HW5

jn2294@nyu.edu

April 2022

Problem Background

This assignment asks us to figure out several ways to interpolate a given function with a fixed number of given points(not necessarily in the domain).

$$f(x;\mu,\sigma)=\frac{1}{\sqrt{2\pi}\sigma}\frac{e^{-\frac{1}{2}\left(\frac{\ln x-\mu}{\sigma}\right)^2}}{x}$$

where μ, σ are parameters and x is independent variable. At $\mu = 0, \sigma = 1$, the function becomes

$$f(x;0{,}1)=\frac{1}{\sqrt{2\pi}}\frac{e^{-\frac{1}{2}(\ln x)^2}}{x}$$

and it looks like

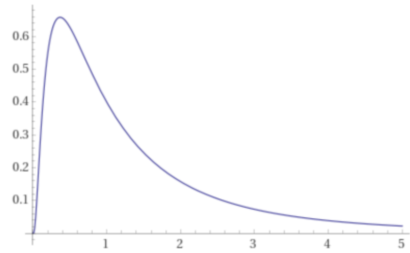

Figure 1: Description for problem 3.1

Problem 5.1

Description

Select 6 points between $x \in [0,5]$ evenly and interpolate these points in a polynomial of the appropriate order. Estimate the upper bound of the interpolation errors.

Figure 2: Description for problem 5.1

Algorithm

Assume that $P(x)$ is the (degree $n-1$ or less) interpolating polynomial fitting the n points $(x_1, y_1), \ldots, (x_n, y_n)$. The interpolation error is

$$f(x) - P(x) = \frac{(x-x_1)(x-x_2)\cdots(x-x_n)}{n!}f^{(n)}(c), \tag{3.6}$$

where c lies between the smallest and largest of the numbers $x, x_1, \ldots, x_n$.

Figure 3: Algorithm for problem 5.1

Code

The source code are previded here at the github repo(hw5.py:189).

Results

- The polynomial interpolation function and its error plot

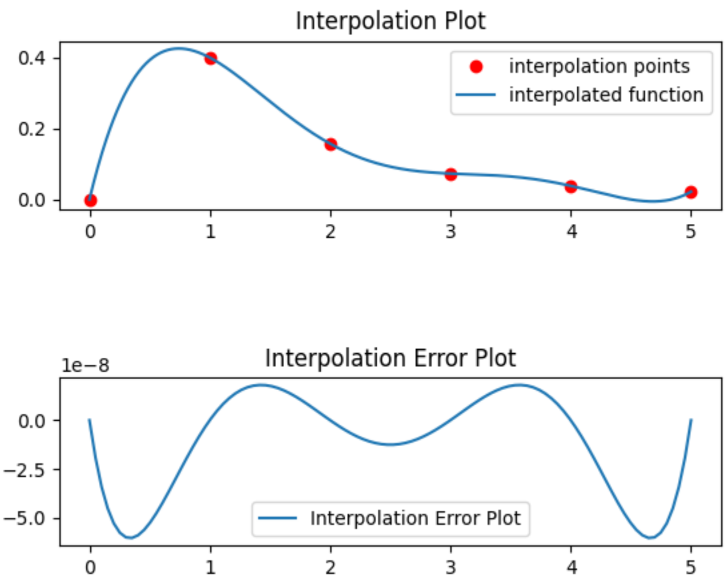

Figure 4: Result plots for problem 5.1

- The upper boundaries at the interpolation points

x	0	1	2	3	4	5
upper bound	15822357.724	-157.172	0.053	0.1154	0.009	-0.001

- The coefficients for interpolation function

coeff	c0	c1	c2	c3	c4	c5
value	0	1.409	-1.546	0.647	-0.120	0.008

Performance

No performance analysis is performed since it doesn't involve heavy computation and only involves simple numpy vector multiplication and small matrix multiplication in normal equation.

Problem 5.2

Description

Problem 5.2 Same as Problem 5.1, select 6 points between $x \in [0, 5]$ as required by Chebyshev interpolation and interpolate these points by Chebyshev polynomials. Estimate the upper bound of the interpolation errors.

Figure 5: Description for problem 5.2

Algorithm

Chebyshev interpolation nodes

On the interval [a,b],

$$x_i = \frac{b+a}{2} + \frac{b-a}{2} \cos \frac{(2i-1)\pi}{2n}$$

for $i = 1, \ldots, n$. The inequality

$$|(x - x_1) \cdots (x - x_n)| \leq \frac{\left(\frac{b-a}{2}\right)^n}{2^{n-1}}$$

holds on $[a, b]$.

Figure 6: Algorithm for problem 5.2

Code

- **Main Program**:(hw5.py:270).
- **Chebshev Interpolation:**(hw5.py:82).

