BACHELOR OF COMPUTER SCIENC
SCHOOL OF COMPUTER SCIENCE
BINA NUSANTARA UNIVERSITY
JAKARTA

ASSESSMENT FORM

Course: MATH6183001 – Scientific Computing

Method of Assessment: Case Study

Semester/Academic Year : 2/2022-2023

Name of Lecturer : Dr. Ir. I Gusti Agung Anom Yudistira, M.Si.

Date : 30 January 2023

Class : Computer Science

Topic : Regression & Interpolation, Taylor Series, Numerical Differentiation, Numerical Integration

Group Members :	1. Fendy Wijaya

Student Outcomes:

(SO 1) Mampu menganalisis masalah komputasi yang kompleks dan mengaplikasikan prinsip komputasi dan keilmuan lain yang sesuai untuk mengidentifikasi solusi.
Able to analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions

Learning Objectives:

(LObj 1.1) Mampu menganalisis masalah komputasi yang kompleks
Able to analyze a complex computing problem

(LObj 1.2) Mampu menerapkan prinsip komputasi dan disiplin ilmu terkait lainnya untuk mengidentifikasi solusi
Able to apply principles of computing and other relevant disciplines to identify solutions

Learning Outcomes :

(LO 1) Melakukan komputasi saintifik dasar menggunakan Pyhton
Compute basic scientific computation using Python
(LO 2) Menyelesaikan Sistem Persamaan Linear, Regresi dan Interpolasi menggunakan komputasi saintifik
Solve the System of Linear Algebraic Equations, Regression and Interpolation through scientific computation
(LO 3) Mengevaluasi penerapan Deret Taylor dan Akar Persamaan dalam komputasi saintifik
Evaluate the application of Taylor Series and Root of Equations in scientific computation
(LO 4) Menjelaskan konsep dasar dan penerapan Turunan Numerik, Integral Numerik, dan Persamaan Diferensial Biasa dalam komputasi saintifik
Explain basic concept and application of Numerical Differentiation, Numerical Integration, and Ordinary Differential Equations in scientific computation

No	Related LO-LOBJ-SO	Assessment criteria	Weight	Excellent (85 - 100)	Good (75-84)	Average (65-74)	Poor (0 - 64)	Score	(Score x Weight)
1	LO2-L.Obj.1.2-SO1	Understanding of Systems of Linear Equations, Regression and Interpolation	**35%**	Able to clearly explain the concept and solve problems in both numerical and computational approaches without errors.	Able to clearly explain the concept and solve problems in either numerical or computational approaches with some errors	Able to clearly explain the concept but unable to solve problems in both approaches	Only able to poorly explain the concept and unable to solve problems in both approaches	100	35
2	LO3-L.Obj.1.1-SO1	Understanding of Taylor Series and Root of Equations	**30%**	Able to clearly explain the concept and solve problems in both numerical and computational	Able to clearly explain the concept and solve	Able to clearly explain the concept but unable to solve	Only able to poorly explain the concept and unable to	100	30

No	Related LO-LOBJ-SO	Assessment criteria	Weight	Excellent (85 - 100)	Good (75-84)	Average (65-74)	Poor (0 - 64)	Score	(Score x Weight)
				approaches without errors.	problems in either numerical or computational approaches with some errors	problems in both approaches	solve problems in both approaches		
3	LO4-L.Obj.1.2-SO1	Understanding of Numerical Differentiation, Integration, and Introductionary ODEs	**35%**	Able to clearly explain the concept and solve problems in both numerical and computational approaches without errors.	Able to clearly explain the concept and solve problems in either numerical or computational approaches with some errors	Able to clearly explain the concept but unable to solve problems in both approaches	Only able to poorly explain the concept and unable to solve problems in both approaches	100	35
		Total Score: ∑(Score x Weight)							100

Remarks:

_____

_____

_____

_____

ASSESSMENT METHOD

Instructions

The deadline of this comprehensive assignment is at the end the semester. Answer the questions below in .PDF format through BINUS Maya. Attach the manual calculation **AND** script that you use. All the answers **must** be rounded according to the given dataset!

1. The relationship between the average temperature on the earth's surface in odd years between 1981 - 1999, is given by the following below: **(35%)**

Year (x)	Temperature (y, ^{o}C)
1981	14.1999
1983	14.2411
1985	14.0342
1987	14.2696
1989	14.197
1991	14.3055
1993	14.1853
1995	14.3577
1997	14.4187
1999	14.3438

   a. Estimate the temperature in even years by linear, quadratic, and cubic interpolation order! Choose the method that you think is appropriate, and explain the difference.

Jawab:

Pada soal ini, saya akan menggunakan metode interpolasi langsung (direct interpolation method). Metode ini mengharuskan kita untuk membuat sebuah sistem persamaan linear berdasarkan data-data yang diberikan, lalu menyelesaikannya untuk mencari koefisien dari setiap variabel x. Keunggulan metode ini adalah lebih cepat digunakan ketika dikomputasi secara manual, sehingga perhitungan manual yang akan saya lakukan menjadi lebih mudah dilakukan. Hal yang membedakan metode ini dengan metode lainnya adalah adanya n+1 buah persamaan linear dari n+1 data yang kita pilih, sehingga dapat dibentuk suatu sistem persamaan berorde n.

Interpolasi dengan direct method yang akan saya lakukan meliputi interpolasi linear, kuadratik, dan kubik. Perbedaannya terletak pada tingkat keakuratan interpolasi. Semakin tinggi orde interpolasi, semakin banyak data point yang dilalui oleh kurva interpolasi, sehingga hasil yang diperoleh juga akan semakin akurat.

Interpolasi Linear

Interpolasi linear dengan menggunakan direct method membutuhkan 2 buah data points untuk memperoleh fungsi interpolasinya. Dalam kasus ini, data yang diberikan adalah tahun ganjil dari 1981 hingga 1999 beserta temperaturnya. Oleh karena hanya 2 data points saja yang diperlukan, maka data points yang digunakan untuk menginterpolasi tahun genap di antara 1981 dan 1999 haruslah tahun 1981 dan 1999 itu sendiri, sehingga proses interpolasi tahun genap dari tahun 1982 hingga 1998 tidak berada di luar jangkauan interval yang telah ditentukan.

Bentuk umum persamaan yang akan diperoleh adalah $y = a_0 + a_1 x$, dengan data points sebagai berikut.

$x_0 = 1981 \rightarrow y = 14{,}1999$

$x_1 = 1999 \rightarrow y = 14{,}3438$

Maka, dapat dibuat sistem persamaan linear sebagai berikut.

$14{,}1999 = a_0 + a_1(1981) \;\rightarrow\; a_0 + 1981a_1 = 14{,}1999$

$14{,}3438 = a_0 + a_1(1999) \;\rightarrow\; a_0 + 1999a_1 = 14{,}3438$

Sistem persamaan linear dua variabel tersebut dapat diubah ke dalam bentuk matriks menjadi:

$$\begin{bmatrix} 1 & 1981 \\ 1 & 1999 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 14{,}1999 \\ 14{,}3438 \end{bmatrix}$$

Dengan menyelesaikan sistem persamaan linear tersebut, diperoleh harga a_0 dan a_1 sebagai berikut:

$$a_0 = -1{,}6371$$

$$a_1 = 0{,}0080$$

Maka, persamaan garis yang akan digunakan untuk menginterpolasi tahun-tahun genap dengan direct method interpolation secara linear adalah $y = -1{,}6371 + 0{,}008x$.

Berdasarkan persamaan tersebut, maka harga temperatur pada tahun genap dapat dicari sebagai berikut:

Untuk $x = 1982$, maka nilai $y = -1{,}6371 + 0{,}008(1982) = 14{,}2189$

Untuk $x = 1984$, maka nilai $y = -1{,}6371 + 0{,}008(1984) = 14{,}2349$

Untuk $x = 1986$, maka nilai $y = -1{,}6371 + 0{,}008(1986) = 14{,}2509$

Untuk $x = 1988$, maka nilai $y = -1{,}6371 + 0{,}008(1988) = 14{,}2669$

Untuk $x = 1990$, maka nilai $y = -1{,}6371 + 0{,}008(1990) = 14{,}2829$

Untuk $x = 1992$, maka nilai $y = -1{,}6371 + 0{,}008(1992) = 14{,}2989$

Untuk $x = 1994$, maka nilai $y = -1{,}6371 + 0{,}008(1994) = 14{,}3149$

Untuk $x = 1996$, maka nilai $y = -1{,}6371 + 0{,}008(1996) = 14{,}3309$

Untuk $x = 1998$, maka nilai $y = -1{,}6371 + 0{,}008(1998) = 14{,}3469$

Interpolasi Kuadratik

Interpolasi kuadratik dengan menggunakan direct method membutuhkan 3 buah data points untuk memperoleh fungsi interpolasinya. Dalam kasus ini, data yang diberikan adalah tahun ganjil dari 1981 hingga 1999 beserta temperaturnya. Oleh karena hanya 3 data points

saja yang diperlukan, maka data points yang digunakan untuk menginterpolasi tahun genap di antara 1981 dan 1999 haruslah tahun 1981 dan 1999 itu sendiri sebagai batas kiri dan kanan, sehingga proses interpolasi tahun genap dari tahun 1982 hingga 1998 tidak berada di luar jangkauan interval yang telah ditentukan, serta 1 data point yang berada di tengah-tengah kumpulan data, yaitu tahun 1989.

Bentuk umum persamaan yang akan diperoleh adalah $y = a_0 + a_1 x + a_2 x^2$, dengan data points sebagai berikut.

$x_0 = 1981 \rightarrow y = 14{,}1999$

$x_1 = 1989 \rightarrow y = 14{,}1970$

$x_2 = 1999 \rightarrow y = 14{,}3438$

Maka, dapat dibuat sistem persamaan linear sebagai berikut.

$14{,}1999 = a_0 + a_1(1981) + a_2(1981)^2 \rightarrow \quad a_0 + 1981a_1 + 1981^2 a_2 = 14{,}1999$

$14{,}1970 = a_0 + a_1(1989) + a_2(1989)^2 \rightarrow \quad a_0 + 1989a_1 + 1989^2 a_2 = 14{,}1970$

$14{,}3438 = a_0 + a_1(1999) + a_2(1999)^2 \rightarrow \quad a_0 + 1999a_1 + 1999^2 a_2 = 14{,}3438$

Sistem persamaan linear dua variabel tersebut dapat diubah ke dalam bentuk matriks menjadi:

$$\begin{bmatrix} 1 & 1981 & 1981^2 \\ 1 & 1989 & 1989^2 \\ 1 & 1999 & 1999^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 14{,}1999 \\ 14{,}1970 \\ 14{,}3438 \end{bmatrix}$$

Dengan menyelesaikan sistem persamaan linear tersebut, diperoleh harga a_0 dan a_1 sebagai berikut:

$$a_0 = 3.307{,}7288$$

$$a_1 = -3{,}3181$$

$$a_2 = 0{,}0008357$$

Maka, persamaan garis yang akan digunakan untuk menginterpolasi tahun-tahun genap dengan direct method interpolation secara kuadratik adalah $y = 3.307{,}7288 - 3{,}3181x + 0{,}0008357x^2$.

