Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки:

Системное и Прикладное Программное Обеспечение

(09.03.04 Програмная инженерия)

Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №4

Вариант №10

Студент

Карташев Владимир Сергеевич,
группа P3215

Преподаватель / Практик

Малышева Татьяна Алексеевна

г. Санкт-Петербург, 2024 г.

Цель работы

Найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

Описание метода, расчетные формулы

Линейная аппроксимация

Рассмотрим в качестве эмпирической формулы линейную функцию:

$$\varphi(x, a, b) = ax + b$$

Сумма квадратов отклонений запишется следующим образом:

$$S = S(a, b) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} [\varphi(x_i) - y_i]^2 = \sum_{i=1}^{n} (ax_i + b - y_i)^2 \to min$$

Для нахождения a и b необходимо найти минимум функции $S(a, b)$.

Необходимое условие существования минимума для функции S:

$$\begin{cases} \dfrac{\partial S}{\partial a} = 0 \\ \dfrac{\partial S}{\partial b} = 0 \end{cases} \quad \text{или} \quad \begin{cases} 2\sum\limits_{i=1}^{n} (ax_i + b - y_i)x_i = 0 \\ 2\sum\limits_{i=1}^{n} (ax_i + b - y_i) = 0 \end{cases}$$

Упростим полученную систему:

$$\begin{cases} a\sum\limits_{i=1}^{n} x_i^2 + b\sum\limits_{i=1}^{n} x_i = \sum\limits_{i=1}^{n} x_i y_i \\ a\sum\limits_{i=1}^{n} x_i + bn = \sum\limits_{i=1}^{n} y_i \end{cases}$$

Линейная аппроксимация

Введем обозначения:

$SX = \sum_{i=1}^{n} x_i$, $SXX = \sum_{i=1}^{n} x_i^2$, $SY = \sum_{i=1}^{n} y_i$, $SXY = \sum_{i=1}^{n} x_i y_i$

Получим систему уравнений для нахождения параметров *a* и *b:*

$$\begin{cases} aSXX + bSX = SXY \\ aSX + bn = SY \end{cases},$$

из которой находим (правило Крамера):

$$\Delta = SXX \cdot n - SX \cdot SX$$
$$\Delta_1 = SXY \cdot n - SX \cdot SY$$
$$\Delta_2 = SXX \cdot SY - SX \cdot SXY$$
$$a = \frac{\Delta_1}{\Delta}, \quad b = \frac{\Delta_2}{\Delta}$$

КВАДРАТИЧНАЯ АППРОКСИМАЦИЯ

Рассмотрим в качестве эмпирической формулы квадратичную функцию:

$$\varphi(x, a_0, a_1, a_2) = a_0 + a_1 x + a_2 x^2$$

Сумма квадратов отклонений запишется следующим образом:

$$S = S(a_0, a_1, a_2) = \sum_{i=1}^{n} (a_0 + a_1 x_i + a_2 x_i^2 - y_i)^2 \rightarrow min$$

Приравниваем к нулю частные производные *S* по неизвестным параметрам, получаем систему линейных уравнений:

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = 2\sum_{i=1}^{n} a_2 x_i^2 + a_1 x_i + a_0 - y_i = 0 \\ \dfrac{\partial S}{\partial a_1} = 2\sum_{i=1}^{n} (a_2 x_i^2 + a_1 x_i + a_0 - y_i) x_i = 0 \\ \dfrac{\partial S}{\partial a_2} = 2\sum_{i=1}^{n} (a_2 x_i^2 + a_1 x_i + a_0 - y_i) x_i^2 = 0 \end{cases}$$

$$\begin{cases} a_0 n + a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i \\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + a_2 \sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} x_i y_i \\ a_0 \sum_{i=1}^{n} x_i^2 + a_1 \sum_{i=1}^{n} x_i^3 + a_2 \sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} x_i^2 y_i \end{cases}$$

Среднеквадратичное отклонение:

$$\delta = \sqrt{\frac{\sum_{i=1}^{n}(\varphi(x_i) - y_i)^2}{n}}$$

Аппроксимация функции

$y = \frac{18x}{x^4+10}$, $x \in [0;\ 4]$, $h = 0.4$

Таблица табулирования

X	0	0.4	0.8	1.2	1.6	2	2.4	2.8	3.2	3.6	4
Y	0	0.72	1.38	1.79	1.74	1.38	1	0.71	0.5	0.36	0.27

Линейная аппроксимация
SX = 0 + 0.4 + 0.8 + 1.2 + 1.6 + 2.0 + 2.4 + 2.8 + 3.2 + 3.6 + 4.0 = 22.0 SXX = 0**2 + 0.4**2 + 0.8**2 + 1.2**2 + 1.6**2 + 2.0**2 + 2.4**2 + 2.8**2 + 3.2**2 + 3.6**2 + 4.0**2 = 61.6 SY = 0 + 0.72 + 1.38 + 1.79 + 1.74 + 1.38 + 1.00 + 0.71 + 0.50 + 0.36 + 0.27 = 9.85 SXY = 0*0 + 0.72*0.4 + 1.38*0.8 + 1.79*1.6 + 1.74*2.0 + 1.38*2.4 + 1.00*2.8 + 0.71*3.2 + 0.50*3.6 + 0.36*3.6 + 0.27*4.0 = 20.29 { 61.6a + 22b = 20.29 { 22a + 11b = 9.85 a = 0.0335, b = 0.8284 P1 (x)=0.0335x + 0.8284

