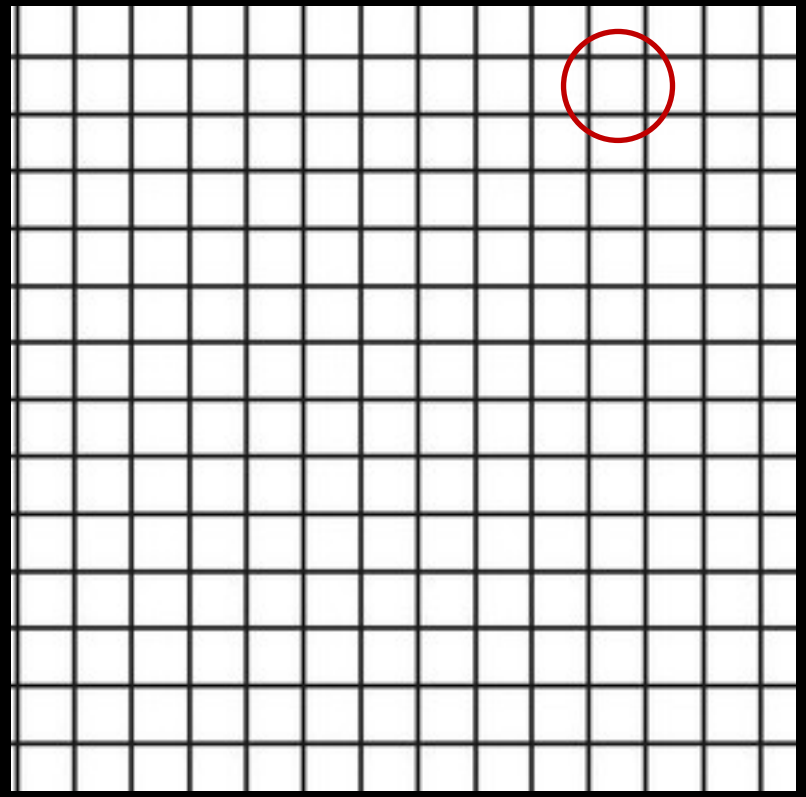
Uncertain Isocontour (2D)

(A slice of hurricane pressure dataset)

