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

ASSESSMENT FORM

Course: MATH6201051 - Scientific Computing

Method of Assessment: Case Study

Semester/Academic Year : 2/2022-2023

Name of Lecturer : **DIMAS ELANG SETYOKO, S.Kom., M.Cs.**
Date : **June 11th, 2023**
Class : **Scientific Computing**
Topic : **Regression and Interpolation**
Taylor Series
Numerical Differentiation
Numerical Integration

Name:	**Michael Geraldin 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

Remarks:

ASSESSMENT METHOD

Instructions

This is Individual Assignment

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 (y)	Temperature (x, ^{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.
   b. Perform a least-square regression of the above data to estimate the temperature in even years.
   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.
   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.

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
   b. Which produces less error for $x=\pi/2$: 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?
   c. Use the same order of Taylor series to approximate cos ($\pi/4$) and determine the truncation error bound. You may include either your manual calculation **OR** Python script for this question

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.
   b. Compared to the methods above, do you think that explicit integration could be more convenient to be done?
   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
$f(x)$	15.180	10.962	1.920	-2.040

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

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.

Evaluation:

Component	Weight
THEORY: ASSIGNMENT	10%
THEORY: FINAL EXAM	30%
THEORY: MID EXAM	20%
LAB: FINAL EXAM	15%
LAB: MID EXAM	15%
AoL - CASE STUDY	10%

Answer

1. A)

- Linear Interpolation ($v(t) = a_0 + a_1 t$), pakai 2 tahun terdekat dari tahun yang ingin dicari

(y)	Temperature
1981	14.1999
1983	14.2411

$t_0 = 1981, v(t_0) = 14{,}1999$

$t_1 = 1983, v(t_1) = 14{,}2411$

$v(1981) = a_0 + a_1(1981) = 14{,}1999$

$v(1982) = a_0 + a_1(1983) = 14{,}2411$

$$\begin{bmatrix} 1 & 1981 \\ 1 & 1983 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 14{,}1999 \\ 14{,}2411 \end{bmatrix}$$

$a_0 + 1981a_1 = 14{,}1999$

$a_0 + 1982a_1 = 14{,}2411$ (Eliminasi)

$2a_1 = 0{,}0412$

$a_1 = 0{,}0206$

$a_0 = -26{,}6087$

$v(1982) = -26{,}6087 + 0{,}0206\ (1982) = 14{,}2205$

Dengan cara yang sama untuk tahun genap lainnya, menggunakan 4 titik terdekatnya, maka akan didapat sebagai berikut.

$v(1984) = a_0 + a_1(1984)$

$v(1984) = 219{,}38245 - 205{,}2448$

$v(1984) = 14{,}13765$

...

Even Years	Temperature
1982	14,2205
1984	14,13765
1986	14,1519
1988	14,2333
1990	14,25125
1992	14,2454
1994	14,2715
1996	14,3882
1998	14,38125

- Quadratic Interpolation ($v(t) = a_0 + a_1 t + a_2 t^2$), pakai 3 tahun terdekat dari tahun yang ingin dicari.

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

  $v(1983) = a_0 + a_1(1983) + a_2(1983)^2 = 14{,}2411$

  $v(1985) = a_0 + a_1(1985) + a_2(1985)^2 = 14{,}0342$

  $$\begin{bmatrix} 1 & 1981 & 3924361 \\ 1 & 1983 & 3932389 \\ 1 & 1985 & 3940225 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 14{,}1999 \\ 14{,}2411 \\ 14{,}0342 \end{bmatrix}$$

  Gauss-Jordan

  $$\begin{bmatrix} 1 & 1981 & 3924361 & 14{,}1999 \\ 1 & 1983 & 3932389 & 14{,}2411 \\ 1 & 1985 & 3940225 & 14{,}0342 \end{bmatrix} R_2 \to R_2 - R_1, R_3 \to R_3 - R_1$$

  ...

  $$\begin{bmatrix} 1 & 0 & 0 & -121853{,}7257375 \\ 0 & 1 & 0 & 122{,}95415 \\ 0 & 0 & 1 & -0{,}0310125 \end{bmatrix}$$

  $a_0 = -121853{,}7257375$
  $a_1 = 122{,}95415$
  $a_2 = -0{,}0310125$

  $v(1982) = -121853{,}7257375 + 122{,}95415(1982) - 0{,}0310125\,(1982)^2 = 14{,}25151$

Untuk 1984 itu bisa juga pakai persamaan yang digunakan dalam 1982 karena 3 titik terdekatnya sama.

$v(1984) = -121853{,}7257375 + 122{,}95415(1984) - 0{,}0310125\,(1984)^2 = 14{,}1686$

$Even\ years$: 1986

$$\begin{bmatrix} 1 & 1983 & 3932289 \\ 1 & 1985 & 3940225 \\ 1 & 1987 & 3948169 \end{bmatrix} = \begin{bmatrix} 14{,}2411 \\ 14{,}0342 \\ 14{,}2696 \end{bmatrix}$$

$a_0 = 217845.0807625$
$a_1 = -219{,}48425$
$a_2 = 0{,}0552875$

$v(1986) = a_0 + a_1(1986) = a_2(1986)^2$
$v(1986) = 217.845{,}0807625 - 435.895{,}7205 + 218.064{,}73635$
$v(1986) = 14{,}0966$

Dengan cara yang sama untuk tahun genap lainnya, menggunakan 4 titik terdekatnya.

$\rightarrow 1988$
$a_0 = -152.071{,}1078$
$a_1 = 153{,}0397$
$a_2 = -0{,}0385$
$v(1988) = a_0 + a1(1988) + a_2(1988)^2$
$v(1988) = 14{,}2718$

$\rightarrow 1990$
$a_0 = 89.553{,}0348625$
$a_1 = -90{,}043$
$a_2 = 0{,}0226375$
$v(1990) = a_0 + a_1(1990) + a_2(1990)^2$
$v(1990) = 14{,}2286$

$\rightarrow 1992$
$a_0 = -113.303{,}0364125$
$a_1 = 113{,}8325$
$a_2 = -0{,}0285875$
$v(1992) = a_0 + a_1(1992) + a_2(1992)^2$
$v(1992) = 14{,}2739$

$\rightarrow 1994$
$a_0 = 145.265{,}868825$
$a_1 = -145{,}7749$
$a_2 = 0{,}03657$
$v(1994) = a_0 + a_1(1994) + a_2(1994)^2$
$v(1994) = 14{,}2349$

$\rightarrow 1996$
$a_0 = -55.523{,}898675$
$a_1 = 55{,}6191$
$a_2 = -0{,}013925$
$v(1996) = a_0 + a_1(1996) + a_2(1996)^2$
$v(1996) = 14{,}4021$

$\rightarrow 1998$
$a_0 = -67.724{,}9446125$
$a_1 = 67{,}8446$
$a_2 = -0{,}0169875$
$v(1998) = a_0 + a_1(1998) + a_2(1998)^2$
$v(1998) = 14{,}3982$

Even Years	Temperature
1982	14,2515
1984	14,1686
1986	14,0966
1988	14,2718
1990	14,2286
1992	14,2739
1994	14,2349
1996	14,4021
1998	14,3982

- Cubic Interpolation ($v(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$), pakai 4 tahun terdekat dari tahun yang ingin dicari

$$v(1981) = a_0 + a_1(1981) + a_2(1981)^2 + a_3(1981)^3 = 14{,}1999$$
$$v(1983) = a_0 + a_1(1983) + a_2(1983)^2 + a_3(1983)^3 = 14{,}2411$$
$$v(1985) = a_0 + a_1(1985) + a_2(1985)^2 + a_3(1985)^3 = 14{,}0342$$
$$v(1987) = a_0 + a_1(1987) + a_2(1987)^2 + a_3(1987)^3 = 14{,}2696$$

$$\begin{bmatrix} 1 & 1981 & 3924361 & 7774159141 \\ 1 & 1983 & 3932289 & 7797729087 \\ 1 & 1985 & 3940225 & 7821346625 \\ 1 & 1987 & 3948169 & 7845011803 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 14{,}1999 \\ 14{,}2411 \\ 14{,}0342 \\ 14{,}2696 \end{bmatrix}$$

