计算方法实验报告

姓名：吕弋卿

学号：210110315

院系：计算机科学与技术学院

专业：计算机类

班级：21 级 3 班

实验报告一

第一部分：问题分析 *（描述并总结出实验题目）*

问题 1：在函数与插值区间相同的前提下，改变插值次数，分别计算几个插值点的函数值，以此探究插值次数是否越大越好。

问题 2：在函数与插值次数与问题 1 相同的前提下，改变插值区间长度，分别计算几个插值点的函数值，以此探究插值区间是否越小越好。

问题 4：通过对于同样的函数，设置不同组别的拉格朗日的插值多项式，分别计算几个点处的函数估计值，而对于同一个点，在不同组别拉格朗日插值多项式中经历了内插和外推的变化，通过比较同一点在内插和外推时的函数估计值来探究拉格朗日插值问题中，内插是否比外推更可靠。

第二部分：数学原理

给定平面上$n+1$个不同的数据点$(x_k, f(x_k))$; $k=0,1,\cdots,n$; $x_i \neq x_j, i \neq j$;

则满足条件：$P_n(x_k) = f(x_k), \quad k=0,1,\cdots,n$

的n次拉格朗日插值多项式$P_n(x) = \sum_{k=0}^{n} f(x_k) l_k(x)$是存在且唯一的。

若$x_k \in [a,b], k=0,1,\cdots,n$, 且$f(x)$充分光滑, 则当$x_k \in [a,b]$时, 有误差估计式：

$$f(x) - P_n(x) = \frac{f^{n+1}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i), \quad \xi \in [a,b]$$

第三部分：程序设计流程

采用 java 语言设计，将项目文件夹导入 Intellij IDEA 运行即可。

主实现类：

```java
/**
 * @author Kosmischer
 * */
public class Calculator {
   private final double[] x;
   private final double[] xi;
   private final int steps;
   private final int bound;

   public Calculator(double[] x, double[] xi, int steps, int
bound){
       this.x = x;
       this.xi = xi;
       this.steps = steps;
       this.bound = bound;
   }

   public void calculate(int type){
       if(type == 1){
          System.out.println("\n插值区间为[" + (-1*bound) + ","
+ bound +  "], n = " + (steps-1) + ", f(x) = 1/(1+x^2)");
       }
       else if(type == 2){
          System.out.println("\n插值区间为[" + (-1*bound) + ","
+ bound +  "], n = " + (steps-1) + ", f(x) = e^x");
       }
       for (double v : x) {
          double y = 0.0;
          for (int k = 0; k < steps; k++) {
             double l = 1.0;
             for (int j = 0; j < steps; j++) {
                if (j != k) {
                   l = l * (v - xi[j]) / (xi[k] - xi[j]);

                }
             }
             if (type == 1) {
                y = y + l * f(xi[k]);
```

```
            } else if (type == 2) {
                y = y + l * g(xi[k]);
            } else if (type == 3){
                y = y + l * h(xi[k]);
            }

        }
        System.out.printf("x = %8.4f, y = %8.4f\n", v, y);
    }
}

public double f(double x){
    return 1/(1+x*x);
}

public double g(double x){
    return Math.exp(x);
}

public double h(double x){
    return Math.sqrt(x);
}

}
```

测试类问题 1（1）

```
/**
 * @author Kosmischer
 * */
public class Question_1_1 {
    public static void main(String[] args) {
        double[] x = {0.75, 1.75, 2.75, 3.75, 4.75};
        double[] x5 = new double[6];
        for(int i=0; i<6; i++){
            x5[i] = -5+(10.0/5)*i;
        }
        Calculator calculator5 = new Calculator(x, x5, 6, 5);
        calculator5.calculate(1);

        double[] x10 = new double[11];
        for(int i=0; i<11; i++){
            x10[i] = -5+(10.0/10)*i;
        }
        Calculator calculator10 = new Calculator(x, x10, 11, 5);
        calculator10.calculate(1);

        double[] x20 = new double[21];
        for(int i=0; i<21; i++){
            x20[i] = -5+(10.0/20)*i;
        }
        Calculator calculator20 = new Calculator(x, x20, 21, 5);
        calculator20.calculate(1);

        System.out.println("\n 实际值为: ");
        for (double v : x) {
            System.out.printf("x = %6.4f, y = %6.4f\n", v,
calculator5.f(v));
        }
    }
}
```

测试类问题 1（2）

```
/**
 * @author Kosmischer
 * */
public class Question_1_2 {
    public static void main(String[] args) {
        double[] x = {-0.95, -0.05, 0.05, 0.95};
        double[] x5 = new double[6];
        for(int i=0; i<6; i++){
            x5[i] = -1+(2.0/5)*i;
        }
        Calculator calculator5 = new Calculator(x, x5, 6, 1);
        calculator5.calculate(2);

        double[] x10 = new double[11];
        for(int i=0; i<11; i++){
            x10[i] = -1+(2.0/10)*i;
        }
        Calculator calculator10 = new Calculator(x, x10, 11, 1);
        calculator10.calculate(2);

        double[] x20 = new double[21];
        for(int i=0; i<21; i++){
            x20[i] = -1+(2.0/20)*i;
        }
        Calculator calculator20 = new Calculator(x, x20, 21, 1);
        calculator20.calculate(2);

        System.out.println("\n实际值为: ");
        for (double v : x) {
            System.out.printf("x = %6.4f, y = %6.4f\n", v,
calculator5.f(v));
        }
    }
}
```

