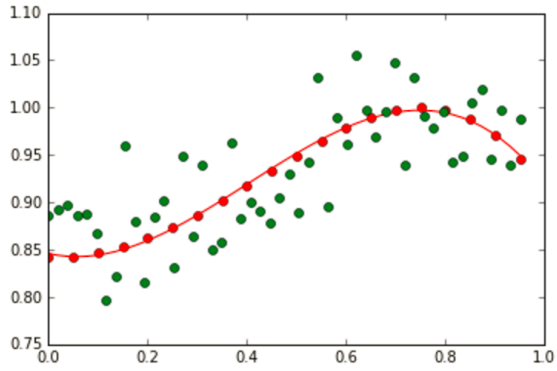
EE5907/EE5027: CA1

Deadline: September 30, 2022 at 5pm

Part 1: MAP

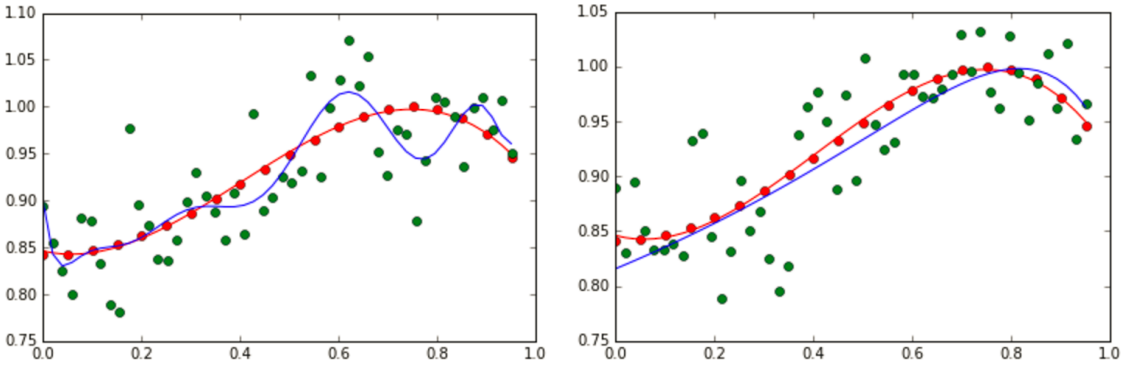
Write a program in python to implement the MAP (or regularization) for polynomial curve fitting problem. Follow the instructions below:

1. Generate 50 2D-data points using the following function: $y = \sin(x^2 + 1)$
2. Add Gaussian random noise to the data
3. Show the original curve line and the noisy data.



The red line is the original curve based on the equation. The green dots are the noisy data.

4. Fit the generated noisy data using the MAP as discussed in class.
5. Compute and display the total absolute error value (between the predicted and the correct ones) of using the computed w.
6. Display the estimated values of w
7. Experiment with your code by changing M and alpha (the coefficient of the regularization/prior term) to various values, and then show the plots. On each the plot, you must show the values of M and alpha.



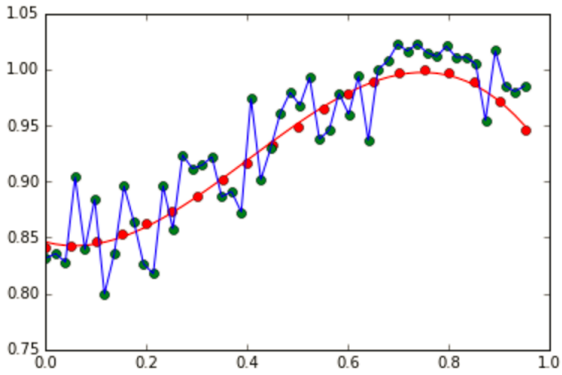
Left: An example of the overfitting problem when M = 10. Right: An example of how the regularization term reduces the overfitting problem (M =10, alpha = 0.4).

