Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа №1

по курсу «Численные методы»

«Приближение функции кубическими сплайнами»

Студент:

Группа: ИУ9-61Б

Преподаватель: Домрачева А.Б.

Москва 2025

1 Постановка задачи

Дано: функция $y = f(x)$ задана конечным набором точек

$y_i = f(x_i), \quad i = \overline{0,n}$ на отрезке $[a,b]$, $a = x_0, \quad b = x_n, \quad x_i = a+ih$, $h = \dfrac{(b-a)}{n}$

x_i	x_0	x_1	$\ldots$	x_{n-1}	x_n
y_i	y_0	y_1	$\ldots$	y_{n-1}	y_n

Найти: интерполяционную функцию $y = g(x)$: $g(x_i) = f(x_i), \quad i = \overline{0,n}$ (т.е. функцию, совпадающую со значениями $y_i = f(x_i), \quad i = \overline{0,n}$ в узлах интерполяции $x_i, \quad i = \overline{0,n}$):

1. Протабулировать функцию $f(x)$ на отрезке $[a,b]$ с шагом $h = \frac{(b-a)}{32}$ и распечатать таблицу (x_i, y_i), $i = \overline{0,n}$.
2. Для заданных узлов (x_i, y_i) построить кубический сплайн (распечатать массивы a, b, c, d).
3. Вычислить значения $f(x)$ в точках $x_i^* = a + \left(i - \frac{1}{2}\right) h, \quad h = \frac{(b-a)}{n}$.
4. Вычислить значения оригинальной функции и сплайна в произвольной точке, задаваемой с экрана.

Индивидуальный вариант (№4): $y = f(x)$ задана функцией: $y = 2x * cos(x/2)$ на отрезке $[0, \pi]$.

2 Основные теоретические сведения

2.1 Метод прогонки

Метод прогонки применяется для решения систем линейных уравнений с трёхдиагональной матрицей коэффициентов. Такая система имеет вид:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = f_i, \quad i = 1, 2, \ldots, n, \tag{1}$$

где a_i, b_i, c_i – коэффициенты системы, а f_i – правая часть.

Прямой ход: на первом этапе метод прогонки преобразует систему к виду, удобному для последовательного нахождения неизвестных. Вводятся новые коэффициенты:

$$\beta_i = \frac{c_i}{b_i - a_i\beta_{i-1}}, \quad i = 1, 2, \ldots, n-1, \tag{2}$$

$$\phi_i = \frac{f_i - a_i\phi_{i-1}}{b_i - a_i\beta_{i-1}}, \quad i = 1, 2, \ldots, n. \tag{3}$$

Обратный ход: после вычисления коэффициентов β_i и ϕ_i производится обратный ход, на котором находятся неизвестные:

$$x_n = \phi_n, \tag{4}$$

$$x_i = \phi_i - \beta_i x_{i+1}, \quad i = n-1, n-2, \ldots, 1. \tag{5}$$

Условия применимости: метод прогонки применим, если выполнены условия:

$$|b_i| > |a_i| + |c_i|, \quad \forall i. \tag{6}$$

2.2 Сплайн-интерполяция

Интерполяционной функцией называется функция $y = g(x)$, проходящая через заданные точки, называемые узлами интерполяции:

$$g(x_i) = f(x_i), \quad i = \overline{0, n}.$$

При этом в промежуточных точках равенство выполняется с некоторой погрешностью

$$g(x_i^*) \approx f(x_i^*).$$

Задача интерполяции заключается в поиске такой функции $y = g(x)$.

Приближение функции кубическим сплайном — пример задачи интерполяции.

Сплайн k-го порядка — функция, проходящая через все узлы (x_i, y_i), $i = \overline{0, n}$, являющаяся многочленом k-ой степени на каждом частичном отрезке разбиения $[x_i, x_{i+1}]$, $x_i = a + ih$, $h = \frac{(b-a)}{n}$, $x_i \in [a, b]$ и имеющая первые p непрерывных на $[a, b]$ производных. $d = k - p$ — дефект сплайна.

Наиболее употребительны сплайны третьего порядка с дефектом $d = 1$ (кубические сплайны).

