Problem Set 5

Debendro Mookerjee
GitHub Account Name: Debendro-dm5790

October 16, 2023

Abstract

This problem set continues the exploration of numerical integration by considering the implementation of the gamma function as well as starting the topic of linear algebra. We again learn about various integration techniques such as rescaling the limits of integration and Gauss-Legendre integration. In the second exercise we see how singular value decomposition (SVD) can be used to fit data.

1 Introduction

An integral can be numerically computed by approximating it as a finite sum with each term being the product of the integrand evaluated at a sample point and the weight at that point. Some approximation methods involve weights with the same value, such as the rectangular rule, while other schemes involve weights with different values, such as the trapezoidal and Simpson's rules. Gauss-Legendre quadrature falls into the latter category. In these integration methods, the sample points x_i are the roots of a Legendre polynomial of a particular order. The weights depend on these polynomials as well.

Before using these powerful integration techniques, it is important to rewrite the integral we want to compute. This can involve expressing the integrand with unitless variables, rescaling the limits of integration into dimensionless ones, rewriting integrals ranging over an infinite interval to one ranging over a finite interval, and choosing the integrand's extremum to be at a particular value of the rescaled variable. Several of these techniques will be used in the first problem.

Consider the following matrix equation: $Ax = b$. Here A is called the design matrix and b is a known vector while the vector x is the only unknown entity. Suppose A is not an invertible matrix and that there is no x that satisfies the matrix equation. During these situations the goal is to determine an x that minimizes the norm of Ax - b. For the case of fitting a model to a two variable data set such as a time-dependent signal the known vector b becomes the data corresponding to the dependent variable (i.e. the signal) and the design matrix becomes a matrix function of the data corresponding to the independent variable (i.e. time). We want to fit the data to some linear combination of functions. Each function is implemented in a column of the design matrix. Performing SVD on the design matrix yields a pseudo inverse of A, allowing us to isolate x. This is the vector of coefficients that are part of the model based on

linear combinations. In the second exercise we will consider polynomial and Fourier fitting.

We will first describe the methods we used to answer the three exercises and then we will provide the suitable outputs.

2 Methods

In the first exercise, which involves numerically determining the gamma function values, we use two functions implemented by Mark Newman: the function gaussxw() which calculates and returns the sample points and the weights for a particular order N, which is the number of desired sample points, and the function gaussxwab() which calculates and returns the sample points and weights scaled to the desired integration limits. These two functions are used for Gauss-Legendre quadrature.

2.1 Exercise 1:

The goal of this exercise is to numerically compute the gamma function

$$\Gamma(a) = \int_0^\infty x^{a-1}e^{-x}dx$$

for various values of the constant a. We first implement a python function that returns the original integrand of the gamma function integral. We make three plots on a single graph for the cases where the a in the integrand is $2, 3$, and 4 for $0 < x < 5$. The generated plots reveal that the maximum occurs at $x = a - 1$.

We will now **compute the maximum** of the integrand $f(x) = x^{a-1}e^{-x}$. We see that the derivative is

$$\frac{df}{dx} = (a-1)x^{a-2}e^{-x} - x^{a-1}e^{-x} = e^{-x}x^{a-2}\big[(a-1) - x\big]$$

Setting this to zero gives us three candidates for the maximum: $x = \infty, x = 0$, and $x = a - 1$. For $x > 0$, the integrand is non-negative and takes on positive values. Therefore, $x = \infty$ cannot be the maximum. To determine the nature of the function at the remaining points we compute the second derivative which is

$$\frac{d^2f}{dx^2} = (a-1)(a-2)x^{a-3}e^{-x} - (a-1)x^{a-2}e^{-x} + x^{a-1}e^{-x} - (a-1)x^{a-2}e^{-x}$$

$$= x^{a-3}e^{-x}\big[(a-1)(a-2) - 2(a-1)x + x^2\big]$$

At $x = a - 1$ we see

$$(a-1)(a-2) - 2(a-1)x + x^2 = (a-1)(a-2) - 2(a-1)^2 + (a-1)^2$$

$$= (a-1)(a-2) - (a-1)^2 = (a-1)(a-2-a+1) = 1 - a$$

Thus **for $a > 1$, $x = a - 1$ is a maximum**.