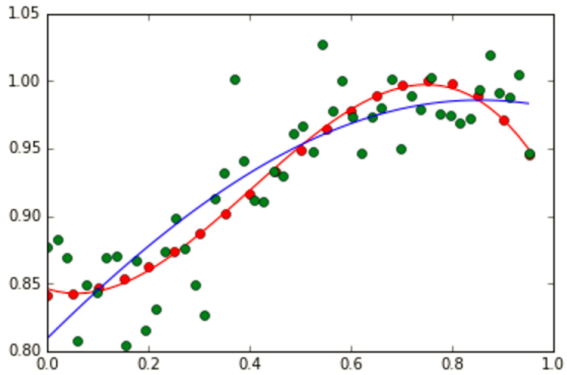
8. From the experiment in #7, discuss how M and alpha influence on the fitting accuracy.

Part 2: BASIS FUNCTION

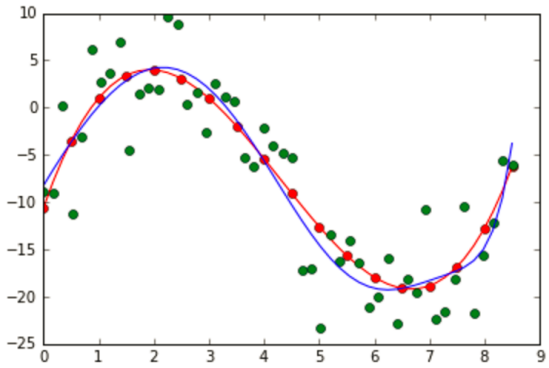
Write a program in python to implement the MLE that employs basis functions to solve the polynomial curve fitting problem. Follow the instructions below:

1. Generate 50 2D-data points using the following function: $y = \sin(x^2 + 1)$
2. Add Gaussian random noise to the data
3. Fit the generated noisy data using the MLE that employs the **Gaussian basis functions** as discussed in class.
4. Show your results for different values of M that generate overfitting and underfitting curves.





5. Change the basis functions to the **sigmoid basis functions**, and show the results for different values of M that generate overfitting and underfitting curves.
6. Change the original curve function to $y = 0.4345x^3 - 5.607x^2 + 16.78x - 10.61$, and use the sigmoid basis function to estimate the best curve fitting from the noisy data.

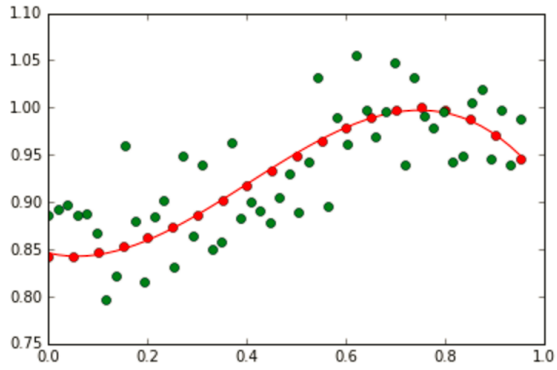


7. Experiment with different parameters of the Gaussian and sigmoid basis functions, and then show the plot. Also, discuss the advantages of these basis functions over polynomial functions.

Part 3: FULL BAYESIAN + PREDICTIVE DISTRIBUTION

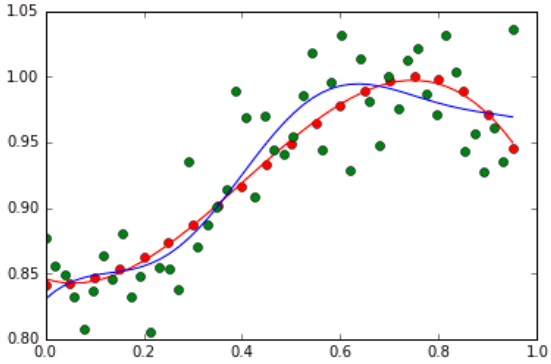
Write a program in python to implement the full Bayesian inference on Gaussian variables for curve fitting problem. Follow the instructions below:

1. Generate 50 2D-data points using the following function: $y = \sin(x^2 + 1)$. Add Gaussian random noise to the data. Show the original curve line and the noisy data.

