Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №5

Вариант 2

Интерполяция функции

Выполнил:

Брагин Роман Андреевич

Проверил:

Рыбаков Степан Дмитриевич

г. Санкт-Петербург

2025

Цель работы:

1. Решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек.
2. Лабораторная работа состоит из двух частей: вычислительной и программной.
3. Для анализа использовать:
    a. многочлен Лагранжа;
    b. многочлен Ньютона;
    c. многочлен Гаусса.

Задание:

	x	y	№ варианта	X_1	X_2
Таблица 1.2	0,50	1,5320	2	0,502	0,645
	0,55	2,5356	7	0,751	0,651
	0,60	3,5406	12	0,523	0,639
	0,65	4,5462	17	0,761	0,661
	0,70	5,5504	22	0,545	0,627
	0,75	6,5559	27	0,783	0,683
	0,80	7,5594	32	0,557	0,641

№ варианта	Метод	№ варианта	Метод	№ варианта	Метод
1	1, 2, 3	10	1, 2, 3	19	1, 2, 3
2	1, 2, 4	11	1, 2, 3	20	1, 2, 4
3	1, 2, 3	12	1, 2, 4	21	1, 2, 3
4	1, 2, 3	13	1, 2, 3	16	1, 2, 3
5	1, 2, 4	14	1, 2, 4	23	1, 2, 4
6	1, 2, 3	15	1, 2, 3	24	1, 2, 3
7	1, 2, 3	16	1, 2, 3	25	1, 2, 3
8	1, 2, 4	17	1, 2, 4	26	1, 2, 4
9	1, 2, 4	18	1, 2, 3	16	1, 2, 4

Вычислительная реализация задачи:

Таблица конечных разностей:

номер	x_i	y_i	delta 1	delta 2	delta 3	delta 4	delta 5	delta 6
0.	0,5	1,532	1,0036	0,0014	-0,0008	-0,0012	0,0059	-0,0166
1.	0,55	2,5356	1,005	0,0006	-0,002	0,0047	-0,0107	
2.	0,6	3,5406	1,0056	-0,0014	0,0027	-0,006		
3.	0,65	4,5462	1,0042	0,0013	-0,0033			
4.	0,7	5,5504	1,0055	-0,002				
5.	0,75	6,5559	1,0035					
6.	0,8	7,5594						

1) Вычисляем значение функции для аргумента X_1 используя формулу Ньютона:
Воспользуемся формулой Ньютона для интерполирования вперед, так как X_1 = 0.502 лежит в левой половине отрезка.

Определим вспомогательный параметр $t = \frac{(x-x_n)}{h} = \frac{(0.502-0.500)}{0.05} = 0.04.$

$$N_6(x) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \frac{t(t-1)(t-2)}{3!}\Delta^3 y_0 + \frac{t(t-1)(t-2)(t-3)}{4!}\Delta^4 y_0 + \frac{t(t-1)(t-2)(t-3)(t-4)}{5!}\Delta^5 y_0 + \frac{t(t-1)(t-2)(t-3)(t-4)(t-5)}{6!}\Delta^6 y_0 = 1.5320 + 0.04 * 1.0036 + \frac{0.04(0.04-1)}{2} * 0.0014 + \frac{0.04\,(0.04-1)\,(0.04-2)}{6} + (-0.0008) + \frac{0.04\,(0.04-1)\,(0.04-2)\,(0.04-3)}{24} * (-0.0012) + \frac{0.04\,(0.04-1)\,(0.04-2)\,(0.04-3)\,(0.04-4)}{120} * 0.0059 + \frac{0.04\,(0.04-1)\,(0.04-2)\,(0.04-3)\,(0.04-4)\,(0.04-5)}{720} * (-0.0166) \approx 1.57226249$$

2) Вычисляем значение функции для аргумента X_2 используя формулу Гаусса:

Ближайшая центральная точка a=0.65 (по таблице), X_2 = 0.645 < 0.65, то есть $x < a \rightarrow$ используем вторую интерполяционную формулу Гаусса.

$$t = \frac{(x-x_0)}{h} = \frac{(0.645-0.65)}{0.05} = -0.1$$

$$P_6(x) = y_0 + t\Delta y_{-1} + \frac{t(t+1)}{2!}\Delta^2 y_{-1} + \frac{(t+1)t(t-1)}{3!}\Delta^3 y_{-2} + \frac{(t+2)(t+1)t(t-1)}{4!}\Delta^4 y_{-2} +$$

$$+\frac{(t+2)(t+1)t(t-1)(t-2)}{5!}\Delta^5 y_{-3}+\frac{(t+3)(t+2)(t+1)t(t-1)(t-2)}{6!}\Delta^6 y_{-3}=$$