测试类问题 2（1）

```
/**
 * @author Kosmischer
 * */
public class Question_2_1 {
    public static void main(String[] args) {
        double[] x = {-0.95, -0.05, 0.05, 0.95};
        double[] x5 = new double[6];
        for(int i=0; i<6; i++){
            x5[i] = -1+(2.0/5)*i;
        }
        Calculator calculator5 = new Calculator(x, x5, 6, 1);
        calculator5.calculate(1);

        double[] x10 = new double[11];
        for(int i=0; i<11; i++){
            x10[i] = -1+(2.0/10)*i;
        }
        Calculator calculator10 = new Calculator(x, x10, 11, 1);
        calculator10.calculate(1);

        double[] x20 = new double[21];
        for(int i=0; i<21; i++){
            x20[i] = -1+(2.0/20)*i;
        }
        Calculator calculator20 = new Calculator(x, x20, 21, 1);
        calculator20.calculate(1);

        System.out.println("\n实际值为: ");
        for (double v : x) {
          System.out.printf("x = %6.4f, y = %6.4f\n", v,
calculator5.f(v));
        }
    }
}
```

测试类问题 2（2）

```
/**
 * @author Kosmischer
 * */
public class Question_2_2 {
    public static void main(String[] args) {
        double[] x = {-4.75, -0.25, 0.25, 4.75};
        double[] x5 = new double[6];
        for(int i=0; i<6; i++){
            x5[i] = -5+(10.0/5)*i;
        }
        Calculator calculator5 = new Calculator(x, x5, 6, 5);
        calculator5.calculate(2);

        double[] x10 = new double[11];
        for(int i=0; i<11; i++){
            x10[i] = -5+(10.0/10)*i;
        }
        Calculator calculator10 = new Calculator(x, x10, 11, 5);
        calculator10.calculate(2);

        double[] x20 = new double[21];
        for(int i=0; i<21; i++){
            x20[i] = -5+(10.0/20)*i;
        }
        Calculator calculator20 = new Calculator(x, x20, 21, 5);
        calculator20.calculate(2);

        System.out.println("\n实际值为: ");
        for (double v : x) {
            System.out.printf("x = %6.4f, y = %6.4f\n", v,
calculator5.g(v));
        }
    }
}
```

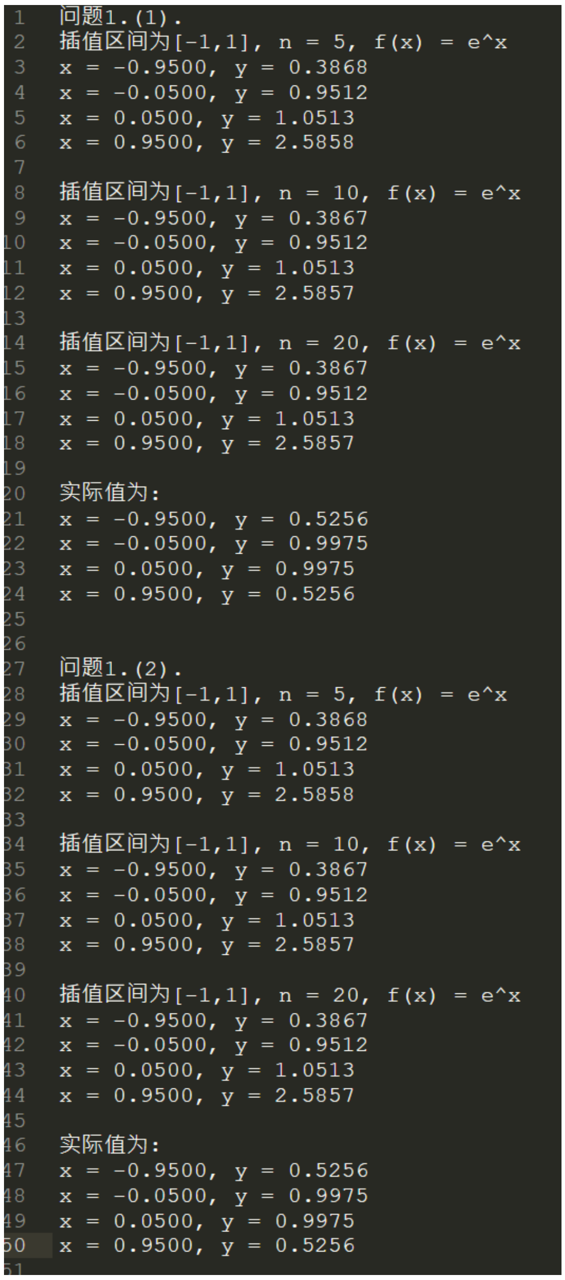
测试类问题 4

```
/**
 * @author Kosmischer
 * */
public class Question_4_1 {
   public static void main(String[] args) {

      double[] x = {5,50,115,185};
      double[] x1 = {1,4,9};
      Calculator calculator1 = new Calculator(x, x1, 3, 4);
      calculator1.calculate(3);

      double[] x2 = {36,49,64};
      Calculator calculator2 = new Calculator(x, x2, 3, 4);
      calculator2.calculate(3);

      double[] x3 = {100,121,144};
      Calculator calculator3 = new Calculator(x, x3, 3, 4);
      calculator3.calculate(3);

      double[] x4 = {169,196,225};
      Calculator calculator4 = new Calculator(x, x4, 3, 4);
      calculator4.calculate(3);

      System.out.println("\n实际值为: ");
      for (double v : x) {
         System.out.printf("x = %8.4f, y = %8.4f\n", v,
calculator1.h(v));
      }
   }
}
```