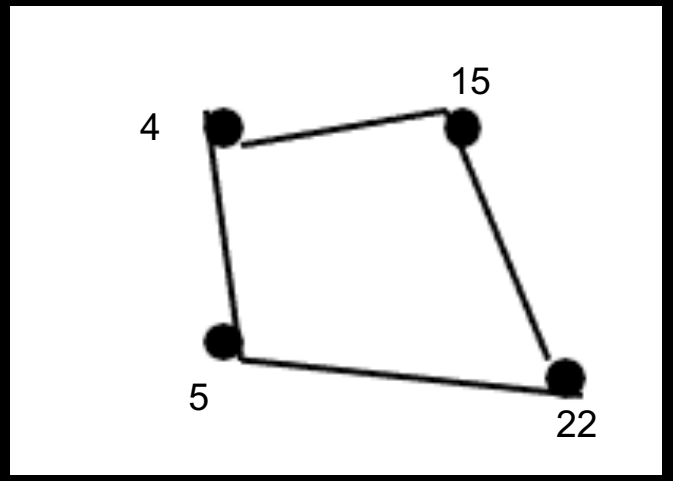
A Regular Grid Dataset



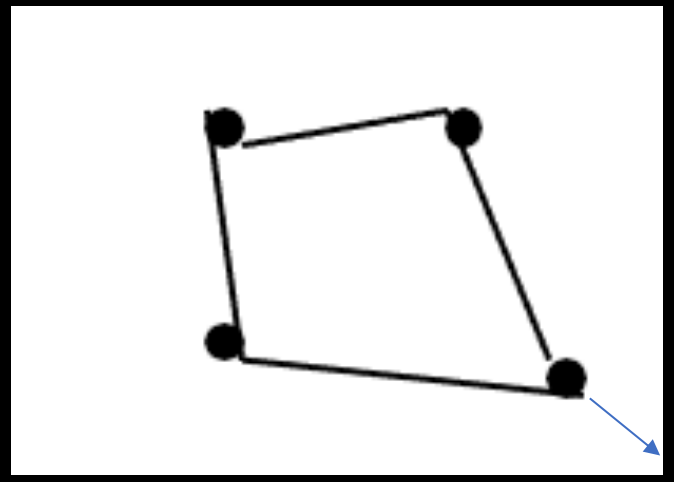
One cell

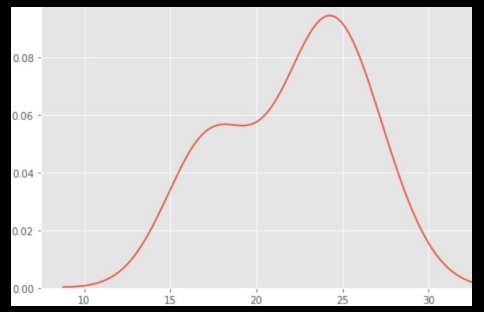
Dataset with and without Uncertainty

Without uncertainty

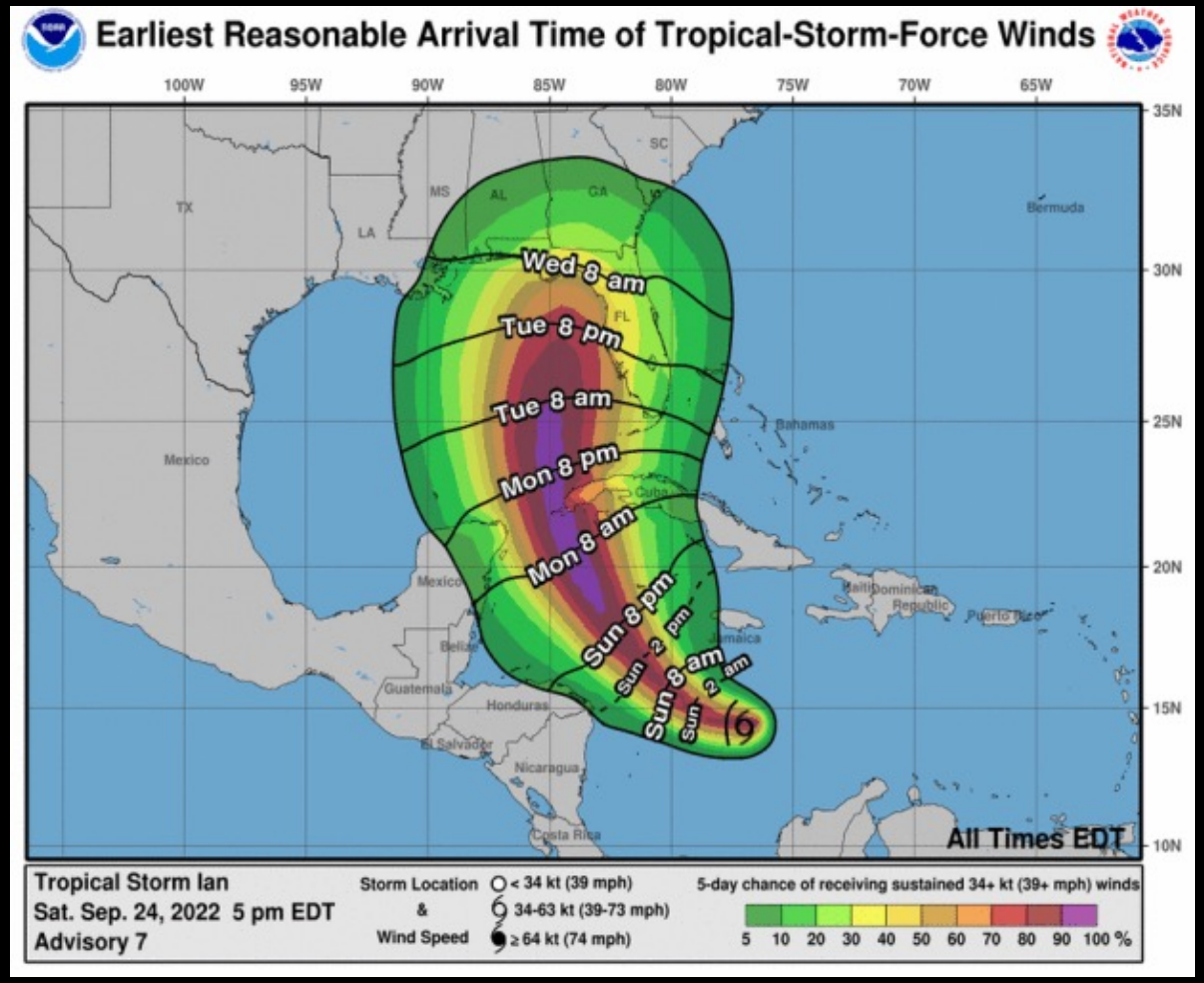