Berdasarkan persamaan tersebut, maka harga temperatur pada tahun genap dapat dicari sebagai berikut:

Untuk $x = 1982$, maka nilai $y = 3.307{,}7288 - 3{,}3181(1982) + 0{,}0008357(1982)^2 = 14{,}1550$

Untuk $x = 1984$, maka nilai $y = 3.307{,}7288 - 3{,}3181(1984) + 0{,}0008357(1984)^2 = 14{,}1475$

Untuk $x = 1986$, maka nilai $y = 3.307{,}7288 - 3{,}3181(1986) + 0{,}0008357(1986)^2 = 14{,}1468$

Untuk $x = 1988$, maka nilai $y = 3.307{,}7288 - 3{,}3181(1988) + 0{,}0008357(1988)^2 = 14{,}1527$

Untuk $x = 1990$, maka nilai $y = 3.307{,}7288 - 3{,}3181(1990) + 0{,}0008357(1990)^2 = 14{,}1654$

Untuk $x = 1992$, maka nilai $y = 3.307{,}7288 - 3{,}3181(1992) + 0{,}0008357(1992)^2 = 14{,}1847$

Untuk $x = 1994$, maka nilai $y = 3.307{,}7288 - 3{,}3181(1994) + 0{,}0008357(1994)^2 = 14{,}2107$

Untuk $x = 1996$, maka nilai $y = 3.307{,}7288 - 3{,}3181(1996) + 0{,}0008357(1996)^2 = 14{,}2434$

Untuk $x = 1998$, maka nilai $y = 3.307{,}7288 - 3{,}3181(1998) + 0{,}0008357(1998)^2 = 14{,}2827$

Interpolasi Kubik

Interpolasi kubik dengan menggunakan direct method membutuhkan 4 buah data points untuk memperoleh fungsi interpolasinya. Dalam kasus ini, data yang diberikan adalah tahun ganjil dari 1981 hingga 1999 beserta temperaturnya. Oleh karena hanya 4 data points saja yang diperlukan, maka data points yang digunakan untuk menginterpolasi tahun genap di antara 1981 dan 1999 haruslah tahun 1981 dan 1999 itu sendiri sebagai batas kiri dan kanan, sehingga proses interpolasi tahun genap dari tahun 1982 hingga 1998 tidak berada di luar jangkauan interval yang telah ditentukan, serta 2 data points yang berada di tengah-tengah kumpulan data, yaitu tahun 1987 dan 1993.

Bentuk umum persamaan yang akan diperoleh adalah $y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$, dengan data points sebagai berikut.

$x_0 = 1981 \rightarrow y = 14{,}1999$

$x_1 = 1987 \rightarrow y = 14{,}2696$

$x_2 = 1993 \rightarrow y = 14{,}1853$

$x_3 = 1999 \rightarrow y = 14{,}3438$

Maka, dapat dibuat sistem persamaan linear sebagai berikut.

$14{,}1999 = a_0 + a_1(1981) + a_2(1981)^2 + a_3(1981)^3 \rightarrow a_0 + 1981a_1 + 1981^2a_2 + 1981^3a_3 = 14{,}1999$

$14{,}2696 = a_0 + a_1(1987) + a_2(1987)^2 + a_3(1987)^3 \rightarrow a_0 + 1987a_1 + 1987^2a_2 + 1987^3a_3 = 14{,}2696$

$14{,}1853 = a_0 + a_1(1993) + a_2(1993)^2 + a_3(1993)^3 \rightarrow a_0 + 1993a_1 + 1993^2a_2 + 1993^3a_3 = 14{,}1853$

$14{,}3438 = a_0 + a_1(1999) + a_2(1999)^2 + a_3(1999)^3 \rightarrow a_0 + 1999a_1 + 1999^2a_2 + 1999^3a_3 = 14{,}3438$

Sistem persamaan linear dua variabel tersebut dapat diubah ke dalam bentuk matriks menjadi:

$$\begin{bmatrix} 1 & 1981 & 1981^2 & 1981^3 \\ 1 & 1987 & 1987^2 & 1987^3 \\ 1 & 1993 & 1993^2 & 1993^3 \\ 1 & 1999 & 1999^2 & 1999^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 14{,}1999 \\ 14{,}2696 \\ 14{,}1853 \\ 14{,}3438 \end{bmatrix}$$

Dengan menyelesaikan sistem persamaan linear tersebut, diperoleh harga a_0 dan a_1 sebagai berikut:

$$a_0 = -2.410.335{,}6462$$

$$a_1 = 3.634{,}9540$$

$$a_2 = -1{,}8272351$$

$$a_3 = 0{,}0003061728$$

Maka, persamaan garis yang akan digunakan untuk menginterpolasi tahun-tahun genap dengan direct method interpolation secara kubik adalah $y = -2.410.335{,}6462 + 3.634{,}9540x - 1{,}8272351x^2 + 0{,}0003061728x^3$.

Berdasarkan persamaan tersebut, maka harga temperatur pada tahun genap dapat dicari sebagai berikut:

Untuk $x = 1982$, maka $y = -2.410.335{,}6462 + 3.634{,}9540(1982) - 1{,}8272351(1982)^2 + 0{,}0003061728(1982)^3 = 14{,}1744$

Untuk $x = 1984$, maka $y = -2.410.335{,}6462 + 3.634{,}9540(1984) - 1{,}8272351(1984)^2 + 0{,}0003061728(1984)^3 = 14{,}2138$

Untuk $x = 1986$, maka $y = -2.410.335{,}6462 + 3.634{,}9540(1986) - 1{,}8272351(1986)^2 + 0{,}0003061728(1986)^3 = 14{,}2140$

Untuk $x = 1988$, maka $y = -2.410.335{,}6462 + 3.634{,}9540(1988) - 1{,}8272351(1988)^2 + 0{,}0003061728(1988)^3 = 14{,}1898$

Untuk $x = 1990$, maka $y = -2.410.335{,}6462 + 3.634{,}9540(1990) - 1{,}8272351(1990)^2 + 0{,}0003061728(1990)^3 = 14{,}1558$

Untuk $x = 1992$, maka $y = -2.410.335{,}6462 + 3.634{,}9540(1992) - 1{,}8272351(1992)^2 + 0{,}0003061728(1992)^3 = 14{,}1267$

Untuk $x = 1994$, maka $y = -2.410.335{,}6462 + 3.634{,}9540(1994) - 1{,}8272351(1994)^2 + 0{,}0003061728(1994)^3 = 14{,}1173$

Untuk $x = 1996$, maka $y = -2.410.335{,}6462 + 3.634{,}9540(1996) - 1{,}8272351(1996)^2 + 0{,}0003061728(1996)^3 = 14{,}1422$

Untuk $x = 1998$, maka $y = -2.410.335{,}6462 + 3.634{,}9540(1998) - 1{,}8272351(1998)^2 + 0{,}0003061728(1998)^3 = 14{,}2162$

b. Perform a least-square regression of the above data to estimate the temperature in even years.

Jawab:

Hasil akhir dari least-square regression adalah suatu persamaan garis $y = a_0 + a_1 x$ yang dapat digunakan untuk mengestimasi data suhu pada tahun genap berdasarkan kumpulan data yang telah diberikan. Untuk itu, kita perlu mencari harga a_0 dan a_1, sehingga diperoleh persamaan regresi yang sesuai.

Langkah pertama adalah mencari harga a_1 dengan menggunakan rumus berikut:

$$a_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} {x_i}^2 - (\sum_{i=1}^{n} x_i)^2}$$

Diketahui bahwa jumlah data (n) = 10, maka variabel-variabel yang belum diketahui dapat dicari nilainya dengan bantuan tabel di bawah.

Year (x)	Temperature (y)	x.y	x^2
1981	14,1999	28.130,0019	3.924.361
1983	14,2411	28.240,1013	3.932.289
1985	14,0342	27.857,8870	3.940.225
1987	14,2696	28.353,6952	3.948.169
1989	14,197	28.237,8330	3.956.121
1991	14,3055	28.482,2505	3.964.081

1993	14,1853	28.271,3029	3.972.049
1995	14,3577	28.643,6115	3.980.025
1997	14,4187	28.794,1439	3.988.009
1999	14,3438	28.673,2562	3.996.001
$\sum_{i=1}^{10} x_i = 19900$	$\sum_{i=1}^{10} y_i = 142{,}5528$	$\sum_{i=1}^{10} x_i y_i = 283.684{,}0834$	$\sum_{i=1}^{10} {x_i}^2 = 39.601.330$

Berdasarkan perhitungan yang telah dilakukan pada tabel di atas, kita dapat mensubstitusikan nilai-nilai yang diperlukan pada rumus a_1 sebagai berikut.

$$a_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} {x_i}^2 - (\sum_{i=1}^{n} x_i)^2}$$

$$a_1 = \frac{10(283.684{,}0834) - (19.900)(142{,}5528)}{10(39.601.330) - (19.900)^2}$$

$$a_1 = 0{,}012156$$

Harga a_1 merupakan nilai kemiringan atau gradien dari garis regresi. Kemudian, untuk mencari harga a_0, digunakan rumus berikut.

$$a_0 = \bar{y} - a_1\bar{x}$$

Sebelum menentukan harga a_0, kita perlu menentukan nilai rata-rata dari data x dan data y, dengan perhitungan sebagai berikut.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{19900}{10} = 1990$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} = \frac{142{,}5528}{10} = 14{,}2553$$

Setelah mendapatkan nilai rata-rata data x dan data y, kedua nilai tersebut dapat disubstitusikan ke dalam rumus a_0.

$$a_0 = \bar{y} - a_1\bar{x}$$

$a_0 = 14{,}2553 - (0{,}012156)(1990)$

$a_0 = -9{,}93514$

Harga a_0 yang diperoleh merupakan y-intercept atau titik potong garis regresi dengan sumbu-y. Dari harga a_0 dan a_1 yang didapat sebelumnya, kita dapat menuliskan persamaan regresi linear dari metode least-square regression sebagai berikut:

$$y = a_0 + a_1 x$$

$$y = -9{,}93514 + 0{,}012156x$$

Persamaan regresi linear tersebut dapat kita manfaatkan untuk memperkirakan estimasi suhu pada tahun-tahun genap, dengan rincian perhitungan sebagai berikut.

Untuk $x = 1982$, maka nilai $y = -9{,}93514 + 0{,}012156(1982) = 14{,}1580$

Untuk $x = 1984$, maka nilai $y = -9{,}93514 + 0{,}012156(1984) = 14{,}1823$

Untuk $x = 1986$, maka nilai $y = -9{,}93514 + 0{,}012156(1986) = 14{,}2067$

Untuk $x = 1988$, maka nilai $y = -9{,}93514 + 0{,}012156(1988) = 14{,}2310$

Untuk $x = 1990$, maka nilai $y = -9{,}93514 + 0{,}012156(1990) = 14{,}2553$

Untuk $x = 1992$, maka nilai $y = -9{,}93514 + 0{,}012156(1992) = 14{,}2796$

Untuk $x = 1994$, maka nilai $y = -9{,}93514 + 0{,}012156(1994) = 14{,}3039$

Untuk $x = 1996$, maka nilai $y = -9{,}93514 + 0{,}012156(1996) = 14{,}3282$

Untuk $x = 1998$, maka nilai $y = -9{,}93514 + 0{,}012156(1998) = 14{,}3525$

Selain perhitungan manual, kita juga dapat melakukan pemrograman Python untuk melakukan least-square regression dengan ketelitian yang lebih tinggi. Program Python untuk menghitung suhu pada tahun genap dengan metode least-square regression ditunjukkan dengan kode program berikut.

```python
# Import library yang diperlukan
import numpy as np
import matplotlib.pyplot as plt

# Masukkan data points tahun dan temperatur yang tertera pada soal
year = [1981, 1983, 1985, 1987, 1989, 1991, 1993, 1995, 1997, 1999]
temperature = [14.1999, 14.2411, 14.0342, 14.2696, 14.197, 14.3055, 14.1853, 14.3577, 14.4187, 14.3438]
```

✓ 0.0s

```python
# Least Square Regression

# Data points yang diberikan diubah ke dalam bentuk array
x = np.array(year)
y = np.array(temperature)

# Tentukan jumlah data yang diberikan (n)
n = len(temperature)

# Hitung jumlah data x, data y, perkalian x dan y, serta x kuadrat
sigma_x = sum(x)
sigma_y = sum(y)
sigma_x2 = sum(x[:]**2)
sigma_xy = sum(x[:]*y[:])

# Hitung harga a1 (gradien/kemiringan garis regresi)
a1 = (n*sigma_xy - sigma_x*sigma_y)/(n*sigma_x2 - (sigma_x**2))

# Hitung rata-rata data x dan data y
x_mean = sigma_x/n
y_mean = sigma_y/n

```

```python
# Hitung harga a0 (y-intercept atau titik potong terhadap sumbu-y)
a0 = y_mean - a1*x_mean

# Keluarkan persamaan garis regresi linear
print(f"Persamaan regresi linear : y = {a0} + {a1} x \n")

# Keluarkan data hasil regresi pada tahun genap dengan menggunakan persamaan regresi linear yang diperoleh
print(" Year  | Temperature")
print("---------------------")
for i in range(n-1):
    print(f" {year[i]+1}  |   {round(a0 + a1*(year[i]+1), 4)}")
print("---------------------")
```

✓ 0.0s

Kode program tersebut akan menghasilkan output sebagai berikut.

```
Persamaan regresi linear : y = -9.934677576074332 + 0.012155757575916748 x

 Year  | Temperature
---------------------
 1982  |   14.158
 1984  |   14.1823
 1986  |   14.2067
 1988  |   14.231
 1990  |   14.2553
 1992  |   14.2796
 1994  |   14.3039
 1996  |   14.3282
 1998  |   14.3525
---------------------
```

c. Perform an analysis of the difference between the results of the regression and interpolations you can above, explain based on the theoretical basis you have learned.