第四部分：实验结果、结论与讨论

问题 1：

控制台输出：

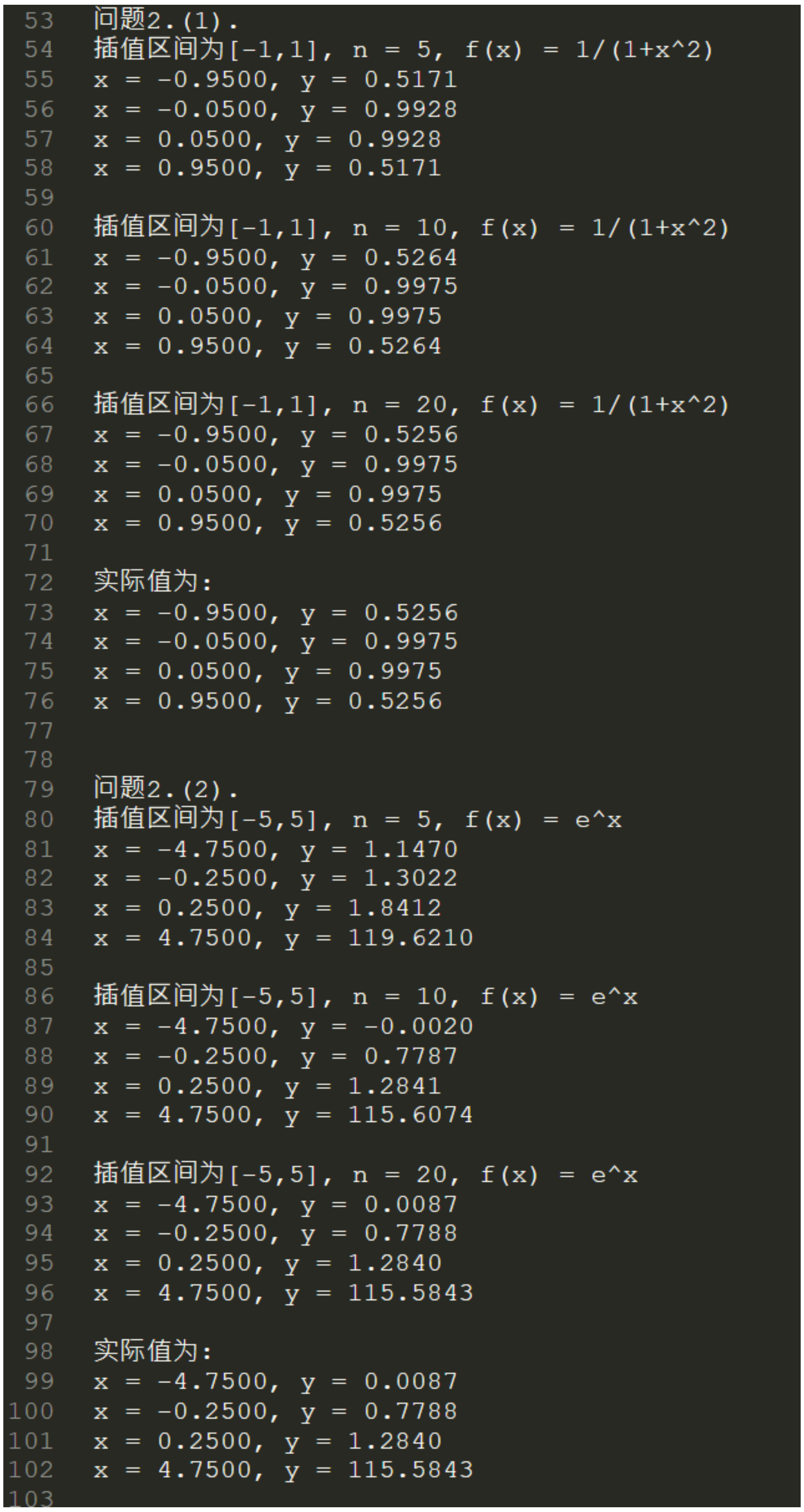


```
问题1.(1).
插值区间为[-1,1], n = 5, f(x) = e^x
x = -0.9500, y = 0.3868
x = -0.0500, y = 0.9512
x = 0.0500, y = 1.0513
x = 0.9500, y = 2.5858

插值区间为[-1,1], n = 10, f(x) = e^x
x = -0.9500, y = 0.3867
x = -0.0500, y = 0.9512
x = 0.0500, y = 1.0513
x = 0.9500, y = 2.5857

插值区间为[-1,1], n = 20, f(x) = e^x
x = -0.9500, y = 0.3867
x = -0.0500, y = 0.9512
x = 0.0500, y = 1.0513
x = 0.9500, y = 2.5857

实际值为:
x = -0.9500, y = 0.5256
x = -0.0500, y = 0.9975
x = 0.0500, y = 0.9975
x = 0.9500, y = 0.5256


问题1.(2).
插值区间为[-1,1], n = 5, f(x) = e^x
x = -0.9500, y = 0.3868
x = -0.0500, y = 0.9512
x = 0.0500, y = 1.0513
x = 0.9500, y = 2.5858

插值区间为[-1,1], n = 10, f(x) = e^x
x = -0.9500, y = 0.3867
x = -0.0500, y = 0.9512
x = 0.0500, y = 1.0513
x = 0.9500, y = 2.5857

插值区间为[-1,1], n = 20, f(x) = e^x
x = -0.9500, y = 0.3867
x = -0.0500, y = 0.9512
x = 0.0500, y = 1.0513
x = 0.9500, y = 2.5857

实际值为:
x = -0.9500, y = 0.5256
x = -0.0500, y = 0.9975
x = 0.0500, y = 0.9975
x = 0.9500, y = 0.5256
```