X	0	0.4	0.8	1.2	1.6	2.0	2.4	2.8	3.2	3.6	4
Y	0	0.72	1.38	1.79	1.74	1.38	1	0.71	0.5	0.36	0.27
P1(X)	0.828	0.842	0.855	0.867	0.882	0.895	0.909	0.922	0.936	0.949	0.962
Ei	0.828	0.124	-0.528	-0.92	-0.858	-0.489	-0.092	0.217	0.434	0.585	0.692

исследуемая функциональная зависимость может быть приближенно описана линейной моделью

$P_1(x) = 0.0335x + 0.8284$, т. к. $P_1(x_i) \approx y_i$, $\varepsilon_i \rightarrow min$

Квадратичная аппроксимация

$$\sum_{1}^{n} x_i = 22$$

$$\sum_{1}^{n} x_i^2 = 61.6$$

$$\sum_{1}^{n} x_i^3 = 193.6$$

$$\sum_{1}^{n} x_i^4 = 648.525$$

$$\sum_{1}^{n} y_i = 9.85$$

$$\sum_{1}^{n} x_i y_i = 20.296$$

$$\sum_{1}^{n} x_i^2 y_i = 39.046$$

{ 11a0 + 22a1 + 61.6a2 = 9.85
{ 22a0 + 61.6a1 + 193.6a2=20.296
{ 61.6a0+193.6a1 + 648.526a3 = 39.046

$a_0 = -1.193$

$a_1 = 3.402$

$a_2 = -0.842$

$P_2(x) = -0.842x^2 + 3.402x - 1.193$

X	0	0.4	0.8	1.2	1.6	2	2.4	2.8	3.2	3.6	4
Y	0	0.72	1.38	1.79	1.74	1.38	1	0.71	0.5	0.36	0.27
$P_2(x)$	-1.193	0.033	0.99	1.677	2.095	2.243	2.122	1.731	1.071	0.141	-1.058
E_i	-1.193	-0.685	-0.394	-0.112	0.355	0.858	1.121	1.026	0.569	-0.223	-1.329

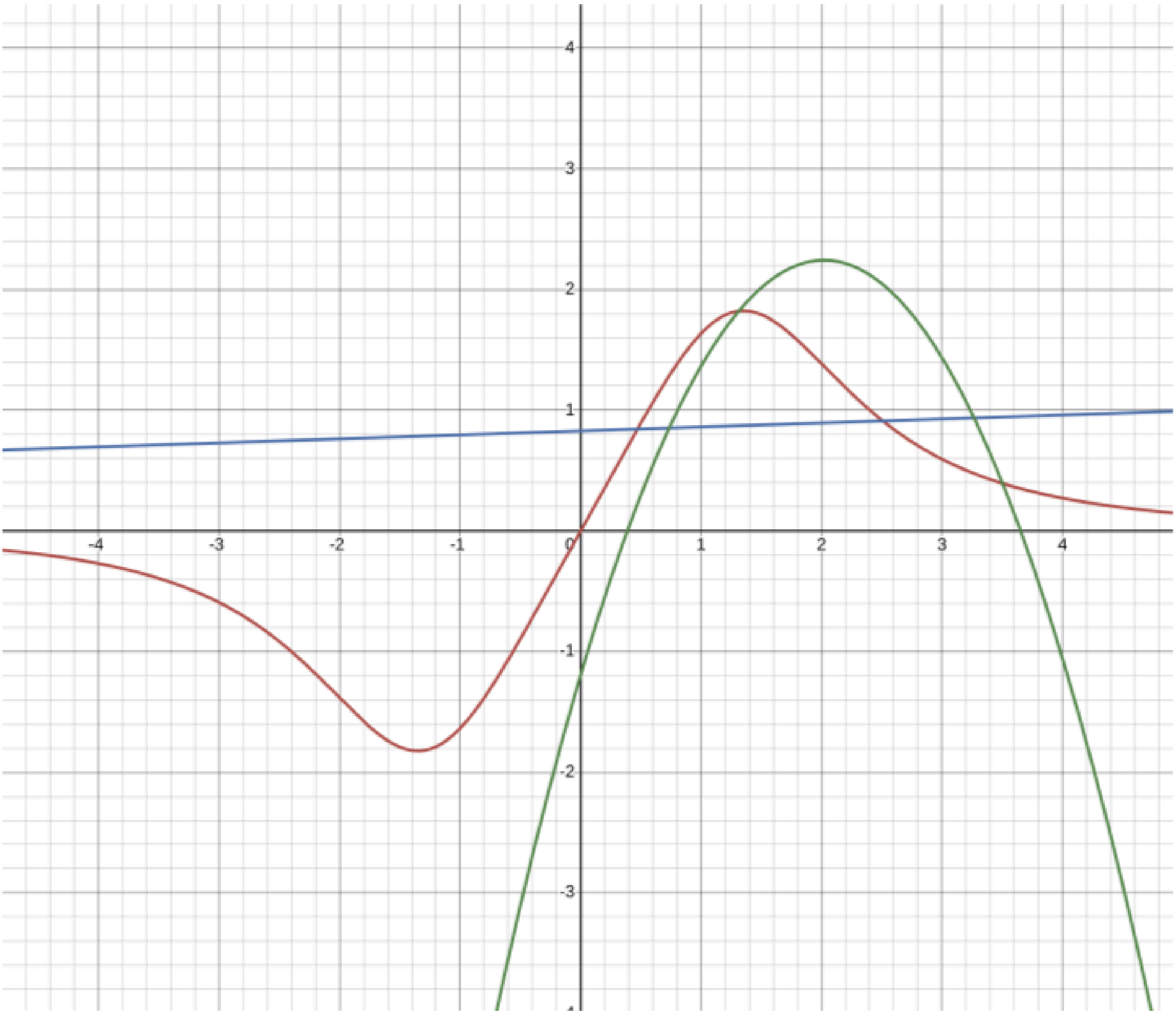
исследуемая функциональная зависимость может быть приближенно описана выбранной моделью, т.к.