Gauss-Jordan

$$\begin{bmatrix} 1 & 1981 & 3924361 & 7774159141 & 14{,}1999 \\ 1 & 1983 & 3932289 & 7797729087 & 14{,}2411 \\ 1 & 1985 & 3940225 & 7821346625 & 14{,}0342 \\ 1 & 1987 & 3948169 & 7845011803 & 14{,}2696 \end{bmatrix} R_2 - R_1 \to R_2, R_3 - R_1 \to$$

$R_3, R_4 - R_1 \to R_4$

$$\begin{bmatrix} 1 & 1981 & 3924361 & 7774159141 & 14{,}1999 \\ 0 & 2 & 7928 & 23569946 & 0{,}0412 \\ 0 & 4 & 15864 & 47187484 & -0{,}1657 \\ 0 & 6 & 23808 & 70852662 & 0{,}0697 \end{bmatrix}$$

...

$$\begin{bmatrix} 1 & 0 & 0 & 0 & -112279076{,}3384875 \\ 0 & 1 & 0 & 0 & \frac{5094035009}{30000} \\ 0 & 0 & 1 & 0 & -85{,}5974625 \\ 0 & 0 & 0 & 1 & \frac{863}{60000} \end{bmatrix}$$

$a_0 = -112279076{,}3384875$

$a_1 = \frac{5094035009}{30000}$

$a_2 = -85{,}5974625$

$a_3 = \frac{863}{60000}$

$v(1982) = -112279076{,}3384875 + \frac{5094035009}{30000}(1982) - 85{,}5974625(1982)^2 + \frac{863}{60000}(1982)^3 = 14{,}2946$

Untuk 1984 itu bisa juga pakai persamaan yang digunakan dalam 1982 karena 4 titik terdekatnya sama.

$v(1984) = -112279076{,}3384875 + \frac{5094035009}{30000}(1984) - 85{,}5974625(1984)^2 + \frac{863}{60000}(1984)^3 = 14{,}1255$

$v(1986) = a_0 + a_1(1986) + a_2(1986)^2 + a_3(1986)^3$

$a_0 = 122475145.40066876$

$a_1 = -184991.34781875$

$a_2 = 93.13938125$

$a_3 = -0.01563125$

$v(1986) = 122475145.40066876 - 184991.34781875(1986) - 93.13938125(1986)^2 - 0.01563125(1986)^3 = 14{,}1435$

Dengan cara yang sama untuk tahun genap lainnya, menggunakan 4 titik terdekatnya, maka akan didapat sebagai berikut.

...

Even Years	Temperature
1982	14,2964
1984	14,1255
1986	14,1435
1988	14,2412
1990	14,2542
1992	14,2414
1994	14,2601
1996	14,4036
1998	14,39977

Estimasi suhu pada tahun-tahun genap dapat dilakukan dengan menggunakan metode interpolasi. Dalam kasus ini, kita memiliki data suhu untuk tahun-tahun ganjil antara 1981 dan 1999. Metode interpolasi yang umum digunakan adalah interpolasi linear, kuadratik, dan kubik. Namun, karena kita hanya memiliki sedikit titik data yang tersedia, interpolasi linear mungkin menjadi metode yang paling sesuai.

Interpolasi linear menggunakan garis lurus untuk menghubungkan titik data yang ada. Dalam hal ini, kita dapat menganggap adanya hubungan linier antara tahun (x) dan suhu (y). Dengan menghubungkan titik data yang kita miliki dengan garis lurus, kita dapat memperkirakan suhu pada tahun-tahun genap.

Interpolasi kuadratik dan kubik akan memperkenalkan tingkat kompleksitas yang lebih tinggi dalam model persamaan. Interpolasi kuadratik menggunakan polinomial orde dua untuk menghubungkan titik data, sementara interpolasi kubik menggunakan polinomial orde tiga. Namun, karena kita hanya memiliki sedikit titik data yang tersedia, menggunakan polinomial yang lebih tinggi dapat menyebabkan overfitting, yaitu model yang terlalu mempertimbangkan noise dalam data. Oleh karena itu, interpolasi linear yang lebih sederhana cenderung memberikan hasil yang lebih konservatif dan lebih sesuai dalam kasus ini.

B) Least-Square Regression ($y = ax + b$)

	x	y	x^2	y^2	x*y
	1981	14,1999	3924361	201,63716	28130,0019
	1983	14,2411	3932289	202,808929	28240,1013
	1985	14,0342	3940225	196,95877	27857,887
	1987	14,2696	3948169	203,621484	28353,6952
	1989	14,197	3956121	201,554809	28237,833
	1991	14,3055	3964081	204,64733	28482,2505
	1993	14,1853	3972049	201,222736	28271,3029
	1995	14,3577	3980025	206,143549	28643,6115
	1997	14,4187	3988009	207,89891	28794,1439
	1999	14,3438	3996001	205,744598	28673,2562
$\sum =$	19900	142,5528	39601330	2032,23828	283684,083

$$a = \frac{\sum xy - \frac{\sum x \sum y}{n}}{\sum x^2 - \frac{(\sum x)^2}{n}} \qquad b = \frac{\sum y - a(\sum x)}{n}$$

$$a = \frac{283684{,}083 - \frac{(19900)(142{,}5528)}{10}}{39601330 - \frac{(19900)^2}{10}} = \frac{4{,}011}{330} = 0{,}01215455$$

$$b = \frac{142{,}5528 - 0{,}01215455(19900)}{10} = -9{,}9322745$$

$y = ax + b$
$y = 0{,}01215455x - 9{,}9322745$
$x = 1982, y = 0{,}01215455(1982) - 9{,}9322745 = 14{,}1580$
$x = 1984, y = 0{,}01215455(1984) - 9{,}9322745 = 14{,}1823$
$x = 1986, y = 0{,}01215455(1986) - 9{,}9322745 = 14{,}2066$
$x = 1988, y = 0{,}01215455(1988) - 9{,}9322745 = 14{,}2309$
$x = 1990, y = 0{,}01215455(1990) - 9{,}9322745 = 14{,}2552$
$x = 1992, y = 0{,}01215455(1992) - 9{,}9322745 = 14{,}2795$
$x = 1994, y = 0{,}01215455(1994) - 9{,}9322745 = 14{,}3038$
$x = 1996, y = 0{,}01215455(1996) - 9{,}9322745 = 14{,}3282$
$x = 1998, y = 0{,}01215455(1998) - 9{,}9322745 = 14{,}3525$

C) Analisis perbedaan antara hasil regresi dan interpolasi yang telah dilakukan berdasarkan dasar teori yang telah dipelajari.

1. Regresi Kuadrat Terkecil (Least Square Regression):

   Regresi kuadrat terkecil adalah metode statistik yang digunakan untuk memodelkan hubungan antara variabel independen dan variabel dependen dengan menggunakan persamaan garis lurus (linear) sebagai model. Dalam regresi kuadrat terkecil, kita mencari garis terbaik yang mewakili data dengan meminimalkan jumlah kuadrat selisih antara nilai yang diobservasi dan nilai yang diprediksi oleh model. Dalam konteks ini, regresi kuadrat terkecil dengan persamaan garis lurus dapat digunakan untuk memprediksi suhu pada tahun-tahun genap.