通过误差分析，插值多项式的次数 n 并不是越大越好，在 n=20 时，(1)中 3.75,4.75 处的误差明显较大。

通过误差分析可知，插值区间越小越好。

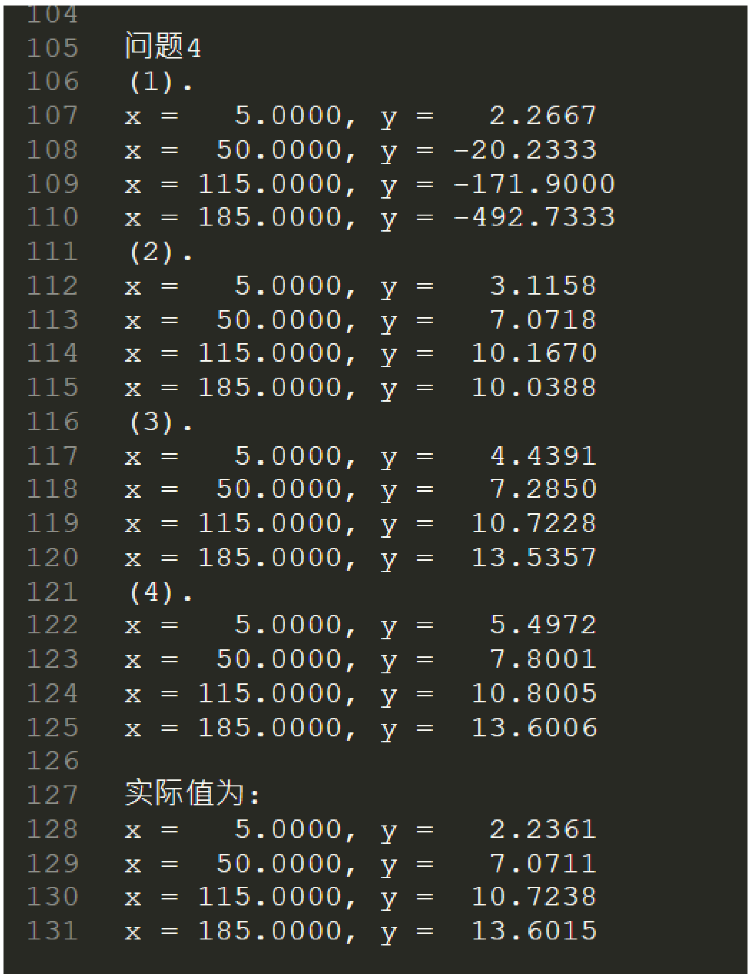
问题 2：



```
问题2.(1).
插值区间为[-1,1], n = 5, f(x) = 1/(1+x^2)
x = -0.9500, y = 0.5171
x = -0.0500, y = 0.9928
x = 0.0500, y = 0.9928
x = 0.9500, y = 0.5171

插值区间为[-1,1], n = 10, f(x) = 1/(1+x^2)
x = -0.9500, y = 0.5264
x = -0.0500, y = 0.9975
x = 0.0500, y = 0.9975
x = 0.9500, y = 0.5264

插值区间为[-1,1], n = 20, f(x) = 1/(1+x^2)
x = -0.9500, y = 0.5256
x = -0.0500, y = 0.9975
x = 0.0500, y = 0.9975
x = 0.9500, y = 0.5256

实际值为:
x = -0.9500, y = 0.5256
x = -0.0500, y = 0.9975
x = 0.0500, y = 0.9975
x = 0.9500, y = 0.5256


问题2.(2).
插值区间为[-5,5], n = 5, f(x) = e^x
x = -4.7500, y = 1.1470
x = -0.2500, y = 1.3022
x = 0.2500, y = 1.8412
x = 4.7500, y = 119.6210

插值区间为[-5,5], n = 10, f(x) = e^x
x = -4.7500, y = -0.0020
x = -0.2500, y = 0.7787
x = 0.2500, y = 1.2841
x = 4.7500, y = 115.6074

插值区间为[-5,5], n = 20, f(x) = e^x
x = -4.7500, y = 0.0087
x = -0.2500, y = 0.7788
x = 0.2500, y = 1.2840
x = 4.7500, y = 115.5843

实际值为:
x = -4.7500, y = 0.0087
x = -0.2500, y = 0.7788
x = 0.2500, y = 1.2840
x = 4.7500, y = 115.5843
```

通过误差分析可知，插值区间越小越好。

问题 4



```
问题4
(1).
x =    5.0000, y =    2.2667
x =   50.0000, y =  -20.2333
x =  115.0000, y = -171.9000
x =  185.0000, y = -492.7333
(2).
x =    5.0000, y =    3.1158
x =   50.0000, y =    7.0718
x =  115.0000, y =   10.1670
x =  185.0000, y =   10.0388
(3).
x =    5.0000, y =    4.4391
x =   50.0000, y =    7.2850
x =  115.0000, y =   10.7228
x =  185.0000, y =   13.5357
(4).
x =    5.0000, y =    5.4972
x =   50.0000, y =    7.8001
x =  115.0000, y =   10.8005
x =  185.0000, y =   13.6006

实际值为:
x =    5.0000, y =    2.2361
x =   50.0000, y =    7.0711
x =  115.0000, y =   10.7238
x =  185.0000, y =   13.6015
```