Jawab:

Perbedaan mendasar antara regresi dengan interpolasi berada pada tingkat keakuratan curve-fitting. Keduanya sama-sama menunjukkan persamaan umum yang mewakili pola perilaku data set yang diberikan, hanya saja interpolasi menggunakan data persis untuk mencari kurva yang akan menggambarkan data secara keseluruhan, sementara regresi hanya menggambarkan garis yang mewakili pola umum tren data yang diberikan.

Regresi merupakan teknik prediksi data yang dilakukan dengan mencari suatu kurva atau garis yang sesuai dengan tren data secara umum. Biasanya, kesalahan yang ditunjukkan cukup signifikan, karena kurva umumnya tidak memotong setiap data point yang diberikan. Teknik ini biasanya digunakan jika data yang diberikan memang tidak memerlukan kepersisan tinggi, atau hanya memerlukan gambaran kasar datanya saja. Berbeda dengan interpolasi, yang merupakan teknik prediksi data yang dilakukan dengan mencari kurva fungsi yang benar-benar persis dengan data point yang diberikan menjadi acuan. Semakin banyak data point yang dilibatkan dalam membuat kurva interpolasi, maka fitting akan menjadi semakin akurat, dan data point akan tepat berpotongan dengan kurva fungsinya.

Perbedaan ini sebenarnya sangat jelas terlihat pada hasil perhitungan interpolasi nomor 1a dan regresi nomor 1b. Ketika kita menghitung secara interpolasi linear, kuadratik, dan kubik, hasil yang diperoleh akan menyesuaikan data point yang digunakan. Misalnya pada interpolasi kubik, saya menggunakan data tahun 1981, 1987, 1993, dan 1999. Apabila nilai tahun tersebut dimasukkan ke dalam fungsi interpolasi sebagai x, maka nilai y yang dihasilkan akan hampir sama persis dengan nilai y yang diberikan oleh soal (pembulatan pada koefisien fungsi polinomial mengakibatkan nilai y tidak sama persis, tetapi hampir sama nilainya).

Hal berbeda terlihat ketika kita menganalisis pola regresi yang didapatkan pada nomor 1b. Garis yang diperoleh merupakan suatu kurva linear yang tidak tepat berpotongan dengan data point. Kita bisa mengujinya dengan memasukkan sembarang nilai x = 1989 pada fungsi regresi yang diperoleh, maka nilai y yang dihasilkan adalah 14,2431. Nilai ini cukup berbeda dengan data pada tabel soal, di mana tahun 1989 mencatat temperatur 14,197°C.

Ilustrasi lebih jelas akan terjawab pada soal 1d.

d. Make a plot that describes the relationship between Temperature (y) and Year (x) as informatively as possible for the reader, based on the results of your analysis using Python library.

Jawab:

Grafik plotting dapat dibuat dengan menggunakan Python library matplotlib.pyplot, di mana library (package) ini dapat melakukan pemetaan (plotting) pada suatu koordinat Cartesius, sehingga kita dapat membaca hubungan yang terbentuk antara tahun dan temperatur dengan plot tersebut.

Berikut adalah kode program untuk memunculkan grafik plotting secara bertahap.

```
# Import library yang diperlukan
import numpy as np
import matplotlib.pyplot as plt

# Masukkan data points tahun dan temperatur yang tertera pada soal
year = [1981, 1983, 1985, 1987, 1989, 1991, 1993, 1995, 1997, 1999]
temperature = [14.1999, 14.2411, 14.0342, 14.2696, 14.197, 14.3055, 14.1853, 14.3577, 14.4187, 14.3438]

# Data points yang diberikan diubah ke dalam bentuk array
x = np.array(year)
y = np.array(temperature)
```

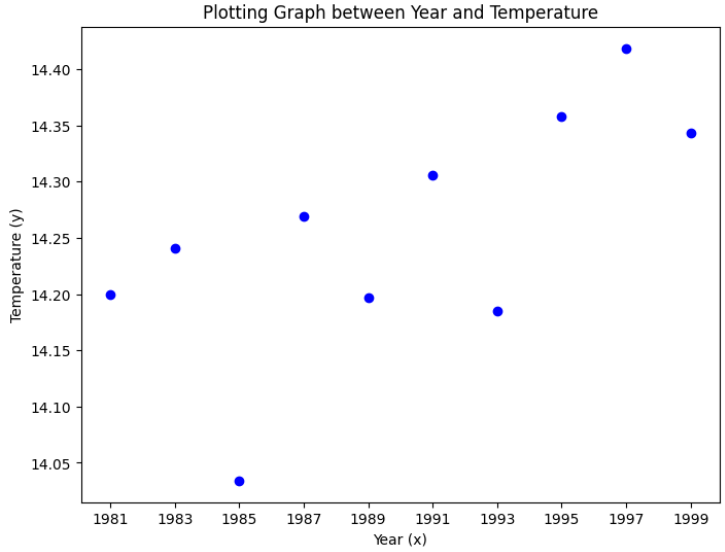
✓ 0.0s

Pertama-tama, kita akan memetakan terlebih dahulu seperti apa bentuk penyebaran data yang diberikan pada soal dengan menggunakan kode program sebagai berikut.

```python
# Plotting Data Points Awal

# Menentukan ukuran grafik plotting
plt.figure(figsize=(8,6))
# Memberi judul, label sumbu-x, dan label sumbu-y
plt.title("Plotting Graph between Year and Temperature")
plt.xlabel("Year (x)")
plt.ylabel("Temperature (y)")
# Mengatur interval nilai yang dimunculkan di sumbu-x
plt.xticks(np.arange(min(x), max(x)+1, 2))
# Menandai titik-titik data sesuai data points pada soal
plt.plot(x, y, "bo")
# Keluarkan plot grafik
plt.show()
```

✓ 0.4s

Output yang dihasilkan ditunjukkan oleh gambar berikut.

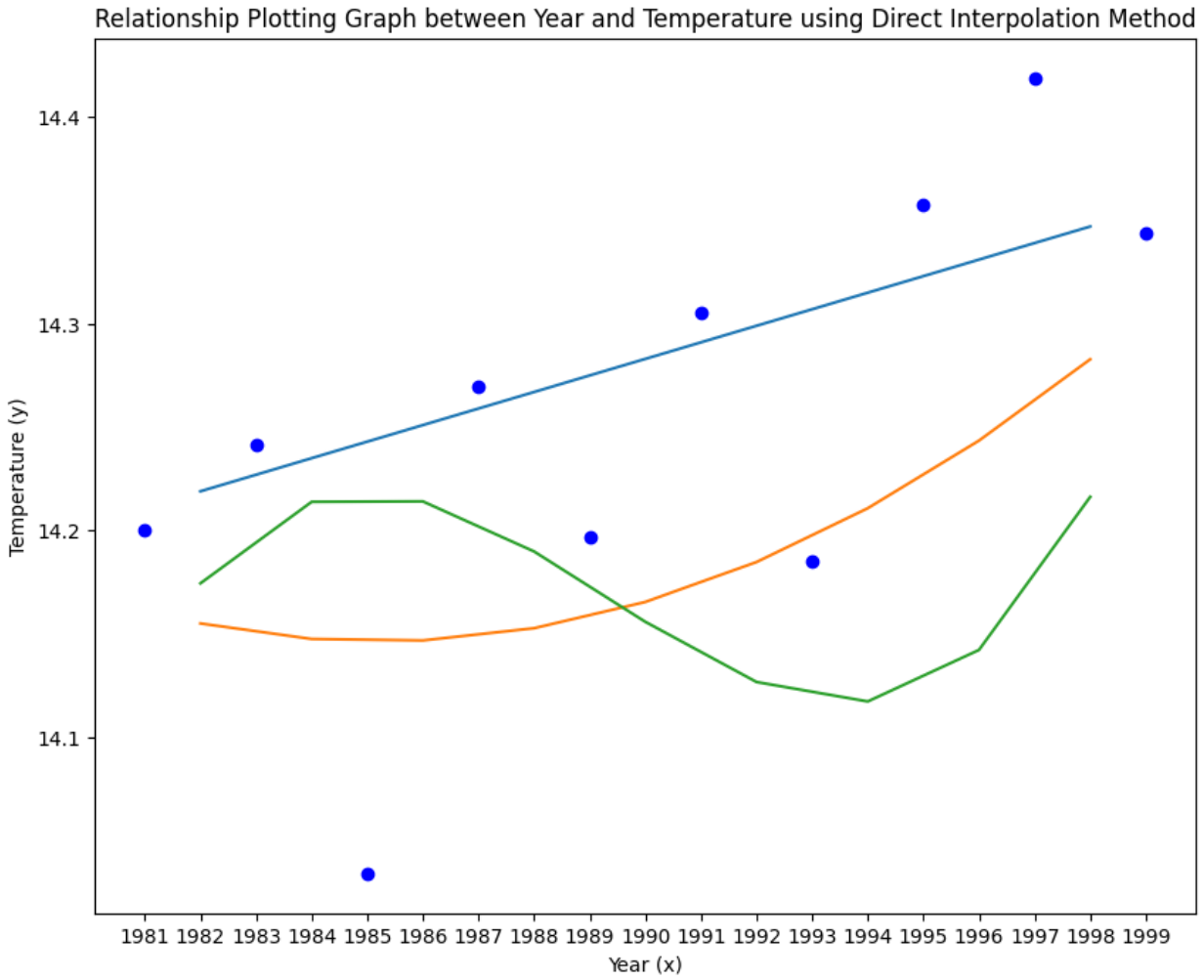


Kemudian, kita akan mencoba menampilkan wujud kurva-kurva interpolasi yang telah kita definisikan pada soal 1a, dengan menggunakan kode program sebagai berikut.

```
# Plotting for Direct Interpolation Method

# Menentukan ukuran grafik plotting
plt.figure(figsize=(10,8))
# Memberi judul, label sumbu-x, dan label sumbu-y
plt.title("Relationship Plotting Graph between Year and Temperature using Direct Interpolation Method")
plt.xlabel("Year (x)")
plt.ylabel("Temperature (y)")
# Mengatur interval nilai yang dimunculkan di sumbu-x
x_interplt = np.array([1982, 1984, 1986, 1988, 1990, 1992, 1994, 1996, 1998])
plt.xticks(np.arange(min(x), max(x)+1, 1))
plt.yticks(np.arange(14, 15, 0.1))
# Menandai titik-titik data sesuai data points pada soal
plt.plot(x, y, "bo")

# Plot kurva fungsi interpolasi
# Interpolasi linear
y_interplt = np.array([14.2189, 14.2349, 14.2509, 14.2669, 14.2829, 14.2989, 14.3149, 14.3309, 14.3469])
plt.plot(x_interplt, y_interplt, "-")
# Interpolasi Kuadratik
y_interplt = np.array([14.1550, 14.1475, 14.1468, 14.1527, 14.1654, 14.1847, 14.2107, 14.2434, 14.2827])
plt.plot(x_interplt, y_interplt, "-")
# Interpolasi Kubik
y_interplt = np.array([14.1744, 14.2138, 14.2140, 14.1898, 14.1558, 14.1267, 14.1173, 14.1422, 14.2162])
plt.plot(x_interplt, y_interplt, "-")

# Keluarkan plot grafik
plt.show()
```

✓ 0.2s

Output yang dihasilkan dari proses interpolasi tersebut adalah sebagai berikut.