2. Interpolasi Kuadratik:

   Interpolasi kuadratik adalah metode untuk memperkirakan nilai di antara titik-titik data yang diberikan dengan menggunakan polinom kuadratik sebagai model. Dalam konteks ini, jika kita ingin memperkirakan suhu pada tahun-tahun

genap berdasarkan data yang hanya tersedia pada tahun-tahun ganjil, kita dapat menggunakan interpolasi kuadratik untuk mengekstrapolasi nilai suhu tersebut.

Perbedaan antara regresi kuadrat terkecil dan interpolasi kuadratik terletak pada model matematis yang digunakan. Regresi kuadrat terkecil menggunakan persamaan garis lurus (linear) sebagai model, sedangkan interpolasi kuadratik menggunakan polinom kuadratik. Dalam regresi kuadrat terkecil, kita mengasumsikan bahwa hubungan antara variabel independen dan dependen adalah linier dalam rentang data yang diberikan. Interpolasi kuadratik, di sisi lain, mencoba untuk memperkirakan bentuk kurva yang memungkinkan data pada titik-titik yang diketahui.

Dalam kasus ini, karena kita hanya memiliki data pada tahun-tahun ganjil, menggunakan regresi kuadrat terkecil dengan persamaan garis lurus (linear) mungkin lebih tepat. Ini berarti kita mengasumsikan hubungan antara tahun dan suhu adalah linier dalam rentang data yang diberikan. Namun, penting untuk dicatat bahwa ini hanya merupakan asumsi model yang digunakan berdasarkan data yang tersedia, dan bukan penegasan bahwa hubungan sebenarnya antara tahun dan suhu selalu linier.

Dalam analisis perbedaan antara hasil regresi dan interpolasi, kita dapat membandingkan estimasi suhu pada tahun-tahun genap yang diberikan oleh kedua metode. Dalam hal ini, hasil regresi kuadrat terkecil dengan persamaan garis lurus akan memberikan prediksi suhu berdasarkan hubungan linier antara tahun dan suhu yang diduga ada. Di sisi lain, interpolasi kuadratik akan memberikan estimasi suhu berdasarkan polinom kuadratik yang melewati titik-titik data yang diketahui. Perbedaan antara kedua metode tersebut dapat mengungkapkan kemungkinan perbedaan dalam bentuk hubungan antara tahun dan suhu pada tahun-tahun genap yang tidak tercakup oleh data yang ada.

Dalam kesimpulan, regresi kuadrat terkecil menggunakan persamaan garis lurus (linear) sebagai model, sedangkan interpolasi kuadratik menggunakan polinom kuadratik. Dalam konteks data ini, regresi kuadrat terkecil dengan persamaan garis lurus mungkin memberikan estimasi yang lebih akurat untuk tahun-tahun genap, mengasumsikan hubungan antara tahun dan suhu adalah linier dalam rentang data

yang diberikan. Namun, penting untuk diingat bahwa ini hanyalah model matematis yang digunakan berdasarkan asumsi yang mungkin tidak selalu berlaku dalam konteks yang lebih luas.

D)

Source Code:

```
# !pip install jupyterthemes
# !jt -t monokai
# !conda update jupyter

import numpy as np
import matplotlib.pyplot as plt

# Data
years = np.array([1981, 1983, 1985, 1987, 1989, 1991, 1993, 1995, 1997,
1999])
temperatures = np.array([14.1999, 14.2411, 14.0342, 14.2696, 14.197,
14.3055, 14.1853, 14.3577, 14.4187, 14.3438])

# Interpolation
even_years = np.arange(1982, 2000, 2)

# Linear Interpolation
linear_interp = np.interp(even_years, years, temperatures)

# Quadratic Interpolation
quadratic_coeffs = np.polyfit(years, temperatures, 2)
quadratic_interp = np.polyval(quadratic_coeffs, even_years)

# Cubic Interpolation
cubic_coeffs = np.polyfit(years, temperatures, 3)
cubic_interp = np.polyval(cubic_coeffs, even_years)

# Least-Square Regression
regression_coeffs = np.polyfit(years, temperatures, 1)
regression_interp = np.polyval(regression_coeffs, even_years)

# Hasil
print("Linear Interpolation:")
for year, temp in zip(even_years, linear_interp):
```

```
    print(f"Year: {year}, Estimated Temperature: {temp:.4f} oC")

print("\nQuadratic Interpolation:")
for year, temp in zip(even_years, quadratic_interp):
    print(f"Year: {year}, Estimated Temperature: {temp:.4f} oC")

print("\nCubic Interpolation:")
for year, temp in zip(even_years, cubic_interp):
    print(f"Year: {year}, Estimated Temperature: {temp:.4f} oC")

print("\nLeast-Square Regression:")
for year, temp in zip(even_years, regression_interp):
    print(f"Year: {year}, Estimated Temperature: {temp:.4f} oC")

# Plot
plt.figure(figsize=(10, 6))
plt.scatter(years, temperatures, label='Data Points')
plt.plot(even_years, linear_interp, label='Linear Interpolation')
plt.plot(even_years, quadratic_interp, label='Quadratic Interpolation')
plt.plot(even_years, cubic_interp, label='Cubic Interpolation')
plt.plot(even_years, regression_interp, label='Least-Square Regression')
plt.xlabel('Year')
plt.ylabel('Temperature (oC)')
plt.title('Temperature vs. Year')
plt.legend()
plt.grid(True)
plt.show()
```

Output:

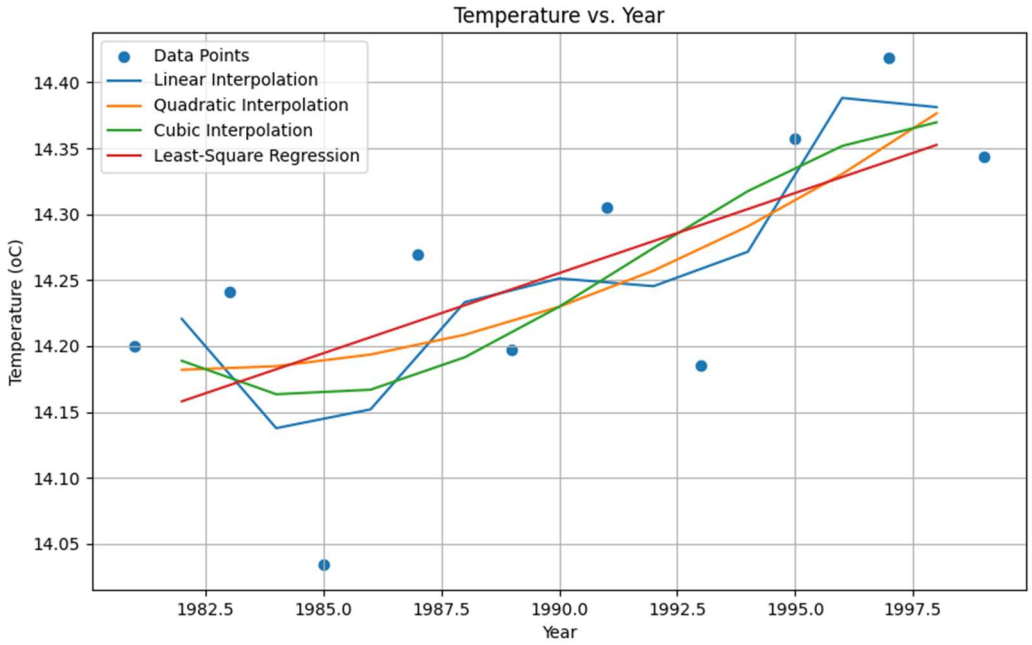
```
Linear Interpolation:
Year: 1982, Estimated Temperature: 14.2205 oC
Year: 1984, Estimated Temperature: 14.1377 oC
Year: 1986, Estimated Temperature: 14.1519 oC
Year: 1988, Estimated Temperature: 14.2333 oC
Year: 1990, Estimated Temperature: 14.2512 oC
Year: 1992, Estimated Temperature: 14.2454 oC
Year: 1994, Estimated Temperature: 14.2715 oC
Year: 1996, Estimated Temperature: 14.3882 oC
Year: 1998, Estimated Temperature: 14.3812 oC

Quadratic Interpolation:
Year: 1982, Estimated Temperature: 14.1820 oC
Year: 1984, Estimated Temperature: 14.1847 oC
Year: 1986, Estimated Temperature: 14.1935 oC
```

```
Year: 1988, Estimated Temperature: 14.2086 oC
Year: 1990, Estimated Temperature: 14.2298 oC
Year: 1992, Estimated Temperature: 14.2572 oC
Year: 1994, Estimated Temperature: 14.2908 oC
Year: 1996, Estimated Temperature: 14.3305 oC
Year: 1998, Estimated Temperature: 14.3765 oC

Cubic Interpolation:
Year: 1982, Estimated Temperature: 14.1887 oC
Year: 1984, Estimated Temperature: 14.1634 oC
Year: 1986, Estimated Temperature: 14.1668 oC
Year: 1988, Estimated Temperature: 14.1915 oC
Year: 1990, Estimated Temperature: 14.2298 oC
Year: 1992, Estimated Temperature: 14.2743 oC
Year: 1994, Estimated Temperature: 14.3175 oC
Year: 1996, Estimated Temperature: 14.3518 oC
Year: 1998, Estimated Temperature: 14.3697 oC

Least-Square Regression:
Year: 1982, Estimated Temperature: 14.1580 oC
Year: 1984, Estimated Temperature: 14.1823 oC
Year: 1986, Estimated Temperature: 14.2067 oC
Year: 1988, Estimated Temperature: 14.2310 oC
Year: 1990, Estimated Temperature: 14.2553 oC
Year: 1992, Estimated Temperature: 14.2796 oC
Year: 1994, Estimated Temperature: 14.3039 oC
Year: 1996, Estimated Temperature: 14.3282 oC
Year: 1998, Estimated Temperature: 14.3525 oC
```