显然，内插比外推更可靠

思考题：

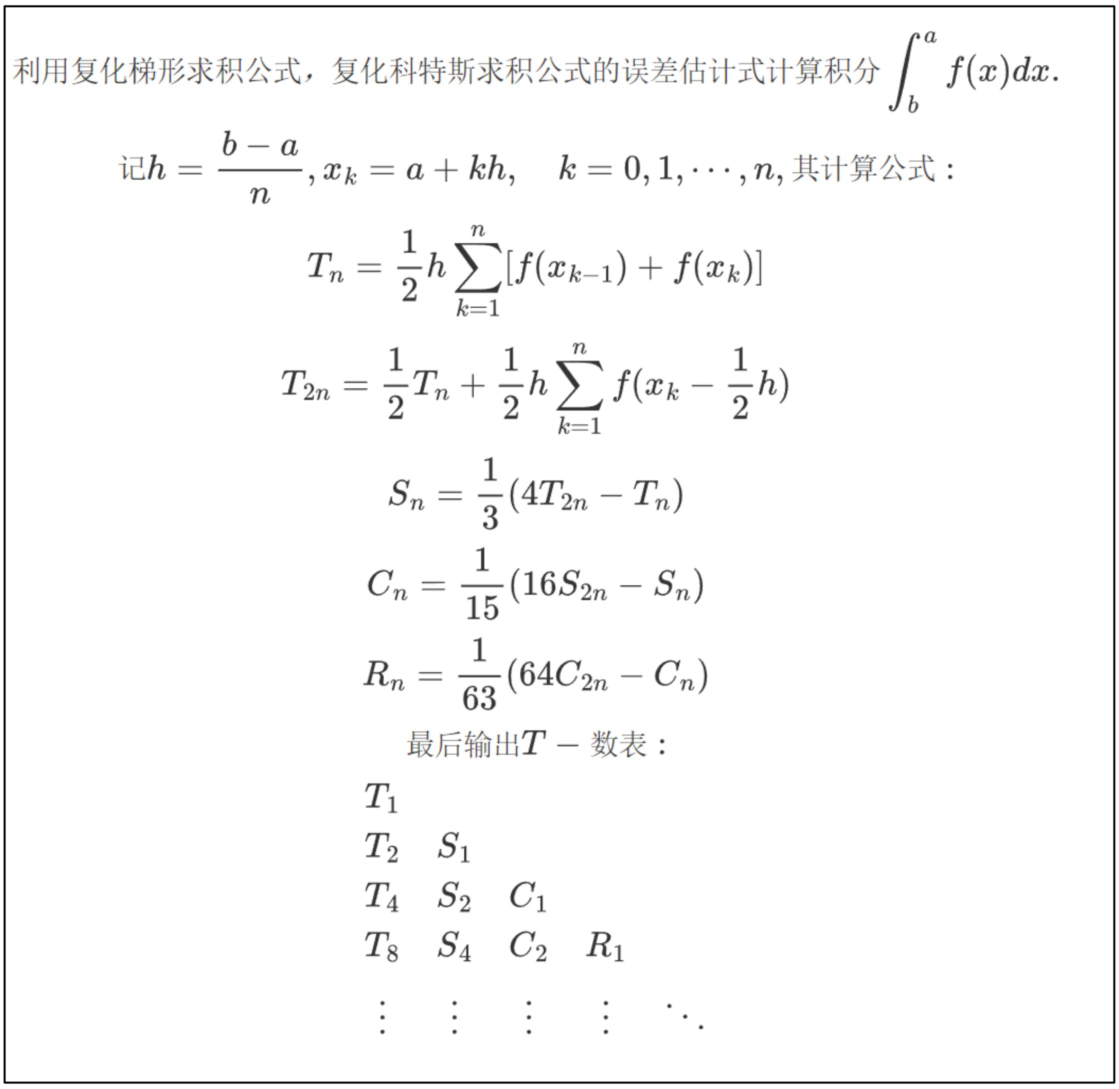
1. 可以调整插值区间，让目标点处于区间的中间位置。
2. 插值区间越小，对于误差估计式$\frac{f^{n+1}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)$，其中$\prod_{i=0}^{n}(x-x_i)$部分值越小，故误差越小。
4. 内插是目标点落在在插值区间内的情况，而外推法是估计插值区间外的函数值。内插法获得的估计值较外推法获得的估计值准确，因为在插值区间内插值多项式的拟合程度高，而在插值区间外拟合程度无法保证。

实验报告二

第一部分：问题分析 *（描述并总结出实验题目）*

实验题目要求利用 Romberg 积分法求四个不同函数的定积分，目的在于巩固学生对于龙贝格积分法的使用，熟悉龙贝格积分法的步骤，并观察二分次数与结果精度的关系。

第二部分：数学原理



利用复化梯形求积公式，复化科特斯求积公式的误差估计式计算积分 $\int_b^a f(x)dx.$

记$h = \dfrac{b-a}{n}, x_k = a + kh, \quad k = 0, 1, \cdots, n,$ 其计算公式：

$$T_n = \frac{1}{2}h\sum_{k=1}^{n}[f(x_{k-1}) + f(x_k)]$$

$$T_{2n} = \frac{1}{2}T_n + \frac{1}{2}h\sum_{k=1}^{n} f(x_k - \frac{1}{2}h)$$

$$S_n = \frac{1}{3}(4T_{2n} - T_n)$$

$$C_n = \frac{1}{15}(16S_{2n} - S_n)$$

$$R_n = \frac{1}{63}(64C_{2n} - C_n)$$

最后输出T − 数表：

$$\begin{array}{lllll}
T_1 & & & & \\
T_2 & S_1 & & & \\
T_4 & S_2 & C_1 & & \\
T_8 & S_4 & C_2 & R_1 & \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

