**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Лабораторная работа №2**

**«Численные методы»**

**Вариант 2**

Ёды Никиты Дмитриевича

студента 3 курса, 6 группы

специальность «прикладная математика»

Преподаватель:

Будник А.М.

**Постановка задачи**

Имеем входные данные:

1. 10 точек на отрезке x ∈ [0.2 1.2];
2. Значения заданных формулой
3. Точки

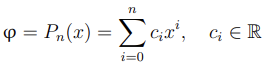
Нам необходимо:

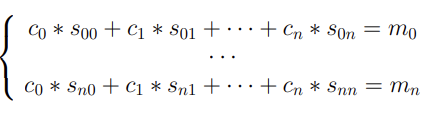
1. Методом наименьших квадратов построить многочлен наилучшего среднеквадратичного приближения 5 степени и вычислить значение в точках
2. Построить интерполяционный многочлен Лагранжа 10 степени, вычислить значение в точках
3. Построить интерполяционный многочлен Ньютона 10 степени, вычислить значение в точках
4. Минимизировать остаток интерполирования многочленами Чебышева;

5. Осуществить интерполирование способом в начале таблицы;

**Метод наименьших квадратов**

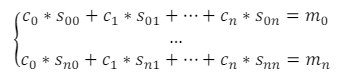
Пусть Φ – пространство, порожденное элементами ,, … ,. –элемент наилучшей аппроксимации 𝑓(𝑥).





Задачу построения можно свести к задаче отыскания 𝑐𝑖 , таких что

В качестве ,, … , возьмем систему многочленов 1, 𝑥, … , 𝑥𝑛. Тогда получим СЛАУ:



**Реализация**

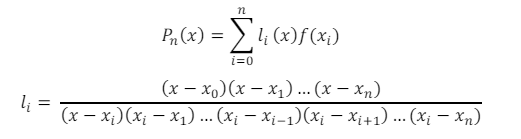
|  |
| --- |
| import numpy as np  from scipy import integrate  def f(x):      return 0.2 \* np.exp(x) + 0.8 \* np.cos(x)  def s(i, j):      return 1. / (i + j + 1)  def m(i):      return integrate.quad(lambda x: pow(x, i) \* (0.2 \* np.exp(x) + 0.8 \* np.cos(x)), 0, 1)  def poly\_val(coefs, x):      i = 0      res = 0.0      for coef in coefs:          res += coef \* pow(x,i)          i += 1      return res  X = [0.27, 0.7, 1.17]  A = np.array([[s(i, j) for i in range(6)] for j in range(6)])  b = np.array([m(i)[0] for i in range(6)])  coefs = np.linalg.solve(A, b)  for x in X:      print('P(', x, ') = ', poly\_val(coefs, x), sep='')      print('f(', x, ') = ', f(x), sep='')      print('|P(', x, ') - f(',x,')| = ', abs( poly\_val(coefs, x) - f(x)))      print() |

**Вывод по полученным результатам**

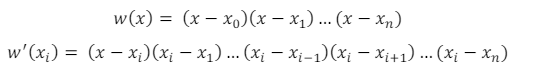
|  |
| --- |
| Метод наименьших квадратов  P(0.27) = 1.0330097943370418  f(0.27) = 1.033009607239362  |P( 0.27 ) - f( 0.27 )| = 1.870976797935242e-07  P(0.7) = 1.0146244524477475  f(0.7) = 1.014624291321686  |P( 0.7 ) - f( 0.7 )| = 1.6112606138207752e-07  P(1.17) = 0.9565341788533362  f(1.17) = 0.9565198751522842  |P( 1.17 ) - f( 1.17 )| = 1.4303701052043039e-05 |

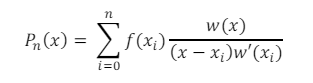
**Интерполяционный многочлен Лагранжа**

Интерполяционный многочлен Лагранжа имеет следующий вид:

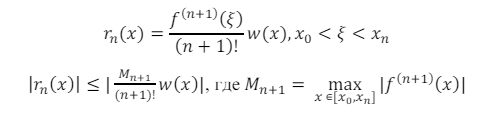


Обозначим через

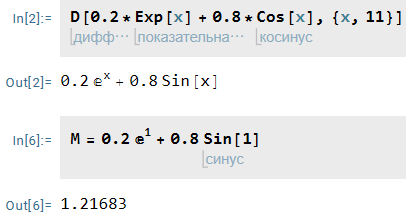




Погрешность интерполяции для многочлена Лагранжа по n+1 узлу (остаток интерполяции):



Вычислим *𝑓(11)(𝑥):*



*Max = 1.21683*

**Реализация**

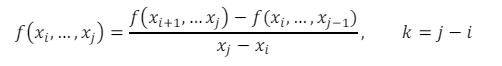
|  |
| --- |
| def w(x, nodes):   res = 1.0   for node in nodes:    res \*= (x - node)   return res  def lagrange\_error(M, x, nodes):   return np.abs(M \* w(x, nodes) / math.factorial(nodes.size))  def lagrange\_value(x, nodes):   res = 0.0   for i in range(nodes.size):    p = 1.0    for j in range(nodes.size):     if i == j:      continue     p \*= (x - nodes[j])     p /= (nodes[i] - nodes[j])    res += p \* f(nodes[i])   return res  interpolation\_nodes = np.linspace(0.0, 1.0, num=11)  for x in X:   print('P(', x, ') = ', lagrange\_value(x,interpolation\_nodes), sep='')   print('f(', x, ') = ', f(x), sep='')   print('r(', x, ') = ', lagrange\_error(1.21,x,interpolation\_nodes), sep='')   print('|P(', x, ') - f(', x, ')| = ', abs(lagrange\_value(x,interpolation\_nodes) - f(x)))   print() |