$$=4.5462+(-0.1)*1.0056+\frac{-0.1\,(-0.1+1)}{2}*(-0.0014)+$$

$$+\frac{(-0.1+1)\,(-0.1)\,(-0.1-1)}{6}*(-0.0020)$$

$$+\frac{(-0.1+2)\,(-0.1+1)\,(-0.1)\,(-0.1-1)}{24}*(0.0047)+$$

$$+\frac{(-0.1+2)\,(-0.1+1)\,(-0.1)\,(-0.1-1)\,(-0.1-2)}{120}*(0.0059)+$$

$$+\frac{(-0.1+3)(-0.1+2)(-0.1+1)(-0.1)(-0.1-1)(-0.1-2)}{720}*(-0.0166)\approx 4.4457138257325$$

Программная реализация задачи:

```
func Bessel(xs, ys []float64, x float64) (float64, error) {
  n := len(xs)
  if n%2 == 0 {
    return 0, errors.New("метод Бесселя требует нечётного количества узлов")
  }
  mid := n / 2
  h := xs[1] - xs[0]
  t := (x - (xs[mid]+xs[mid-1])/2) / h

  diffTable := utils.BuildFiniteDiffs(ys)

  sum := (diffTable[mid-1][0] + diffTable[mid][0]) / 2
  p := 1.0
  fact := 1.0
  var k int
  for i := 1; i < n; i++ {
    if i%2 == 1 {
      k = (i + 1) / 2
      p *= t - 0.5
    } else {
      k = i / 2
      p *= t*t - float64(k*k)
    }
    fact *= float64(i)
    idx := mid - (i / 2) - 1
    if idx < 0 || idx >= len(diffTable) {
      break
    }
    sum += p * diffTable[idx][i] / fact
  }

  return sum, nil
}
```

```
func Gauss(xs, ys []float64, x float64) (float64, error) {
  n := len(xs)
  if n%2 == 0 {
```

```
    return 0, errors.New("метод Гаусса требует нечётного количества узлов")
  }

  mid := n / 2
  h := xs[1] - xs[0]
  t := (x - xs[mid]) / h
  diffTable := utils.BuildFiniteDiffs(ys)

  sum := diffTable[mid][0]
  fact := 1.0
  p := 1.0
  for i := 1; i < n; i++ {
    if i%2 == 1 {
      k := (i + 1) / 2
      p *= (t - float64(k-1))
    } else {
      k := i / 2
      p *= (t + float64(k))
    }
    fact *= float64(i)
    idx := mid - (i / 2)
    sum += p * diffTable[idx][i] / fact
  }

  return sum, nil
}
```

```
func Lagrange(xs, ys []float64, x float64) float64 {
  n := len(xs)
  result := 0.0

  for i := 0; i < n; i++ {
    li := 1.0
    for j := 0; j < n; j++ {
      if j != i {
        li *= (x - xs[j]) / (xs[i] - xs[j])
      }
    }
    result += li * ys[i]
  }
  return result
}
```

```
func Newton(xs, ys []float64, x float64) float64 {
  n := len(xs)
  coeff := make([]float64, n)
  copy(coeff, ys)

  for j := 1; j < n; j++ {
    for i := n - 1; i >= j; i-- {
      coeff[i] = (coeff[i] - coeff[i-1]) / (xs[i] - xs[i-j])
    }
  }

  result := coeff[0]
  mult := 1.0
```

```
  for i := 1; i < n; i++ {
    mult *= (x - xs[i-1])
    result += coeff[i] * mult
  }

  return result
}
```

```
func Stirling(xs, ys []float64, x float64) (float64, error) {
  n := len(xs)
  if n%2 == 0 {
    return 0, errors.New("метод Стирлинга требует нечётного количества узлов")
  }
  mid := n / 2
  h := xs[1] - xs[0]
  t := (x - xs[mid]) / h
  diffTable := utils.BuildFiniteDiffs(ys)

  sum := diffTable[mid][0]
  p := 1.0
  fact := 1.0
  for i := 1; i < n; i++ {
    if i%2 == 0 {
      k := i / 2
      p *= (t*t - float64(k*k))
    } else {
      p *= t
    }
    fact *= float64(i)
    idx := mid - (i / 2)
    sum += p * diffTable[idx][i] / fact
  }

  return sum, nil
}
```

Вывод:

В рамках данной работы была проведена аппроксимация функций с использованием различных подходов: линейного, квадратичного, кубического, экспоненциального и логарифмического. На основе этих методов был разработана программа на языке Golang, реализующий метод наименьших квадратов и визуализирующий исходную функцию вместе с аппроксимирующими кривыми.

Проведенный анализ позволил определить наиболее точную модель аппроксимации, рассчитать среднеквадратичное отклонение, а также

вычислить коэффициент корреляции Пирсона для линейной зависимости.