Results

- The polynomial interpolation function and its error plot

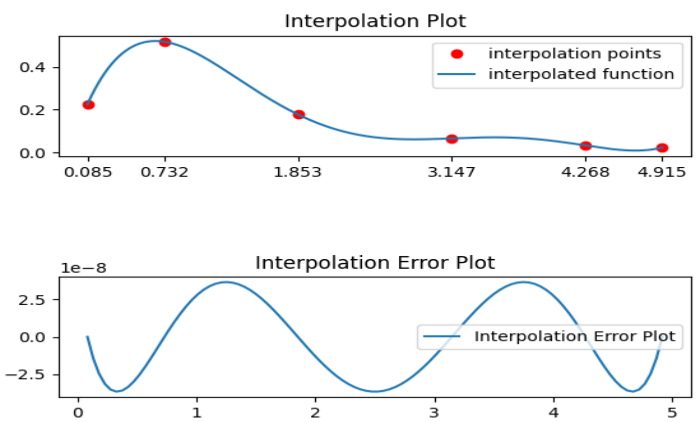

Figure 7: Result plots for problem 5.2

- The upper boundaries at the interpolation points

x	0.085	0.732	1.853	3.147	4.268	4.915
upper bound	20409201.458	-1040.988	-0.371	0.075	0.003	0.003

- The coefficients for interpolation function

coeff	c0	c1	c2	c3	c4	c5
value	0.109	1.520	-1.806	0.779	-0.146	0.010

Performance

No performance analysis is performed since it doesn't involve heavy computation and only involves simple numpy vector multiplication and small matrix multiplication in normal equation.

Problem 5.3

Description

Problem 5.3 Using data from Problem 5.1, select 6 points between $x \in [0, 5]$ evenly and fit these points in the following form

$$y = c_0 + c_1 x + c_2 x^2 + c_3 x^3$$

Also, compute the RMSE for this fit.

Figure 8: Description for problem 5.3

Algorithm

Interpolation polynomial is in the form

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

p interpolates the data points means that

$$p(x_i) = y_i \qquad \text{for all } i \in \{0, 1, \ldots, n\}.$$

If we substitute the above data into the polynomial, we get a system of linear equation for the coefficients a_k:

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \ldots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \ldots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \ldots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

Solving this system for a_k, we construct the interpolant $p(x)$.

Figure 9: Algorithm for problem 5.3

Code

- **Main Program**:(hw5.py:282).
- **Normal Equation Solution :**(hw5.py:168).

Results

- **Curve Fitting:**

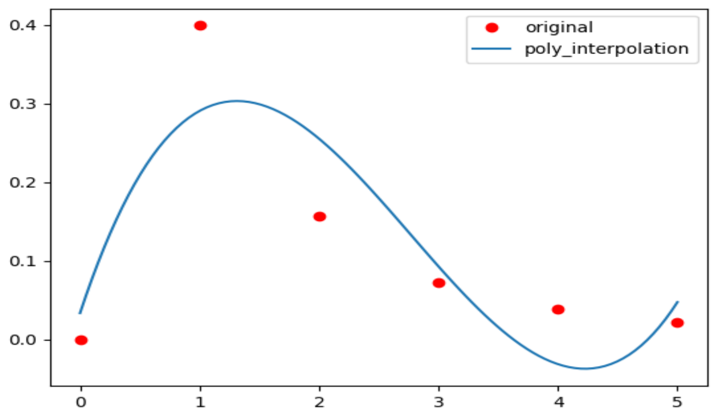

Figure 10: Result plot for problem 5.3

- **Coefficients:**

coeff	c_0	c_1	c_2	c_3
value	0.02751258	-0.22859882	0.45798736	0.03355724

- **RSME:** 0.06887457675451285

Performance

No performance analysis is performed since it doesn't involve heavy computation and only involves simple numpy vector multiplication and small matrix multiplication in normal equation.

Problem 5.4

Description

Problem 5.4 Using data from Problem 5.1, select 6 points between $x \in [0, 5]$ evenly and fit these points in the following form

$$y = c_1 x\, e^{c_2 x}$$

Also, compute the RMSE for this fit.

Figure 11: Description for problem 5.4

Algorithm

$$\ln y = \ln c_1 + \ln t + c_2 t$$
$$k + c_2 t = \ln y - \ln t,$$

Now, we can construct

$$A = \begin{bmatrix} 1 & t_1 \\ \vdots & \vdots \\ 1 & t_m \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} \ln y_1 - \ln t_1 \\ \vdots \\ \ln y_m - \ln t_m \end{bmatrix}$$

Figure 12: Algorithm for problem 5.4

Code

Main Program:(hw5.py:307).

Results

- **Curve Fitting:**

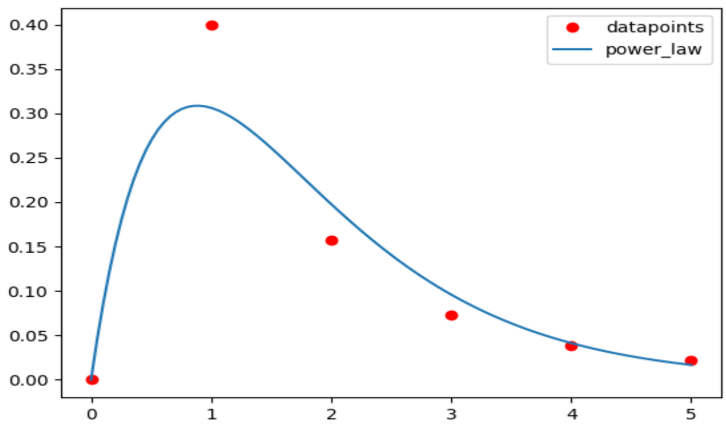

Figure 13: Result plot for problem 5.3

- **Coefficients:**

coeff	c_1	c_2
value	0.9474407258985583	-1.1296988632916292

- **RSME:** 0.2169227011386179

Performance

No performance analysis is performed since it doesn't involve heavy computation and only involves simple numpy vector multiplication and small matrix multiplication in normal equation.