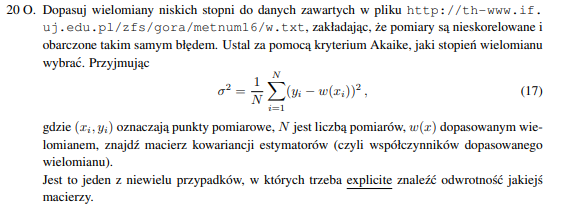
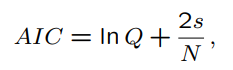
**Zadanie 20**



**Kod w języku Python:**

|  |
| --- |
| *import* numpy *as* np   *def* loadData():  plik = open("dane.txt")  tmp\_points = []  tmp\_values = []  *for* linia *in* plik:  tmp = linia.strip().split()  tmp\_points.append(float(tmp[0]))  tmp\_values.append(float(tmp[1]))  *return* tmp\_points, tmp\_values   *def* Akaike(*stopien*, *points*, *values*):  tmpArray = [[0] \* (*stopien* + 1) *for* i *in* range(*stopien* + 1)]  *for* i *in* range(*stopien* + 1):  *for* j *in* range(*stopien* + 1):  tmpArray[i][j] = pow(*points*[i], *stopien* - j)   A = np.array(tmpArray)  b = [0] \* (*stopien* + 1)  *for* i *in* range(len(b)):  b[i] = *values*[i]   a = np.linalg.solve(A, b)  Q = 0   *for* i *in* range(*stopien*):  tmp = 0  *for* j *in* range(*stopien* + 1):  tmp += (a[*stopien* - j] \* pow(*points*[i], *stopien* - 1))  tmp -= *values*[i]  Q += tmp   *if* Q > 0:  Q = np.log(Q) + float(2 \* s / len(*values*))  *return* Q  *else*:  *return* 0   *def* calculateW(*n*, *points*):  sigma2 = 0  sigmaX2 = 0  *for* i *in* range(len(*points*)):  sigmaX2 += pow(*points*[i], i + 1)  sigma2 += *points*[i]  sigmaX2 \*= *n* W = sigmaX2 - sigma2  *return* W   *def* calculateWa(*n*, *points*, *values*):  sigmaXY = 0  sigmaX = 0  sigmaY = 0  *for* i *in* range(len(*points*)):  sigmaXY += *points*[i] \* *values*[i]  sigmaX += *points*[i]  sigmaY += *values*[i]  sigmaXY \*= *n* Wa = sigmaXY - sigmaX \* sigmaY  *return* Wa   *def* calculateWb(*points*, *values*):  sigmaX2 = 0  sigmaY = 0  sigmaX = 0  sigmaXY = 0  *for* i *in* range(len(*values*)):  sigmaX2 += pow(*points*[i], 2)  sigmaY += *values*[i]  sigmaX += *points*[i]  sigmaXY += *points*[i] \* *values*[i]   Wb = sigmaX2 \* sigmaY - sigmaX \* sigmaXY  *return* Wb   *def* function(*a*, *x*):  result = 0  *for* i *in* range(len(*a*)):  result += *a*[i] \* pow(*x*, len(*a*) - 1)  *return* result   *def* var(*points*, *values*, *a*):  result = 0  *for* i *in* range(len(*points*)):  tmp = *values*[i] - function(*a*, *points*[i])  result += pow(tmp, 2)  result /= len(*points*)  *return* result;   *def* createMatrix(*points*):  A = [[0] \* 2 *for* i *in* range(len(*points*))]  *for* i *in* range(len(*points*)):  A[i][0] = *points*[i]  A[i][1] = 1  *return* A   # Wczytywanie danych z pliku tmp\_points, tmp\_values = loadData() points = np.array(tmp\_points) values = np.array(tmp\_values)  s = 10 # Stopien wielomianu dla ktorgo szukamy AIC  print("\nWartosc kryterium Akaike'a dla stopnia:") akaikeValues = [0] \* s *for* i *in* range(s):  akaikeValues[i] = Akaike(i + 1, points, values) # 0 gdy wynik bylby ujemny  print(i + 1, ") Q = ", akaikeValues[i])  n = 1 # Wybieramy stopien wielomianu *for* i *in* range(s - 1):  *if* akaikeValues[i + 1] < akaikeValues[i]:  *if* akaikeValues[i + 1] > 0:  n = i + 2  print("Wybieramy wielomian ", n, " stopnia\n") # Bedzie to 1 stopien  a = [0] \* (n + 1) # Tablica wspolczynnikow a i b  # Obliczanie wspolczynnikow a i b W = calculateW(n, points) Wa = calculateWa(n, points, values) Wb = calculateWb(points, values)  a[0] = float(Wa / W) a[1] = float(Wb / W)  print("Obliczone wspolczynniki wielomianu:", "\na = ", a[0], "\nb = ", a[1])  # Obliczanie wariancji Var = var(points, values, a) print("\nObliczona wariancja: ", Var)  # Obliczanie macierzy kowariancji A = np.array(createMatrix(points)) Cp = np.dot(A.transpose(), A) Cp = np.linalg.inv(Cp) Cp = Cp \* Var  print("\nMacierz kowariancji:\n", Cp)  **Wyniki:** |

Pomiary są nieskorelowane i obarczone takim samym błędem. Wykorzystując liniowe zagadnienie najmniejszych kwadratów minimalizowałem formę kwadratową po as, as-1,… a0. Obliczając kolejne pochodne po as, as-1,… a0, otrzymujemy s+1 równań z s+1 niewiadomymi. Do rozwiązania zagadnień numerycznych i algebraicznych użyłem biblioteki NumPy. Znając wektor współczynników jesteśmy w stanie obliczyć AIC.



Wartość kryterium Akaike była najmniejsza dla wielomianu pierwszego stopnia. Do obliczenia wartośći współczynników a i b wykorzystałem regułę Cramer’a:

C:\Users\Bartek\Downloads\CodeCogsEqn (1).png

C:\Users\Bartek\Downloads\CodeCogsEqn (2).png

C:\Users\Bartek\Downloads\CodeCogsEqn (3).png

Aby znaleźć macierz kowariancji estymatorów należy wyliczyć wariancje ze wzoru podanego w treści zadania oraz macierz An x 2 równań liniowych dla n punktów, w których wykonano pomiary. Ponieważ pomiary były nieskorelowane i obarczone takim samym błędem macierz kowariancji upraszcza się do wzoru:

