Assignment Solution

SE2324: Mathematical Foundation of Computer Sciences(Spring 2021)

Student Name, ID1001001001

May 23, 2021

1 README

...

2 NumPy Warm-up (15 points)

Consider the following one-dimensional *Gaussian probability distribution*, also known as the Normal distribution or bell curve distribution:

$$G(x \mid \mu, \sigma) = \frac{1}{Z} \exp\left[-\frac{1}{2}\frac{1}{\sigma^2}(x-\mu)^2\right] \quad \text{where } Z = \sqrt{\frac{\pi}{\frac{1}{2}\frac{1}{\sigma^2}}}$$

Here, the function $G : \mathbb{R} \to \mathbb{R}$ takes as input a scalar x, and produces as output a scalar. The particular bell shape of G is determined by two parameters: the mean $\mu \in \mathbb{R}$ and the variance $\sigma^2 \in \mathbb{R}$

Numerically verify the following identity in Python:

$$\int_{\mathbb{R}} G(x)dx = 1$$

2.1 (10 points) Choose a few different one-dimensional Gaussian functions (by choosing different mean and variance values), plot them.

Draw four gaussian distribution graph based on 3σ-rule, the four graphs are shown as follows:

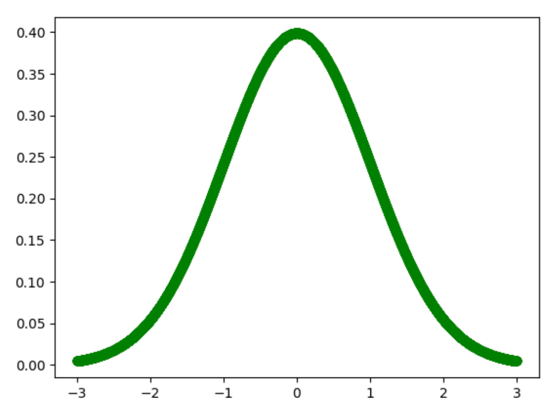

Figure 1 *Gaussian Distribution*($\mu = 0$ *and* $\sigma = 1$)

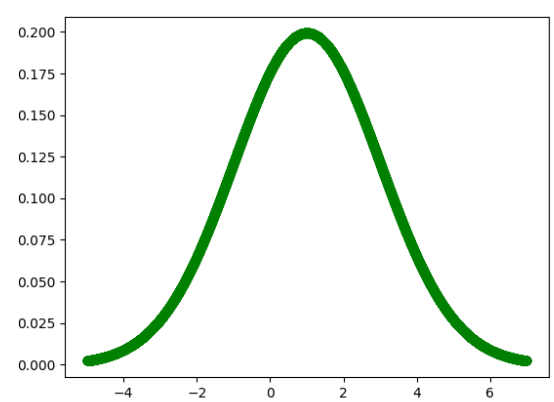

Figure 2 *Gaussian Distribution*($\mu = 1$ *and* $\sigma = 2$)

Put them together:

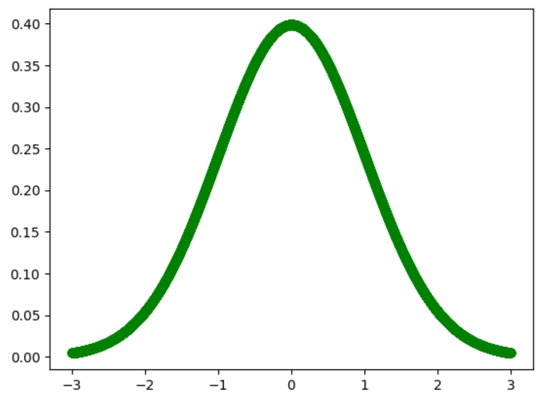

Figure 3 *Gaussian Distribution*($\mu = 5$ *and* $\sigma = 0.5$)

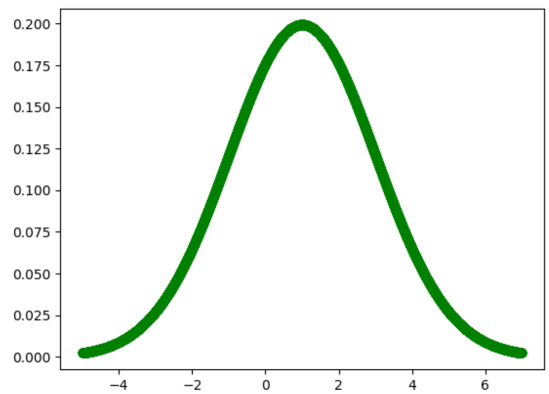

Figure 4 *Gaussian Distribution*($\mu = 10$ *and* $\sigma = 4$)

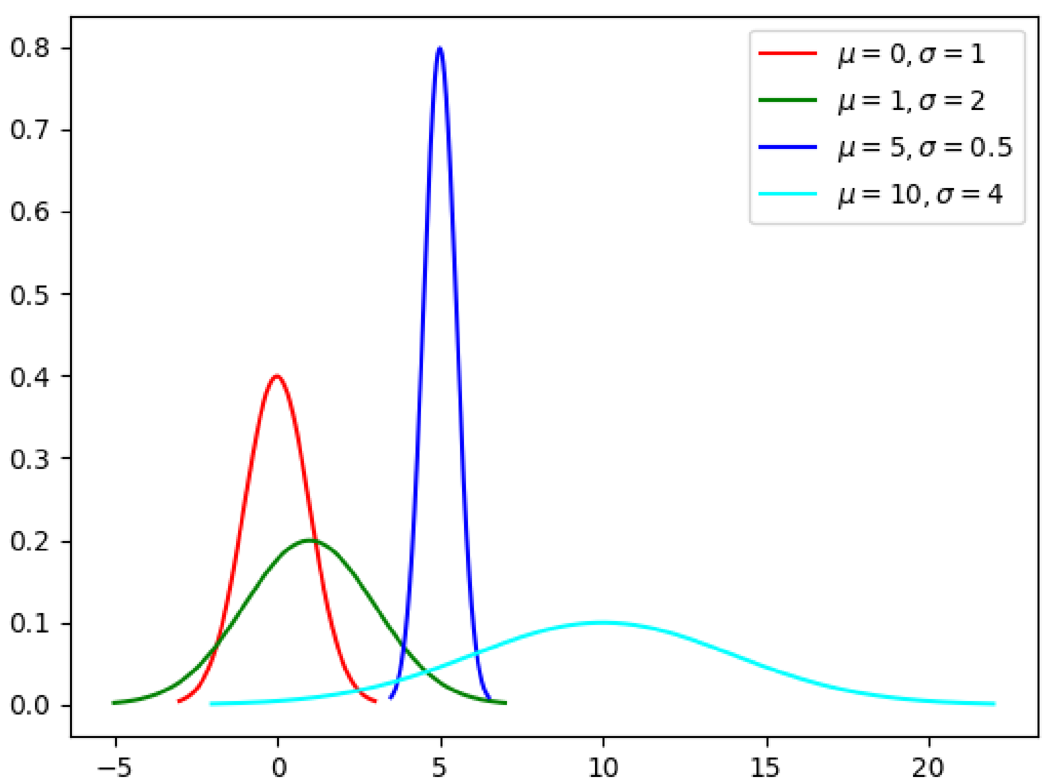

Figure 5 *Gaussian Distribution*($\mu = 10$ *and* $\sigma = 4$)

2.2 (10 points) Verify the above identity for each Gaussian function.

Using knowledge of calculus,I write a programme to verify the above identity for the four function mentioned above.The running result of the programme are as follows:

```
The integral of this function (mu = 0,sigma = 1) is:  0.9999266581491894
The integral of this function (mu = 1,sigma = 2) is:  0.9999266581491894
The integral of this function (mu = 5,sigma = 0.5) is:  0.9999266581491894
The integral of this function (mu = 10,sigma = 4) is:  0.9999266581491894
```

Figure 6 *Running result of the programme*

The integral region is $[\mu-4\sigma,\mu+4\sigma]$,according to the attributions of gaussian function,the result would be extremely close to 1.Apparently,with the region strenching,the result would infinitely approach to 1,which numerically verify the above identity.

Then I try to use Kahan Algorithm to reduce error,the running results are as follows:

```
The integral of this function (mu = 0,sigma = 1) is:  0.9999266581491878
The integral of this function (mu = 1,sigma = 2) is:  0.9999266581491878
The integral of this function (mu = 5,sigma = 0.5) is:  0.9999266581491878
The integral of this function (mu = 10,sigma = 4) is:  0.9999266581491878
```

Figure 7 *Running result of the programme*

The result would be much more precise with the help of kahan algorithm.

3 Numerics and Linear Algebra (60 points)

In this question, you will explore how the condition number of a matrix can have practical influence on which algorithms (e.g. LU, Cholesky) you can use to solve linear systems of the form: $A\vec{x} = \vec{b}$ where $A \in \mathbb{R}^{n\times n}$. We will study the Vandermonde matrix, as well as a matrix constructed from a finite Fourier Series basis. In this question, you will interpolate samples of the analytical function

$$f(x) = \frac{1}{1+x^2}$$

for $x \in [0,1]$. Your monomial interpolants (with $N = n+1$ terms) are given by

$$g_V(x) = \sum_{j=0}^{N} c_j x^j$$

and

$$g_F(x) = \sum_{j=1}^{N/2} c_j \sin(j\pi x) + \sum_{j=N/2+1}^{N} c_j \cos((j-N/2)\pi x).$$

