**Kelas A11. 4608/ Kelompok 05**

Anggota :

MUHAMMAD FARIS ASSAMI A11.2022.14647

EPHESIANS PRISMARANTHA A11.2022.14632

ABDUL ROZAK SAPUTRA A11.2022.14663

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS DIAN NUSWANTORO**

**2024**

**Laporan Proyek Akhir   
Mata Kuliah Komputasi Numerik**



|  |
| --- |
| 1. **Problem #1: Diferensiasi Numerik Dikerjakan oleh: Abdul Rozak Saputra / A11.2022.14663** |
| **Instruksi:**   1. **Salinkan source code atau tabel yang dibuat untuk menyelesaikan problem.** 2. **Jelaskan bagian-bagian pada source code atau formula perhitungan yang ada pada tabel**   **Input:**  **Code Selisih Maju:**  def f(x):      fungsi\_x = x\*\*2 + 10\*x + 9      return fungsi\_x  *# RSM = Rumus Selisih Maju*  def selisih\_maju(f,x, h = 1e-5):      RSM = (f(x + h) - f(x)) / h      return RSM  titik\_x = [1, 2, 3]  h = 1e-5  def turunan\_fungsi(x):      turunan\_fx = 2\*x + 10      return turunan\_fx  print("Turunan numerik dan analitik serta error relatif pada titik-titik yang diberikan:")  print("-" \* 60)  for x in titik\_x:      turunan\_numerik = selisih\_maju(f, x, h)      analisis\_numerik = turunan\_fungsi(x)      relative\_error\_differens = abs(turunan\_numerik - analisis\_numerik) / abs(analisis\_numerik) \* 100      print(f"untuk x : {x}")      print(f"turunan numerik : {turunan\_numerik:.5f}")      print(f"turunan analisis : {analisis\_numerik}")      print(f"nilai error relatif : {relative\_error\_differens:.5f}%")      print(f"-" \* 60)  **Code Validasi:**  def f(x):      sebuah\_fungsi\_2 = x\*\*2 + x\*10      return sebuah\_fungsi\_2  def turunan\_fungsi(x):      turunan\_fungsi\_2 = x\*2 + 10      return turunan\_fungsi\_2  def turunan\_numerik(x, h= 1e-5):      CDM = (f(x + h) - f(x - h)) / (2\*h)      return CDM  def nilai\_error\_relatif(analitik, numerik):      nilai\_error = abs(analitik - numerik) / abs(analitik) \* 100      return nilai\_error  titik\_nilai = [1, 2, 3]  results = []  print("-" \* 60)  for z in titik\_nilai:      analitik = turunan\_fungsi(z)      numerik  = turunan\_numerik(z)      error = nilai\_error\_relatif(analitik, numerik)      results.append((z, analitik, numerik, error))      print(f"x = {z}")      print(f"turunan analitik f'({z}) = {analitik}")      print(f"turunan numerik f'({z}) = {numerik}")      print(f"nilai error relatif = {error:.10f}")      print("-" \* 60) Bagian 1: Definisi Fungsi **1. Fungsi f(x) :**  def f(x):      fungsi\_x = x\*\*2 + 10\*x + 9      return fungsi\_x  Fungsi ini mendefinisikan persamaan kuadrat f(x)=x² + 10x + 9 f(x) = x² + 10x + 9 f(x)=x ² + 10x + 9.  **2. Fungsi turunan\_fungsi(x):**  def turunan\_fungsi(x):      turunan\_fx = 2\*x + 10      return turunan\_fx  Fungsi ini mendefinisikan turunan analitik dari f(x)f(x)f(x), yaitu f′(x)=2x+10f'(x) = 2x + 10f′(x)=2x+10.  **Bagian 2: Rumus Selisih Maju (Forward Difference)**  **3. Fungsi selisih\_maju(f, x, h):**  def selisih\_maju(f,x, h = 1e-5):      RSM = (f(x + h) - f(x)) / h      return RSM  Fungsi ini menggunakan metode selisih maju untuk menghitung turunan numerik.  **Bagian 3: Menghitung dan Menampilkan Hasil**  **4. Perhitungan pada titik-titik tertentu**:  titik\_x = [1, 2, 3]  h = 1e-5  for x in titik\_x:      turunan\_numerik = selisih\_maju(f, x, h)      analisis\_numerik = turunan\_fungsi(x)      relative\_error\_differens = abs(turunan\_numerik - analisis\_numerik) / abs(analisis\_numerik) \* 100      print(f"untuk x : {x}")      print(f"turunan numerik : {turunan\_numerik:.5f}")      print(f"turunan analisis : {analisis\_numerik}")      print(f"nilai error relatif : {relative\_error\_differens:.5f}%")      print(f"-" \* 60)  Pada bagian ini, turunan numerik dan analitik dihitung untuk titik-titik tertentu x=1,2,3x = 1, 2, 3x=1,2,3. Selanjutnya, error relatif dihitung dan hasilnya ditampilkan.  **Bagian 4: Validasi dengan Metode Selisih Tengah (Central Difference)**  **5. Fungsi turunan\_numerik(x, h) dengan metode selisih tengah**:  def turunan\_numerik(x, h= 1e-5):      CDM = (f(x + h) - f(x - h)) / (2\*h)      return CDM  Fungsi ini menggunakan metode selisih tengah untuk menghitung turunan numerik.  **6. Fungsi nilai\_error\_relatif(analitik, numerik):**  def nilai\_error\_relatif(analitik, numerik):      nilai\_error = abs(analitik - numerik) / abs(analitik) \* 100      return nilai\_error  Fungsi ini menghitung error relatif antara nilai turunan analitik dan numerik.  **7. Menghitung dan Menampilkan Hasil dengan Metode Selisih Tengah**:  titik\_nilai = [1, 2, 3]  results = []  print("-" \* 60)  for z in titik\_nilai:      analitik = turunan\_fungsi(z)      numerik  = turunan\_numerik(z)      error = nilai\_error\_relatif(analitik, numerik)      results.append((z, analitik, numerik, error))      print(f"x = {z}")      print(f"turunan analitik f'({z}) = {analitik}")      print(f"turunan numerik f'({z}) = {numerik}")      print(f"nilai error relatif = {error:.10f}")      print("-" \* 60)  Bagian ini menghitung turunan numerik menggunakan metode selisih tengah untuk titik-titik x=1,2,3x = 1, 2, 3x=1,2,3 dan membandingkannya dengan nilai turunan analitik, kemudian menampilkan hasilnya. |
| **Instruksi: Tuliskan luaran atau output berdasarkan source code/tabel yang dibuat**  **Output:**  Turunan numerik dan analitik serta error relatif pada titik-titik yang diberikan:  ------------------------------------------------------------  untuk x : 1  turunan numerik : 12.00001  turunan analisis : 12  nilai error relatif : 0.00008%  ------------------------------------------------------------  untuk x : 2  turunan numerik : 14.00001  turunan analisis : 14  nilai error relatif : 0.00007%  ------------------------------------------------------------  untuk x : 3  turunan numerik : 16.00001  turunan analisis : 16  nilai error relatif : 0.00006%  ------------------------------------------------------------  x = 1  turunan analitik f'(1) = 12  turunan numerik f'(1) = 11.999999999989795  nilai error relatif = 0.0000000001  ------------------------------------------------------------  x = 2  turunan analitik f'(2) = 14  turunan numerik f'(2) = 14.000000000002897  nilai error relatif = 0.0000000000  ------------------------------------------------------------  x = 3  turunan analitik f'(3) = 16  turunan numerik f'(3) = 15.999999999749546  nilai error relatif = 0.0000000016  ------------------------------------------------------------   |  |  | | --- | --- | | **Hasil Numerik** | **Error Relatif (%)** | | **12.00001** | **0,00008%** | | **14.00001** | **0,00007%** | | **16.00001** | **0,00006%** | |
| 1. **Problem #2: Integrasi Numerik Dikerjakan oleh: Ephesians Prismaranatha / A11.2022.14632** |