**Вывод по полученным результатам**

|  |
| --- |
| Лагранж  P(0.27) = 1.0330096072393602  f(0.27) = 1.033009607239362  r(0.27) = 3.0217219233764112e-15  |P( 0.27 ) - f( 0.27 )| = 1.7763568394002505e-15  P(0.7) = 1.014624291321686  f(0.7) = 1.014624291321686  r(0.7) = 1.0177044392397262e-29  |P( 0.7 ) - f( 0.7 )| = 0.0  P(1.17) = 0.9565198751046964  f(1.17) = 0.9565198751522842  r(1.17) = 7.516971950491497e-11  |P( 1.17 ) - f( 1.17 )| = 4.758782257141547e-11 |

**Построить интерполяционный многочлен Ньютона 10 степени**

Введем понятие разделенных разностей. Разделенная разность нулевого порядка совпадает с 𝑓(𝑥𝑖). Разделенная разность k-го порядка определяется следующим образом:



Интерполяционным многочленом Ньютона с разделенными разностями

называется интерполяционный многочлен следующего вида:



**Реализация**

|  |
| --- |
| def build\_table(nodes): # построение таблицы разделенных разностей   table = np.zeros(shape=(nodes.size, nodes.size))   for i in range(nodes.size):    table[i, 0] = f(nodes[i])   for i in range(1, nodes.size):    for j in range(nodes.size - i):     table[j, i] = (table[j + 1, i - 1] - table[j, i - 1]) / (i \* 0.1)   return table  def newton\_value(x, rr\_table, nodes):   n = rr\_table.shape[0]   p = 1.0   res = rr\_table[0, 0]   for i in range(1, n):    p \*= (x - nodes[i - 1])    res += p \* rr\_table[0, i]   return res  rr\_table = build\_table(interpolation\_nodes)  for x in X:   print('P(', x, ') = ', newton\_value(x,rr\_table, interpolation\_nodes), sep='')   print('f(', x, ') = ', f(x), sep='')   print('|P(', x, ') - f(', x, ')| = ', abs(newton\_value(x,rr\_table, interpolation\_nodes) - f(x)))   print() |

**Вывод по полученным результатам**

|  |
| --- |
| Ньютон  P(0.27) = 1.03300960723936  f(0.27) = 1.033009607239362  |P( 0.27 ) - f( 0.27 )| = 1.9984014443252818e-15  P(0.7) = 1.014624291321686  f(0.7) = 1.014624291321686  |P( 0.7 ) - f( 0.7 )| = 0.0  P(1.17) = 0.9565198751046964  f(1.17) = 0.9565198751522842  |P( 1.17 ) - f( 1.17 )| = 1.5100143357926754e-1 |

**Минимизировать остаток интерполирования многочленами Чебышева**

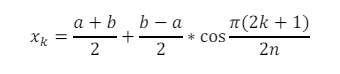
Многочлены Чебышева имеют следующий вид:



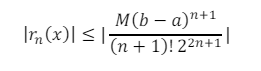
Если в качестве узлов интерполирования взять корни многочлена Чебышева, то остаток интерполирования будут минимальным.

Для 𝑥 ∈ [𝑎, 𝑏] корни многочлена Чебышева можно найти следующим

образом:



По полученным узлам можно построить интерполяционный многочлен, остаток которого будет оцениваться следующим образом:



**Реализация**

|  |
| --- |
| def chebyshev\_nodes(n, a, b):    res = []    for i in range(n):     x = (a + b) / 2 + (b - a) / 2 \* np.cos(np.pi \* (2 \* i + 1) / (2 \* n))     res.append(x)    return np.array(res)   print('\n\nУзлы выбранные наилучшим образом: ')   best\_nodes = chebyshev\_nodes(11, 0, 1)   print(best\_nodes)   for x in X:    print('P(', x, ') = ', lagrange\_value(x, best\_nodes), sep='')    print('f(', x, ') = ', f(x), sep='')    print('r(', x, ') = ', lagrange\_error(0.3, x, best\_nodes), sep='')    print('|P(', x, ') - f(', x, ')| = ', abs(lagrange\_value(x, best\_nodes) - f(x)))    print() |

**Вывод по полученным результатам**

|  |
| --- |
| Узлы выбранные наилучшим образом:  [0.99491072 0.954816 0.87787479 0.77032041 0.64086628 0.5  0.35913372 0.22967959 0.12212521 0.045184 0.00508928]  P(0.27) = 1.0330096072393693  f(0.27) = 1.033009607239362  r(0.27) = 3.0635102159496062e-15  |P( 0.27 ) - f( 0.27 )| = 7.327471962526033e-15  P(0.7) = 1.0146242913216776  f(0.7) = 1.014624291321686  r(0.7) = 3.5221161872162815e-15  |P( 0.7 ) - f( 0.7 )| = 8.43769498715119e-15  P(1.17) = 0.9565198751208399  f(1.17) = 0.9565198751522842  r(1.17) = 1.2270575969676284e-11  |P( 1.17 ) - f( 1.17 )| = 3.1444291614945996e-11 |