Assuming $a > 1$, we see that $f(0) = 0$ and $f(a-1) = (a-1)^{a-1}e^{1-a} > 0$. Hence **for $a > 1$ the maximum of $f(x)$ occurs at $x = a - 1$.**

In order to perform numerical integration accurately, we need to make sure our implementation avoids numerical errors. The current form of the integrand can lead to numerical overflow and/or underflow in the computer. For instance as x gets larger and larger, x^{a-1} becomes very larger while e^{-x} becomes very small. In order to make the integral less difficult to evaluate, we can rewrite the integrand as

$$f(x) = e^{(a-1)\ln x - x}$$

We note that both terms in the exponent grow as x grows, with x growing faster than $\ln(x)$. Before this revision, the computer had to compute x^{a-1} and e^{-x} separately before multiplying them together, leading to the possibility of multiplying a very large number with a very small number. The computer does not evaluate this new form of the integral in the same manner. It computes $\ln(x)$ and x separately, subtracts them, and then exponentiates the result. The direct multiplication of a very large number and a very small number is avoided and so the new version is better than the old one. We note that in the exponent, we add two numbers with dissimilar magnitudes and opposite signs. This does not lead to severe numerical errors; only when two numbers with similar magnitudes and opposite signs are added, errors blow up.

For numerical integration, it is appropriate to recast the integration that is over an infinite interval to one that is over a finite interval. For this exercise, we will consider integrating from 0 to 1. In order to map the limits of integration from $\{0, \infty\}$ to $\{0, 1\}$ we make the following change of variables:

$$z = \frac{x}{x+c}$$

Thus we see that

$$\frac{dz}{dx} = \frac{(x+c) - x}{(x+c)^2} = \frac{c}{(x+c)^2} \rightarrow dz = \frac{cdx}{(x+c)^2}$$

We also see that the transformation implies that

$$zx + zc = x \rightarrow (z-1)x = -zc \rightarrow x = \frac{zc}{1-z}$$

Therefore we have

$$x + c = \frac{zc}{1-z} + \frac{c(1-z)}{1-z} = \frac{c}{1-z}$$

Thus we see that

$$dz = \frac{cdx}{\left(\frac{c^2}{(1-z)^2}\right)} = \frac{(1-z)^2 dx}{c}$$

or

$$dx = \frac{c}{(1-z)^2} dz$$

Thus we see the following for any integrand $f(x)$:

$$\int_0^\infty f(x)dx = \int_0^1 f\left(\frac{zc}{1-z}\right)\frac{c}{(1-z)^2}dz$$

For this exercise we want $z = \frac{1}{2}$ at $x = a - 1$. **Since $z = \frac{x}{x+c}$ this demands that $c = a - 1$.** Hence when $f(x)$ is the gamma function integrand, we want to compute the original integral as

$$\Gamma(a) = \int_0^1 f\left(\frac{zc}{1-z}\right)\frac{c}{(1-z)^2}dz\bigg|_{c=a-1} = \int_0^1 exp\left[(a-1)\ln\left(\frac{z(a-1)}{1-z}\right) - \frac{z(a-1)}{1-z}\right]\frac{a-1}{(1-z)^2}dz$$

We finally use Gauss-Legendre quadrature to compute this gamma function integral for a = 3/2, 3, 6 and 10.

2.2 Exercise 2:

We first use Python's pandas module to read the data file signal.data and convert it into a .txt file using the tocsv() method. Yes this is possible! We then open the text file signal.txt and fill the time and signal values into the Python list time and signal. We then convert these lists into numpy arrays called time and signal. We then plot the signal and save the plot as a .png file called SignalPlot.png.

We then implement singular value decomposition to determine the best cubic fit to the signal data. **We first re-express the time data by subtracting from each time value the mean and divide that by the standard deviation of the time data.** The adjusted time values are stored in a numpy array called tp. We then implement a design matrix A whose columns are the functions 1, tp, tp^2, and tp^3 and perform singular value decomposition to determine the coefficients in front of 1, tp, tp^2, and tp^3. We also print out the singular values and the condition number, which the largest singular value divided by the smallest one.