Use $M = m+1$ uniformly sampled positions

$$x_i = ih,\ i = 0\ldots m,$$

with spacing $h = 1/m$, to generate M samples of the test function f,

$$f_i = f(x_i),\ i = 0 \dots m.$$

To estimate the polynomial coefficients $\vec{c}$, you will assemble and solve the linear systems

$$V\vec{c} = \vec{f}$$

and

$$F\vec{c} = \vec{f}$$

where V is the M-by-N Vandermonde matrix with entries

$$V_{ij} = (x_i)^j$$

and F is the finite Fourier Series basis matrix with entries

$$F_{i,j-1} = \begin{cases} \sin(j\pi x_i), & \text{if } 1 \le j \le N/2 \\ \cos((j - N/2)\pi x_i), & N/2 + 1 \le j \le N \end{cases}$$

For simplicity, assume $M = N$ for the entire problem.

3.1 (25 points) Use an LU solve (*scipy.linalg.lu* from SciPy package) to estimate the monomial coefficients $\vec{c}$. (You can solve a linear system $A\vec{x} = \vec{b}$ using NumPy API *numpy.linalg.solve*().) Report the residual L2 norm ($||r||_2 = ||Ax - b||_2$) for both linear systems when $N = 8$ and $N = 16$.

I choose to use the function linalg.lu with parament permute_l = False,which will calculate the matrices: P,L,U(P refers to permutation). With matrix P, the calculation would avoid deviding zero or a very tiny number.

This function will factorize the vandermonde matrix V into P, L and U, namely $N = PLU$

Thus the linear function $Vc = f$ would become $PLUc = f$,namely $LUc = P^{-1}f$

Then one function has been split into two:$Ly = M(let\ M = P^{-1}f)$ and $Uc = y$. Solving out these two function, then we will get the coeffecient matrix c

By doing so, the original $O(n^3)$ gaussian elimination algorithm has become $O(n^2)$, which is much more efficient.The residual L2 norm is shown as follows, the algorithm's performance is pretty good.

```
For Vc = f
When N = 8:
c =  [ 1.00000000e+00 -1.92980788e-03 -9.64341068e-01 -2.55940537e-01
  1.93414812e+00 -1.83752964e+00  7.29012818e-01 -1.03419883e-01]
Residual N2 norm =  7.021666937153402e-16
For Fc = f
When N = 8:
c =  [ 1.35488014 -0.04116264 -0.35430384  0.0015691   0.26057558  0.83058785
 -0.01057558 -0.08058785]
Residual N2 norm =  6.106226635438361e-16
```

Figure 8 *Running result of the programme*

```
For Vc = f
When N = 16:
c =  [ 1.00000000e+00  2.66980597e-07 -1.00001351e+00  2.93918340e-04
  9.96295647e-01  3.05535757e-02 -1.17587846e+00  7.32813488e-01
 -1.25244707e+00  5.13169268e+00 -9.56975249e+00  1.01178263e+01
 -6.70185081e+00  2.79542307e+00 -6.78607930e-01  7.36512945e-02]
Residual N2 norm =  2.575143099812927e-15
For Fc = f
When N = 16:
c =  [ 1.42902802e+00 -5.56448788e-02 -6.69283694e-01  8.98075222e-03
  1.45594520e-01 -7.09313611e-04 -9.39339281e-03  1.01294769e-05
  2.70446195e-01  1.05644809e+00 -2.32032131e-02 -3.50662948e-01
  2.87594191e-03  4.52018325e-02 -1.18923742e-04 -9.86977914e-04]
Residual N2 norm =  1.821747095556374e-15
```

Figure 9 *Running result of the programme*

3.2 (10 points) Using the *numpy.linalg.cond* function in NumPy, plot N vs. $cond(V)$ and N vs. $cond(F)$ for $N = 4, 6, 8, ...32$. Write a couple of sentences explaining the reasons for the trends in these two plots. Hints: Use a logarithmic scale in y axis for better clarity. Also, try wrapping the creation of V and F into functions that you can call repeatedly to generate the required output data. These functions will be helpful for the next part.

The graph of cond(V) and cond(F) is shown as follows:

For linear function $Ax = b$, $cond(A)$ can determine how sensitive the function is. A very big condition number means that very tiny changes of b would lead to greate change of x

Both $cond(V)$ and $cond(F)$ explodes with N increasing, that's because when N become too big, overfitting would usually occur. When overfitting, a tiny change will make big differences, the system is rather sensitive, thus possessing high condition number.