P2 (xi) ≈ Yi , εi →min

Среднеквадратичные отклонения
Для линейной аппроксимации: δ = 0.593

Для квадратичной аппроксимации: δ = 0.819
У лин. аппроксимации значение меньше, значит, этот метод оказался точнее.

График



Листинг программы

Экспоненциальная функция:

```
func solveSystem(sumX, sumXX, sumLnY, sumXLnY, n float64) (float64, float64) {
   return utils.Rounding(math.Exp((sumLnY*sumXX - sumXLnY*sumX) / (n*sumXX -
math.Pow(sumX, 2)))),
      utils.Rounding((n*sumXLnY - sumLnY*sumX) / (n*sumXX - math.Pow(sumX, 2)))
}

func p(a, b float64, xValues []float64) []float64 {
   pValues := make([]float64, len(xValues))
   for i := 0; i < len(xValues); i++ {
      pValues[i] = utils.Rounding(a * math.Pow(math.E, b*xValues[i]))
   }
   return pValues
}

func ExponentialFunc(xValues, yValues []float64) (float64, []float64) {
   fmt.Println()
   fmt.Println()
   fmt.Println("Аппроксимация с помощью экспоненциальной функции:")
   if err := utils.Verify(xValues); err != nil {
      fmt.Println(err)
      return 0, []float64{}
   }

   sumX := utils.SumPow(xValues, 1)
   sumXX := utils.SumPow(xValues, 2)
   sumLnY := utils.SumLnPow(yValues, 1)
   sumXLnY := utils.SumCoupleALnB(xValues, yValues)
   n := float64(len(xValues))

   a, b := solveSystem(sumX, sumXX, sumLnY, sumXLnY, n)

   pValues := p(a, b, xValues)
   if utils.CheckNaN(pValues) {
      fmt.Println("Ошибка рассчета - pValues")
      return 0, []float64{}
   }
   eValues := utils.Difference(pValues, yValues)
   if utils.CheckNaN(pValues) {
      fmt.Println("Ошибка рассчета - eValues")
      return 0, []float64{}
   }
   s := utils.SumPow(eValues, 2)
   stDev := utils.StandardDeviation(pValues, yValues)
   detCoefficient := utils.DeterminationCoefficient(pValues, yValues)

   fmt.Printf("ΣX: %g, ΣX2: %g, ΣLn(Y): %g, ΣXLn(Y): %g, N: %g \n", sumX, sumXX, sumLnY,
sumXLnY, n)
   //fmt.Println()
   fmt.Printf("a: %g, b: %g \n", a, b)
   //fmt.Println()
   fmt.Println("X:", xValues)
   fmt.Println("Y:", yValues)
   fmt.Println("P:", pValues)
   fmt.Println("E:", eValues)
   //fmt.Println()
   fmt.Println("Мера отклонения:", s)
   fmt.Println("Стандартное отклонение:", stDev)
   fmt.Println("Достоверность аппроксимации:", detCoefficient)
   fmt.Println(utils.DeterminationMessage(detCoefficient))

   path := "output/exponential.png"
   f := func(x float64) float64 {
      return a * math.Exp(b*x)
```

```
    }
    graph.Create(path, xValues, yValues, f, "exponential")
    fmt.Println("График построен:", path)

    return detCoefficient, []float64{a, b}
}
```