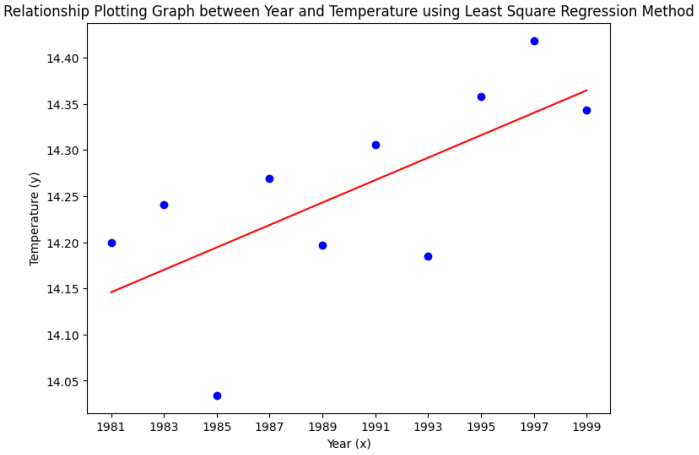
Garis biru menunjukkan fungsi interpolasi linear, kurva kuning menunjukkan fungsi interpolasi kuadratik, sementara kurva hijau menunjukkan fungsi interpolasi kubik.

Selanjutnya, kita akan melakukan plotting hasil regresi linear yang telah dilakukan pada soal 1b, dengan kode program sebagai berikut.

```
# Plotting for Least Square Regression

# Menentukan ukuran grafik plotting
plt.figure(figsize=(8,6))
# Memberi judul, label sumbu-x, dan label sumbu-y
plt.title("Relationship Plotting Graph between Year and Temperature using Least Square Regression Method")
plt.xlabel("Year (x)")
plt.ylabel("Temperature (y)")
# Mengatur interval nilai yang dimunculkan di sumbu-x
plt.xticks(np.arange(min(year), max(year)+1, 2))
# Menandai titik-titik data sesuai data points pada soal
plt.plot(x, y, "bo")
# Plot persamaan garis regresi
plt.plot(x, -9.934677576074332 + 0.012155757575916748*x, "r")
# Keluarkan plot grafik
plt.show()
```

✓ 0.5s

Output yang dihasilkan adalah sebagai berikut.



Setelah melakukan plotting dari proses interpolasi dan regresi yang telah dilakukan, kita dapat melihat bahwa ada kecenderungan antara tahun terhadap temperatur, di mana temperatur cenderung mengalami kenaikan seiring bertambahnya tahun. Argumen ini dapat dilihat dari adanya garis interpolasi linear maupun regresi linear yang bergradien positif. Selain itu, kurva interpolasi kuadratik dan kubik yang dihasilkan juga menunjukkan adanya dominansi interval kemonotonan naik pada kurva dibanding interval kemonotonan turunnya. Oleh karena itu, hubungan antara suhu terhadap waktu (tahun) merupakan hubungan yang berkorelasi positif, meski agak tersebar. Semakin besar tahun, maka suhu juga akan cenderung mengalami kenaikan pada range interval yang diberikan.

2. Compute the fourth order Taylor expansion for sin(x) and cos(x) and sin(x)cos(x) around 0. **(30%)**
   a. Write down your manual calculation **AND** Python script to answer above's question

Jawab:

Fungsi sin(x)

Berikut ini adalah perhitungan manual ekspansi Taylor berorde 4 untuk **sin(x)** di sekitar x = 0.

Pertama-tama, nilai fungsi sin(x) dan semua turunannya akan disubstitusi dengan 0 untuk dimasukkan ke dalam rumus deret Taylor.

$f(0) = \sin 0 = 0$

$f'(x) = \cos x \ \rightarrow \ f'(0) = \cos 0 = 1$

$f''(x) = -\sin x \ \rightarrow \ f''(0) = -\sin 0 = 0$

$f'''(x) = -\cos x \ \rightarrow \ f'''(0) = -\cos 0 = -1$

$f^{(4)}(x) = \sin x \ \rightarrow \ f^{(4)}(0) = \sin 0 = 0$

Selanjutnya, hasil yang telah diperoleh akan dimasukkan ke dalam rumus di bawah ini.

$$f(x+h) = f(x) + f'(x)\cdot h + \frac{f''(x)\cdot h^2}{2!} + \frac{f'''(x)\cdot h^3}{3!} + \frac{f^{(4)}(x)\cdot h^4}{4!} + \cdots$$

$$f(0+h) = f(0) + f'(0)\cdot h + \frac{f''(0)\cdot h^2}{2!} + \frac{f'''(0)\cdot h^3}{3!} + \frac{f^{(4)}(0)\cdot h^4}{4!} + \cdots$$

$$f(h) = 0 + 1\cdot h + \frac{0\cdot h^2}{2!} + \frac{(-1)\cdot h^3}{3!} + \frac{0\cdot h^4}{4!} + \cdots$$

$$f(h) = 0 + h + 0 - \frac{h^3}{3!} + 0 + \cdots$$

$$f(h) = h - \frac{h^3}{3!} + \cdots$$

Dengan mengubah variabel h sebagai x, kita dapat memperoleh bentuk umum deret Taylor untuk sin(x) sebagai berikut:

$$f(x) = \sin x = x - \frac{x^3}{3!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2n+1}}{(2n+1)!}$$

Dengan memotong deret sampai turunan keempat, diperoleh pendekatan (aproksimasi) fungsi sin(x) di sekitar x = 0 sebagai:

$$f(x) = \sin x \approx x - \frac{x^3}{3!}$$

atau dapat ditulis juga sebagai:

$$f(x) = \sin x \approx x - \frac{1}{6}x^3$$

Pembuatan ekspansi deret Taylor sin(x) juga dapat dikomputasikan melalui program Python melalui kode program berikut.

```
1  # Import library yang diperlukan
2  import numpy as np
3  import matplotlib.pyplot as plt
✓ 0.0s
```

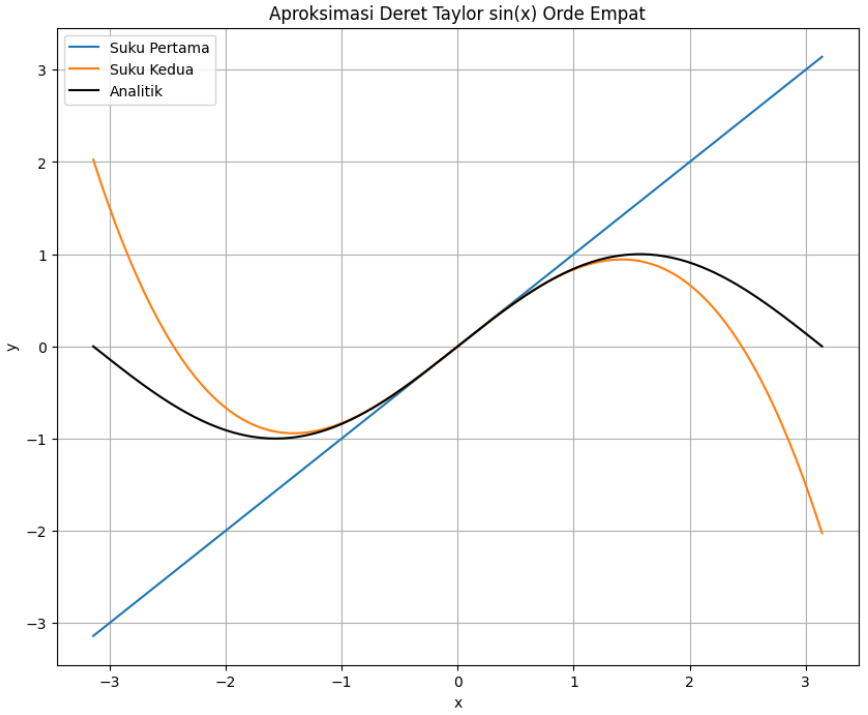
```python
# sin(x)

# Inisialisasi titik-titik absis koordinat penyusun kurva
x = np.linspace(-np.pi, np.pi, 200)
# Inisialisasi titik ordinat seluruhnya sebagai 0
y = np.zeros(len(x))

# Memberi label untuk legenda dan mengatur ukuran grafik plotting
labels = ["Suku Pertama", "Suku Kedua", "Suku Ketiga", "Suku Keempat"]
plt.figure(figsize=(10,8))

# Inisialisasi pengiterasi suku deret ke-n mulai dari 0
n = 0

# Lakukan iterasi selama orde deret lebih kecil atau sama dengan 4
while(2*n+1 <= 4):
    # Nilai y ditambah sesuai rumus deret Taylor sin(x) pada n tertentu
    y = y + ((-1)**n*x**(2*n+1))/(np.math.factorial(2*n+1))
    # Keluarkan suku deret dan plot pada grafik
    print(f"Suku ke-{n+1} : {((-1)**n)}x^{2*n+1}/{(2*n+1)}!")
    plt.plot(x, y, label=labels[n])
    # Tambahkan pengiterasi n untuk melanjutkan proses loop
    n += 1

# Plot grafik sin(x) sesungguhnya
plt.plot(x, np.sin(x), "k", label="Analitik")

```

```
28  # Lengkapi kelengkapan grafik lainnya
29  plt.title("Aproksimasi Deret Taylor sin(x) Orde Empat")
30  plt.grid()
31  plt.xlabel("x")
32  plt.ylabel("y")
33  plt.legend()
34  plt.show()
```

✓ 0.2s

Output yang dihasilkan berupa suku-suku deret Taylor sin(x) dan plot grafiknya sebagai berikut.

```
...  Suku ke-1 : 1x^1/1!
     Suku ke-2 : -1x^3/3!
```



Fungsi cos(x)

Berikut ini adalah perhitungan manual ekspansi Taylor berorde 4 untuk **cos(x)** di sekitar x = 0.

Pertama-tama, nilai fungsi cos(x) dan semua turunannya akan disubstitusi dengan 0 untuk dimasukkan ke dalam rumus deret Taylor.

$f(0) = \cos 0 = 1$

$f'(x) = -\sin x \;\rightarrow\; f'(0) = -\sin 0 = 0$

$f''(x) = -\cos x \;\rightarrow\; f''(0) = -\cos 0 = -1$

$f'''(x) = \sin x \;\rightarrow\; f'''(0) = \sin 0 = 0$

$f^{(4)}(x) = \cos x \;\rightarrow\; f^{(4)}(0) = \cos 0 = 1$

Selanjutnya, hasil yang telah diperoleh akan dimasukkan ke dalam rumus di bawah ini.

$$f(x+h) = f(x) + f'(x)\cdot h + \frac{f''(x)\cdot h^2}{2!} + \frac{f'''(x)\cdot h^3}{3!} + \frac{f^{(4)}(x)\cdot h^4}{4!} + \cdots$$

$$f(0+h) = f(0) + f'(0)\cdot h + \frac{f''(0)\cdot h^2}{2!} + \frac{f'''(0)\cdot h^3}{3!} + \frac{f^{(4)}(0)\cdot h^4}{4!} + \cdots$$

$$f(h) = 1 + 0\cdot h + \frac{(-1)\cdot h^2}{2!} + \frac{0\cdot h^3}{3!} + \frac{1\cdot h^4}{4!} + \cdots$$

$$f(h) = 1 + 0 - \frac{h^2}{2!} + 0 + \frac{h^4}{4!} + \cdots$$

$$f(h) = 1 - \frac{h^2}{2!} + \frac{h^4}{4!} + \cdots$$

Dengan mengubah variabel h sebagai x, kita dapat memperoleh bentuk umum deret Taylor untuk cos(x) sebagai berikut:

$$f(x) = \cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2n}}{(2n)!}$$

Dengan memotong deret sampai turunan keempat, diperoleh pendekatan (aproksimasi) fungsi cos(x) di sekitar x = 0 sebagai:

$$f(x) = \cos x \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!}$$

atau dapat ditulis juga sebagai:

$$f(x) = \cos x \approx 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4$$

Pembuatan ekspansi deret Taylor cos(x) juga dapat dikomputasikan melalui program Python melalui kode program berikut.

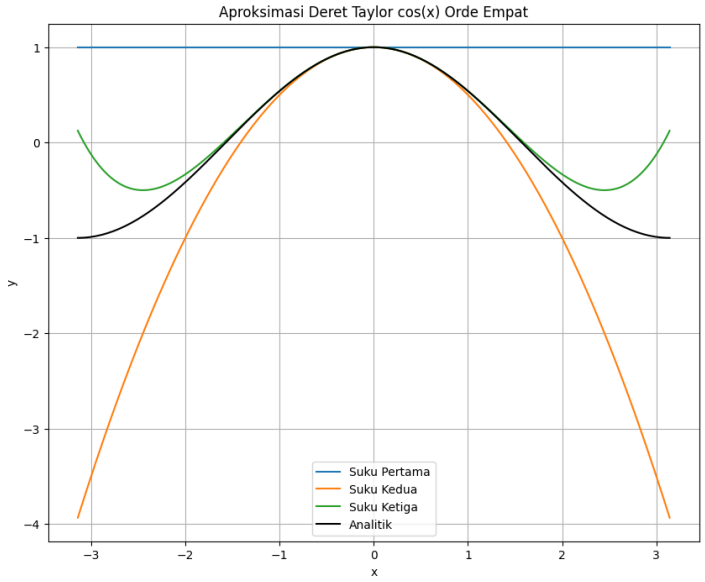
```python
# cos(x)

# Inisialisasi titik-titik absis koordinat penyusun kurva
x = np.linspace(-np.pi, np.pi, 200)
# Inisialisasi titik ordinat seluruhnya sebagai 0
y = np.zeros(len(x))

# Memberi label untuk legenda dan mengatur ukuran grafik plotting
labels = ["Suku Pertama", "Suku Kedua", "Suku Ketiga", "Suku Keempat"]
plt.figure(figsize=(10,8))

# Inisialisasi pengiterasi suku deret ke-n mulai dari 0
n = 0

# Lakukan iterasi selama orde deret lebih kecil atau sama dengan 4
while(2*n <= 4):
    # Nilai y ditambah sesuai rumus deret Taylor cos(x) pada n tertentu
    y = y + ((-1)**n*x**(2*n))/(np.math.factorial(2*n))
    # Keluarkan suku deret dan plot pada grafik
    print(f"Suku ke-{n+1} : {((-1)**n)}x^{2*n}/{(2*n)}!")
    plt.plot(x, y, label=labels[n])
    # Tambahkan pengiterasi n untuk melanjutkan proses loop
    n += 1

```

```python
# Plot grafik cos(x) sesungguhnya
plt.plot(x, np.cos(x), "k", label="Analitik")

# Lengkapi kelengkapan grafik lainnya
plt.title("Aproksimasi Deret Taylor cos(x) Orde Empat")
plt.grid()
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()
```

✓ 0.2s

Output yang dihasilkan berupa suku-suku deret Taylor cos(x) dan plot grafiknya sebagai berikut.

```
Suku ke-1 : 1x^0/0!
Suku ke-2 : -1x^2/2!
Suku ke-3 : 1x^4/4!
```



Fungsi sin(x)cos(x)

Berikut ini adalah perhitungan manual ekspansi Taylor berorde 4 untuk **sin(x)cos(x)** di sekitar x = 0.

Pertama-tama, nilai fungsi sin(x)cos(x) dan semua turunannya akan disubstitusi dengan 0 untuk dimasukkan ke dalam rumus deret Taylor.

$$f(x) = \sin x \cos x = \frac{1}{2} \cdot 2 \sin x \cos x = \frac{1}{2} \sin 2x$$

$$f(0) = \sin 0 \cos 0 = 0$$

$$f'(x) = \cos 2x \;\rightarrow\; f'(0) = \cos 0 = 1$$

$$f''(x) = -2 \sin 2x \;\rightarrow\; f''(0) = -2 \sin 0 = 0$$

$$f'''(x) = -4 \cos 2x \;\rightarrow\; f'''(0) = -4 \cos 0 = -4$$

$$f^{(4)}(x) = 8 \sin 2x \;\rightarrow\; f^{(4)}(0) = 8 \sin 0 = 0$$

Selanjutnya, hasil yang telah diperoleh akan dimasukkan ke dalam rumus di bawah ini.

$$f(x+h) = f(x) + f'(x) \cdot h + \frac{f''(x) \cdot h^2}{2!} + \frac{f'''(x) \cdot h^3}{3!} + \frac{f^{(4)}(x) \cdot h^4}{4!} + \cdots$$

$$f(0+h) = f(0) + f'(0) \cdot h + \frac{f''(0) \cdot h^2}{2!} + \frac{f'''(0) \cdot h^3}{3!} + \frac{f^{(4)}(0) \cdot h^4}{4!} + \cdots$$

$$f(h) = 0 + 1 \cdot h + \frac{0 \cdot h^2}{2!} + \frac{(-4) \cdot h^3}{3!} + \frac{0 \cdot h^4}{4!} + \cdots$$

$$f(h) = 0 + h + 0 - \frac{4h^3}{3!} + 0 + \cdots$$

$$f(h) = h - \frac{4h^3}{3!} + \cdots$$

Dengan mengubah variabel h sebagai x, kita dapat memperoleh bentuk umum deret Taylor untuk sin(x)cos(x) sebagai berikut:

$$f(x) = \sin x \cos x = x - \frac{4x^3}{3!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot 4^n \cdot x^{2n+1}}{(2n+1)!}$$

Dengan memotong deret sampai turunan keempat, diperoleh pendekatan (aproksimasi) fungsi sin(x)cos(x) di sekitar x = 0 sebagai:

$$f(x) = \sin x \cos x \approx x - \frac{4x^3}{3!}$$

atau dapat ditulis juga sebagai:

$$f(x) = \sin x \cos x \approx x - \frac{2}{3}x^3$$

Pembuatan ekspansi deret Taylor sin(x)cos(x) juga dapat dikomputasikan melalui program Python melalui kode program berikut.

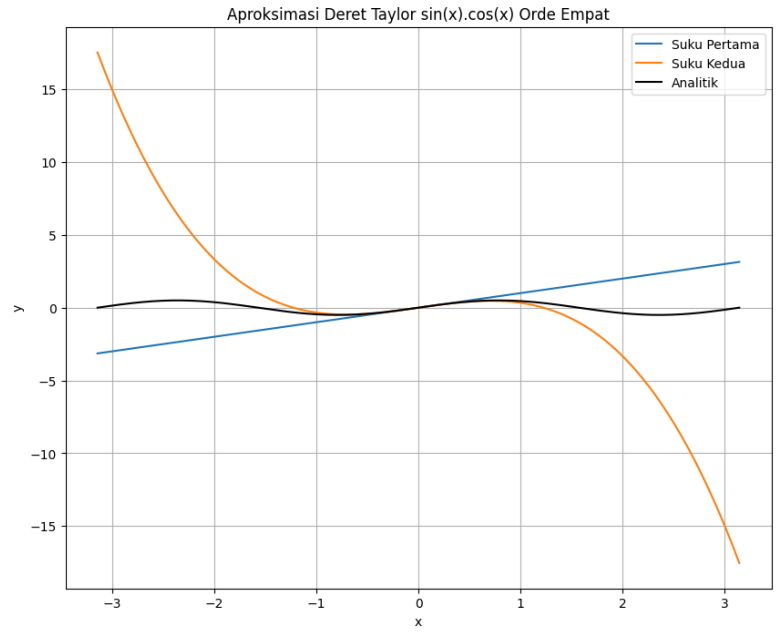
```
# sin(x)cos(x)

# Inisialisasi titik-titik absis koordinat penyusun kurva
x = np.linspace(-np.pi, np.pi, 200)
# Inisialisasi titik ordinat seluruhnya sebagai 0
y = np.zeros(len(x))

# Memberi label untuk legenda dan mengatur ukuran grafik plotting
labels = ["Suku Pertama", "Suku Kedua", "Suku Ketiga", "Suku Keempat"]
plt.figure(figsize=(10,8))

# Inisialisasi pengiterasi suku deret ke-n mulai dari 0
n = 0

```

```
# Lakukan iterasi selama orde deret lebih kecil atau sama dengan 4
while(2*n+1 <= 4):
    # Nilai y ditambah sesuai rumus deret Taylor sin(x).cos(x) pada n tertentu
    y = y + ((-1)**n*4**n*x**(2*n+1))/(np.math.factorial(2*n+1))
    # Keluarkan suku deret dan plot pada grafik
    print(f"Suku ke-{n+1} : {((-1)**n)*(4**n)}x^{2*n+1}/{(2*n+1)}!")
    plt.plot(x, y, label=labels[n])
    # Tambahkan pengiterasi n untuk melanjutkan proses loop
    n += 1

# Plot grafik sin(x).cos(x) sesungguhnya
plt.plot(x, np.sin(x)*np.cos(x), "k", label="Analitik")

# Lengkapi kelengkapan grafik lainnya
plt.title("Aproksimasi Deret Taylor sin(x).cos(x) Orde Empat")
plt.grid()
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()
```

✓ 0.2s

Output yang dihasilkan berupa suku-suku deret Taylor sin(x)cos(x) dan plot grafiknya sebagai berikut.

```
...  Suku ke-1 : 1x^1/1!
     Suku ke-2 : -4x^3/3!
```



b. Which produces less error for x=π/2: computing the Taylor expansion for sin and cos separately then multiplying the result together, or computing the Taylor expansion for the product first then plugging in x?

Jawab:

Untuk menguji besarnya error yang dihasilkan oleh kedua kondisi tersebut, kita dapat melakukan komputasi di program Python. Sebagai langkah awal, kita perlu melakukan import library (package) yang akan digunakan.

```
1  # Import library yang diperlukan
2  import numpy as np
```

✓ 0.1s

Selanjutnya, kita dapat mengomputasikan kedua kondisi yang ditanyakan sebagai berikut.

(1) Menghitung ekspansi Taylor untuk sin(x) dan cos(x) secara terpisah, lalu mengalikan hasilnya

```
# sin(x) dan cos(x) dihitung terpisah dari deret Taylor masing-masing

# Inisialisasi nilai x yang akan disubstitusi, dan inisialisasi y = 0
x = np.pi/2
y = 0

# Inisialisasi pengiterasi suku deret ke-n mulai dari 0
n = 0

# Lakukan iterasi selama orde deret lebih kecil atau sama dengan 4
while(2*n+1 <= 4):
    # Nilai y ditambah sesuai rumus deret Taylor sin(x) pada n tertentu dan masukkan x = pi/2
    y = y + ((-1)**n*x**(2*n+1))/(np.math.factorial(2*n+1))
    # Tambahkan pengiterasi n untuk melanjutkan proses loop
    n += 1

# Masukkan nilai y akhir iterasi ke dalam variabel taylorSin, lalu keluarkan hasilnya
taylorSin = y
print(f"Harga aproksimasi deret Taylor untuk sin(pi/2) = {taylorSin}")

# Inisialisasi ulang nilai y = 0 dan pengiterasi n = 0
y = 0
n = 0

# Lakukan iterasi selama orde deret lebih kecil atau sama dengan 4
while(2*n <= 4):
    # Nilai y ditambah sesuai rumus deret Taylor cos(x) pada n tertentu dan masukkan x = pi/2
    y = y + ((-1)**n*x**(2*n))/(np.math.factorial(2*n))
    # Tambahkan pengiterasi n untuk melanjutkan proses loop
    n += 1

```

```
# Masukkan nilai y akhir iterasi ke dalam variabel taylorCos, lalu keluarkan hasilnya
taylorCos = y
print(f"Harga aproksimasi deret Taylor untuk cos(pi/2) = {taylorCos}")

# Hitung hasil perkalian dari aproksimasi sin(pi/2) dan cos(pi/2), lalu keluarkan hasilnya
aproksimasi = taylorSin*taylorCos
print(f"Harga aproksimasi sin(pi/2).cos(pi/2) = {aproksimasi}")

# Hitung true error dengan membandingkan hasil perkalian aproksimasi dengan true value, keluarkan hasilnya
trueError = np.abs(np.sin(x)*np.cos(x) - aproksimasi)
print(f"Error = {trueError}")
```

✓ 0.0s

Output yang dihasilkan oleh kode tersebut adalah sebagai berikut.

```
Harga aproksimasi deret Taylor untuk sin(pi/2) = 0.9248322292886504
Harga aproksimasi deret Taylor untuk cos(pi/2) = 0.019968957764878226
Harga aproksimasi sin(pi/2).cos(pi/2) = 0.018467935726263235
Error = 0.018467935726263172
```