With uncertainty
(The data value is represented by a distribution)



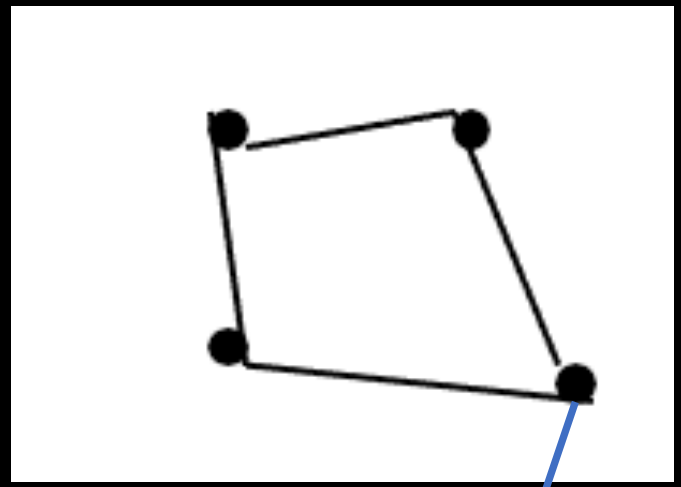


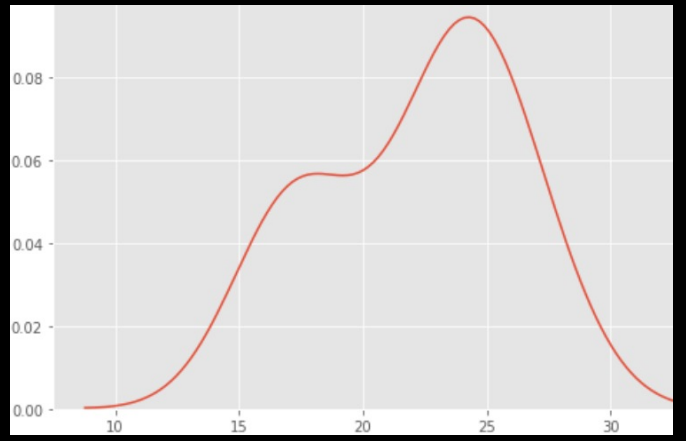
An Example of Uncertain Information (dataset)



What does a distribution mean?