In addition, fourier series fit the function $f(x) = \frac{1}{1+x^2}$ better than polynomial. Thus, facing tiny changes, it's system is less sensitive, which means lower condition number. That explains why $cond(F)$ is always lower than $cond(V)$

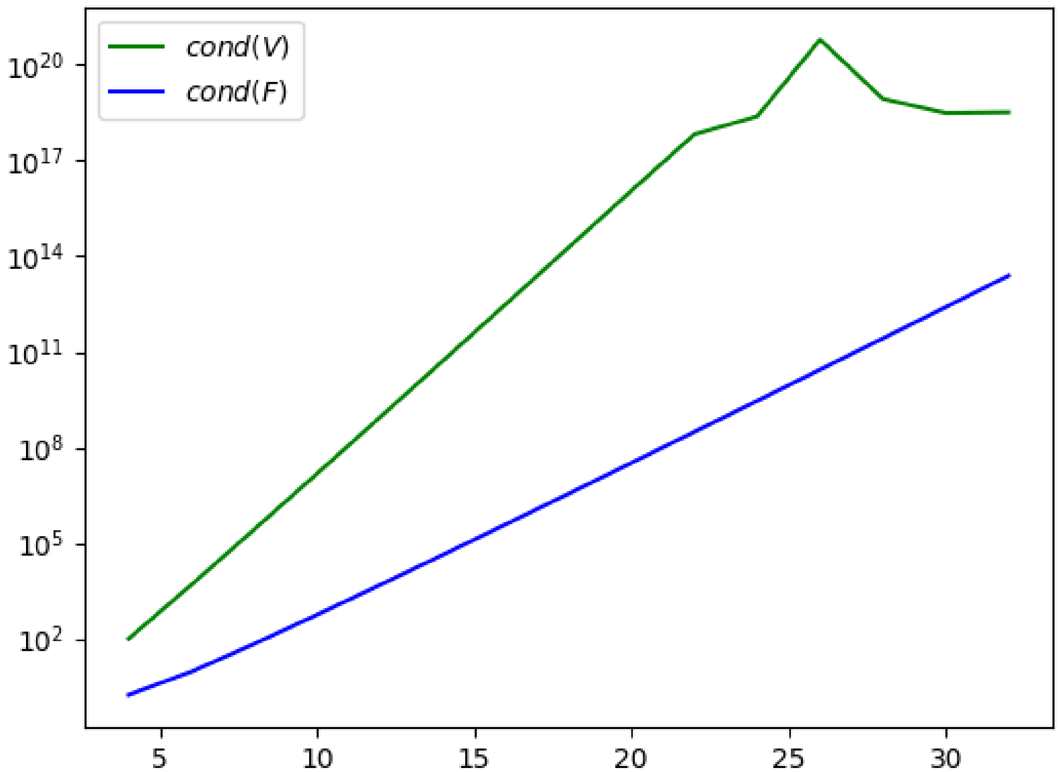

Figure 10 *cond*(V) *and cond*(F)

3.3 (15 points) A necessary condition for being able to use Cholesky factorization is that the matrix must be positive definite. Construct $A_V = V^T V$ and $A_F = F^T F$ for $N = 4, 6, \ldots 32$ (a total of 30 matrices). Mathematically, when would these matrices be positive definite? Explain. Using the *isposdef*() function introduced in Appendix **??**, check to which matrices NumPy reports as positive definite. Create a table of values that includes the following columns: N, *isposdef*(A_V), *isposdef*(A_F), cond(V), cond(F). What is the largest value of N where A_V is positive definite, and what is the condition number of that V? What is the largest value of N where A_F is positive definite, and what is the condition number of that F? Are these condition numbers connected in some way? If so, how?

3.4 (10 points) For $N = 8$, transform the linear systems above into positive definite systems according to Question 3.3, and use Cholesky factorization (*numpy.linalg.cholesky*()) to solve them. Report the residual L2 norm for each solution. Compare the residuals to Question 3.1: how does Cholesky compare to LU?

4 Least Squares Problems and QR (25 points)

In the question above, a $M \times M$ square linear system is solved for the interpolation coeffcients. The resulting interpolation function can provide a solution that pass through all the sample points. However, in some applications, finding such a solution could be too time-consuming and unnecessary, or even impossible (Consider two sample with the same x value and different y values). In such cases, we usually reduce the number of the interpolants (let $M > N$), and solve the resulting over-determined linear system using least square method. Instead of looking for the exact solution to an over-determined system, we look for the solution with the smallest error vector $V\vec{c} - \vec{f}$ (or $F\vec{c} - \vec{f}$) by minimizing the overall backward error:

$$\text{minimize } \|V\vec{c} - \vec{f}\|_2^2$$

4.1 (15 points) Solve the least square system with QR decomposition(*numpy.linalg.qr*()) when $M = 16$, $N = 4, 8$.

4.2 (10 points) Plot the g_V, g_F when $M = 16$, $N = 4, 8$, compare them with the analytical function $f(x)$ and the interpolation function obtained in Question 3.1.