第三部分：程序设计流程

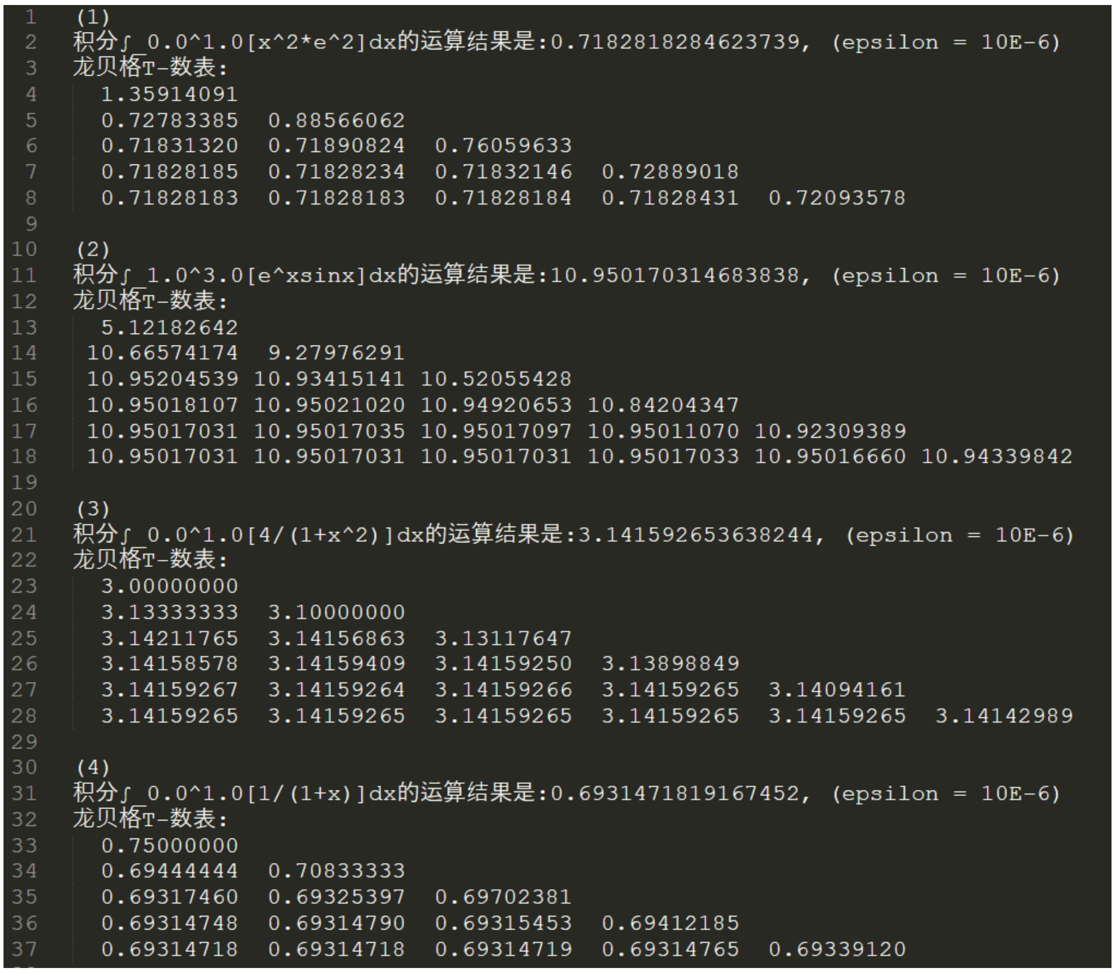
主要实现类

```
/**
 * @author Kosmischer
 * */
public class Calculator {
    private final double a;
    private final double b;
    private final double epsilon;


    public Calculator(double a, double b, double epsilon){
        this.a = a;
        this.b = b;
        this.epsilon = epsilon;
    }

    public void calculate(int serial){
        double[][] T;
        double h;
        double sigma;
        int i;
        int ii;
        boolean termination = false;

        T = new double[100][100];
        h = b - a;
        i = 1;
        T[0][0] = 0.5 * h *(f(a, serial) + f(b, serial));

        while(!termination){
            sigma = 0;
            ii = (int) Math.pow(2,i-1);
            for(int k=1; k<=ii; k++){
                sigma += f((a+(k-0.5)*h) , serial);
            }
            T[0][i] = 0.5 * T[0][i-1] + 0.5 * h * sigma;
            for(int m=1; m<=i; m++){
                int k = i - m;
                T[m][k] = ((Math.pow(4,m))*T[m-1][k+1]-T[m-1][k])
/ ((Math.pow(4,m))-1);
            }
            if(Math.abs(T[i][0] - T[i-1][0]) < epsilon){
                termination = true;
                output(serial,i,T[i][0],T);
                continue;
            }
            h = 0.5 * h;
            i++;
        }

    }

    public double f(double x, int type){
        switch (type){
            case 1: return x*x*Math.exp(x);
            case 2: return Math.exp(x)*Math.sin(x);
            case 3: return 4/(1+x*x);
            case 4: return 1/(1+x);
            default: return 0;
        }
    }

    public void output(int serial, int i, double res,
double[][] T){
        String prefix = "\n积分∫_"+a+"^"+b;
        String functionName="";
        switch (serial){
            case 1: functionName = "[x^2*e^2]dx"; break;
            case 2: functionName = "[e^xsinx]dx"; break;
            case 3: functionName = "[4/(1+x^2)]dx"; break;
            case 4: functionName = "[1/(1+x)]dx"; break;
            default: break;
        }
        String suffix = "的运算结果是:" + res + ", (epsilon = 10E-
6)";
        System.out.println(prefix+functionName+suffix);
        System.out.println("龙贝格 T-数表:");
        for(int index = 0; index <=i; index++){
            for(int iii=0; iii<=i; iii++){
                for(int jjj=0; jjj<=i; jjj++){
                    if(iii + jjj == index){
                        System.out.printf("%12.8f",T[jjj][iii]);
                    }
                }
            }
            System.out.println();
        }
        System.out.println();
    }

}
```