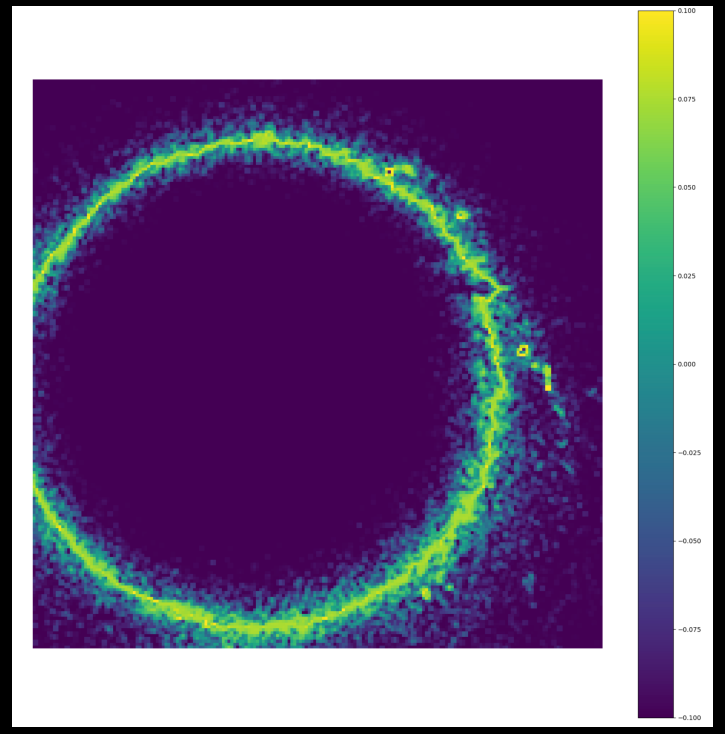
- If you draw 100 samples from the distribution, you could get many sample with value around 25, only a few sample with values around 30, and almost no samples with values less than 10





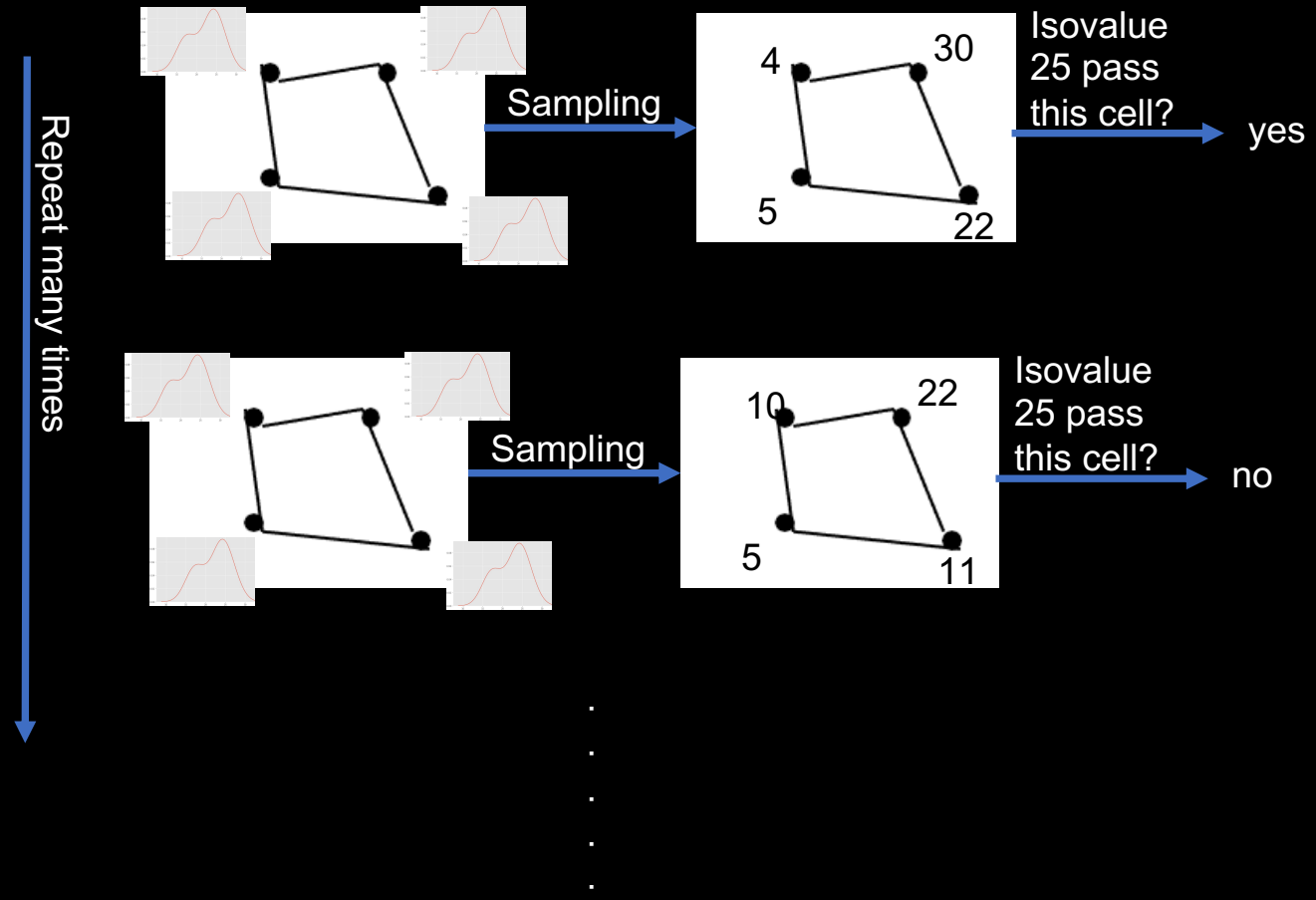
How to compute an ‘isosurface’ from an uncertain dataset