Gambar 1 - Output(Plot)

2. A) Around 0, $a = 0$

$$\boldsymbol{T = f(a) + f'(a)(x-a)^1 + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^n(a)}{n!}(x-a)^n}$$

$f(x) = \sin x \qquad f(0) = 0$
$f'(x) = \cos x \qquad f'(0) = 1$
$f''(x) = -\sin x \qquad f''(0) = 0$
$f'''(x) = -\cos x \qquad f'''(0) = -1$
$f^{(4)}(x) = \cos x \qquad f^{(4)}(0) = 0$

$$\sin x = 0 + 1(x-0)^1 + \frac{0}{2!}(x-0)^2 + \frac{(-1)}{3!}(x-0)^3 + \frac{0}{4!}(x-0)^4$$

$$\sin x = x - \frac{x^3}{6}$$

$f(x) = \cos x \qquad f(0) = 1$
$f'(x) = -\sin x \qquad f'(0) = 0$
$f''(x) = -\cos x \qquad f''(0) = -1$
$f'''(x) = \sin x \qquad f'''(0) = 0$
$f^{(4)}(x) = \cos x \qquad f^{(4)}(0) = 1$

$$\cos x = 1 + 0(x-0)^1 + \frac{(-1)}{2!}(x-0)^2 + \frac{0}{3!}(x-0)^3 + \frac{1}{4!}(x-0)^4$$

$$\cos x = 1 - \frac{x^2}{2} + \frac{x^4}{24}$$

$f(x) = sin\,\mathrm{x}\cos\mathrm{x} \qquad f(0) = 0$
$f'(x) = \cos 2x \qquad f'(0) = 1$
$f''(x) = -2\sin 2x \qquad f''(0) = 0$
$f'''(x) = -4\cos 2x \qquad f'''(0) = -4$
$f^{(4)}(x) = 8\sin 2x \qquad f^{(4)}(0) = 0$

$$\sin x \cos x = 0 + 1(x-0)^1 + \frac{0}{2!}(x-0)^2 + \frac{(-4)}{3!}(x-0)^3 + \frac{0}{4!}(x-0)^4$$

$$\sin x \cos x = x - \frac{2x^3}{3}$$

Source Code:

```
import sympy as sp
import math

x = sp.symbols('x')
f = sp.sin(x)

# Taylor expansion sin(x)
sin_taylor = sp.series(f, x, 0, 5).removeO()
print("Ekpansi Taylor untuk sin(x):", sin_taylor)
```

```
# Taylor expansion cos(x)
f = sp.cos(x)
cos_taylor = sp.series(f, x, 0, 5).removeO()
print("Ekpansi Taylor untuk cos(x):", cos_taylor)

# Taylor expansion sin(x)cos(x)
f = sp.sin(x) * sp.cos(x)
sincos_taylor = sp.series(f, x, 0, 5).removeO()
print("Ekpansi Taylor untuk sin(x)cos(x):", sincos_taylor)
```

Output:

```
Ekpansi Taylor untuk sin(x): -x**3/6 + x
Ekpansi Taylor untuk cos(x): x**4/24 - x**2/2 + 1
Ekpansi Taylor untuk sin(x)cos(x): -2*x**3/3 + x
```

B)

Exact Value:

$$\sin\frac{\pi}{2}\cos\frac{\pi}{2} = \sin\frac{\pi}{2}\cos\frac{\pi}{2} = (1)(0) = 0$$

Terpisah:

$$\sin\frac{\pi}{2} = \frac{\pi}{2} - \frac{\left(\frac{\pi}{2}\right)^3}{6} = 0{,}924832$$

$$\cos\frac{\pi}{2} = 1 - \frac{\left(\frac{\pi}{2}\right)^2}{2} + \frac{\left(\frac{\pi}{2}\right)^4}{24} = 0{,}019969$$

$$\sin\frac{\pi}{2}\cos\frac{\pi}{2} = (0{,}924832)(0{,}019969) = 0{,}01846797$$

$$Error = \left|\sin\frac{\pi}{2}\cos\frac{\pi}{2}_{Terpisah} - \sin\frac{\pi}{2}\cos\frac{\pi}{2}_{ExactValue}\right| = |0{,}01846797 - 0| = 0{,}01846797$$

Gabung:

$$\sin\frac{\pi}{2}\cos\frac{\pi}{2} = \frac{\pi}{2} - \frac{2\left(\frac{\pi}{2}\right)^3}{3} = -1{,}01306$$

$$Error = \left|\sin\frac{\pi}{2}\cos\frac{\pi}{2}_{Gabung} - \sin\frac{\pi}{2}\cos\frac{\pi}{2}_{ExactValue}\right| = |-1{,}01306 - 0| = 1{,}01306$$

Source Code:

```
# Exact value sin(π/2) * cos(π/2)
exact_value = sp.sin(sp.pi/2) * sp.cos(sp.pi/2)

x = sp.symbols('x')

# Hasil sin(π/2) * cos(π/2) pakai ekspansi taylor terpisah sin(x)
dan cos(x)
taylor_terpisah = sp.lambdify(x, sin_taylor * cos_taylor)
result_terpisah = taylor_terpisah(math.pi/2)

# Hasil sin(π/2) * cos(π/2) pakai ekspansi taylor gabung
sin(x)cos(x)
taylor_sincos = sp.lambdify(x, sincos_taylor)
result_sincos = taylor_sincos(math.pi/2)

print("Result pakai sin(x) and cos(x) terpisah:",
result_terpisah)
print("Result pakai ekspansi Taylor sin(x)cos(x):",
result_sincos)
print("Exact value:",exact_value,"\n")

# Error
# error_separate = abs((exact_value -
result_separate)/exact_value)*100
# error_product = abs((exact_value -
result_product)/exact_value)*100

# Pakai yang diatas itu karena nanti hasilnya bakalan tak hingga
# karena penyebut bakalan 0 karena exact valuenya kan 0
# jadi tinggal pakai exact_value - hasil aja

error_terpisah = abs(exact_value - result_terpisah)
error_sincos = abs(exact_value - result_sincos)

print("Error ekspansi Taylor yang terpisah sin(x) dan cos(x):",
error_terpisah)
print("Error ekspansi Taylor yang gabung sin(x)cos(x):",
error_sincos)
```