We then superimpose the cubic model on the original signal data and plot the residuals. **The residual plot shows a sinusoidal pattern instead of a randomized scattering of residuals, suggesting that the cubic model is poor. We also computed the chi-square statistic and found it to be about 4582.7. A large chi-square suggests that the model is poor.**

We then implement a polynomial model of degree 24 by redefining the design matrix and using the same adjusted time values. We plot the original signal and the new model. **From this plot, we see that the model roughly captures the oscillating behavior of the signal. From the residual plot for this new model, we see that the residuals have no pattern, a sign that the model is not as poor as the previous one. Superimposing the cubic residuals with the new residuals, we see that the new model's residuals are usually less than those of the cubic model. We also determine the condition number, which is approximately 6,267,712,864.4. Although this condition number is large it does not exceed the dynamic range of the 32-bit floating point numbers**. Thus the condition number is OK. We also compute and print the chi-square statistic, which is about 3720.87. Although it is larger, the statistic for this model is less than that of the cubic model.

To find the trigonometric fit, we determine and subtract off any linear trend by using singular value decomposition. To determine the Fourier frequencies in the signal without the linear trend, we perform a fast Fourier transform. The frequencies returned are not omega, they are f. **The frequency magnitudes are between 0 and 0.5 Hertz, assuming time is measured**

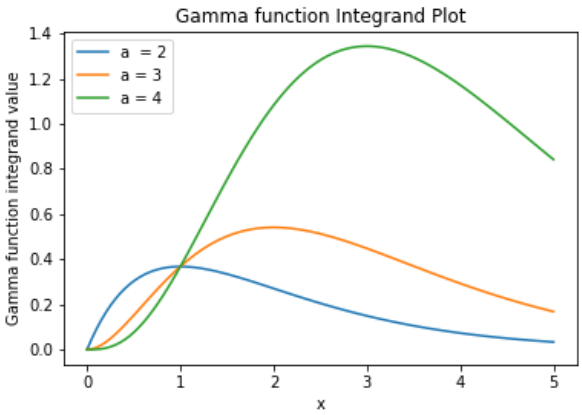
in seconds. Because the frequencies are bounded, there is an overall period to the signal, which can be seen by just looking at the original signal data.

We also make a plot of the frequencies versus the FFT amplitudes. We see that the plot is symmetric. **To determine the trigonometric model, we take the absolute values of the frequencies obtained from the FFT and create a design matrix of sines and cosines with all of the frequencies**. We perform singular value decomposition on the design matrix to determine the model. We superimpose this model onto the orginal signal and see almost perfect agreement. This reveals that the FFT was done correctly

3 Results

3.1 Exercise 1:

Here is the plot of the gamma function integrand for $a = 2, 3$ and 4.



The plots of the three integrands. We see that the maximum is located around $x = a - 1$ for each case.

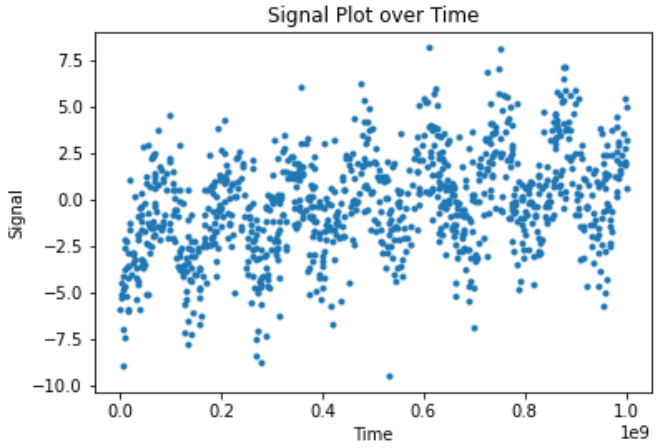
Here are the computed values of $\Gamma(3/2), \Gamma(3), \Gamma(6)$, and $\Gamma(10)$. The values seen below agree quite well with the expected values.

```
Gamma(3/2) is 0.8862694302378622
Gamma(3) is 2.0000022604228165
Gamma(6) is 119.99997528617405
Gamma(10) is 362880.23320973973
```

Values agree quite well with the expected values of 0.886, 2, 120, and 362880.

3.2 Exercise 2:

Here is the plot of the signal.



The original signal to be studied

Here is the relevant information for the cubic fitting.