- In an uncertain dataset, you can not get a ‘certain’ isosurface (3D) or isocontour (2D)
- You can only compute a probability field to represent 'where (which cell)' has a higher chance that the isovalue will pass
- The figure shows a probability field of isovalue with value ‘0’ of an uncertain hurricane pressure dataset



Basic Idea to Compute Uncertain Isosurface

To compute the probability that isovalue 25 pass a cell?



Calculate the ratio that the isovalue passes this cell in the resampling process

If you run this process for every cell in the uncertain dataset, you will get a probability field

Homework

- Clearly, the resampling process has some drawbacks
  - We have to resample a lot of times to allow the result converge
  - Then, it take a lot of time (many cells * number of resampling)

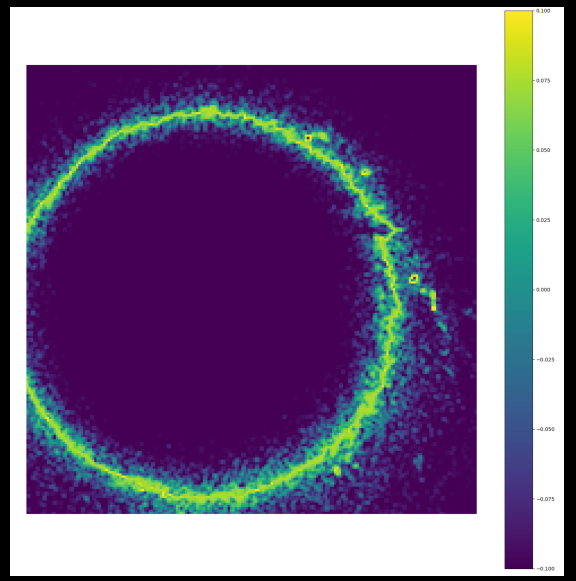
Homework: Files

- Isocontour.ipynb: template
- rawData.npy and rawDataSd.npy: uncertain dataset
  - It is an uncertain hurricane pressure dataset (2D regular grid)
  - Data value on every grid point is represented by a Gaussian distribution (mean and standard deviation)

Homework: Requirement

- Complete Isocontour.ipynb to
  - Compute probability field of an isovalue efficiently
  - You can still use the above resampling process to complete this homework. But we will take some points from you

Example: uncertain isosurface with value 0



Template

Main procedure. You can change the argument in "computePlotIsoContour" to test different isovalues