Линейная функция:

```
func solveSystem(sumX, sumXX, sumY, sumXY, n float64) (float64, float64) {
    return utils.Rounding((n*sumXY - sumX*sumY) / (n*sumXX - sumX*sumX)),
        utils.Rounding((sumXX*sumY - sumX*sumXY) / (n*sumXX - sumX*sumX))
}

func p(a, b float64, xValues []float64) []float64 {
    pValues := make([]float64, len(xValues))
    for i := 0; i < len(xValues); i++ {
        pValues[i] = utils.Rounding(a*xValues[i] + b)
    }
    return pValues
}

func correlationCoefficient(xValues, yValues []float64) float64 {
    averageX := utils.Average(xValues)
    averageY := utils.Average(yValues)

    var numerator float64 = 0
    for i := 0; i < len(xValues); i++ {
        numerator += (xValues[i] - averageX) * (yValues[i] - averageY)
    }

    var sumX float64 = 0
    for i := 0; i < len(xValues); i++ {
        sumX += math.Pow(xValues[i]-averageX, 2)
    }

    var sumY float64 = 0
    for i := 0; i < len(xValues); i++ {
        sumY += math.Pow(yValues[i]-averageY, 2)
    }

    var denominator float64 = math.Sqrt(sumX * sumY)
    return utils.Rounding(numerator / denominator)
}

func LinearFunc(xValues, yValues []float64) (float64, []float64) {
    fmt.Println()
    fmt.Println()
    fmt.Println("Аппроксимация с помощью линейной функции:")

    sumX := utils.SumPow(xValues, 1)
    sumXX := utils.SumPow(xValues, 2)
    sumY := utils.SumPow(yValues, 1)
    sumXY := utils.SumCouple(xValues, 1, yValues, 1)
    n := float64(len(xValues))

    a, b := solveSystem(sumX, sumXX, sumY, sumXY, n)

    pValues := p(a, b, xValues)
    if utils.CheckNaN(pValues) {
        fmt.Println("Ошибка рассчета - pValues")
        return 0, []float64{}
    }
    eValues := utils.Difference(pValues, yValues)
    if utils.CheckNaN(pValues) {
        fmt.Println("Ошибка рассчета - eValues")
        return 0, []float64{}
```

```
    }
    s := utils.SumPow(eValues, 2)
    stDev := utils.StandardDeviation(pValues, yValues)
    corrCoefficient := correlationCoefficient(xValues, yValues)
    detCoefficient := utils.DeterminationCoefficient(pValues, yValues)

    fmt.Printf("ΣX: %g, ΣXX: %g, ΣY: %g, ΣXY: %g, N: %g \n", sumX, sumXX, sumY, sumXY, n)
    //fmt.Println()
    fmt.Printf("a: %g, b: %g \n", a, b)
    //fmt.Println()
    fmt.Println("X:", xValues)
    fmt.Println("Y:", yValues)
    fmt.Println("P:", pValues)
    fmt.Println("E:", eValues)
    //fmt.Println()
    fmt.Println("Мера отклонения:", s)
    fmt.Println("Стандартное отклонение:", stDev)
    fmt.Println("Коэффициент корреляции Пирсона:", corrCoefficient)
    fmt.Println("Достоверность аппроксимации:", detCoefficient)
    fmt.Println(utils.DeterminationMessage(detCoefficient))

    path := "output/linear.png"
    f := func(x float64) float64 {
        return a*x + b
    }
    graph.Create(path, xValues, yValues, f, "linear")
    fmt.Println("График построен:", path)

    return detCoefficient, []float64{a, b}
}
```