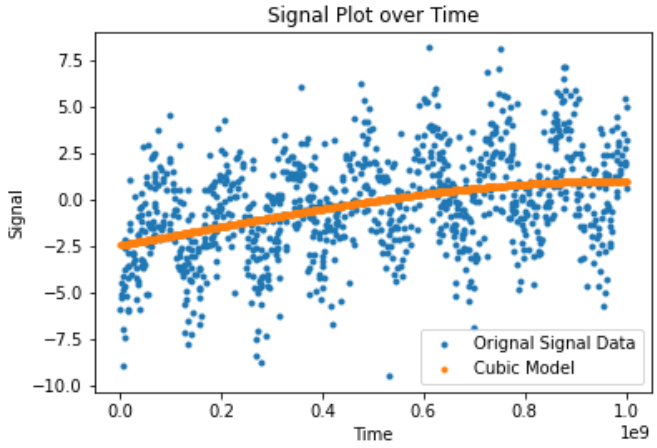
```
Coefficients of 1, t, t^2, and t^3 of best fit are
[-0.07814308  1.16874341 -0.2223711  -0.05547099]
The singular values are
[68.00312171 49.58519301 17.73756424 11.48710154]
The condition number is
5.919954783396497
```

The condition number is small, which is a good sign...for now.

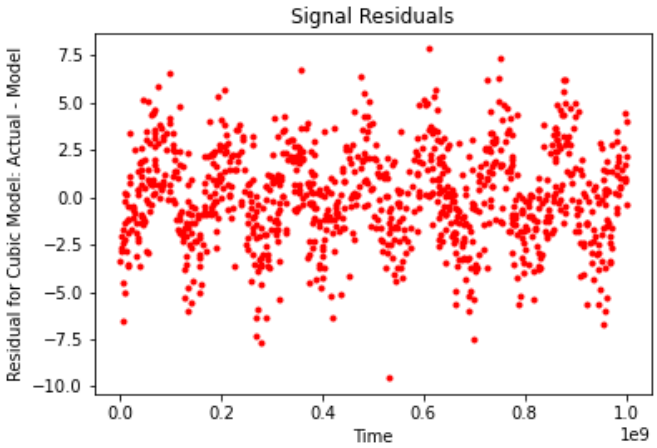
Here is the plot of the cubic model on the data along with the corresponding residuals and the chi-square statistic, which can be calculated from

$$\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i}$$

where O_i is the observed value and E_i is the expected/model value. This formula can be found in https://www.bmj.com/about-bmj/resources-readers/publications/statistics-square-one/8-chi-squared-tests.



As we can see, the cubic model does not capture the behavior of the actual signal



The residuals are not randomly scattered at all. The cubic model is a terrible approximation

```
Chi-square is 4582.711851078894
```

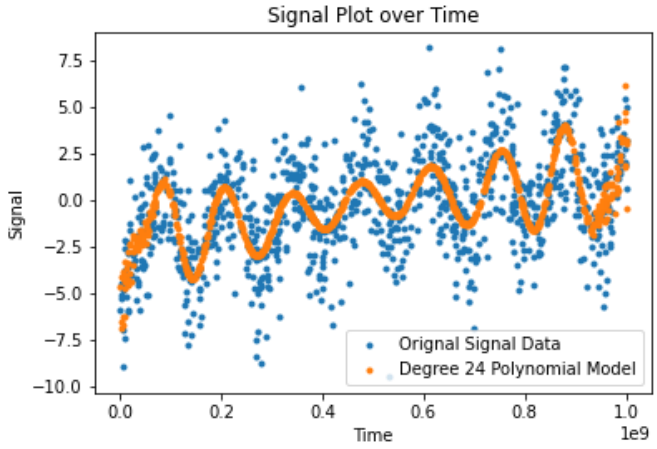
On top of all of this, the chi-squared statistic is very large.

Below is the relevant information on the degree-24 polynomial fitting.