(Do not remove/change Initialize() and plt.show())

```
data2D = 0
plot = 0
dataSd2D = 0


##### x, y: location. Return: mean and standard deviation to represent the data at [x,y]
##### DO NOT modify this function
def getDataValue(x, y):
    return data2D[x, y], dataSd2D[x, y]

##### data loading and setup/plot image
##### DO NOT modify this function
def Initialize():
    global data2D
    global data2DPlot
    global plot
    global dataSd2D
    data2D = np.load("rawData.npy").transpose()
    dataSd2D = np.load("rawDataSd.npy").transpose()
    plot = np.zeros((data2D.shape[0]-1, data2D.shape[1]-1))

    plt.rcParams['figure.figsize'] = [20, 20]
    plt.axis('off')

##### (TODO) WORK on this function
##### compute and draw the uncertain isocontour of the given datavalue ("isovalue")
##### you should use "getDataVlue()" to get the data (Gaussian distribution) you want
##### Store the probability field in 'plot' to display
##### I do not mind the computation is efficnet or not
def computePlotIsoContour( isovalue ):
    #########TODO

    plt.imshow(plot, cmap='viridis', vmin=0, vmax=1) ###draw the probability field in 'plot'
    plt.colorbar()

##### main
Initialize()

##### You can modify this function call to test your program on different isovalues
computePlotIsoContour(0)

plt.show()
```

Template

This is the function you should complete. (I do not mind the efficiency of your implementation)

```
data2D = 0
plot = 0
dataSd2D = 0


##### x, y: location. Return: mean and standard deviation to represent the data at [x,y]
##### DO NOT modify this function
def getDataValue(x, y):
    return data2D[x, y], dataSd2D[x, y]

##### data loading and setup/plot image
##### DO NOT modify this function
def Initialize():
    global data2D
    global data2DPlot
    global plot
    global dataSd2D
    data2D = np.load("rawData.npy").transpose()
    dataSd2D = np.load("rawDataSd.npy").transpose()
    plot = np.zeros((data2D.shape[0]-1, data2D.shape[1]-1))

    plt.rcParams['figure.figsize'] = [20, 20]
    plt.axis('off')

##### (TODO) WORK on this function
##### compute and draw the uncertain isocontour of the given datavalue ("isovalue")
##### you should use "getDataVlue()" to get the data (Gaussian distribution) you want
##### Store the probability field in 'plot' to display
##### I do not mind the computation is efficnet or not
def computePlotIsoContour( isovalue ):
    #########TODO

    plt.imshow(plot, cmap='viridis', vmin=0, vmax=1) ###draw the probability field in 'plot'
    plt.colorbar()

##### main
Initialize()

##### You can modify this function call to test your program on different isovalues
computePlotIsoContour(0)

plt.show()
```

Template

You can get the Gaussian distribution (mean and standard deviation) on a grid point (x,y) by this function

```
data2D = 0
plot = 0
dataSd2D = 0


##### x, y: location. Return: mean and standard deviation to represent the data at [x,y]
##### DO NOT modify this function
def getDataValue(x, y):
    return data2D[x, y], dataSd2D[x, y]

##### data loading and setup/plot image
##### DO NOT modify this function
def Initialize():
    global data2D
    global data2DPlot
    global plot
    global dataSd2D
    data2D = np.load("rawData.npy").transpose()
    dataSd2D = np.load("rawDataSd.npy").transpose()
    plot = np.zeros((data2D.shape[0]-1, data2D.shape[1]-1))

    plt.rcParams['figure.figsize'] = [20, 20]
    plt.axis('off')

##### (TODO) WORK on this function
##### compute and draw the uncertain isocontour of the given datavalue ("isovalue")
##### you should use "getDataVlue()" to get the data (Gaussian distribution) you want
##### Store the probability field in 'plot' to display
##### I do not mind the computation is efficnet or not
def computePlotIsoContour( isovalue ):
    #########TODO

    plt.imshow(plot, cmap='viridis', vmin=0, vmax=1) ###draw the probability field in 'plot'
    plt.colorbar()

##### main
Initialize()

##### You can modify this function call to test your program on different isovalues
computePlotIsoContour(0)

plt.show()
```