Логарифмическая функция:

```
func solveSystem(sumLnX, sumLnXLnX, sumY, sumYLnX, n float64) (float64, float64) {
    return utils.Rounding((sumY*sumLnXLnX - sumYLnX*sumLnX) / (n*sumLnXLnX -
math.Pow(sumLnX, 2))),
        utils.Rounding((n*sumYLnX - sumY*sumLnX) / (n*sumLnXLnX - math.Pow(sumLnX, 2)))
}

func p(a, b float64, xValues []float64) []float64 {
    pValues := make([]float64, len(xValues))
    for i := 0; i < len(xValues); i++ {
        pValues[i] = utils.Rounding(a + b*math.Log(xValues[i]))
    }
    return pValues
}

func LogarithmicFunc(xValues, yValues []float64) (float64, []float64) {
    fmt.Println()
    fmt.Println()
    fmt.Println("Аппроксимация с помощью логарифмической функции:")

    if err := utils.Verify(xValues); err != nil {
        fmt.Println(err)
        return 0, []float64{}
    }

    sumLnX := utils.SumLnPow(xValues, 1)
    sumLnXLnX := utils.SumLnPow(xValues, 2)
    sumY := utils.SumPow(yValues, 1)
    sumYLnX := utils.SumCoupleALnB(yValues, xValues)
    n := float64(len(xValues))

    a, b := solveSystem(sumLnX, sumLnXLnX, sumY, sumYLnX, n)

    pValues := p(a, b, xValues)
    if utils.CheckNaN(pValues) {
```

```go
        fmt.Println("Ошибка рассчета - pValues")
        return 0, []float64{}
    }
    eValues := utils.Difference(pValues, yValues)
    if utils.CheckNaN(pValues) {
        fmt.Println("Ошибка рассчета - eValues")
        return 0, []float64{}
    }

    s := utils.SumPow(eValues, 2)
    stDev := utils.StandardDeviation(pValues, yValues)
    detCoefficient := utils.DeterminationCoefficient(pValues, yValues)

    fmt.Printf("ΣLn(X): %g, ΣLn(X)Ln(X): %g, ΣY: %g, ΣYLn(X): %g, N: %g \n", sumLnX,
sumLnXLnX, sumY, sumYLnX, n)
    //fmt.Println()
    fmt.Printf("a: %g, b: %g \n", a, b)
    //fmt.Println()
    fmt.Println("X:", xValues)
    fmt.Println("Y:", yValues)
    fmt.Println("P:", pValues)
    fmt.Println("E:", eValues)
    //fmt.Println()
    fmt.Println("Мера отклонения:", s)
    fmt.Println("Стандартное отклонение:", stDev)
    fmt.Println("Достоверность аппроксимации:", detCoefficient)
    fmt.Println(utils.DeterminationMessage(detCoefficient))

    path := "output/logarithmic.png"
    f := func(x float64) float64 {
        return a*math.Log(x) + b
    }
    graph.Create(path, xValues, yValues, f, "logarithmic")
    fmt.Println("График построен:", path)

    return detCoefficient, []float64{a, b}
}
```

Полиномиальная функция 2-й степени:

```go
func solveSystem(sumX, sumX2, sumX3, sumX4, sumY, sumXY, sumX2Y, n float64) (float64,
float64, float64) {
    // Формирование матрицы коэффициентов системы
    A := [][]float64{
        {float64(n), sumX, sumX2},
        {sumX, sumX2, sumX3},
        {sumX2, sumX3, sumX4},
    }

    // Формирование матрицы свободных членов
    B := []float64{sumY, sumXY, sumX2Y}

    // Решение системы уравнений
    coefficients := utils.GaussianElimination(A, B)

    // Возвращение коэффициентов
    return utils.Rounding(coefficients[0]),
        utils.Rounding(coefficients[1]),
        utils.Rounding(coefficients[2])
}

func p(a0, a1, a2 float64, xValues []float64) []float64 {
    pValues := make([]float64, len(xValues))
    for i := 0; i < len(xValues); i++ {
        pValues[i] = utils.Rounding(a0 + a1*xValues[i] + a2*math.Pow(xValues[i], 2))
    }
    return pValues
```

```
}

func PolynomialFunction2ndDegree(xValues, yValues []float64) (float64, []float64) {
   fmt.Println()
   fmt.Println()
   fmt.Println("Аппроксимация с помощью полиномиальной функции 2-й степени:")

   sumX := utils.SumPow(xValues, 1)
   sumX2 := utils.SumPow(xValues, 2)
   sumX3 := utils.SumPow(xValues, 3)
   sumX4 := utils.SumPow(xValues, 4)
   sumY := utils.SumPow(yValues, 1)
   sumXY := utils.SumCouple(xValues, 1, yValues, 1)
   sumX2Y := utils.SumCouple(xValues, 2, yValues, 1)
   n := float64(len(xValues))

   a0, a1, a2 := solveSystem(sumX, sumX2, sumX3, sumX4, sumY, sumXY, sumX2Y, n)

   pValues := p(a0, a1, a2, xValues)
   if utils.CheckNaN(pValues) {
      fmt.Println("Ошибка рассчета - pValues")
      return 0, []float64{}
   }
   eValues := utils.Difference(pValues, yValues)
   if utils.CheckNaN(pValues) {
      fmt.Println("Ошибка рассчета - eValues")
      return 0, []float64{}
   }
   s := utils.SumPow(eValues, 2)
   stDev := utils.StandardDeviation(pValues, yValues)
   detCoefficient := utils.DeterminationCoefficient(pValues, yValues)

   fmt.Printf("ΣX: %g, ΣX2: %g, ΣX3: %g, ΣX4: %g \n", sumX, sumX2, sumX3, sumX4)
   fmt.Printf("ΣY: %g, ΣXY: %g, ΣX2Y: %g, N: %g \n", sumY, sumXY, sumX2Y, n)
   //fmt.Println()
   fmt.Printf("a0: %g, a1: %g, a2: %g \n", a0, a1, a2)
   //fmt.Println()
   fmt.Println("X:", xValues)
   fmt.Println("Y:", yValues)
   fmt.Println("P:", pValues)
   fmt.Println("E:", eValues)
   //fmt.Println()
   fmt.Println("Мера отклонения:", s)
   fmt.Println("Стандартное отклонение:", stDev)
   fmt.Println("Достоверность аппроксимации:", detCoefficient)
   fmt.Println(utils.DeterminationMessage(detCoefficient))

   path := "output/p_f_2_d.png"
   f := func(x float64) float64 {
      return a0 + a1*x + a2*x*x
   }
   graph.Create(path, xValues, yValues, f, "p_f_2_d")
   fmt.Println("График построен:", path)

   return detCoefficient, []float64{a0, a1, a2}
}
```