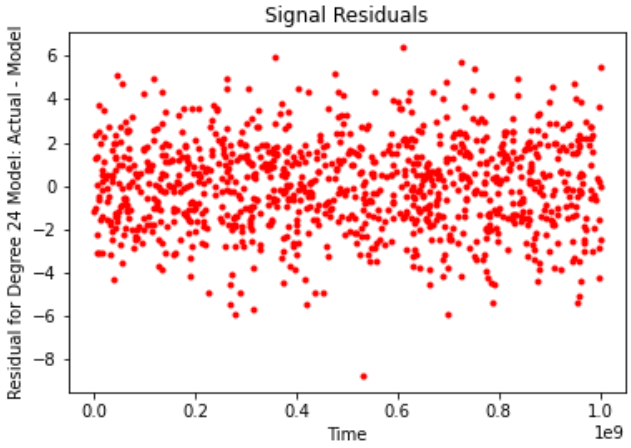
```
Coefficients of  best fit are
[ 6.16090098e-01 -1.07837077e+01 -4.96713341e+01  3.46083443e+02
  7.11384814e+02 -2.81799302e+03 -3.86877944e+03  1.03388324e+04
  1.04287909e+04 -2.07347815e+04 -1.59838841e+04  2.50957224e+04
  1.49476521e+04 -1.93126983e+04 -8.77617573e+03  9.65678583e+03
  3.20435045e+03 -3.12135713e+03 -6.80306677e+02  6.27699475e+02
  6.60686770e+01 -7.11797927e+01  9.53438376e-01  3.46221395e+00
 -5.06630569e-01]
The singular values are
[2.18086972e+06 1.23945048e+06 4.01324412e+04 2.38013792e+04
 1.60844092e+03 1.03754367e+03 1.46107177e+02 1.02947815e+02
 3.21707374e+01 2.33207453e+01 1.77558081e+01 8.95507136e+00
 5.97613436e+00 2.66573329e+00 1.74007126e+00 6.54879450e-01
 4.18476102e-01 1.33964393e-01 8.66854285e-02 2.37509264e-02
 1.41861375e-02 3.24336392e-03 1.92634535e-03 3.89579709e-04
 3.47953036e-04]
The condition number is
6267712864.403497
```

Although condition number is large, it is less than the dynamic range of the 32-bit floating point numbers, which we used when writing this program.

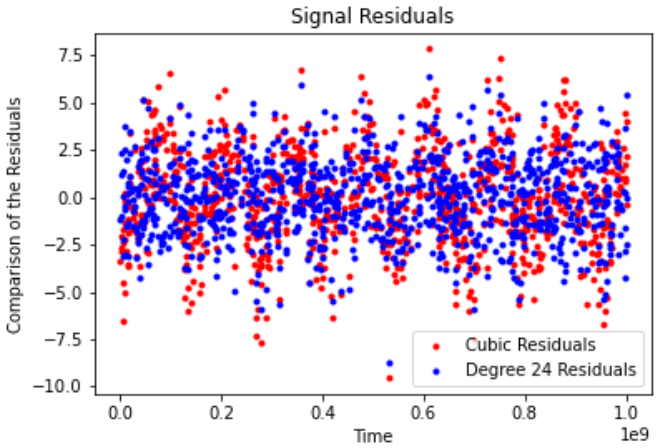
Here are the plots of the new model and the new residuals. We also provide a plot that compares the residuals of the two polynomial fits.



As we can see, the cubic model captures the behavior of the actual signal better than the cubic model did



The residuals are randomly scattered. The degree-24 model is a better approximation