测试类：

```
/**
 * @author Kosmischer
 * */
public class Main {
    public static void main(String[] args) {
        Calculator calculator1 = new Calculator(0, 1, 1.0E-6);
        calculator1.calculate(1);

        Calculator calculator2 = new Calculator(1, 3, 1.0E-6);
        calculator2.calculate(2);

        Calculator calculator3 = new Calculator(0, 1, 1.0E-6);
        calculator3.calculate(3);

        Calculator calculator4 = new Calculator(0, 1, 1.0E-6);
        calculator4.calculate(4);
    }
}
```

第四部分：实验结果、结论与讨论



```
(1)
积分∫_0.0^1.0[x^2*e^2]dx的运算结果是:0.7182818284623739, (epsilon = 10E-6)
龙贝格T-数表:
  1.35914091
  0.72783385  0.88566062
  0.71831320  0.71890824  0.76059633
  0.71828185  0.71828234  0.71832146  0.72889018
  0.71828183  0.71828183  0.71828184  0.71828431  0.72093578

(2)
积分∫_1.0^3.0[e^xsinx]dx的运算结果是:10.950170314683838, (epsilon = 10E-6)
龙贝格T-数表:
  5.12182642
 10.66574174  9.27976291
 10.95204539 10.93415141 10.52055428
 10.95018107 10.95021020 10.94920653 10.84204347
 10.95017031 10.95017035 10.95017097 10.95011070 10.92309389
 10.95017031 10.95017031 10.95017031 10.95017033 10.95016660 10.94339842

(3)
积分∫_0.0^1.0[4/(1+x^2)]dx的运算结果是:3.141592653638244, (epsilon = 10E-6)
龙贝格T-数表:
  3.00000000
  3.13333333  3.10000000
  3.14211765  3.14156863  3.13117647
  3.14158578  3.14159409  3.14159250  3.13898849
  3.14159267  3.14159264  3.14159266  3.14159265  3.14094161
  3.14159265  3.14159265  3.14159265  3.14159265  3.14159265  3.14142989

(4)
积分∫_0.0^1.0[1/(1+x)]dx的运算结果是:0.6931471819167452, (epsilon = 10E-6)
龙贝格T-数表:
  0.75000000
  0.69444444  0.70833333
  0.69317460  0.69325397  0.69702381
  0.69314748  0.69314790  0.69315453  0.69412185
  0.69314718  0.69314718  0.69314719  0.69314765  0.69339120
```

上面是用龙贝格积分法求得的积分近似解。

思考题：

在实验中，二分次数越多，计算精度呈幂次趋势提高。

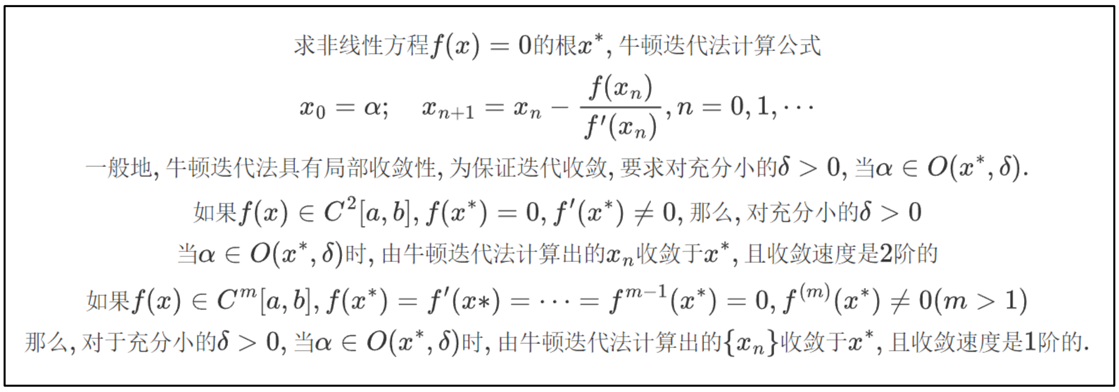
实验报告三

第一部分：问题分析 *（描述并总结出实验题目）*

实验题目要求利用 Newton 迭代法求四个不同方程的近似解，目的在于巩固学生对于 Newton 迭代法的使用，熟悉 Newton 迭代法的步骤。

题目 1 要求探究确定初值的原则，题目 2 探究对于理论上的同一解由于方程形式不同在计算过程中的精度问题。

第二部分：数学原理



求非线性方程$f(x)=0$的根x^*, 牛顿迭代法计算公式

$$x_0=\alpha;\quad x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}, n=0,1,\cdots$$

一般地, 牛顿迭代法具有局部收敛性, 为保证迭代收敛, 要求对充分小的$\delta>0$, 当$\alpha\in O(x^*,\delta)$.

如果$f(x)\in C^2[a,b], f(x^*)=0, f'(x^*)\neq 0$, 那么, 对充分小的$\delta>0$

当$\alpha\in O(x^*,\delta)$时, 由牛顿迭代法计算出的x_n收敛于x^*, 且收敛速度是2阶的

如果$f(x)\in C^m[a,b], f(x^*)=f'(x*)=\cdots=f^{m-1}(x^*)=0, f^{(m)}(x^*)\neq 0(m>1)$

那么, 对于充分小的$\delta>0$, 当$\alpha\in O(x^*,\delta)$时, 由牛顿迭代法计算出的$\{x_n\}$收敛于x^*, 且收敛速度是1阶的.