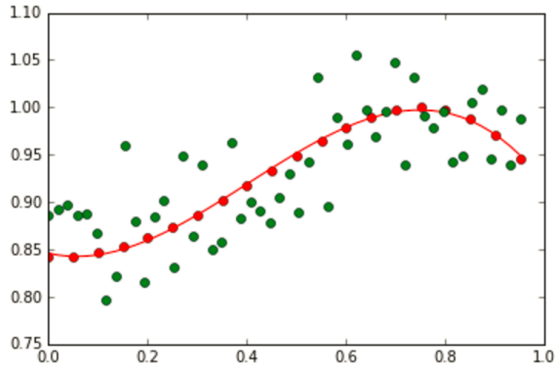


The red line is the original curve based on the equation. The green dots are the noisy data.

2. Compute w based on the full Bayesian inference (by using basis functions like discussed in class). Display the estimated values of w.
3. Experiment with your code by changing alpha and beta. Discuss the meaning of them with respect to the curve fitting results.
4. Show your best fitting, similar to:

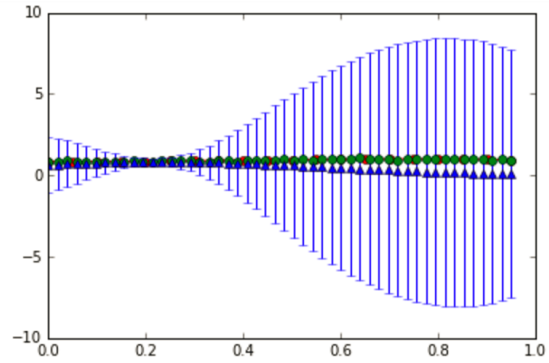


5. Explain how useful p(w|t) for the training and testing stages.
6. Generate 50 2D-data points using the following function: $y = \sin(x^2 + 1)$. Add Gaussian random noise to the data. Show the original curve line and the noisy data.

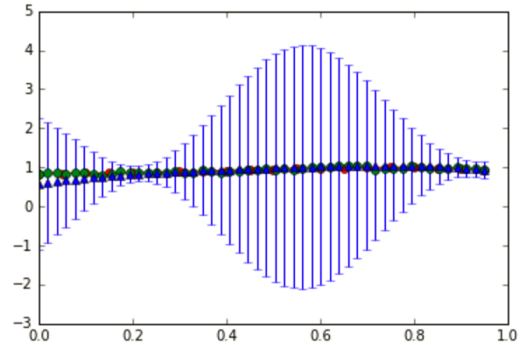


The red line is the original curve based on the equation. The green dots are the noisy data.

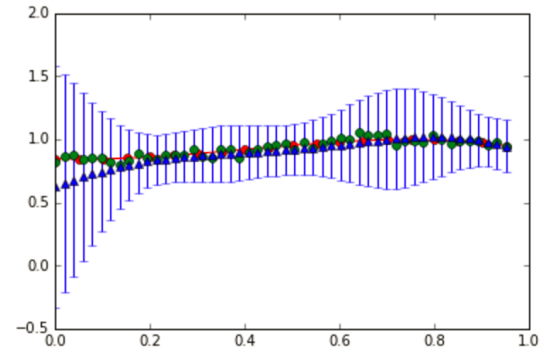
7. Compute the predictive distribution of every input data sequentially, where each input data is taken randomly from the noise data. Show your best prediction results for all 50 data one by one:



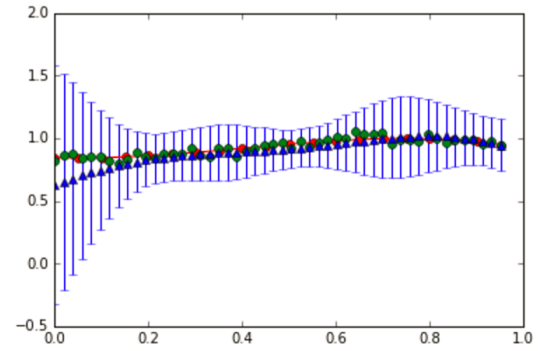
[11]
1



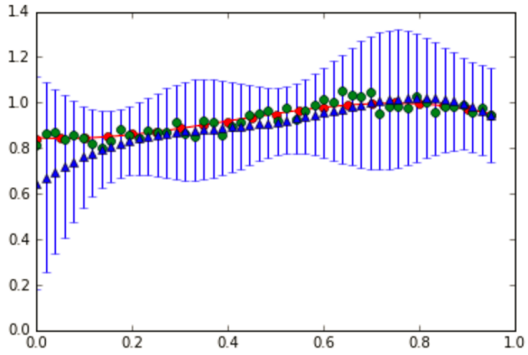
[11, 47]
2



[11, 47, 25]
3

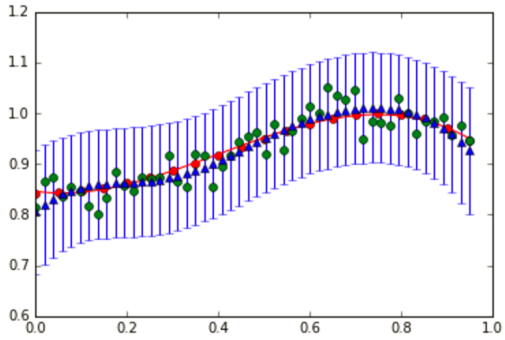


[11, 47, 25, 26]
4

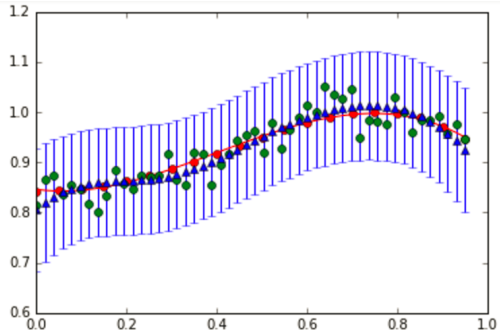


```
[11, 47, 25, 26, 7]
5
```

...
...
...



```
[11, 47, 25, 26, 7, 16, 5, 19, 17, 39, 11, 31, 3, 12, 21, 5, 5, 29, 1, 29, 4, 6, 3, 4, 14, 12, 0, 18, 9, 6, 19, 13,
 1, 13, 1, 4, 1, 2, 8, 7, 2, 2, 5, 2, 5, 1, 2, 0, 1]
49
```



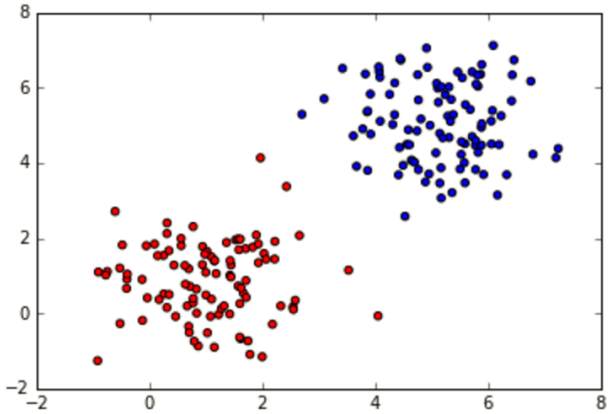
```
[11, 47, 25, 26, 7, 16, 5, 19, 17, 39, 11, 31, 3, 12, 21, 5, 5, 29, 1, 29, 4, 6, 3, 4, 14, 12, 0, 18, 9, 6, 19, 13,
 1, 13, 1, 4, 1, 2, 8, 7, 2, 2, 5, 2, 5, 1, 2, 0, 1, 0]
50
finish!
```

8. Explain why the predictive distribution is better than the original form of the full Bayesian inference.
9. Discuss the differences between p(t_new|t) and p(w|t).