(2) Menghitung ekspansi Taylor untuk sin(x)cos(x), lalu mensubstitusikan nilai x ke dalam deret tersebut

```
# Menghitung langsung dari deret Taylor sin(x).cos(x)

# Inisialisasi nilai x yang akan disubstitusi, dan inisialisasi y = 0
x = np.pi/2
y = 0

# Inisialisasi pengiterasi suku deret ke-n mulai dari 0
n = 0

```

```python
# Lakukan iterasi selama orde deret lebih kecil atau sama dengan 4
while(2*n+1 <= 4):
    # Nilai y ditambah sesuai rumus deret Taylor sin(x).cos(x) pada n tertentu dan masukkan x = pi/2
    y = y + ((-1)**n*4**n*x**(2*n+1))/(np.math.factorial(2*n+1))
    # Tambahkan pengiterasi n untuk melanjutkan proses loop
    n += 1

# Hitung hasil aproksimasi sin(pi/2).cos(pi/2), lalu keluarkan hasilnya
aproksimasi = y
print(f"Harga aproksimasi sin(pi/2).cos(pi/2) = {aproksimasi}")

# Hitung true error dengan membandingkan hasil perkalian aproksimasi dengan true value, lalu keluarkan hasilnya
trueError = np.abs(np.sin(x)*np.cos(x) - aproksimasi)
print(f"Error = {trueError}")
```

✓ 0.0s

Output yang dihasilkan oleh kode tersebut adalah sebagai berikut.

```
Harga aproksimasi sin(pi/2).cos(pi/2) = -1.0130600632300881
Error = 1.0130600632300881
```

Berdasarkan perhitungan komputasi melalui program Python yang telah dilakukan, kita peroleh data bahwa error aproksimasi $\sin(\pi/2)\cos(\pi/2)$ dengan menghitung secara terpisah nilai $\sin(\pi/2)$ dan $\cos(\pi/2)$, lalu mengalikannya adalah 0,0185. Sementara itu, error aproksimasi dengan langsung mensubstitusi nilai $\pi/2$ pada deret $\sin(x)\cos(x)$ adalah 1,0131. Dengan demikian, kita dapat menyimpulkan bahwa **error yang lebih kecil diperoleh ketika kita mengomputasikan hasil aproksimasi deret Taylor sin(x) dan cos(x) pada x = π/2, lalu mengalikannya**.

c. Use the same order of Taylor series to approximate cos (π/4) and determine the truncation error bound. You may include either your manual calculation **OR** Python script for this question

Jawab:

Untuk mendapatkan nilai aproksimasi dari cos (π/4), kita hanya perlu mensubstitusikan nilai x = π/4 ke dalam persamaan deret Taylor untuk cos(x) yang telah ditentukan sebelumnya. Secara manual, kita dapat menuliskannya sebagai berikut.

$$\cos x \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!}$$

$$\cos\frac{\pi}{4} \approx 1 - \frac{1}{2!}\cdot\left(\frac{\pi}{4}\right)^2 + \frac{1}{4!}\cdot\left(\frac{\pi}{4}\right)^4$$

$$\cos\frac{\pi}{4} \approx 0{,}70742920670977$$

Sementara itu, untuk menentukan besarnya truncation error bound dari aproksimasi yang telah dilakukan, kita dapat melakukan perhitungan sebagai berikut.

Pertama-tama, dicari turunan selanjutnya dari fungsi cos(x) setelah turunan terakhir pada saat memformulasikan deret Taylor. Ketika membuat deret Taylor dari cos(x), turunan terakhir yang dilakukan adalah turunan keempat. Maka kita akan menghitung turunan kelima dari fungsi cos(x). Diketahui bahwa $f^{(4)}(x) = \cos x$, maka $f^{(5)}(x) = -\sin x$.

Kemudian, kita akan menggunakan rumus berikut untuk mencari truncation error bound dari aproksimasi tersebut.

$$E_n(x) = \frac{f^{(n+1)}(c)\cdot(x-a)^{n+1}}{(n+1)!}$$

dengan n adalah orde turunan terakhir dari deret Taylor, a adalah nilai acuan pendekatan deret Taylor (a = 0), dan x adalah nilai yang ingin diaproksimasi terhadap fungsi. Rumus tersebut dapat dimodifikasi menjadi:

$$|E_n(x)| \leq \frac{M|x-a|^{n+1}}{(n+1)!}$$

dengan M adalah nilai maksimum yang dapat dicapai oleh turunan ke n+1 dari fungsi tersebut. Kita dapat mensubstitusikan semua nilai sebagai berikut.

$$|E_n(x)| \leq \frac{M|x-0|^{4+1}}{(4+1)!}$$

$$|E_n(x)| \leq \frac{M|x|^5}{5!}$$

$$|E_n(x)| \leq \frac{M\left|\frac{\pi}{4}\right|^5}{5!}$$

Nilai maksimum yang dapat dicapai oleh sin(x) adalah 1, sehingga nilai M = 1.

$$|E_n(x)| \leq \frac{1 \cdot \left|\frac{\pi}{4}\right|^5}{5!}$$

$$|E_n(x)| \leq 0{,}0024904$$

Melalui perhitungan ini, diperoleh bahwa aproksimasi cos (π/4) ≈ 0,70742920670977 memiliki truncation error bound yang nilainya di bawah 0,0024904. Kita dapat membuktikan perhitungan ini melalui program Python sebagai berikut.

```python
# Import library yang diperlukan
import numpy as np
```

✓ 0.4s

```python
# Inisialisasi nilai x yang akan disubstitusi, dan inisialisasi y = 0
x = np.pi/4
y = 0

# Inisialisasi pengiterasi suku deret ke-n mulai dari 0
n = 0

```

```python
# Lakukan iterasi selama orde deret lebih kecil atau sama dengan 4
while(2*n <= 4):
    # Nilai y ditambah sesuai rumus deret Taylor cos(x) pada n tertentu dan masukkan x = pi/4
    y = y + ((-1)**n*x**(2*n))/(np.math.factorial(2*n))
    # Tambahkan pengiterasi n untuk melanjutkan proses loop
    n += 1

# Keluarkan hasil aproksimasi cos(pi/4)
print(f"Harga aproksimasi deret Taylor untuk cos(pi/4) = {y}")

# Hitung truncation error dengan membandingkan aproksimasi dengan true value, lalu keluarkan hasilnya
trueValue = np.cos(x)
print(f"True Value = {trueValue}")
truncationError = np.abs(np.cos(x) - y)
print(f"Truncation Error = {truncationError}")
```

✓ 0.0s

Output yang dihasilkan oleh kode program tersebut ditunjukkan sebagai berikut.

```
Harga aproksimasi deret Taylor untuk cos(pi/4) = 0.707429206709773
True Value = 0.7071067811865476
Truncation Error = 0.000322425523225478
```

Program Python tersebut menunjukkan keluaran hasil error sekitar 0,0003224, yang nilainya terbukti lebih kecil daripada batasan error yang telah dihitung secara manual sebelumnya, yaitu 0,0024904.

3. Given that $f(x) = x^3 - 0.3x^2 - 8.56x + 8.448$. **(35%)**

   a. Approximate $\int_0^{2\pi} f(x)\,dx$ with 20 evenly-spaced grid points over the whole interval using Riemann Integral, Trapezoid Rule, and Simpson's Rule. Explain the difference behind each of the method.

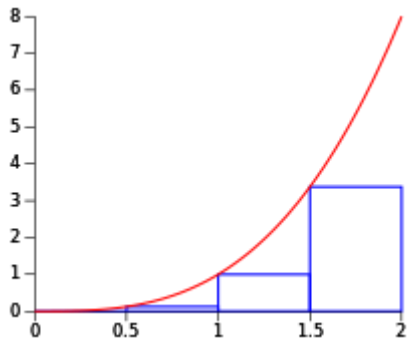
   Jawab:

Fungsi $f(x) = x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448$ akan diintegralkan dengan batas-batas dari 0 hingga 2π. Aproksimasi akan dilakukan dengan tiga metode berbeda, yaitu integral Riemann, aturan Trapezoid, dan aturan Simpson's dengan 20 buah grid points. Apabila terdapat n buah grid points pada proses pembagian partisi suatu luasan daerah fungsi, maka akan ada n-1 buah partisi yang dihasilkan. Maka dari itu, dari 20 buah grid points yang ditentukan, maka jumlah partisi yang membagi daerah di bawah kurva sebanyak 19 buah. Berdasarkan informasi awal ini, kita akan melakukan aproksimasi dengan membagi-bagi luasan di bawah kurva menjadi 19 buah partisi, lalu dihitung luasannya masing-masing, dan menjumlahkannya untuk memperoleh nilai pendekatan luas daerah di bawah kurva. Oleh karena integral tentu setara nilainya dengan luasan daerah di bawah kurva dari batas a ke b, maka nilai luas yang diperoleh dari perhitungan partisi-partisi tersebut akan menghampiri nilai integral tentu f(x) yang diberikan. Ketiga metode yang digunakan memiliki perbedaan pada bentuk partisi yang digunakan untuk membagi-bagi luasan daerah di bawah kurva. Perbedaan secara lebih rinci akan dijelaskan sebagai berikut.

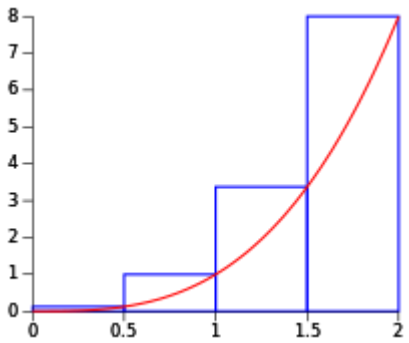
Integral Riemann

Metode integral Riemann merupakan metode pengintegrasian suatu fungsi secara numerik dengan membagi area di bawah kurva menggunakan *partisi-partisi yang berbentuk persegi panjang* dengan lebar yang sama besarnya untuk setiap partisi. Pada setiap partisi dibentuk persegi panjang setinggi kurva. Luas setiap partisi diperoleh dengan mengalikan panjang dan lebar masing-masing persegi panjang. Jumlah masing-masing luas tersebut digunakan untuk menaksir integral suatu fungsi dengan interval tertentu.

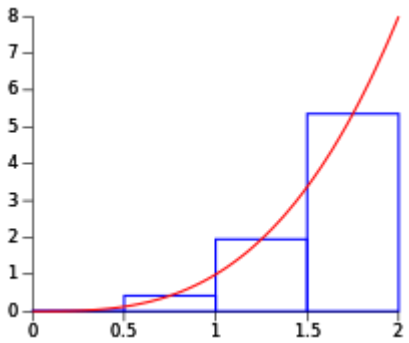
Partisi berbentuk persegi panjang yang digunakan untuk menghampiri kurva fungsi juga dapat dibedakan menjadi 3 jenis, yaitu Riemann kiri, Riemann kanan, dan Riemann tengah. Ketiganya berbeda pada posisi tinggi persegi panjang yang berimpit pada kurva fungsi. Pada Riemann kiri, setiap partisi persegi panjang bagian pojok kirinya selalu tepat berimpit dengan titik yang berada di kurva fungsi. Sedangkan Riemann kanan, bagian pojok kanan persegi panjangnya yang berimpit pada titik yang berada di kurva fungsi. Begitupun Riemann tengah, tepat bagian tengah dari pojok kiri dan pojok kanan persegi panjang yang berimpit pada titik yang berada di kurva fungsi. Ilustrasi sederhananya dapat digambarkan sebagai berikut (hanya ilustrasi, bukan kurva pada soal).