| **Instruksi:**   1. **Salinkan source code atau tabel yang dibuat untuk menyelesaikan problem.** 2. **Jelaskan bagian-bagian pada source code atau formula perhitungan yang ada pada tabel**   **Input:** |
| **Instruksi: Tuliskan luaran atau output berdasarkan source code/tabel yang dibuat**  **Output:**   |  |  | | --- | --- | | **Hasil Numerik** | **Error Relatif (%)** | |  |  | |
| 1. **Problem #3: Regresi Linear Sederhana dan Interpolasi Dikerjakan oleh: Muhammad Faris Assami / A11.2022.14647** |
| **Instruksi:**   1. **Salinkan source code atau tabel yang dibuat untuk menyelesaikan problem.** 2. **Jelaskan bagian-bagian pada source code atau formula perhitungan yang ada pada tabel**   **Input:**  **Regresi Linear Sederhana**  **1. Inisialisasi Data:**  data\_regresi = {      'no': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],      'jarak': [2, 8, 30, 7, 15, 5, 10, 2, 2, 15, 14, 32],      'waktu': [10, 15, 50, 13, 30, 10, 20, 5, 5, 30, 30, 55]  }  data\_x = data\_regresi['jarak']  data\_y = data\_regresi['waktu']  n = len(data\_x)  **2.** **Menghitung Mean dan Sigma:**  sigma\_x = sum(data\_x)  sigma\_y = sum(data\_y)  sigma\_xy = sum(x \* y for x, y in zip(data\_x, data\_y))  sigma\_xx = sum(x \*\* 2 for x in data\_x)  mean\_x = sigma\_x / n  mean\_y = sigma\_y / n  **3. Menghitung Koefisien aaa dan bbb:**  b = (n \* sigma\_xy - sigma\_x \* sigma\_y) / (n \* sigma\_xx - sigma\_x\*\*2)  a = mean\_y - b \* mean\_x  **4. Prediksi dan Evaluasi:**  def prediksi\_waktu(x):      return a + b \* x  new\_jarak = 10  prediksi = prediksi\_waktu(new\_jarak)  prediksi\_waktu\_2 = a + b \* 10  nilai\_aktual = data\_regresi['waktu'][6]  error\_relatif = abs(prediksi\_waktu\_2 - nilai\_aktual) / nilai\_aktual  print(f"hasil prediksi waktu dengan jarak 10 km : {prediksi:.2f} menit")  print(f"Nilai aktual waktu tempuh pada jarak 10 km: {nilai\_aktual} menit")  print(f"Error relatif: {error\_relatif:.2%}") Interpolasi Linear: **1. Inisialisasi Data:**  data\_regresi = {      'no': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],      'jarak': [2, 8, 30, 7, 15, 5, 10, 2, 2, 15, 14, 32],      'waktu': [10, 15, 50, 13, 30, 10, 20, 5, 5, 30, 30, 55],  }  **2. Menghitung Nilai Interpolasi:**  x = 10  x1, y1 = data\_regresi['jarak'][1], data\_regresi['waktu'][1]  x2, y2 = data\_regresi['jarak'][10], data\_regresi['waktu'][10]  y = y1 + ((x - x1) \* (y2 - y1) / (x2 - x1))  **3. Evaluasi:** Interpolasi Kuadrat **1. Inisialisasi Data:**  data\_regresi = {      'no': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],      'jarak': [2, 8, 30, 7, 15, 5, 10, 2, 2, 15, 14, 32],      'waktu': [10, 15, 50, 13, 30, 10, 20, 5, 5, 30, 30, 55]  }  **2. Menentukan Titik Data dan Matriks:**  x1, y1 = data\_regresi['jarak'][1], data\_regresi['waktu'][1]  x2, y2 = data\_regresi['jarak'][6], data\_regresi['waktu'][6]  x3, y3 = data\_regresi['jarak'][9], data\_regresi['waktu'][9]  A = [      [x1\*\*2, x1, 1],      [x2\*\*2, x2, 1],      [x3\*\*2, x3, 1]  ]  B = [y1, y2, y3]  **3. Menggunakan Metode Gauss-Jordan untuk Menyelesaikan Sistem Persamaan Linear:**  def gauss\_jordan(A, B):      n = len(B)      for i in range(n):          max\_row = max(range(i, n), key=lambda r: abs(A[r][i]))          A[i], A[max\_row] = A[max\_row], A[i]          B[i], B[max\_row] = B[max\_row], B[i]          pivot = A[i][i]          A[i] = [x / pivot for x in A[i]]          B[i] /= pivot          for j in range(n):              if i != j:                  factor = A[j][i]                  A[j] = [A[j][k] - factor \* A[i][k] for k in range(n)]                  B[j] -= factor \* B[i]      return B  koefisien = gauss\_jordan(A, B)  a, b, c = koefisien  **4. Prediksi dan Evaluasi:**  x = 10  prediksi\_waktu\_3 = a \* x\*\*2 + b \* x + c  nilai\_asli = 20  nilai\_relative\_error = abs(prediksi\_waktu\_3 - nilai\_asli) / nilai\_asli \* 100  print(f"Koefisien: a= {a:.2f}, b= {b:.2f}, c= {c:.2f}")  print(f"Prediksi waktu tempuh untuk jarak 10 km: {prediksi\_waktu\_3:.0f} menit")  print(f"Nilai aktual: {nilai\_asli} menit")  print(f"Error relatif: {nilai\_relative\_error:.2f}%") |
| **Instruksi: Tuliskan luaran atau output berdasarkan source code/tabel yang dibuat**  **Output:**  hasil prediksi waktu dengan jarak 10 km : 19.77 menit  Nilai aktual waktu tempuh pada jarak 10 km: 20 menit  Error relatif: 1.13%  Interpolasi Linier:  Prediksi waktu dengan jarak 10km : 20.00 menit  error relative : 0.00%  nilai waktu pada jarak 10km : 20 menit  x1 = 8, x2 = 14, y = 15, y2 = 30  Interpolasi Kuadrat:  Koefisien: a= -0.07, b= 3.79, c= -10.71  Prediksi waktu tempuh untuk jarak 10 km: 20 menit  Nilai aktual: 20 menit  Error relatif: 0.00%   |  |  |  | | --- | --- | --- | | **Metode** | **Hasil Numerik** | **Error Relatif (%)** | | **Regresi Linear Sederhana** | **19.77** | **1,15%** | | **Interpolasi Linear** | **20.00** | **0,00%** | | **Interpolasi Kuadratik** | **20.00** | **0,00%** | |
| 1. **Problem #4: Regresi Linear Berganda Dikerjakan oleh: Nama … / NIM** |
| **Instruksi:**   1. **Salinkan source code atau tabel yang dibuat untuk menyelesaikan problem.** 2. **Jelaskan bagian-bagian pada source code atau formula perhitungan yang ada pada tabel**   **Input:** |
| **Instruksi: Tuliskan luaran atau output berdasarkan source code/tabel yang dibuat**  **Output:**  **Persamaan Matriks:**  **Persamaan Regresi:** |
| 1. **Problem Terapan #1: Diferensial Numerik Dikerjakan oleh: Abdul Rozak Saputra/ A11.2022.14663** |
| **Instruksi:**   1. **Salinkan source code atau tabel yang dibuat untuk menyelesaikan problem.** 2. **Jelaskan bagian-bagian pada source code atau formula perhitungan yang ada pada tabel**   **Input:**  **Code No 1 :**  data\_Differensial = {      'frame': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],      'sumbu\_x': [100, 144, 196, 256, 324, 400, 484, 576, 676, 784],  }  *# Data yg diketahui*  long\_roads = 50  pixel\_resolution = 1080  frame\_sec = 60  real\_range\_by\_pixel = long\_roads / pixel\_resolution  velocity\_KMH = []  for i in range(len(data\_Differensial['sumbu\_x']) - 1):      pixel = data\_Differensial['sumbu\_x'][i + 1] - data\_Differensial['sumbu\_x'][i]      meter = pixel \* real\_range\_by\_pixel      time = 1 / frame\_sec      velocity\_MPS = meter / time      velocity\_KMH.append(velocity\_MPS \* 3.6)  for i, velocity in enumerate(velocity\_KMH):      print(f"kecepatan antara frame {i + 1} dengan {i + 2} adalah {velocity:.2f} km/h") Penjelasan:  1. **data\_Differensial**: Dictionary yang menyimpan data frame dan posisi sumbu-x (pixel). 2. **long\_roads**: Panjang jalan dalam meter (50 meter). 3. **pixel\_resolution**: Resolusi pixel (1080 pixel). 4. **frame\_sec**: Jumlah frame per detik (60 frame per detik). 