Полиномиальная функция 3-й степени:

```
func solveSystem(sumX, sumX2, sumX3, sumX4, sumX5, sumX6, sumY, sumXY, sumX2Y, sumX3Y, n
float64) (float64, float64, float64, float64) {
   // Формирование матрицы коэффициентов системы
   A := [][]float64{
      {float64(n), sumX, sumX2, sumX3},
      {sumX, sumX2, sumX3, sumX4},
      {sumX2, sumX3, sumX4, sumX5},
      {sumX3, sumX4, sumX5, sumX6},
```

```go
	}

	// Формирование матрицы свободных членов
	B := []float64{sumY, sumXY, sumX2Y, sumX3Y}

	// Решение системы уравнений
	coefficients := utils.GaussianElimination(A, B)

	// Возвращение коэффициентов
	return utils.Rounding(coefficients[0]),
		utils.Rounding(coefficients[1]),
		utils.Rounding(coefficients[2]),
		utils.Rounding(coefficients[3])
}

func p(a0, a1, a2, a3 float64, xValues []float64) []float64 {
	pValues := make([]float64, len(xValues))
	for i := 0; i < len(xValues); i++ {
		pValues[i] = utils.Rounding(a0 + a1*xValues[i] + a2*math.Pow(xValues[i], 2) +
a3*math.Pow(xValues[i], 3))
	}
	return pValues
}

func PolynomialFunction3rdDegree(xValues, yValues []float64) (float64, []float64) {
	fmt.Println()
	fmt.Println()
	fmt.Println("Аппроксимация с помощью полиномиальной функции 3-й степени:")

	sumX := utils.SumPow(xValues, 1)
	sumX2 := utils.SumPow(xValues, 2)
	sumX3 := utils.SumPow(xValues, 3)
	sumX4 := utils.SumPow(xValues, 4)
	sumX5 := utils.SumPow(xValues, 5)
	sumX6 := utils.SumPow(xValues, 6)
	sumY := utils.SumPow(yValues, 1)
	sumXY := utils.SumCouple(xValues, 1, yValues, 1)
	sumX2Y := utils.SumCouple(xValues, 2, yValues, 1)
	sumX3Y := utils.SumCouple(xValues, 3, yValues, 1)
	n := float64(len(xValues))

	a0, a1, a2, a3 := solveSystem(sumX, sumX2, sumX3, sumX4, sumX5, sumX6, sumY, sumXY,
sumX2Y, sumX3Y, n)

	pValues := p(a0, a1, a2, a3, xValues)
	if utils.CheckNaN(pValues) {
		fmt.Println("Ошибка рассчета - pValues")
		return 0, []float64{}
	}
	eValues := utils.Difference(pValues, yValues)
	if utils.CheckNaN(pValues) {
		fmt.Println("Ошибка рассчета - eValues")
		return 0, []float64{}
	}
	s := utils.SumPow(eValues, 2)
	stDev := utils.StandardDeviation(pValues, yValues)
	detCoefficient := utils.DeterminationCoefficient(pValues, yValues)

	fmt.Printf("ΣX: %g, ΣX2: %g, ΣX3: %g, ΣX4: %g, ΣX5: %g, ΣX6: %g \n", sumX, sumX2,
sumX3, sumX4, sumX5, sumX6)
	fmt.Printf("ΣY: %g, ΣXY: %g, ΣX2Y: %g, ΣX3Y: %g, N: %g \n", sumY, sumXY, sumX2Y,
sumX3Y, n)
	//fmt.Println()
	fmt.Printf("a0: %g, a1: %g, a2: %g, a3: %g \n", a0, a1, a2, a3)
	//fmt.Println()
	fmt.Println("X:", xValues)
	fmt.Println("Y:", yValues)
	fmt.Println("P:", pValues)
```

```
    fmt.Println("E:", eValues)
    //fmt.Println()
    fmt.Println("Мера отклонения:", s)
    fmt.Println("Стандартное отклонение:", stDev)
    fmt.Println("Достоверность аппроксимации:", detCoefficient)
    fmt.Println(utils.DeterminationMessage(detCoefficient))

    path := "output/p_f_3_d.png"
    f := func(x float64) float64 {
        return a0 + a1*x + a2*x*x + a3*x*x*x
    }
    graph.Create(path, xValues, yValues, f, "p_f_2_d")
    fmt.Println("График построен:", path)

    return detCoefficient, []float64{a0, a1, a2, a3}
}
```