Riemann Kiri



Riemann Kanan



Riemann Tengah

Saya akan menggunakan program Python untuk mengaproksimasi besarnya luasan daerah yang dibentuk oleh 19 partisi tersebut, baik dengan Riemann kiri, kanan, dan tengah. Berikut adalah kode program untuk menaksir luasan daerah di bawah kurva $f(x) = x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448$ dari x = 0 hingga x = 2π dengan menggunakan **metode integral Riemann**.

```python
# Import library yang diperlukan
import numpy as np

# Deklarasikan fungsi f(x) yang akan digunakan
def f(x):
    return x**3 - 0.3*x**2 - 8.56*x + 8.448
```

✓ 0.0s

```python
# Riemann Integral

# Inisialisasi batas atas dan batas bawah pengintegralan
batas_bawah = 0
batas_atas = 2*np.pi
# Inisialisasi jumlah titik yang akan membagi luasan partisi
grid_points = 20

```

```python
# Hitung lebar partisi (h)
lebar_partisi = (batas_atas - batas_bawah)/(grid_points - 1)

# Tentukan nilai absis setiap partisi
x = np.linspace(batas_bawah, batas_atas, grid_points)
# Tentukan nilai ordinat setiap partisi
y = f(x)

# Tentukan koordinat titik tengah setiap partisi
x_mid = (x[:grid_points-1] + x[1:])/2
y_mid = f(x_mid)

# Hitung luas seluruh daerah dengan menggunakan ujung kiri partisi (Left Riemann)
leftRiemann = lebar_partisi * sum(y[:grid_points-1])
# Hitung luas seluruh daerah dengan menggunakan ujung kanan partisi (Right Riemann)
rightRiemann = lebar_partisi * sum(y[1:])
# Hitung luas seluruh daerah dengan menggunakan titik tengah partisi (Mid Riemann)
midRiemann = lebar_partisi * sum(y_mid)

# Keluarkan semua hasil perhitungan integral Riemann
print(f"Hasil integrasi numerik dengan Left Riemann sebesar {leftRiemann}")
print(f"Hasil integrasi numerik dengan Right Riemann sebesar {rightRiemann}")
print(f"Hasil integrasi numerik dengan Mid Riemann sebesar {midRiemann}")
```

✓ 0.0s

Output yang dihasilkan oleh kode program tersebut adalah sebagai berikut.

```
Hasil integrasi numerik dengan Left Riemann sebesar 219.82600410076233
Hasil integrasi numerik dengan Right Riemann sebesar 280.1520639859213
Hasil integrasi numerik dengan Mid Riemann sebesar 248.42158035858884
```

Aturan Trapezoid

Aturan Trapezoid merupakan metode pengintegrasian suatu fungsi secara numerik dengan membagi area di bawah kurva menggunakan *partisi-partisi yang berbentuk trapesium* dengan lebar yang sama besarnya untuk setiap partisi. Pada setiap partisi dibentuk trapesium yang sisi sejajarnya setinggi kurva. Luas setiap partisi diperoleh dengan menghitung luas setiap trapesium yang terbentuk. Jumlah masing-masing luas tersebut digunakan untuk menaksir integral suatu fungsi dengan interval tertentu.

Saya akan menggunakan program Python untuk mengaproksimasi besarnya luasan daerah yang dibentuk oleh 19 partisi tersebut dengan aturan Trapezoid. Berikut adalah kode program untuk menaksir luasan daerah di bawah kurva $f(x) = x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448$ dari x = 0 hingga x = 2π dengan menggunakan **aturan Trapezoid**.

```python
# Import library yang diperlukan
import numpy as np

# Deklarasikan fungsi f(x) yang akan digunakan
def f(x):
    return x**3 - 0.3*x**2 - 8.56*x + 8.448
```

✓ 0.0s

```python
# Trapezoid Rule

# Inisialisasi batas atas dan batas bawah pengintegralan
batas_bawah = 0
batas_atas = 2*np.pi
# Inisialisasi jumlah titik yang akan membagi luasan partisi
grid_points = 20

# Hitung lebar partisi (h)
lebar_partisi = (batas_atas - batas_bawah)/(grid_points - 1)

```

```
# Tentukan nilai absis setiap partisi
x = np.linspace(batas_bawah, batas_atas, grid_points)
# Tentukan nilai ordinat setiap partisi
y = f(x)

# Hitung luas seluruh daerah dengan menggunakan aturan Trapezoid
trapezoid = lebar_partisi * (y[0] + y[grid_points-1] + 2*sum(y[1:grid_points-1]))/2

# Keluarkan hasil perhitungan Trapezoid Rule
print(f"Hasil integrasi numerik dengan Trapezoid Rule sebesar {trapezoid}")
```

✓ 0.0s

Output yang dihasilkan oleh kode program tersebut adalah sebagai berikut.

```
...  Hasil integrasi numerik dengan Trapezoid Rule sebesar 249.98903404334183
```

Aturan Simpson

Aturan Simpson merupakan metode pengintegrasian suatu fungsi secara numerik dengan membagi area di bawah kurva menggunakan *partisi-partisi yang merupakan suatu kurva polinomial berderajat dua atau tiga* dengan lebar yang sama besarnya untuk setiap partisi. Apabila titik-titik data dihampiri dengan suatu kurva polinomial berderajat dua, maka aturan ini dispesifikkan sebagai Aturan Simpson 1/3. Sementara jika titik-titik data dihampiri dengan suatu kurva polinomial berderajat tiga, maka aturan ini dispesifikkan sebagai Aturan Simpson 3/8. Pada setiap partisi dibentuk sebuah parabola yang mengikuti kelengkungan kurva. Luas setiap partisi diperoleh dengan menghitung luas setiap cekungan yang terbentuk. Jumlah masing-masing luas tersebut digunakan untuk menaksir integral suatu fungsi dengan interval tertentu.

Saya akan menggunakan program Python untuk mengaproksimasi besarnya luasan daerah yang dibentuk oleh 19 partisi tersebut dengan aturan Simpson, baik Simpson 1/3 maupun Simpson 3/8.

Berikut adalah kode program untuk menaksir luasan daerah di bawah kurva $f(x) = x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448$ dari x = 0 hingga x = 2π dengan menggunakan **aturan Simpson 1/3**.

```python
# Import library yang diperlukan
import numpy as np

# Deklarasikan fungsi f(x) yang akan digunakan
def f(x):
    return x**3 - 0.3*x**2 - 8.56*x + 8.448
```

✓ 0.0s

```python
# Simpson 1/3 Rule

# Inisialisasi batas atas dan batas bawah pengintegralan
batas_bawah = 0
batas_atas = 2*np.pi
# Inisialisasi jumlah titik yang akan membagi luasan partisi
grid_points = 20

# Hitung lebar partisi (h)
h = (batas_atas - batas_bawah)/(grid_points - 1)

# Tentukan nilai absis setiap partisi
x = np.linspace(batas_bawah, batas_atas, grid_points)
# Tentukan nilai ordinat setiap partisi
y = f(x)

# Hitung luas seluruh daerah dengan menggunakan aturan Simpson 1/3
simpson = (h/3)*(y[0] + 2*sum(y[2:grid_points-1:2]) + 4*sum(y[1:grid_points-1:2]) + y[grid_points-1])

# Keluarkan hasil perhitungan Simpson 1/3 Rule
print(f"Hasil integrasi numerik dengan Simpson 1/3 Rule sebesar {simpson}")
```

✓ 0.0s

Output yang dihasilkan oleh kode program tersebut adalah sebagai berikut.

```
...   Hasil integrasi numerik dengan Simpson 1/3 Rule sebesar 229.83706760693207
```

Sementara berikut ini merupakan kode program untuk menaksir luasan daerah di bawah kurva $f(x) = x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448$ dari x = 0 hingga x = 2π dengan menggunakan **aturan Simpson 3/8**.

```python
# Simpson 3/8 Rule

# Inisialisasi batas atas dan batas bawah pengintegralan
batas_bawah = 0
batas_atas = 2*np.pi
# Inisialisasi jumlah titik yang akan membagi luasan partisi
grid_points = 20

# Hitung lebar partisi (h)
h = (batas_atas - batas_bawah)/(grid_points - 1)

# Tentukan nilai absis setiap partisi
x = np.linspace(batas_bawah, batas_atas, grid_points)
# Tentukan nilai ordinat setiap partisi
y = f(x)

# Hitung luas seluruh daerah dengan menggunakan aturan Simpson 3/8
simpson = (3*h/8)*(y[0] + 3*sum(y[1:grid_points-1:3]) + 3*sum(y[2:grid_points-1:3]) + 2*sum(y[3:grid_points-1:3]) + y

# Keluarkan hasil perhitungan Simpson 3/8 Rule
print(f"Hasil integrasi numerik dengan Simpson 3/8 Rule sebesar {simpson}")
```

✓ 0.0s

Output yang dihasilkan oleh kode program tersebut adalah sebagai berikut.

```
...   Hasil integrasi numerik dengan Simpson 3/8 Rule sebesar 234.64102072007026
```

b. Compared to the methods above, do you think that analitycal integration could be more convenient to be done?

Jawab:

Pada kasus di soal ini, proses integrasi secara analitik (eksplisit) masih lebih mudah dan nyaman untuk dilakukan dibanding mengintegrasikannya secara numerik seperti yang telah dilakukan pada soal 3a. Alasannya, karena fungsi f(x) yang diberikan masih berupa suatu fungsi polinomial berorde 3. Fungsi polinomial terbilang sangat mudah untuk diintegralkan secara analitik, sehingga nilai eksak dari integral tentu yang ditanyakan juga dapat dengan mudah ditentukan menggunakan formula yang ada. Berbeda halnya dengan teknik pengintegralan secara numerik, di mana kita perlu membagi partisi sebanyak 19 buah, lalu menghitung luasan masing-masing partisi. Apabila dilakukan secara manual, hal ini tentu merepotkan karena banyaknya partisi yang perlu dihitung luasnya. Meskipun saya telah menggunakan program Python untuk mengerjakan soal nomor 3a, tetap saja saya perlu mengubah algoritma pengerjaan manual tersebut ke dalam kode-kode Python terlebih dahulu. Apabila saya tidak mahir dalam menggunakan bahasa Python, pastinya saya akan kesulitan untuk mengomputasikan perhitungan luas yang nilainya pun masih belum terlalu dekat dengan nilai aslinya.

Akan tetapi, metode numerik tentunya akan sangat berguna jika fungsi yang akan diintegralkan merupakan fungsi eksponensial yang tidak bisa diintegralkan secara analitik karena terlalu kompleks. Dalam hal ini, fungsi polinomial masih bukan tergolong fungsi yang sulit diintegralkan secara eksplisit dan analitik.

c. Use polynomial interpolation to compute $f'(x)$ and $f''(x)$ at $x = 0$, using the discrete data below

x	-1.1	-0.3	0.8	1.9
$f(x)$	16.170	10.962	1.920	-2.040

Jawab:

Interpolasi polinomial dapat dilakukan untuk mencari formula simbolik aproksimasi untuk $f(x)$, sehingga kita dapat menurunkannya secara langsung di titik x = 0. Dengan 4 buah data points, kita dapat melakukan interpolasi kubik, dengan rincian langkah sebagai berikut.

Bentuk polinomial kubik yang menjadi tujuan akhir perhitungan adalah $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$. Setiap harga x dan f(x) disubstitusikan ke dalam persamaan tersebut sehingga membentuk sebuah sistem persamaan linear empat variabel.

$f(-1{,}1) = a_0 + a_1(-1{,}1) + a_2(-1{,}1)^2 + a_3(-1{,}1)^3 \quad \longrightarrow \quad a_0 - 1{,}1a_1 + 1{,}21a_2 - 1{,}331a_3 = 16{,}170$

$f(-0{,}3) = a_0 + a_1(-0{,}3) + a_2(-0{,}3)^2 + a_3(-0{,}3)^3 \quad \longrightarrow \quad a_0 - 0{,}3a_1 + 0{,}09a_2 - 0{,}027a_3 = 10{,}962$

$f(0{,}8) = a_0 + a_1(0{,}8) + a_2(0{,}8)^2 + a_3(0{,}8)^3 \quad \longrightarrow \quad a_0 + 0{,}8a_1 + 0{,}64a_2 + 0{,}512a_3 = 1{,}920$

$f(1{,}9) = a_0 + a_1(1{,}9) + a_2(1{,}9)^2 + a_3(1{,}9)^3 \quad \longrightarrow \quad a_0 + 1{,}9a_1 + 3{,}61a_2 + 6{,}859a_3 = -2{,}040$

Sistem persamaan linear tersebut dapat diubah ke dalam bentuk matriks sebagai berikut.

$$\begin{bmatrix} 1 & -1{,}1 & 1{,}21 & -1{,}331 \\ 1 & -0{,}3 & 0{,}09 & -0{,}027 \\ 1 & 0{,}8 & 0{,}64 & 0{,}512 \\ 1 & 1{,}9 & 3{,}61 & 6{,}859 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 16{,}170 \\ 10{,}962 \\ 1{,}920 \\ -2{,}040 \end{bmatrix}$$

Apabila sistem persamaan linear tersebut diselesaikan, diperoleh harga-harga koefisien fungsi sebagai berikut.

$$a_0 = 8{,}448$$

$$a_1 = -8{,}56$$

$$a_2 = -0{,}3$$

$$a_3 = 1$$

Maka, fungsi yang diperoleh berdasarkan hasil interpolasi adalah $f(x) = 8{,}448 - 8{,}56x - 0{,}3x^2 + x^3$, atau bisa disusun dari derajat tertinggi menjadi $f(x) = x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448$.