5. **real\_range\_by\_pixel**: Jarak nyata per pixel, dihitung dengan membagi panjang jalan dengan resolusi pixel. 6. **velocity\_KMH**: List kosong untuk menyimpan kecepatan dalam km/jam. 7. **for loop**: Loop untuk menghitung kecepatan antara frame berturut-turut:    * **pixel**: Selisih posisi sumbu-x antara dua frame.    * **meter**: Jarak sebenarnya antara dua frame dalam meter.    * **time**: Waktu yang diperlukan untuk bergerak dari satu frame ke frame berikutnya.    * **velocity\_MPS**: Kecepatan dalam meter per detik.    * **velocity\_KMH**: Kecepatan dalam kilometer per jam, dihitung dengan mengalikan kecepatan dalam meter per detik dengan 3.6. 8. **print loop**: Loop untuk mencetak kecepatan antara frame berturut-turut.   **Code 2 :**  data\_Differensial = {      'frame': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],      'sumbu\_x': [100, 144, 196, 256, 324, 400, 484, 576, 676, 784],  }  velocity\_Limit = 40  long\_roads = 50  pixel\_resolution = 1080  frame\_sec = 60  real\_range\_by\_pixel = long\_roads / pixel\_resolution  velocity\_KMH = []  for i in range(len(data\_Differensial['sumbu\_x']) - 1):      pixel = data\_Differensial['sumbu\_x'][i + 1] - data\_Differensial['sumbu\_x'][i]      meter = pixel \* real\_range\_by\_pixel      time = 1 / frame\_sec      velocity\_MPS = meter / time      velocity\_KMH.append(velocity\_MPS \* 3.6)  traffic\_ticket = False  for velocity in velocity\_KMH:      if velocity > velocity\_Limit:          traffic\_ticket = True          break  if traffic\_ticket:      print(f"Mobil tersebut perlu di tilang ")  else:      print(f"Mobil tersebut tidak perlu di tilang") Penjelasan:  1. **data\_Differensial**: Dictionary yang menyimpan data frame dan posisi sumbu-x (pixel). 2. **velocity\_Limit**: Batas kecepatan dalam km/jam (40 km/jam). 3. **long\_roads**: Panjang jalan dalam meter (50 meter). 4. **pixel\_resolution**: Resolusi pixel (1080 pixel). 5. **frame\_sec**: Jumlah frame per detik (60 frame per detik). 6. **real\_range\_by\_pixel**: Jarak nyata per pixel, dihitung dengan membagi panjang jalan dengan resolusi pixel. 7. **velocity\_KMH**: List kosong untuk menyimpan kecepatan dalam km/jam. 8. **for loop**: Loop untuk menghitung kecepatan antara frame berturut-turut:    * **pixel**: Selisih posisi sumbu-x antara dua frame.    * **meter**: Jarak sebenarnya antara dua frame dalam meter.    * **time**: Waktu yang diperlukan untuk bergerak dari satu frame ke frame berikutnya.    * **velocity\_MPS**: Kecepatan dalam meter per detik.    * **velocity\_KMH**: Kecepatan dalam kilometer per jam, dihitung dengan mengalikan kecepatan dalam meter per detik dengan 3.6. 9. **traffic\_ticket**: Boolean yang digunakan untuk menentukan apakah kendaraan perlu diberi tilang. 10. **if loop**: Loop untuk memeriksa apakah kecepatan melebihi batas kecepatan:     * Jika ada kecepatan yang melebihi batas, **traffic\_ticket** di-set ke True. 11. **if statement**: Mengecek nilai **traffic\_ticket** dan mencetak hasil. |
