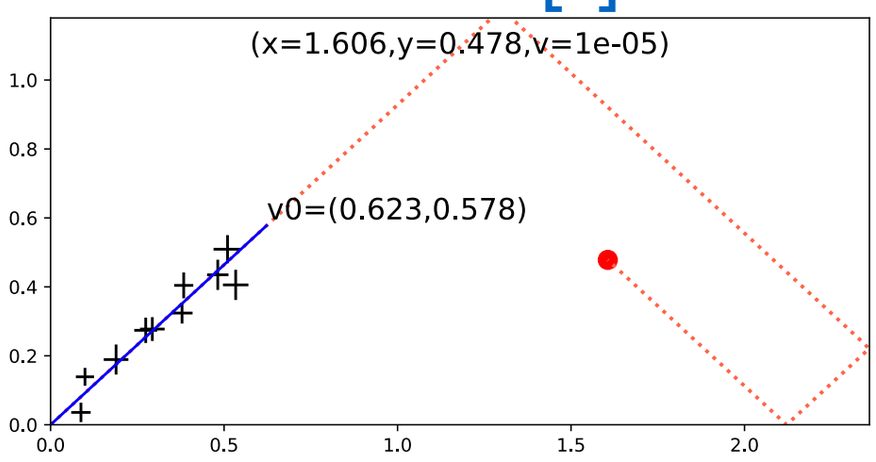
#0: data[0]



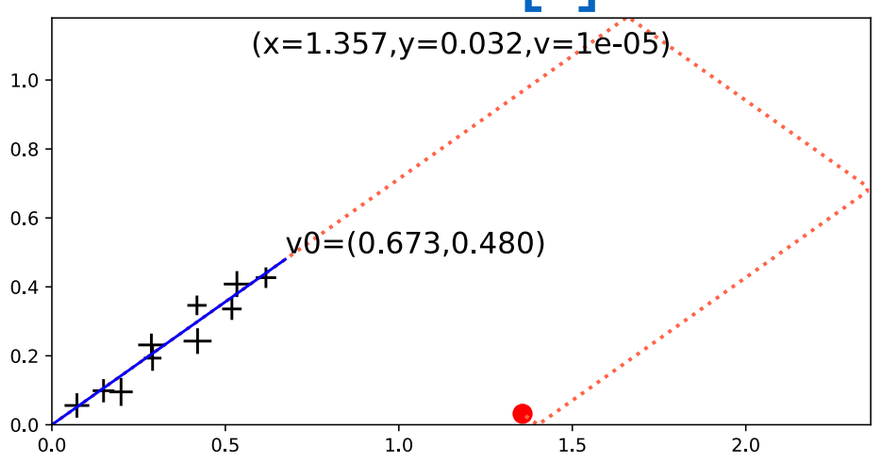
#1: data[1]



#2: data[2]



#3: data[3]



Making the plot is not necessary. You just need to return the ending position.