Степенная функция:

```
func solveSystem(sumLnX, sumLnY, sumLnXY, sumLnXX, n float64) (float64, float64) {
    return utils.Rounding(math.Exp((sumLnY*sumLnXX - sumLnXY*sumLnX) / (n*sumLnXX -
sumLnX*sumLnX))),
        utils.Rounding((n*sumLnXY - sumLnX*sumLnY) / (n*sumLnXX - sumLnX*sumLnX))
}

func p(a, b float64, xValues []float64) []float64 {
    pValues := make([]float64, len(xValues))
    for i := 0; i < len(xValues); i++ {
        pValues[i] = utils.Rounding(a * math.Pow(xValues[i], b))
    }
    return pValues
}

func PowerFunc(xValues, yValues []float64) (float64, []float64) {
    fmt.Println()
    fmt.Println()
    fmt.Println("Аппроксимация с помощью степенной функции:")
    if err := utils.Verify(xValues); err != nil {
        fmt.Println(err)
        return 0, []float64{}
    }

    sumLnX := utils.SumLnPow(xValues, 1)
    sumLnXX := utils.SumLnPow(xValues, 2)
    sumLnY := utils.SumLnPow(yValues, 1)
    sumLnXY := utils.SumCoupleLn(xValues, yValues)
    n := float64(len(xValues))

    a, b := solveSystem(sumLnX, sumLnY, sumLnXY, sumLnXX, n)

    pValues := p(a, b, xValues)
    if utils.CheckNaN(pValues) {
        fmt.Println("Ошибка рассчета - pValues")
        return 0, []float64{}
    }
    eValues := utils.Difference(pValues, yValues)
    if utils.CheckNaN(pValues) {
        fmt.Println("Ошибка рассчета - eValues")
        return 0, []float64{}
    }
    s := utils.SumPow(eValues, 2)
    stDev := utils.StandardDeviation(pValues, yValues)
    detCoefficient := utils.DeterminationCoefficient(pValues, yValues)

    fmt.Printf("ΣLn(X): %g, ΣLn(Y): %g, ΣLn(XY): %g, ΣLn(XX): %g, N: %g \n", sumLnX,
sumLnY, sumLnXY, sumLnXX, n)
    //fmt.Println()
```

```go
	fmt.Printf("a: %g, b: %g \n", a, b)
	//fmt.Println()
	fmt.Println("X:", xValues)
	fmt.Println("Y:", yValues)
	fmt.Println("P:", pValues)
	fmt.Println("E:", eValues)
	//fmt.Println()
	fmt.Println("Мера отклонения:", s)
	fmt.Println("Стандартное отклонение:", stDev)
	fmt.Println("Достоверность аппроксимации:", detCoefficient)
	fmt.Println(utils.DeterminationMessage(detCoefficient))

	path := "output/pow.png"
	f := func(x float64) float64 {
		return a * math.Pow(x, b)
	}
	graph.Create(path, xValues, yValues, f, "pow")
	fmt.Println("График построен:", path)

	return detCoefficient, []float64{a, b}
}
```

Примеры и результаты работы программы

Пример входных данных:

файл с названием input

содержимое файла input:

```
X 1.1 2.3 3.7 4.5 5.4 6.8 7.5
Y 3.5 4.1 5.2 6.9 8.3 14.8 21.2
```