| **Instruksi: Tuliskan luaran atau output berdasarkan source code/tabel yang dibuat**  **Output:**  **No 1:**  kecepatan antara frame 1 dengan 2 adalah 440.00 km/h  kecepatan antara frame 2 dengan 3 adalah 520.00 km/h  kecepatan antara frame 3 dengan 4 adalah 600.00 km/h  kecepatan antara frame 4 dengan 5 adalah 680.00 km/h  kecepatan antara frame 5 dengan 6 adalah 760.00 km/h  kecepatan antara frame 6 dengan 7 adalah 840.00 km/h  kecepatan antara frame 7 dengan 8 adalah 920.00 km/h  kecepatan antara frame 8 dengan 9 adalah 1000.00 km/h  kecepatan antara frame 9 dengan 10 adalah 1080.00 km/h  No 2 :  Mobil tersebut perlu di tilang   |  |  | | --- | --- | | **Interval Frame** | **Kecepatan (kilometer/jam)** | | **Frame 1-2** | **440,00** | | **Frame 2-3** | **520,00** | | **Frame 3-4** | **600,00** | | **Frame 4-5** | **680,00** | | **Frame 5-6** | **760,00** | | **Frame 6-7** | **840,00** | | **Frame 7-8** | **920.00** | | **Frame 8-9** | **1000,00** | | **Frame 9-10** | **1080,00** |   **Jadi, mobil tersebut perlu di tilang** |
| 1. **Problem Terapan #2: Integrasi Numerik dan Interpolasi Dikerjakan oleh: Nama … / NIM** |
| **Instruksi:**   1. **Salinkan source code atau tabel yang dibuat untuk menyelesaikan problem.** 2. **Jelaskan bagian-bagian pada source code atau formula perhitungan yang ada pada tabel**   **Input:** |
| **Instruksi: Tuliskan luaran atau output berdasarkan source code/tabel yang dibuat**  **Output:**   |  |  | | --- | --- | | **Metode Integrasi** | **Perkiraan total jarak lintasan (meter)** | | **Riemann** |  | | **Trapezoid** |  |  |  |  | | --- | --- | | **Metode Interpolasi** | **Perkiraan posisi mobil pada 5,5 detik** | | **Linear** |  | | **Kuadratik** |  | |
| 1. **Problem Terapan #3: Regresi Linear Sederhana dan Interpolasi Dikerjakan oleh: Muhammad Faris Assami / A11.2022.14647** |
| **Instruksi:**   1. **Salinkan source code atau tabel yang dibuat untuk menyelesaikan problem.** 2. **Jelaskan bagian-bagian pada source code atau formula perhitungan yang ada pada tabel**   **Input:** Bagian 1: Regresi LinearData Regresi data\_Regresi = {      'bulan': [1, 2, 3, 4, 5, 6, 7 ,8 ,9 ,10],      'harga': [39.4, 53.5, 58, 42.5, 44.4, 60.3, 40.8, 63.9, 55.8, 44.4, 75.2, 59.4],      'penjualan': [3615, 3842, 3729, 2980, 3108, 4168, 3473, 4224, 3672, 3459, 3600, 4200]  }  **Data Initialization**: The data\_Regresi dictionary stores monthly sales data (bulan), product prices (harga), and sales volumes (penjualan). Extracting Prices and Demands price = data\_Regresi['harga']  demands = data\_Regresi['penjualan']  **Extract Data**: Extracts price and demand data from the dictionary. Calculating Means mean\_price = sum(price) / len(price)  mean\_demands = sum(demands) / len(demands)  **Mean Calculation**: Calculates the mean price and mean demand.  **Computing Slope (b) and Intercept (a)**  x = sum((price[i] - mean\_price) \* (demands[i] - mean\_demands) for i in range(len(price)))  y = sum((price[i] - mean\_price)\*\*2 for i in range(len(price)))  b = x / y  a = mean\_demands - b \* mean\_price  **Memprediksi Permintaan**  predicted\_price = 50  predicted\_demands = a + b \* predicted\_price  print(f"prediksi penjualan logam mulia adalah {predicted\_demands:.0f} unit")   * **Prediksi** : Memprediksi permintaan pada harga tertentu menggunakan persamaan regresi linierDan=A+BXkamu = a + bxDan=A+b x. * **Output** : Mencetak permintaan yang diprediksi.  