Output:

```
Result pakai sin(x) and cos(x) terpisah: 0.018467935726263183
Result pakai ekspansi Taylor sin(x)cos(x): -1.0130600632300881
Exact value: 0

Error ekspansi Taylor yang terpisah sin(x) dan cos(x):
0.0184679357262632
Error ekspansi Taylor yang gabung sin(x)cos(x): 1.01306006323009
```

C)

Approximation $\cos\frac{\pi}{4}$**:**

$$\cos\frac{\pi}{4} = 1 - \frac{\left(\frac{\pi}{4}\right)^2}{2} + \frac{\left(\frac{\pi}{4}\right)^4}{24} = 0{,}707429$$

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

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

dimana M adalah nilai maximum dari $|f^{(n+1)}|$ ***dari interval***

$Interval: \left[0, \frac{\pi}{4}\right]$

$|f^{(4+1)}(0)| = |-\sin 0| = 0$

$|f^{(4+1)}\left(\frac{\pi}{4}\right)| = \left|-\sin\frac{\pi}{4}\right| = |-0{,}707106781186547| = 0{,}707106781186547$

$Maka: M = 0{,}707106781186547$

$$|E_n\left(\frac{\pi}{4}\right)| = \frac{0{,}707106781186547\left|\frac{\pi}{4} - 0\right|^{(4+1)}}{(4+1)!} = \frac{0{,}707106781186547\left|\frac{\pi}{4}\right|^{(5)}}{(5)!}$$

$$= 0{,}00176097$$

Source Code:

```
x = sp.symbols('x')
f = sp.cos(x)

# Taylor expansion cos(x)
print("Ekspansi Taylor cos(x):", cos_taylor)

# Approximating cos(π/4) pakai ekspansi Taylornya
approximation = cos_taylor.subs(x, math.pi/4)
print("Approximation cos(π/4):", approximation)
```

```
# Turunan ke-(n+1) dari cos(x)
f_prime_5 = sp.diff(f, x, 5)

# Maximum value dari turunan ke-(n+1) dengan interval [0, π/4]
max_derivative = max(abs(f_prime_5.subs(x, c)) for c in [0,
math.pi/4])
i=1;

print("\nNilai turunan ke-(n+1) pada interval[0, π/4]:")
for c in [0,math.pi/4]:
    print(str(i)+".",abs(f_prime_5.subs(x,c)))
    i+=1
print()
# Truncation error bound
truncation_error_bound = max_derivative * (abs(math.pi/4 - 0))**5
/ math.factorial(5)
print("Truncation error bound:", truncation_error_bound)
```

Output:

```
Ekspansi Taylor cos(x): x**4/24 - x**2/2 + 1
Approximation cos(π/4): 0.707429206709773

Nilai turunan ke-(n+1) pada interval[0, π/4]:
1. 0
2. 0.707106781186547

Truncation error bound: 0.00176097488841343
```

3. A) $f(x) = x^3 - 0.3x^2 - 8.56x + 8.448$

   $Interval\ [0,2\pi]$

   $n = 20$

   $\Delta x = \frac{b-a}{n}, \Delta x = (2\pi - 0) / 20 = \pi / 10$

   - ***Riemann Integration***

   Left Riemann

   $\boldsymbol{x_{left}} = \left[\mathbf{0}, \frac{\boldsymbol{\pi}}{\mathbf{10}}, \frac{\mathbf{2}\boldsymbol{\pi}}{\mathbf{10}}, \frac{\mathbf{3}\boldsymbol{\pi}}{\mathbf{10}}, \dots, \frac{\mathbf{18}\boldsymbol{\pi}}{\mathbf{10}}, \frac{\mathbf{19}\boldsymbol{\pi}}{\mathbf{10}}\right]$

   $\int_0^{2\pi} f(x)\,dx = \Delta x * (f(x_0) + f(x_1) + f(x_2) + \dots + f(x_{19}))$

$\int_0^{2\pi} f(x)\,dx = \Delta x * (f(0) + f(\frac{\pi}{10}) + f(\frac{2\pi}{10}) + \ldots + f(\frac{19\pi}{10}))$

$\int_0^{2\pi} f(x)\,dx = \frac{\pi}{10} * (8{,}448 + 5{,}76019 + 3{,}19921 + \ldots + 159{,}33641)$

$\int_0^{2\pi} f(x)\,dx = 221{,}2322711083824$

Mid Riemann

$\boldsymbol{x_{mid}} = [\frac{\boldsymbol{\pi}}{\boldsymbol{20}}, \frac{\boldsymbol{3\pi}}{\boldsymbol{20}}, \frac{\boldsymbol{5\pi}}{\boldsymbol{20}}, \ldots, \frac{\boldsymbol{37\pi}}{\boldsymbol{20}}, \frac{\boldsymbol{39\pi}}{\boldsymbol{20}}]$

$\int_0^{2\pi} f(x)\,dx = \Delta x * (f(\frac{\pi}{20}) + f(\frac{3\pi}{20}) + f(\frac{5\pi}{20}) + \ldots + f(\frac{39\pi}{20}))$

$\int_0^{2\pi} f(x)\,dx = \frac{\pi}{10} * (7{,}09987 + 4{,}45222 + 2{,}02441 + \ldots + 174{,}65745)$

$\int_0^{2\pi} f(x)\,dx = 248{,}4725226033433$

Right Riemann

$\boldsymbol{x_{right}} = [\frac{\boldsymbol{\pi}}{\boldsymbol{10}}, \frac{\boldsymbol{2\pi}}{\boldsymbol{10}}, \frac{\boldsymbol{3\pi}}{\boldsymbol{10}}, \ldots, \frac{\boldsymbol{19\pi}}{\boldsymbol{10}}, \boldsymbol{2\pi}]$

$\int_0^{2\pi} f(x)\,dx = \Delta x * (f(\frac{\pi}{10}) + f(\frac{2\pi}{10}) + f(\frac{3\pi}{10}) + \ldots + f(2\pi))$

$\int_0^{2\pi} f(x)\,dx = \frac{\pi}{10} * (5{,}76019 + 3{,}19921 + 0{,}95108 + \ldots + 190{,}87062)$

$\int_0^{2\pi} f(x)\,dx = 278{,}5420279992834$

- ***Trapezoida Rule's***

$\int_{\boldsymbol{a}}^{\boldsymbol{b}} \boldsymbol{f(x)\,dx} = (\frac{\boldsymbol{\Delta x}}{\boldsymbol{2}}) * (\boldsymbol{f(x_0)} + \boldsymbol{2f(x_1)} + \boldsymbol{2f(x_2)} + \ldots + \boldsymbol{2f(x_{n-1})} + \boldsymbol{f(x_n)})$

$\int_0^{2\pi} f(x)\,dx = \left(\frac{\pi}{20}\right) * \left(f(0) + 2f\left(\frac{\pi}{10}\right) + 2f\left(\frac{2\pi}{10}\right) + \ldots + 2f\left(\frac{19\pi}{10}\right) + f(2\pi)\right)$

$\int_0^{2\pi} f(x)\,dx = \left(\frac{\pi}{20}\right) * (8{,}448 + 2(5{,}76019) + 2(3{,}19921) + \ldots + 2(159{,}33641) + 190{,}87062)$