Работа программы:

```
Аппроксимация с помощью линейной функции:
ΣX: 31.3, ΣXX: 172.09, ΣY: 64, ΣXY: 368.03, N: 7
a: 2.5474, b: -2.2476
X: [1.1 2.3 3.7 4.5 5.4 6.8 7.5]
Y: [3.5 4.1 5.2 6.9 8.3 14.8 21.2]
P: [0.5545 3.6114 7.1778 9.2157 11.5084 15.0747 16.8579]
E: [-2.9455 -0.4886 1.9778 2.3157 3.2084 0.2747 -4.3421]
Мера отклонения: 47.412
Стандартное отклонение: 2.602525767678116
Коэффициент корреляции Пирсона: 0.9026
Достоверность аппроксимации: 0.8147514578521735
Удовлетворительная точность аппроксимации
График построен: output/linear.png


Аппроксимация с помощью полиномиальной функции 2-й степени:
ΣX: 31.3, ΣX2: 172.09, ΣX3: 1049.047, ΣX4: 6779.4325
ΣY: 64, ΣXY: 368.03, ΣX2Y: 2355.717, N: 7
a0: 6.3654, a1: -2.6867, a2: 0.6016
X: [1.1 2.3 3.7 4.5 5.4 6.8 7.5]
Y: [3.5 4.1 5.2 6.9 8.3 14.8 21.2]
P: [4.138 3.3685 4.6605 6.4577 9.3999 15.9138 20.0552]
E: [0.638 -0.7315 -0.5395 -0.4423 1.0999 1.1138 -1.1448]
Мера отклонения: 5.1897
Стандартное отклонение: 0.8610395112885354
Достоверность аппроксимации: 0.9797226650910713
Высокая точность аппроксимации
График построен: output/p_f_2_d.png


Аппроксимация с помощью полиномиальной функции 3-й степени:
```

ΣX: 31.3, ΣX2: 172.09, ΣX3: 1049.047, ΣX4: 6779.4325, ΣX5: 45466.1494, ΣX6: 312660.209
ΣY: 64, ΣXY: 368.03, ΣX2Y: 2355.717, ΣX3Y: 15850.9961, N: 7
a0: 0.7752, a1: 3.3245, a2: -1.0398, a3: 0.1272
X: [1.1 2.3 3.7 4.5 5.4 6.8 7.5]
Y: [3.5 4.1 5.2 6.9 8.3 14.8 21.2]
P: [3.3433 4.4687 5.284 6.2706 8.4364 15.2972 20.8827]
E: [-0.1567 0.3687 0.084 -0.6294 0.1364 0.4972 -0.3173]
Мера отклонения: 0.9302
Стандартное отклонение: 0.3645323873520313
Достоверность аппроксимации: 0.9963655651235488
Высокая точность аппроксимации
График построен: output/p_f_3_d.png

Аппроксимация с помощью экспоненциальной функции:
ΣX: 31.3, ΣX2: 172.09, ΣLn(Y): 14.108813856646814, ΣXLn(Y): 72.07144604352129, N: 7
a: 2.1497, b: 0.2796
X: [1.1 2.3 3.7 4.5 5.4 6.8 7.5]
Y: [3.5 4.1 5.2 6.9 8.3 14.8 21.2]
P: [2.9238 4.0894 6.0487 7.565 9.7296 14.3911 17.5022]
E: [-0.5762 -0.0106 0.8487 0.665 1.4296 -0.4089 -3.6978]
Мера отклонения: 17.3793
Стандартное отклонение: 1.5756774370772357
Достоверность аппроксимации: 0.9322112798400866
Удовлетворительная точность аппроксимации
График построен: output/exponential.png

Аппроксимация с помощью логарифмической функции:
ΣLn(X): 9.358854105460436, ΣLn(X)Ln(X): 15.255173133797939, ΣY: 64, ΣYLn(X): 106.0134877437097, N: 7
a: -0.8248, b: 7.4553
X: [1.1 2.3 3.7 4.5 5.4 6.8 7.5]
Y: [3.5 4.1 5.2 6.9 8.3 14.8 21.2]
P: [-0.1142 5.3848 8.9292 10.3885 11.7478 13.4664 14.1969]
E: [-3.6142 1.2848 3.7292 3.4885 3.4478 -1.3336 -7.0031]
Мера отклонения: 103.4989
Стандартное отклонение: 3.8451999944271895
Достоверность аппроксимации: 0.5956079691919786
Низкая точность аппроксимации

```
График построен: output/logarithmic.png



Аппроксимация с помощью степенной функции:
ΣLn(X): 9.358854105460436, ΣLn(Y): 14.108813856646814, ΣLn(XY):
21.24453593693386, ΣLn(XX): 15.25517313379739, N: 7
a: 2.3506, b: 0.8683
X: [1.1 2.3 3.7 4.5 5.4 6.8 7.5]
Y: [3.5 4.1 5.2 6.9 8.3 14.8 21.2]
P: [2.5534 4.8447 7.3206 8.6769 10.1652 12.4178 13.5206]
E: [-0.9466 0.7447 2.1206 1.7769 1.8652 -2.3822 -7.6794]
Мера отклонения: 77.232
Стандартное отклонение: 3.3216170730534245
Достоверность аппроксимации: 0.7016123597739155
Низкая точность аппроксимации
График построен: output/pow.png



---
Наилучшая аппроксимирующая функция:
Полиномиальная функция 3-й степени: 0.7752 + 3.3245*x +
-1.0398*x^2 + 0.1272*x^3
Достоверность аппроксимации: 0.9963655651235488
```

Выводы

В результате выполнения данной лабораторной работы были изучены методы для нахождения аппроксимирующих функций, приближающим функцию, заданную множеством её точек.