На каждом частичном отрезке разбиения кубический сплайн описывается

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

$$x \in [x_i, x_{i+1}], \quad i = \overline{0, n-1}$$

На частные многочлены накладываются условия:

1. Сплайн проходит через все узлы

$$S_i(x_i) = y_i, \quad i = \overline{0, n-1}; \quad S_{n-1}(x_n) = y_n$$

2. Условие гладкости на краях

$$S_0''(x_0) = 0; \quad S_{n-1}''(x_n) = 0$$

3. Непрерывность сплайна и его первых двух производных в промежуточных узлах

$$S_{i-1}'(x_i) = S_i'(x_i);$$

$$S_{i-1}''(x_i) = S_i''(x_i);$$

$$i = \overline{0, n-1}$$

Эти условия позволяют выразить коэффициенты a_i, b_i, d_i и приводят к трехдиагональной СЛАУ относительно коэффициента c_i:

$$a_i = y_i, \quad i = \overline{0, n-1};$$

$$b_i = \frac{y_{i+1} - y_i}{h} - \frac{h}{3}(c_{i+1} + 2c_i), \quad i = \overline{0, n-2};$$

$$b_{n-1} = \frac{y_{n-1} - y_{n-1}}{h} - \frac{2h}{3}c_{n-1};$$

$$d_i = \frac{c_{i+1} - c_i}{3h}, \quad i = \overline{0, n-2};$$

$$d_{n-1} = -\frac{c_n}{3h}$$

СЛАУ с трехдиагональной матрицей относительно коэффициента c_i:

$$c_{i-1} + 4c_i + c_{i+1} = \frac{3(y_{i+1} - 2y_i + y_{i-1})}{h^2}, \quad i = \overline{1, n-1};$$

$$c_0 = c_n = 0,$$

где $h = x_{i+1} - x_i$, $i = \overline{0, n-1}$ - постоянный шаг интерполяции.

3 Реализация

```go
package main

import (
  "fmt"
)

func forward(a, b, c, d []float64) ([]float64, []float64) {
  n := len(d)
  alpha := make([]float64, n-1)
  beta := make([]float64, n)

  alpha[0] = c[0] / b[0]
  beta[0] = d[0] / b[0]

  for i := 1; i < n-1; i++ {
    alpha[i] = c[i] / (b[i] - a[i-1]*alpha[i-1])
  }

  for i := 1; i < n; i++ {
    beta[i] = (d[i] - a[i-1]*beta[i-1]) / (b[i] - a[i-1]*alpha[i-1])
  }

  return alpha, beta
}

func backward(alpha, beta []float64) []float64 {
  n := len(beta)
  x := make([]float64, n)
  x[n-1] = beta[n-1]

  for i := n - 2; i >= 0; i-- {
    x[i] = beta[i] - alpha[i]*x[i+1]
  }
```

```
34
35    return x
36  }
37
38  func progonka(a, b, c, d []float64) []float64 {
39    alpha, beta := forward(a, b, c, d)
40    x := backward(alpha, beta)
41    return x
42  }
```