$\int_0^{2\pi} f(x)\,dx = 249{,}88714955383287$

- ***Simpson 1/3 Rule's***

$\int_{\boldsymbol{a}}^{\boldsymbol{b}} \boldsymbol{f(x)dx} = (\frac{\boldsymbol{\Delta x}}{\boldsymbol{3}}) * (\boldsymbol{f(x_0)} + \boldsymbol{4f(x_1)} + \boldsymbol{2f(x_2)} + \boldsymbol{4f(x_3)} \ldots + \boldsymbol{2f(x_{n-2})} + \boldsymbol{4f(x_{n-1})} + \boldsymbol{f(x_n)})$

$\int_{\boldsymbol{a}}^{\boldsymbol{b}} \boldsymbol{f(x)dx} = (\frac{\boldsymbol{\Delta x}}{\boldsymbol{3}}) * (\boldsymbol{f(x_0)} + \boldsymbol{4(f(x_1) + f(x_3) + \cdots)} + \boldsymbol{2(f(x_2) + f(x_4)} + \cdots) + \boldsymbol{f(x_n)})$

$\int_0^{2\pi} f(x)\,dx = (\frac{\Delta x}{3}) * (f(0) + 4f\left(\frac{\pi}{10}\right) + 2f\left(\frac{2\pi}{10}\right) + 4f\left(\frac{3\pi}{10}\right) \ldots + 2f\left(\frac{18\pi}{10}\right) + 4f\left(\frac{19\pi}{10}\right) + f(2\pi))$

$\int_0^{2\pi} f(x)\,dx = (\frac{\pi}{30}) * (8{,}448 + 4(5{,}76019) + 2(3{,}19921) + 4(0{,}95108) \ldots + 2(131{,}27769) + 4(159{,}33641) + 190{,}87062)$

$\int_0^{2\pi} f(x)\,dx = 248{,}9440649201731$

- ***Simpson 3/8 Rule's***

$\int_a^b \boldsymbol{f(x)dx} = (\frac{\boldsymbol{3\Delta x}}{\boldsymbol{8}}) * (\boldsymbol{f(x_0) + 3(f(x_1) + f(x_2) + f(x_4) \dots) + 2(f(x_3) + f(x_6) + \cdots) + f(x_n)})$

$\int_0^{2\pi} f(x)\,dx = (\frac{3\Delta}{8}) * (f(0) + 3\left(f\left(\frac{\pi}{10}\right) + f\left(\frac{2\pi}{10}\right) + f\left(\frac{4\pi}{10}\right) \dots\right) + 2(f\left(\frac{3\pi}{10}\right) + f\left(\frac{6\pi}{10}\right) + \cdots) + f(2\pi))$

$\int_0^{2\pi} f(x)\,dx = (\frac{3\pi}{80}) * (8{,}448 + 3(5{,}76019 + 3{,}19921 + (-0{,}798153) \dots) + 2(0{,}95108 + (-2{,}05578) + \cdots) + 190{,}87062)$

$\int_0^{2\pi} f(x)\,dx = 242{,}7324330000515$

Source Code (1):

```
#RIEMANN INTEGRATION

import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return x**3 - 0.3*x**2 - 8.56*x + 8.448

def riemann_left(f, a, b, n):
    dx = (b - a) / n
    x = np.linspace(a, b, n+1)[:-1]
    y = f(x)
    return dx * np.sum(y)

def riemann_midpoint(f, a, b, n):
    dx = (b - a) / n
    x = np.linspace(a + dx/2, b - dx/2, n)
    y = f(x)
    return dx * np.sum(y)

def riemann_right(f, a, b, n):
    dx = (b - a) / n
    x = np.linspace(a, b, n+1)[1:]
    y = f(x)
    return dx * np.sum(y)

a = 0
```

```
b = 2 * np.pi
n = 20

x = np.linspace(a, b, 100)
y = f(x)

left_riemann_approximation = riemann_left(f, a, b, n)
midpoint_riemann_approximation = riemann_midpoint(f, a, b, n)
right_riemann_approximation = riemann_right(f, a, b, n)
print("Riemann Integration:")
print("Approximation using Left Riemann Sum:",
left_riemann_approximation)
print("Approximation using Midpoint Riemann Sum:",
midpoint_riemann_approximation)
print("Approximation using Right Riemann Sum:",
right_riemann_approximation)

fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(8, 12))

# Left
ax1.plot(x, y, label='f(x)')
ax1.fill_between(x, y, alpha=0.3)
ax1.bar(x[:-1], f(x[:-1]), width=(b-a)/n, align='edge',
alpha=0.3, label='Left Riemann Sum')
ax1.legend()
ax1.set_xlabel('x')
ax1.set_ylabel('f(x)')
ax1.set_title('Approximation using Left Riemann Sum')
ax1.grid(True)

# Mid
ax2.plot(x, y, label='f(x)')
ax2.fill_between(x, y, alpha=0.3)
ax2.bar(x[:-1]+(b-a)/(2*n), f(x[:-1]+(b-a)/(2*n)), width=(b-a)/n,
align='center', alpha=0.3, label='Midpoint Riemann Sum')
ax2.legend()
ax2.set_xlabel('x')
ax2.set_ylabel('f(x)')
ax2.set_title('Approximation using Midpoint Riemann Sum')
```

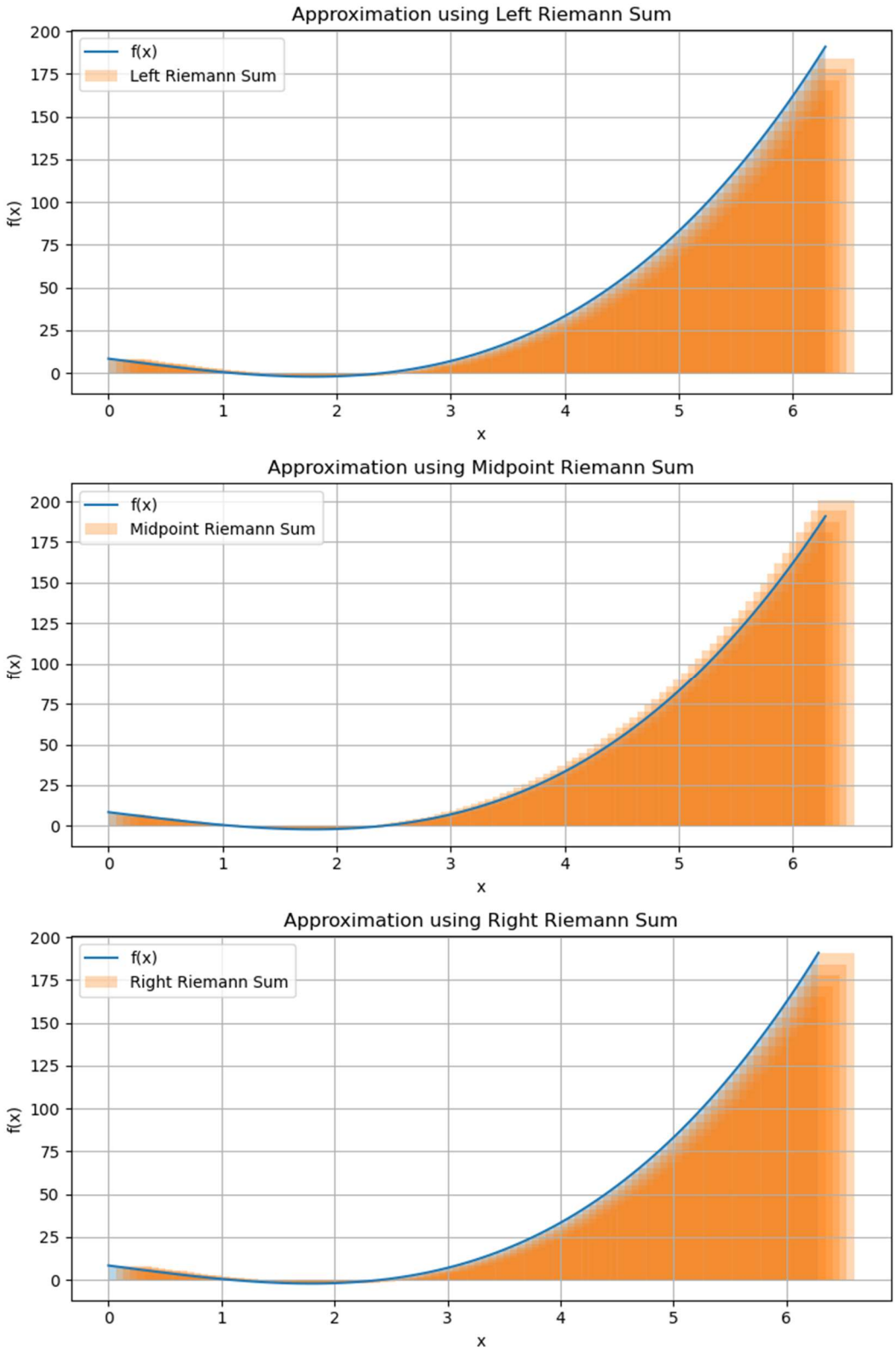
```
ax2.grid(True)

# Right
ax3.plot(x, y, label='f(x)')
ax3.fill_between(x, y, alpha=0.3)
ax3.bar(x[1:], f(x[1:]), width=(b-a)/n, align='edge', alpha=0.3,
label='Right Riemann Sum')
ax3.legend()
ax3.set_xlabel('x')
ax3.set_ylabel('f(x)')
ax3.set_title('Approximation using Right Riemann Sum')
ax3.grid(True)

plt.tight_layout()
plt.show()
```