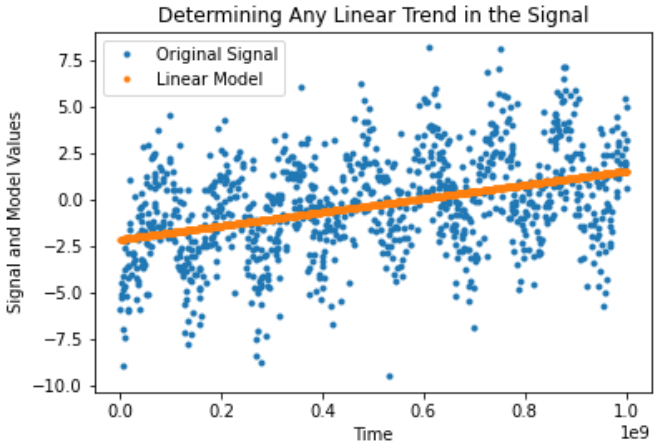


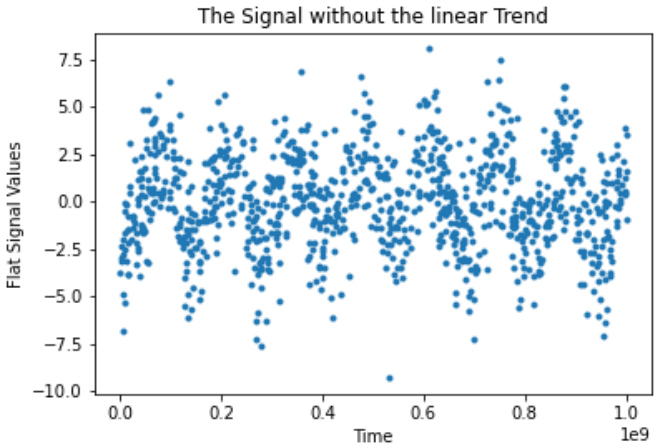
Overall, the new residuals have smaller magnitudes than the previous residuals

Finally, we provide the chi-square of this model. Again, it is not great but it is less than that of the cubic model.

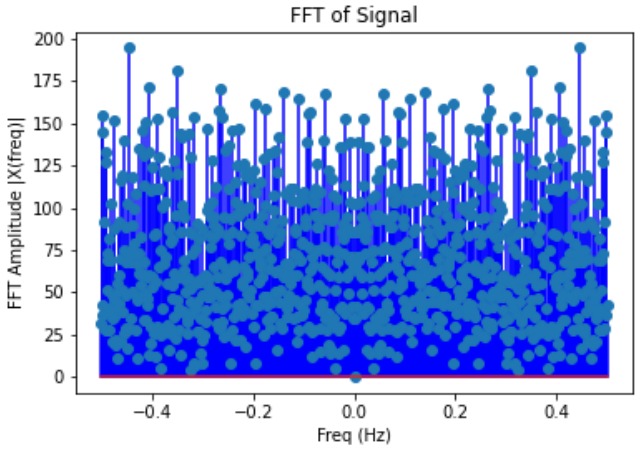
```
Chi-square is 3720.8739444236685
```

Here are the plots showing how we determined and removed the linear trend in the signal.

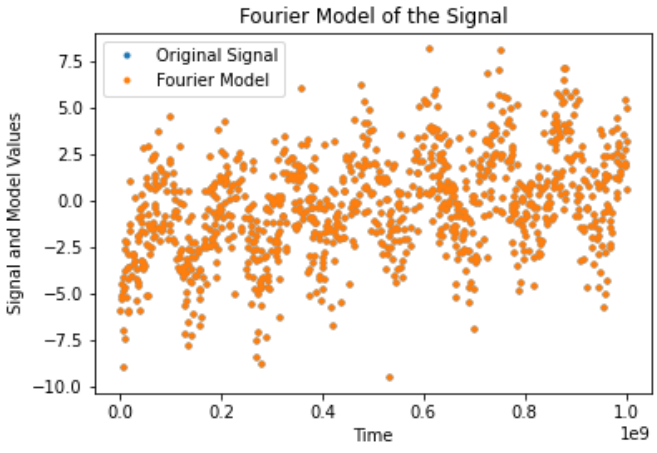


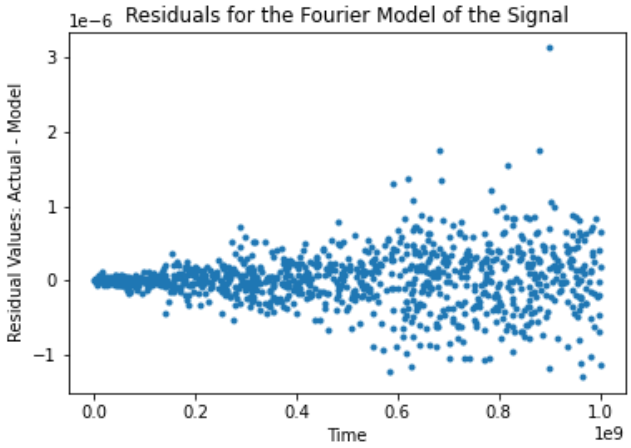


Here is the FFT of the signal.



Finally, here are the plots of the Fourier model and its residuals.





Notice that the residuals are of the order 10^{-6}