Bagian 2: Interpolasi KuadratData Regresi data\_Regresi = {      'bulan': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],      'harga': [39.4, 53.5, 58, 42.5, 44.4, 60.3, 40.8, 63.9, 55.8, 44.4, 75.2, 59.4],      'penjualan': [3615, 3842, 3729, 2980, 3108, 4168, 3473, 4224, 3672, 3459, 3600, 4200]  }  *# x1 = 39.4, x2 = 53.5, x3 = 58*  *# y1 = 3615, y2 = 3842, y3 = 3729*  x1, y1 = data\_Regresi['harga'][0], data\_Regresi['penjualan'][0]  x2, y2 = data\_Regresi['harga'][1], data\_Regresi['penjualan'][1]  x3, y3 = data\_Regresi['harga'][2], data\_Regresi['penjualan'][2]   **Inisialisasi Data** : data\_RegresiKamus menyimpan data penjualan bulanan ( bulan), harga produk ( harga), dan volume penjualan ( penjualan).   **Ekstrak Titik Data** : Mengekstrak titik data tertentu untuk interpolasi kuadrat.  **Fungsi untuk Interpolasi Kuadrat**  def interpolasi\_kuadrat(x1, y1, x2, y2, x3, y3):      A = [          [x1\*\*2, x1, 1],          [x2\*\*2, x2, 1],          [x3\*\*2, x3, 1],      ]      B = [y1, y2, y3]      determinan\_A = A[0][0] \* (A[1][1] \* A[2][2] - A[1][2] \* A[2][1]) - A[0][1] \* (A[1][0] \* A[2][2] - A[1][2] \* A[2][0]) + A[0][2] \* (A[1][0] \* A[2][1] - A[1][1] \* A[2][0])      Adjoin\_A = [          [(A[1][1] \* A[2][2] - A[1][2] \* A[2][1]), -(A[0][1] \* A[2][2] - A[0][2] \* A[2][1]), (A[0][1] \* A[1][2] - A[0][2] \* A[1][1])],          [-(A[1][0] \* A[2][2] - A[1][2] \* A[2][0]), (A[0][0] \* A[2][2] - A[0][2] \* A[2][0]), -(A[0][0] \* A[1][2] - A[0][2] \* A[1][0])],          [(A[1][0] \* A[2][1] - A[1][1] \* A[2][0]), -(A[0][0] \* A[2][1] - A[0][1] \* A[2][0]), (A[0][0] \* A[1][1] - A[0][1] \* A[1][0])]      ]      invA = [[Adjoin\_A[j][i] / determinan\_A for i in range(3)] for j in range(3)]        a = sum(invA[0][j] \* B[j] for j in range(3))      b = sum(invA[1][j] \* B[j] for j in range(3))      c = sum(invA[2][j] \* B[j] for j in range(3))      return a, b, c   * **Pengaturan Matriks** : Menyiapkan matriksAAAdan vektorBBBuntuk menyelesaikan persamaan kuadrat. * **Perhitungan Determinan** : Menghitung determinan matriksAAA. * **Matriks Adjoint** : Menghitung adjoint matriksAAA. * **Matriks Invers** : Menghitung invers matriksAAAmenggunakan adjoint dan determinan. * **Perhitungan Koefisien** : Memecahkan koefisienAAABahasa Indonesia:BBB, DanCCCdari persamaan kuadrat.   **Memprediksi Permintaan dengan Interpolasi Kuadrat**  a, b, c = interpolasi\_kuadrat(x1, y1, x2, y2, x3, y3)  prediksi\_harga = 50  prediksi\_penjualan = a \* prediksi\_harga\*\*2 + b \* prediksi\_harga + c  print(f"Prediksi penjualan {prediksi\_penjualan:.2f} unit")   * **Prediksi** : Menggunakan persamaan kuadratDan=AX2+BX+Cpersamaan y = ax^2 + bx + cDan=sebuah x2+b x+Cuntuk memprediksi permintaan pada harga tertentu. * **Output** : Mencetak permintaan yang diprediksi. |
| **Instruksi: Tuliskan luaran atau output berdasarkan source code/tabel yang dibuat**  **Output:**   |  |  | | --- | --- | | **Metode** | **Prediksi banyak penjualan (kg)** | | **Regresi Linear Sederhana** | **3604** | | **Interpolasi Kuadratik** | **3867.85** | |
| 1. **Problem Terapan #4: Regresi Linear Berganda Dikerjakan oleh: Nama … / NIM** |
| **Instruksi:**   1. **Salinkan source code atau tabel yang dibuat untuk menyelesaikan problem.** 2. **Jelaskan bagian-bagian pada source code atau formula perhitungan yang ada pada tabel**   **Input:** |
| **Instruksi: Tuliskan luaran atau output berdasarkan source code/tabel yang dibuat**  **Output:**  **Persamaan Matriks:**  **Persamaan Regresi:**  **Lama proses rendering pada unit computer tersebut: … jam** |