Листинг 1: Метод прогонки

```
1   package main
2
3   import (
4     "fmt"
5     "math"
6   )
7
8   type CubSpline struct {
9     x, y        []float64
10    a, b, c, d []float64
11  }
12
13  func NewCubSpline(x, y []float64, h float64) *CubSpline {
14    n := len(x)
15    if n != len(y) || n < 2 {
16      panic("Invalid input data")
17    }
18
19    a := make([]float64, n-1)
20    b := make([]float64, n)
21    c := make([]float64, n-1)
22    d := make([]float64, n)
23
24    b[0] = 1.0
25    b[n-1] = 1.0
26
27    for i := 1; i < n-1; i++ {
28      a[i-1] = h
29      b[i] = 4 * h
30      c[i] = h
31      d[i] = 3 * ((y[i+1]-y[i])/h - (y[i]-y[i-1])/h)
32    }
33
34    cCoeffs := progonka(a, b, c, d)
35
```

```
  spline := &CubSpline{
    x: x,
    y: y,
    a: make([]float64, n-1),
    b: make([]float64, n-1),
    c: cCoeffs,
    d: make([]float64, n-1),
  }

  for i := 0; i < n-1; i++ {
    spline.a[i] = y[i]
    if i < n-2 {
      spline.b[i] = (y[i+1]-y[i])/h - h*(2*cCoeffs[i]+cCoeffs[i+1])/3
      spline.d[i] = (cCoeffs[i+1] - cCoeffs[i]) / (3 * h)
    } else {
      spline.b[i] = (y[i+1]-y[i])/h - 2*h*cCoeffs[i]/3
      spline.d[i] = -cCoeffs[i] / (3 * h)
    }
  }

  return spline
}

func (s *CubSpline) Interpolate(xVal float64) float64 {
  // Поиск интервала
  i := 0
  for i < len(s.x)-1 && xVal > s.x[i+1] {
    i++
  }

  dx := xVal - s.x[i]
  return s.a[i] + s.b[i]*dx + s.c[i]*dx*dx + s.d[i]*dx*dx*dx
}

func generateNodesWithStep(f func(float64) float64, a, b float64, n int) ([]
    float64, []float64, []float64, float64) {
  h := (b - a) / float64(n)
  x := make([]float64, n)
  y := make([]float64, n)
  xPrime := make([]float64, n)

  for i := 0; i < n; i++ {
    x[i] = a + float64(i)*h
    y[i] = f(x[i])
    xPrime[i] = a + (float64(i)-0.5)*h
  }
```

```
81
82    return x, y, xPrime, h
83  }
84
85  func main() {
86    // Задаем функцию
87    // Вариант 4
88    f := func(x float64) float64 {
89      return 2 * x * math.Cos(x/2)
90    }
91
92    // Задаем отрезок
93    a, b := 0.0, math.Pi
94
95    // Задаем шаг интерполяции
96    n := 32
97
98    // Генерация узлов с заданным шагом
99    x, y, xPrime, h := generateNodesWithStep(f, a, b, n)
100
101   // Построение сплайна
102   spline := NewCubSpline(x, y, h)
103
104   fmt.Println("Коэффициенты сплайна:")
105   fmt.Printf("%-15s %16s %16s %16s %16s\n", "Интервал", "a", "b", "c", "d")
106   fmt.Println("
        ------------------------------------------------------------------------------
      ")
107
108   for i := 0; i < len(x)-1; i++ {
109     fmt.Printf("[%.10f, %.10f] %16.10f %16.10f %16.10f %16.10f\n",
110       x[i], x[i+1], spline.a[i], spline.b[i], spline.c[i], spline.d[i])
111   }
112
113   fmt.Println("Таблица значений:")
114   fmt.Printf("%2s %16s %16s %16s %16s\n", "i", "x_i‘", "f(x_i‘)", "S(x_i‘)", "
      Погрешность")
115   fmt.Println("
        ------------------------------------------------------------------------------
      ")
116
117   for i := 0; i < len(x); i++ {
118     fxPrime := f(xPrime[i])
119     sxPrime := spline.Interpolate(xPrime[i])
120     errAbs := math.Abs(fxPrime - sxPrime)
121     fmt.Printf("%2d %16.10f %16.10f %16.10f %16.10f\n",
```

```
122         i, xPrime[i], fxPrime, sxPrime, errAbs)
123     }
124
125     var point float64
126     fmt.Print("Введите произвольную точку для вычисления значений функции и сплайна: ")
127     fmt.Scan(&point)
128
129     originalValue := f(point)
130     splineValue := spline.Interpolate(point)
131     errAbs := math.Abs(originalValue - splineValue)
132
133     fmt.Printf("Значение оригинальной функции в точке %.10f: %.10f\n", point,
        originalValue)
134     fmt.Printf("Значение сплайна в точке %.10f: %.10f\n", point, splineValue)
135     fmt.Printf("Погрешность в точке %.10f: %.10f\n", point, errAbs)
136 }
```

Листинг 2: Сплайн-интерполяция

4 Результаты

Для заданных узлов интерполяции (x_i, y_i) построен кубический сплайн с коэффициентами, представленными в таблице 1.

Значения функции и результаты интерполяции в точках x_i^* представлены в таблице 2.

5 Вывод

В рамках лабораторной работы был исследован метод аппроксимации функции с использованием кубической сплайн-интерполяции. На основе заданных узлов интерполяции был построен сплайн третьего порядка, а также вычислены значения функции в серединах интервалов между узлами. В результате тестирования было установлено, что значения функции и сплайна полностью совпадают в узлах интерполяции, что подтверждает корректность метода. Таким образом, метод кубической сплайн-интерполяции демонстрирует высокую точность в узлах, но требует учета возможных неточностей в промежуточных точках.