Turunan pertama dari fungsi $f(x)$ didefinisikan sebagai berikut:

$$f'(x) = \frac{d}{dx} f(x) = \frac{d}{dx}(x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448)$$

$$f'(x) = 3x^2 - 0{,}6x - 8{,}56$$

Untuk x = 0, diperoleh harga:

$f'(0) = 3(0)^2 - 0{,}6(0) - 8{,}56$

$f'(0) = -8{,}56$

Sementara itu, turunan kedua dari fungsi $f(x)$ didefinisikan sebagai berikut:

$f''(x) = \frac{d}{dx} f'(x) = \frac{d}{dx}(3x^2 - 0{,}6x - 8{,}56)$

$f''(x) = 6x - 0{,}6$

Untuk x = 0, diperoleh harga:

$f''(0) = 6(0) - 0{,}6$

$f''(0) = -0{,}6$

d. Calculate the accuracy result compared to the initial $f(x)$

Jawab:

Setelah melakukan proses diferensial dan integrasi secara numerik, kita akan mencoba menghitung seberapa akurat nilai-nilai yang diperoleh dari perhitungan numerik jika dibandingkan dengan perhitungan analitik berdasarkan fungsi f(x) yang diberikan pada soal.

Fungsi $f(x)$ didefinisikan sebagai $f(x) = x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448$. Fungsi pada soal ini rupanya sama persis dengan hasil interpolasi polinomial berorde tiga yang telah dilakukan pada soal 3c. Akibatnya, ketika fungsi ini diturunkan secara analitik, maka diperoleh $f'(x) = 3x^2 - 0{,}6x - 8{,}56$. Untuk x = 0, maka diperoleh $f'(0) = -8{,}56$. Begitu pula ketika diturunkan sekali lagi secara analitik, maka diperoleh $f''(x) = 6x - 0{,}6$. Untuk x = 0, diperoleh $f''(0) = -0{,}6$. Nilai-nilai yang diperoleh berdasarkan perhitungan analitik ini tentu **sama persis** dengan hasil yang diperoleh dari soal 3c karena fungsi f(x) yang ditemukan dari interpolasi polinomial **tepat sama persis dengan fungsi f(x) awal**, sehingga proses diferensial numerik dengan interpolasi polinomial pada soal 3c memiliki tingkat keakuratan 100%.

Sementara itu, untuk proses integrasi numerik, ada beberapa metode yang telah digunakan pada soal 3a, yaitu integral Riemann (kiri, kanan, dan tengah), aturan Trapezoid, dan aturan Simpson (Simpson 1/3 dan Simpson 3/8). Kita akan mencoba mengintegralkan fungsi tersebut secara analitik dan mencari nilai integral tentu dari batas 0 hingga 2π, kemudian menghitung true error dari kedua jenis perhitungan yang telah dilakukan (numerik 19 panel dan analitik).

Perhitungan integral tentu secara analitik dijabarkan sebagai berikut.

$$\int_0^{2\pi} x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448\ dx = \frac{1}{3+1}x^{3+1} - \frac{0{,}3}{2+1}x^{2+1} - \frac{8{,}56}{1+1}x^{1+1} + \frac{8{,}448}{0+1}x^{0+1}\ \Big|_0^{2\pi}$$

$$\int_0^{2\pi} x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448\ dx = \frac{1}{4}x^4 - \frac{0{,}3}{3}x^3 - \frac{8{,}56}{2}x^2 + \frac{8{,}448}{1}x^1\ \Big|_0^{2\pi}$$

$$\int_0^{2\pi} x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448\ dx = \frac{1}{4}x^4 - 0{,}1x^3 - 4{,}28x^2 + 8{,}448x\ \Big|_0^{2\pi}$$

$$\int_0^{2\pi} x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448\ dx = \left(\frac{1}{4}(2\pi)^4 - 0{,}1(2\pi)^3 - 4{,}28(2\pi)^2 + 8{,}448(2\pi)\right) - \left(\frac{1}{4}(0)^4 - 0{,}1(0)^3 - 4{,}28(0)^2 + 8{,}448(0)\right)$$

$$\int_0^{2\pi} x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448\ dx = \left(\frac{1}{4}(2\pi)^4 - 0{,}1(2\pi)^3 - 4{,}28(2\pi)^2 + 8{,}448(2\pi)\right) - 0$$

$$\int_0^{2\pi} x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448\ dx = \frac{1}{4}(2\pi)^4 - 0{,}1(2\pi)^3 - 4{,}28(2\pi)^2 + 8{,}448(2\pi)$$

$$\int_0^{2\pi} x^3 - 0{,}3x^2 - 8{,}56x + 8{,}448\ dx = 248{,}944$$

Hasil pengintegralan secara analitik ini telah dibulatkan ke dalam 3 tempat desimal. Perhitungan ini kemudian akan dijadikan sebagai true value dalam proses menghitung true error. Berikut ini dirincikan kembali hasil perhitungan aproksimasi dengan berbagai metode pada soal 3a, yang dibulatkan ke dalam 3 tempat desimal:

- Left Riemann = 219,826
- Right Riemann = 280,152
- Mid Riemann = 248,422
- Trapezoid Rule = 249,989
- Simpson 1/3 Rule = 229,837
- Simpson 3/8 Rule = 234,641

Kemudian, dari nilai-nilai aproksimasi tersebut, akan dihitung besarnya absolute relative true error sebagai berikut.

- Integral Riemann

  Left Riemann memiliki tingkat error terhadap true value sebesar:

  $$|\epsilon_t| = \left|\frac{\text{True Value} - \text{Approximate Value}}{\text{True Value}}\right| \times 100\%$$

  $$|\epsilon_t| = \left|\frac{248{,}944 - 219{,}826}{248{,}944}\right| \times 100\%$$

  $$|\epsilon_t| = \left|\frac{29{,}118}{248{,}944}\right| \times 100\% = \mathbf{11{,}70\%}$$

  Right Riemann memiliki tingkat error terhadap true value sebesar:

  $$|\epsilon_t| = \left|\frac{\text{True Value} - \text{Approximate Value}}{\text{True Value}}\right| \times 100\%$$

  $$|\epsilon_t| = \left|\frac{248{,}944 - 280{,}152}{248{,}944}\right| \times 100\%$$

$$|\epsilon_t| = \left|\frac{-31{,}208}{248{,}944}\right| \times 100\% = \mathbf{12{,}54\%}$$

Mid Riemann memiliki tingkat error terhadap true value sebesar:

$$|\epsilon_t| = \left|\frac{\text{True Value} - \text{Approximate Value}}{\text{True Value}}\right| \times 100\%$$

$$|\epsilon_t| = \left|\frac{248{,}944 - 248{,}422}{248{,}944}\right| \times 100\%$$

$$|\epsilon_t| = \left|\frac{0{,}522}{248{,}944}\right| \times 100\% = \mathbf{0{,}21\%}$$

- Trapezoid Rule

  Aturan Trapezoid memiliki tingkat error terhadap true value sebesar:

$$|\epsilon_t| = \left|\frac{\text{True Value} - \text{Approximate Value}}{\text{True Value}}\right| \times 100\%$$

$$|\epsilon_t| = \left|\frac{248{,}944 - 249{,}989}{248{,}944}\right| \times 100\%$$

$$|\epsilon_t| = \left|\frac{-1{,}045}{248{,}944}\right| \times 100\% = \mathbf{0{,}42\%}$$

- Simpson's Rule

  Aturan Simpson 1/3 memiliki tingkat error terhadap true value sebesar:

$$|\epsilon_t| = \left|\frac{\text{True Value} - \text{Approximate Value}}{\text{True Value}}\right| \times 100\%$$

$$|\epsilon_t| = \left|\frac{248{,}944 - 229{,}837}{248{,}944}\right| \times 100\%$$

$$|\epsilon_t| = \left|\frac{19{,}107}{248{,}944}\right| \times 100\% = \mathbf{7{,}68\%}$$

Aturan Simpson 3/8 memiliki tingkat error terhadap true value sebesar:

$$|\epsilon_t| = \left|\frac{\text{True Value} - \text{Approximate Value}}{\text{True Value}}\right| \times 100\%$$

$$|\epsilon_t| = \left|\frac{248{,}944 - 234{,}641}{248{,}944}\right| \times 100\%$$

$$|\epsilon_t| = \left|\frac{14{,}303}{248{,}944}\right| \times 100\% = \mathbf{5{,}75\%}$$

Berdasarkan perhitungan error tersebut, dapat disimpulkan bahwa tingkat akurasi aproksimasi integrasi numerik secara berturut-turut dari yang paling akurat hingga ke yang paling tidak akurat adalah sebagai berikut:

Mid Riemann > Trapezoid Rule > Simpson 3/8 Rule > Simpson 1/3 Rule > Left Riemann > Right Riemann

Note for Lecturers:

1. The lecturers are advised to assess student's understanding towards the topics included in the assignment.
2. The students will submit their answer in .PDF format through BINUS Maya.
3. The deadline of this comprehensive assignment is at the end the semester.

Catatan dari Mahasiswa untuk Dosen:

Izin, Pak Anom. Saya ingin memberi sedikit masukan untuk soal AoL ini agar ada perubahan untuk penugasan AoL ke depannya.

1. Soal nomor 1 sudah saya konfirmasi dengan Bapak, bahwa tahun harus menjadi data pada sumbu x. Sementara soal aslinya malah menjadikan tahun sebagai sumbu y, temperatur sebagai sumbu x.

2. Soal nomor 3c, saya mencoba menanyakan di grup tapi belum dibalas. Jadi saya mencoba menjelaskan asumsi saya di sini. Untuk x = –1,1 tertulis f(x) = 15,180. Padahal, jika nilai –1,1 dimasukkan ke dalam f(x), maka nilai f(–1,1) seharusnya 16,170. Bukan 15,180. Saya melakukan perubahan kecil pada soal, saya menggunakan f(–1,1) = 16,170 sesuai perhitungan sebenarnya. Jadi, semua perhitungan yang saya lakukan di soal 3c sudah menggunakan angka sebenarnya, yaitu 16,170 supaya perhitungan menjadi tepat. Akan tetapi, perbedaan versi angka ini telah mengakibatkan kebingungan di antara mahasiswa-mahasiswa kelas LD01, sehingga ada yang tetap menggunakan angka soal 15,180, dan ada juga yang menggunakan angka sesungguhnya 16,170. Hal ini tentu menyebabkan ketidakseragaman pengerjaan soal di antara mahasiswa kelas LD01. Setelah saya teliti, saya menyadari bahwa nilai f(–2,2) = 15,180. Dari sini, saya berpikir bahwa soal nomor 3c sepertinya salah ketik. Itulah sebabnya, saya sendiri menggunakan angka 16,170 sebagai angka sesungguhnya. Akan tetapi, saya menyadari bahwa mengubah soal bukan hal yang lumrah, oleh sebab itulah saya mengonfirmasi kepada Bapak pada kolom masukan ini.

3. Beberapa soal tidak secara eksplisit menuliskan mode pengerjaan untuk soal tersebut. Misalnya nomor 1 dan 3. Hampir semua mahasiswa cukup bingung dengan mode pengerjaan untuk soal tersebut, apakah dikerjakan manual atau dikerjakan melalui coding. Mungkin lebih baik jika soal diperjelas dengan keterangan mode pengerjaannya.

Sekian beberapa masukan dari saya. Semoga Bapak dapat memaklumi beberapa perbedaan versi jawaban atau angka soal yang ditemukan pada sejumlah mahasiswa akibat kekurangan tersebut. Terima kasih atas kesempatan memberi masukannya, Pak. Semoga Bapak sehat selalu dan terus semangat dalam mengajar dan mencerdaskan generasi muda bangsa.