Basic Info

Name: Ni Jiasheng

Environment: python3.8.5 with pycharm

Problems to do: Problem2 and Problem3

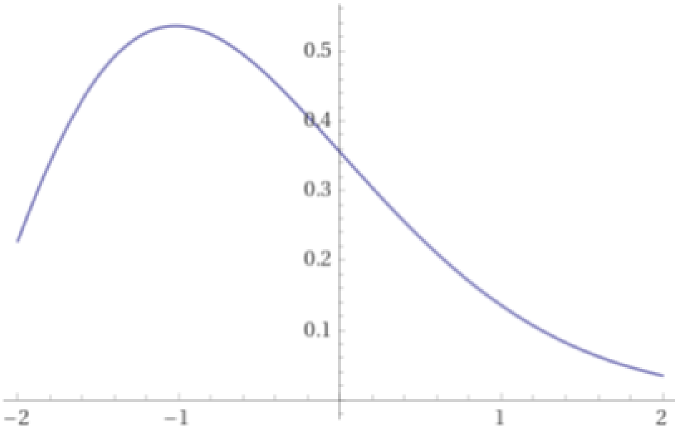
Problem 2

Problem Description

Problem M.2 (50 points) The very famed Airy function is defined in an integral

$$\mathrm{Ai}(x) = \frac{1}{\pi}\int_0^\infty \cos\left(\frac{t^3}{3} + xt\right) dt$$

and, for $x \in (-2, 2)$, it looks like



I calculated the 5 data points tabulated below:

x	-2	-1	0	1	2
$\mathrm{Ai}(x)$	0.227407	0.535561	0.355028	0.135292	0.034924

Write a program to best fit these 5 points with the following given function:

$$\mathrm{f}(x) = c_1 x\, e^{c_2 x}$$

Program

```
    def
problem2(self,datapoints,f=None,option="normal"):
        assert datapoints is not None

        if option == "normal":

            # Adjust the x axis to lie in domain

            x_data = np.array(list(map(lambda
x:x[0],datapoints)))
            y_data = np.array(list(map(lambda
x:x[1],datapoints)))


            # 1. Construct AX=b with order = 1
            A = self.construct_A(datapoints,1)
            b =
self.construct_b(datapoints,"polynomial")

            # 2 and 3 Use Normal Equation to solve
the linear system
            x,rsme = self.normalEquation(A,b)

            k, c2 = x
            c1 = np.exp(k)

            f = lambda t, a, b: a * t * np.exp(b *
t)

            # Draw original points
            plt.plot(x_data,y_data , "o",
label="datapoints")
```

```
            #
            x_points = np.linspace(x_data[0], 
y_data[-1], 100)
            y_hat = f(x_points, c1, c2)
            plt.plot(x_points, y_hat, "g", 
label="fitted")

            plt.legend()
            plt.show()

        else:
            assert f is not None

 self.polynomialFittingPackage(f,datapoints)
```

Algorithm

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

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

Results

Obtain AX=b

```
    def construct_A(self,datapoints,order):
        # k + c2x = lny-lnx

        res_list = []
        for pair in datapoints:
            x = pair[0]
            tmp_list = []
            for t in range(order, -1, -1):
                tmp_list.append(x ** t)
            res_list.append(tmp_list)

        return np.array(res_list)



    def
construct_b(self,datapoints,option="polynomial"):
        b_res = []
        for pair in datapoints:
            y = pair[1]
            x = pair[0]
            if option == "exponential":
                b_res.append(math.log(y) -
math.log(x))
            elif option == "polynomial":
                b_res.append(y)

        return np.array(b_res)
```

Obtain Normal Equation

```
 def normalEquation(self, A, b):
        # x = (A^T*A)^(-1)*(A^T*y)

        # 1. Obtain Normal Equation Solution
        x = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, b))

        # 2. Compute RMSE
        # 2.1 b is the true value, Ax = b_hat is the LS estimation
        rsme = np.sqrt(np.mean(np.square(np.dot(A, x) - b)))
        print("Solution: {}".format(x))
        print("RMSE: {}".format(rsme))
        return x, rsme
```

Solve Coefficient

c1	-0.06861319
c2	-0.43241871

RSME

RMSE: 0.13457383719185537

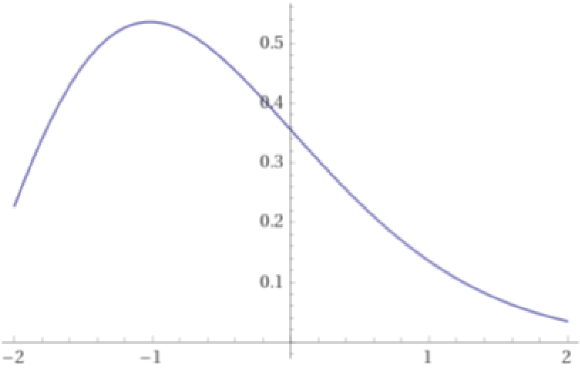
Problem 3

Problem Description

Problem M.3 (50 points) The very famed Airy function is defined in an integral

$$\mathrm{Ai}(x) = \frac{1}{\pi}\int_0^\infty \cos\left(\frac{t^3}{3} + xt\right) dt$$

and, for $x \in (-2, 2)$, it looks like



I calculated the 5 data points tabulated below:

x	-2	-1	0	1	2
$\mathrm{Ai}(x)$	0.227407	0.535561	0.355028	0.135292	0.034924

Write a program to implement any of the interpolation algorithms, including (1) Lagrange interpolation, (2) Newton's divided differences, and (3) Cubic spline, to interpolate these 5 points with

$$\mathrm{P}_4(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4$$

Program

The following is the main function

```
    def problem3(self,datapoints):
        A = self.construct_A(datapoints,4)
        b =
self.construct_b(datapoints,"polynomial")

        # Coefficient
        coeff = np.linalg.solve(A, b)

        # Plot the graph
        # x0 is the independent variables, func is
the interpolation function
        x0, func = self.formatPoly(4,coeff)
        self.plotInterpolation(datapoints, func, x0)
```

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 $p(x)$.

Results&Performance

Coefficient

a0	0.355028
a1	-0.25080575
a2	-0.007480125000000006
a3	0.05067125
a4	-0.012121375

Equation

The obtained equation is `-0.012121375*x**4 + 0.05067125*x**3 - 0.007480125000000006*x**2 - 0.25080575*x + 0.355028`

Plot