Output:

```
Riemann Integration:
Approximation using Left Riemann Sum: 221.2322711083824
Approximation using Midpoint Riemann Sum: 248.4725226033433
Approximation using Right Riemann Sum: 278.5420279992834
```



Gambar 2 - Output(Plot)

Source Code (2):

```
import math

exact_value = (1/4)*((2*math.pi)**4) - (0.3/3)*((2*math.pi)**3) -
(8.56/2)*((2*math.pi)**2) + 8.448*(2*math.pi) - ((1/4)*(0)**4 -
(0.3/3)*(0)**3 - (8.56/2)*(0)**2 + 8.448*(0))

# Trapezoidal Rule
def trapezoidal_rule(f, a, b, n):
    h = (b - a) / n
    sum_f = f(a) + f(b)
    for i in range(1, n):
        sum_f += 2 * f(a + i * h)
    return (h / 2) * sum_f

# Simpson's 1/3 Rule
def simpsons_1_3(f, a, b, n):
    h = (b - a) / n
    sum_f = f(a) + f(b)
    for i in range(1, n):
        if i % 2 == 0:
            sum_f += 2 * f(a + i * h)
        else:
            sum_f += 4 * f(a + i * h)
    return (h / 3) * sum_f

# Simpson's 3/8 Rule
def simpsons_3_8(f, a, b, n):
    h = (b - a) / n
    sum_f = f(a) + f(b)
    for i in range(1, n):
        if i % 3 == 0:
            sum_f += 2 * f(a + i * h)
        else:
            sum_f += 3 * f(a + i * h)
    return (3 * h / 8) * sum_f
```

```
# Calculation
a = 0
b = 2 * math.pi
n = 20

trapezoidal_approximation = trapezoidal_rule(f, a, b, n)
simpsons_1_3_approximation = simpsons_1_3(f, a, b, n)
simpsons_3_8_approximation = simpsons_3_8(f, a, b, n)

# Print the results
print("Approximations:")
print("Trapezoidal Rule's:", trapezoidal_approximation)
print("Simpson 1/3 Rule's:", simpsons_1_3_approximation)
print("Simpson 3/8 Rule's:", simpsons_3_8_approximation)
print("Exact value:",exact_value)
```

Output:

```
Approximations:
Trapezoidal Rule's: 249.88714955383287
Simpson 1/3 Rule's: 248.9440649201731
Simpson 3/8 Rule's: 242.7324330000515
Exact value: 248.94406492017316
```

Berikut penjelasan tentang perbedaan masing-masing metode tersebut:

1. Riemann Integral:
   - Left Riemann Sum: Pada metode ini, luas di bawah kurva dihitung dengan menggunakan nilai fungsi pada titik ujung kiri dari setiap subinterval. Dalam hal ini, batas bawah subinterval digunakan sebagai perkiraan nilai fungsi.
   - Midpoint Riemann Sum: Pada metode ini, luas di bawah kurva dihitung dengan menggunakan nilai fungsi pada titik tengah setiap subinterval. Dalam hal ini, nilai fungsi pada titik tengah subinterval digunakan sebagai perkiraan nilai fungsi.
   - Right Riemann Sum: Pada metode ini, luas di bawah kurva dihitung dengan menggunakan nilai fungsi pada titik ujung kanan dari setiap subinterval. Dalam hal ini, batas atas subinterval digunakan sebagai perkiraan nilai fungsi.
2. Trapezoidal Rule:
   - Metode ini menggunakan garis lurus antara titik-titik grid yang berdekatan untuk membentuk trapesium. Luas di bawah kurva diperkirakan dengan menjumlahkan

luas semua trapesium di seluruh interval. Metode ini memberikan perkiraan yang lebih baik dibandingkan dengan metode Riemann karena memperhitungkan kemiringan kurva di antara titik grid.

3. Simpson's Rule:
   - 1/3 Simpson's Rule: Metode ini menggunakan polinomial orde dua antara tiga titik grid berurutan untuk membentuk kurva dalam setiap subinterval. Luas di bawah kurva dihitung menggunakan rumus yang menggabungkan polinomial orde dua tersebut.
   - Simpson's Rule: Metode ini menggunakan polinomial orde tiga antara empat titik grid berurutan untuk membentuk kurva dalam setiap subinterval. Luas di bawah kurva dihitung menggunakan rumus yang menggabungkan polinomial orde tiga tersebut.

Perbedaan antara metode-metode di atas terletak pada cara pendekatan luas di bawah kurva. Riemann Integral menggunakan sumbu vertikal dan membagi interval menjadi subinterval dengan lebar yang sama. Trapezoidal Rule menggunakan garis lurus yang menghubungkan titik grid berdekatan, sedangkan Simpson's Rule menggunakan polinomial orde dua atau tiga untuk membentuk kurva yang lebih akurat.

Secara umum, semakin tinggi orde polinomial yang digunakan, semakin akurat perkiraan integralnya. Simpson's Rule (terutama 3/8 Simpson's Rule) memberikan perkiraan yang lebih akurat daripada Riemann Integral dan Trapezoidal Rule, karena mempertimbangkan polinomial orde yang lebih tinggi dalam pendekatannya.

Dalam hal kenyamanan penggunaan, jika fungsi dapat diintegrasikan secara eksplisit, metode integrasi eksplisit akan lebih nyaman karena memberikan solusi yang tepat. Namun, dalam banyak kasus, fungsi kompleks sulit atau bahkan tidak mungkin diintegrasikan secara eksplisit, dan metode numerik seperti yang dijelaskan di atas menjadi pilihan yang lebih praktis untuk memperoleh perkiraan integral.

B) Dibandingkan dengan metode-metode yang disebutkan sebelumnya (Riemann Integral, Trapezoidal Rule, dan Simpson's Rule), integrasi eksplisit dapat menjadi lebih nyaman dilakukan tergantung pada sifat fungsi yang diintegralkan dan ketersediaan teknik-teknik integrasi eksplisit yang sesuai.