Таблица 1: Коэффициенты кубического сплайна

Интервал	a	b	c	d
[0.0000, 0.0982]	0.0000	2.0000	0.0000	-0.2499
[0.0982, 0.1963]	0.1961	1.9928	-0.0736	-0.2489
[0.1963, 0.2945]	0.3908	1.9711	-0.1469	-0.2469
[0.2945, 0.3927]	0.5827	1.9351	-0.2196	-0.2439
[0.3927, 0.4909]	0.7703	1.8850	-0.2915	-0.2399
[0.4909, 0.5890]	0.9523	1.8208	-0.3621	-0.2350
[0.5890, 0.6872]	1.1274	1.7429	-0.4314	-0.2291
[0.6872, 0.7854]	1.2941	1.6516	-0.4988	-0.2222
[0.7854, 0.8836]	1.4512	1.5472	-0.5643	-0.2145
[0.8836, 0.9817]	1.5975	1.4302	-0.6275	-0.2059
[0.9817, 1.0799]	1.7316	1.3010	-0.6881	-0.1964
[1.0799, 1.1781]	1.8526	1.1603	-0.7459	-0.1861
[1.1781, 1.2763]	1.9591	1.0084	-0.8007	-0.1750
[1.2763, 1.3744]	2.0502	0.8461	-0.8523	-0.1631
[1.3744, 1.4726]	2.1249	0.6741	-0.9003	-0.1506
[1.4726, 1.5708]	2.1823	0.4929	-0.9447	-0.1374
[1.5708, 1.6690]	2.2214	0.3035	-0.9851	-0.1235
[1.6690, 1.7671]	2.2416	0.1065	-1.0215	-0.1091
[1.7671, 1.8653]	2.2421	-0.0972	-1.0536	-0.0942
[1.8653, 1.9635]	2.2223	-0.3068	-1.0814	-0.0788
[1.9635, 2.0617]	2.1817	-0.5214	-1.1046	-0.0629
[2.0617, 2.1598]	2.1198	-0.7401	-1.1231	-0.0467
[2.1598, 2.2580]	2.0363	-0.9620	-1.1369	-0.0301
[2.2580, 2.3562]	1.9309	-1.1861	-1.1457	-0.0139
[2.3562, 2.4544]	1.8034	-1.4115	-1.1498	0.0052
[2.4544, 2.5525]	1.6537	-1.6371	-1.1483	0.0145
[2.5525, 2.6507]	1.4819	-1.8621	-1.1440	0.0608
[2.6507, 2.7489]	1.2881	-2.0850	-1.1261	-0.0309
[2.7489, 2.8471]	1.0726	-2.3070	-1.1352	0.3925
[2.8471, 2.9452]	0.8355	-2.5186	-1.0196	-1.1065
[2.9452, 3.0434]	0.5774	-2.7508	-1.3455	4.5684

Таблица 2: Значения функции и сплайна в точках x_i^*

i	x_i^*	$f(x_i^*)$	$S(x_i^*)$	Погрешность
0	-0.0490873852	-0.0981452020	-0.0981451946	0.0000000074
1	0.0490873852	0.0981452020	0.0981451946	0.0000000074
2	0.1472621556	0.2937262849	0.2937262626	0.0000000223
3	0.2454369261	0.4871822523	0.4871822153	0.0000000370
4	0.3436116965	0.6771058444	0.6771057928	0.0000000516
5	0.4417864669	0.8621039936	0.8621039275	0.0000000661
6	0.5399612373	1.0408034340	1.0408033537	0.0000000803
7	0.6381360078	1.2118562448	1.2118561506	0.0000000943
8	0.7363107782	1.3739453075	1.3739451996	0.0000001079
9	0.8344855486	1.5257896591	1.5257895379	0.0000001212
10	0.9326603190	1.6661497234	1.6661495894	0.0000001341
11	1.0308350895	1.7938324026	1.7938322561	0.0000001465
12	1.1290098599	1.9076960106	1.9076958522	0.0000001584
13	1.2271846303	2.0066550333	2.0066548636	0.0000001698
14	1.3253594007	2.0896846978	2.0896845172	0.0000001806
15	1.4235341712	2.1558253354	2.1558251446	0.0000001908
16	1.5217089416	2.2041865241	2.2041863238	0.0000002003
17	1.6198837120	2.2339509944	2.2339507852	0.0000002092
18	1.7180584824	2.2443782867	2.2443780694	0.0000002173
19	1.8162332529	2.2348081460	2.2348079212	0.0000002248
20	1.9144080233	2.2046636430	2.2046634122	0.0000002309
21	2.0125827937	2.1534540089	2.1534537700	0.0000002389
22	2.1107575641	2.0807771752	2.0807769395	0.0000002357
23	2.2089323346	1.9863220081	1.9863217375	0.0000002706
24	2.3071071050	1.8698702295	1.8698700705	0.0000001590
25	2.4052818754	1.7312980169	1.7312974264	0.0000005904
26	2.5034566458	1.5705772754	1.5705782844	0.0000010090
27	2.6016314163	1.3877765779	1.3877716111	0.0000049668
28	2.6998061867	1.1830617674	1.1830791003	0.0000173329
29	2.7979809571	0.9566962183	0.9566303294	0.0000658889
30	2.8961557275	0.7090407562	0.7092854614	0.0002447052
31	2.9943304980	0.4405532327	0.4396387955	0.0009144372