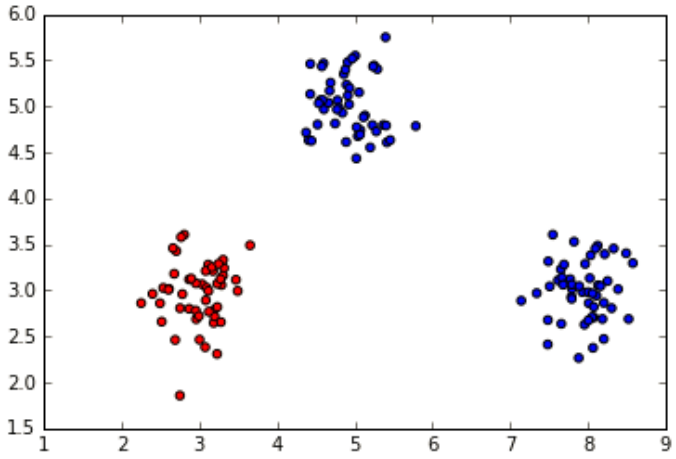
Part 4: CLASSIFICATION USING FULL BAYESIAN + PREDICTIVE DISTRIBUTION

1. This is an example code to generate classification data of two classes

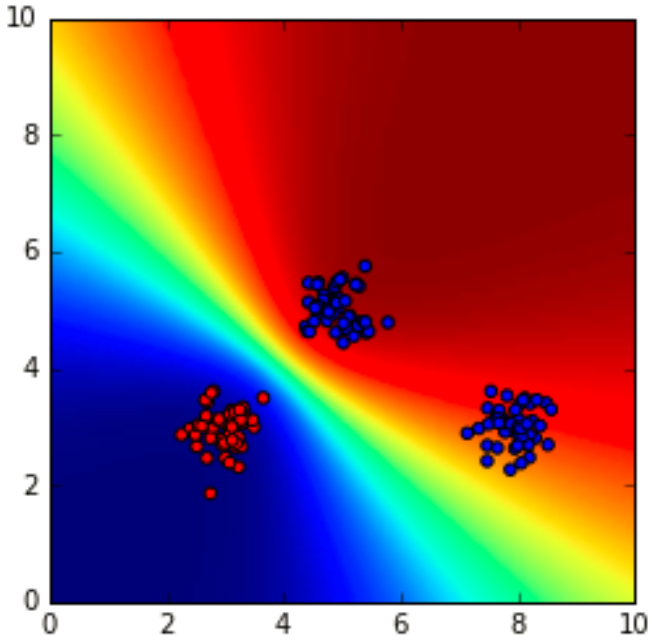
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np


N1 = 100
N2 = 100
K = 2
sigma = 1.0

mean = (1, 1)
cov = [[sigma, 0], [0, sigma]]
X1 = np.random.multivariate_normal(mean, cov, N1)
c1 = ['red'] * len(X1)

mean = (5, 5)
cov = [[sigma, 0], [0, sigma]]
X2 = np.random.multivariate_normal(mean, cov, N2)
c2 = ['blue'] * len(X2)

X = np.concatenate((X1,X2))
color = np.concatenate((c1,c2))

T = 0* np.ones([len(X),K])
for n in range(0,len(X)):
    if(n<len(X1)):
        T[n][0] = 1
    if(n>=N1 and n<len(X1)+len(X2)):
        T[n][1] = 1
T = T.astype(int)

plt.scatter(X[:, 0], X[:, 1],  marker='o', c=color)
plt.show()
```



2. Based on the code above, generate data similar to:



3. Implement the predictive distribution, where the output is a decision whether a given point belongs to C1 or C2. Based in the generated data and your implementation, compute the probability map:



4. Ask the user to enter any new value of x, and your task is to provide the decision whether it belongs to C1 or C2, and its uncertainty information.

Submission

Submit both your code and results in an ipython-notebook format via Luminus by the deadline.