Keuntungan utama dari integrasi eksplisit adalah kemampuannya untuk memberikan solusi yang tepat secara analitis, yaitu dalam bentuk persamaan matematika yang dapat dievaluasi secara langsung. Jika fungsi yang diintegralkan memiliki bentuk matematika yang dapat dipecahkan dengan mudah, atau jika ada rumus-rumus khusus yang dikenal untuk integran tersebut, maka integrasi eksplisit bisa lebih nyaman dilakukan. Dalam beberapa kasus, integrasi eksplisit juga dapat memberikan wawasan tentang sifat-sifat matematis dan geometris dari fungsi tersebut.

Namun, ada situasi di mana integrasi eksplisit menjadi sulit atau bahkan tidak mungkin dilakukan. Beberapa alasan yang mungkin termasuk kompleksitas fungsi yang diintegralkan, ketidakteraturan fungsi, keberadaan integral tak-terbatas, atau bahkan ketiadaan teknik-teknik integrasi eksplisit yang tepat. Dalam kasus-kasus seperti ini, metode numerik seperti Riemann Integral, Trapezoidal Rule, atau Simpson's Rule menjadi pilihan yang lebih realistis dan efektif untuk menghitung integral.
Keputusan tentang metode yang paling nyaman untuk digunakan dalam mengintegrasikan suatu fungsi tergantung pada sifat fungsi itu sendiri, ketersediaan teknik-teknik integrasi eksplisit yang tepat, dan kebutuhan akurasi solusi. Jika integrasi eksplisit memungkinkan dan memberikan solusi yang tepat dengan biaya yang wajar, itu bisa menjadi pilihan yang lebih nyaman. Namun, jika fungsi kompleks atau tidak dapat diintegrasikan secara eksplisit, metode numerik menjadi pilihan yang lebih praktis dan efektif.

C)

$\boldsymbol{x}$	-1.1	-0.3	0.8	1.9
$\boldsymbol{f(x)}$	15.180	10.962	1.920	-2.040

$$P_2(x) = a_0 + a_1x + a_2x + a_3x^3, x = -1.1, -0.3, 0.8, 1.9$$

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \sum x_i^5 \\ \sum x_i^3 & \sum x_i^4 & \sum x_i^5 & \sum x_i^6 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \\ \sum y_i x_i^2 \\ \sum y_i x_i^3 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 1{,}3 & 5{,}55 & 6{,}013 \\ 1{,}3 & 5{,}55 & 6{,}013 & 14{,}9139 \\ 5{,}55 & 6{,}013 & 14{,}9139 & 23{,}47573 \\ 6{,}013 & 14{,}9139 & 23{,}47573 & 49{,}080315 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 26{,}022 \\ -22{,}3266 \\ 13{,}21878 \\ -33{,}509874 \end{bmatrix}$$

Eliminasi:

$a_0 = 8{,}547$

$a_1 = -\frac{127769}{15200} = -8{,}40585526$

$a_2 = -\frac{78}{95} = -0{,}82105263$

$a_3 = \frac{185}{152} = 1{,}21710526$

$P_2(x) = 8{,}547 - 8{,}40585526x - 0{,}82105263x^2 + 1{,}21710526x^3, \rightarrow$

$f(x)\ dari\ interpolasi\ polinomial$

$P_2'(x) = a_1 + 2a_2x$

$P_2''(x) = 2a_2$

$f'(0) = P_2'(0) = -8{,}40585526 + 2(-0{,}82105263)(0)$

$f'(0) = -8{,}40585526$

$f''(0) = P_2''(0) = 2(-0{,}82105263)$

$f''(0) = -1{,}64210526$

Source Code:

```
import numpy as np
from scipy.interpolate import lagrange

x_data = np.array([-1.1, -0.3, 0.8, 1.9])
f_data = np.array([15.180, 10.962, 1.920, -2.040])

# Interpolasi polinomial
poly = lagrange(x_data, f_data)

# Menghitung f'(x) dan f''(x) di x = 0
x_interp = 0

# Menghitung turunan pertama (f'(x))
f_prime = np.polyval(np.polyder(poly), x_interp)

# Menghitung turunan kedua (f''(x))
f_double_prime = np.polyval(np.polyder(poly, 2), x_interp)

print("f'(0) =", f_prime)
print("f''(0) =", f_double_prime)
```

Output:

```
f'(0) = -8.405855263157896
f''(0) = -1.6421052631578998
```

D) $\boldsymbol{P_2(x) = 8,547 - 8,40585526x - 0,82105263x^2 + 1,21710526x^3}$

$P_2(-1{,}1) = 8{,}547 - 8{,}40585526(-1{,}1) - 0{,}82105263(-1{,}1)^2 + 1{,}21710526(-1{,}1)^3$

$P_2(-1{,}1) = 15{,}18$

$Selisih_1 = P_2(-1{,}1) - f(-1{,}1) = 15{,}18 - 15{,}18 = 0$

$P_2(-0{,}3) = 8{,}547 - 8{,}40585526(-0{,}3) - 0{,}82105263(-0{,}3)^2 + 1{,}21710526(-0{,}3)^3$

$P_2(-0{,}3) = 10{,}962$

$Selisih_2 = P_2(-0{,}3) - f(-0{,}3) = 10{,}962 - 10{,}962 = 0$

$P_2(0{,}8) = 8{,}547 - 8{,}40585526(0{,}8) - 0{,}82105263(0{,}8)^2 + 1{,}21710526(0{,}8)^3$

$P_2(0{,}8) = 1{,}92$

$Selisih_3 = P_2(0{,}8) - f(0{,}8) = 1{,}92 - 1{,}92 = 0$

$P_2(1{,}9) = 8{,}547 - 8{,}40585526(1{,}9) - 0{,}82105263(1{,}9)^2 + 1{,}21710526(1{,}9)^3$

$P_2(1{,}9) = -2{,}04$

$Selisih_4 = P_2(1{,}9) - f(1{,}9) = -2{,}04 - (-2{,}04) = 0$

$Akurasi = \frac{Selisih_1 + Selisih_2 + Selisih_3 + Selisih_4}{4} = \frac{0+0+0+0}{4} = 0$

Source Code:

```
# Menghitung nilai f(x) menggunakan polinomial interpolasi

f_interp = np.polyval(poly, x_data)



# Menghitung akurasi (selisih absolut) dibandingkan dengan nilai
f(x) awal

print("\nNilai f(x) menggunakan polinomial
interpolasi:",f_interp)

print("Nilai f(x) awal pada soal:",f_data)

accuracy = np.abs(f_interp - f_data)



# MASIH TERDAPAT SELISIH PADAHAL F(X) DARI POLINOMIAL INTERPOLASI
DAN F(X) AWAL NYA SUDAH SAMA,
```

```
# INI TERJADI KARENA SEHARUSNYA NILAI F(X) POLINOMIAL INTERPOLASI
DAN F(X) AWAL ITU

# TIDAK BENAR BENAR SAMA PERSIS KARENA ADANYA PEMBULATAN YANG
MEMBUAT ADA SELISIH

# WALAUPUN NILAI SELISIHNYA SANGAT KECIL, YANG MEMBUAT JUGA
AKURASINYA JUGA ADA NILAINYA

# DAN SANGAT KECIL JUGA



print("Selisih:", accuracy)

print("Akurasi:",sum(accuracy)/4)
```

Output:

```
Nilai f(x) menggunakan polinomial interpolasi: [15.18  10.962
1.92  -2.04 ]

Nilai f(x) awal pada soal: [15.18  10.962  1.92  -2.04 ]

Selisih: [1.77635684e-15 0.00000000e+00 1.77635684e-15
2.66453526e-15]

Akurasi: 1.5543122344752192e-15
```

Note:
Python Code:

AOL Scientific Computing_ Michael Geraldin Wijaya_2602238021.ipynb