第三部分：程序设计流程

主实现类：

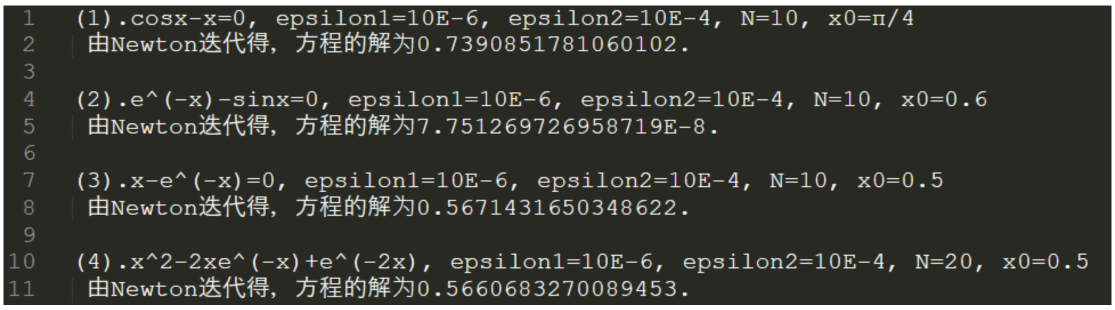
```
/**
 * @author Kosmischer
 * */
public class Calculator {
    private double alpha;
    private double epsilon1;
    private double epsilon2;
    private int N;

    public Calculator(double alpha, double epsilon1, double
epsilon2, int N){
        this.alpha = alpha;
        this.epsilon1 = epsilon1;
        this.epsilon2 = epsilon2;
        this.N = N;
    }

    public void calculate(int serial){
        double belta = 0;
        double F = 0;
        double DF = 0;
        double Tol = 0;
        int endFlag = 0;

        for(int n=1 ; n<N && endFlag==0 ; n++){
            F = f(alpha, serial);
            DF = df(alpha, serial);
            if(Math.abs(F) < epsilon1){
                endFlag = 1;
                output(serial, alpha);
                continue;
            }
            if(Math.abs(DF) < epsilon2){
                endFlag = 2;
                System.out.println("Iteration Failed !");
                continue;
            }
            belta = alpha - F/DF;
            Tol = Math.abs(belta-alpha);
            if(Tol < epsilon1){
                endFlag = 1;
                output(serial, belta);
                continue;
            }
            alpha = belta;
        }
    }

    public double f(double x, int serial){
        switch (serial){
            case 1: return Math.cos(x) - x;
            case 2: return Math.exp(-1*x) - Math.cos(x);
            case 3: return x - Math.exp(-1*x);
            case 4: return x*x - 2*x*Math.exp(-1*x) + Math.exp(-
2*x);
            default:return 0;
        }
    }

    public double df(double x, int serial){
        switch (serial){
            case 1: return -1*Math.sin(x) - 1;
            case 2: return -1*Math.exp(-1*x) - Math.sin(x);
            case 3: return 1 + Math.exp(-1*x);
            case 4: return 2*x - 2*Math.exp(-1*x) +
2*x*Math.exp(-1*x) - 2*Math.exp(-2*x);
            default:return 0;
        }
    }

    public void output(int serial, double res){
        switch (serial){
            case 1: System.out.println("\n(" + serial + ").cosx-
x=0, epsilon1=10E-6, epsilon2=10E-4, N=10, x0=π/4\n 由 Newton 迭
代得，方程的解为" + res + "."); break;
            case 2: System.out.println("\n(" + serial + ").e^(-
x)-sinx=0, epsilon1=10E-6, epsilon2=10E-4, N=10, x0=0.6\n 由
Newton 迭代得，方程的解为" + res + "."); break;
            case 3: System.out.println("\n(" + serial + ").x-
e^(-x)=0, epsilon1=10E-6, epsilon2=10E-4, N=10, x0=0.5\n 由
Newton 迭代得，方程的解为" + res + "."); break;
            case 4: System.out.println("\n(" + serial + ").x^2-
2xe^(-x)+e^(-2x), epsilon1=10E-6, epsilon2=10E-4, N=20,
x0=0.5\n 由 Newton 迭代得，方程的解为" + res + "."); break;
```

```
            default: break;
        }
    }
}
```

测试类：

```
/**
 * @author Kosmischer
 * */
public class Main {
    public static void main(String[] args) {

        Calculator calculator1 = new Calculator(Math.PI/4, 10E-
6, 10E-4, 10);
        Calculator calculator2 = new Calculator(0.6, 10E-6, 10E-
4, 10);
        Calculator calculator3 = new Calculator(0.5, 10E-6, 10E-
4, 10);
        Calculator calculator4 = new Calculator(0.5, 10E-6, 10E-
4, 20);

        calculator1.calculate(1);
        calculator2.calculate(2);
        calculator3.calculate(3);
        calculator4.calculate(4);
    }
}
```

第四部分：实验结果、结论与讨论



```
(1).cosx-x=0, epsilon1=10E-6, epsilon2=10E-4, N=10, x0=π/4
 由Newton迭代得, 方程的解为0.7390851781060102.

(2).e^(-x)-sinx=0, epsilon1=10E-6, epsilon2=10E-4, N=10, x0=0.6
 由Newton迭代得, 方程的解为7.751269726958719E-8.

(3).x-e^(-x)=0, epsilon1=10E-6, epsilon2=10E-4, N=10, x0=0.5
 由Newton迭代得, 方程的解为0.5671431650348622.

(4).x^2-2xe^(-x)+e^(-2x), epsilon1=10E-6, epsilon2=10E-4, N=20, x0=0.5
 由Newton迭代得, 方程的解为0.5660683270089453.
```

思考题：

1. 确定初值应当选取根附近的任意一点作为初值。实际计算中应当通过二分法等方法初步确定近似值，然后在附近取初值进行迭代计算。

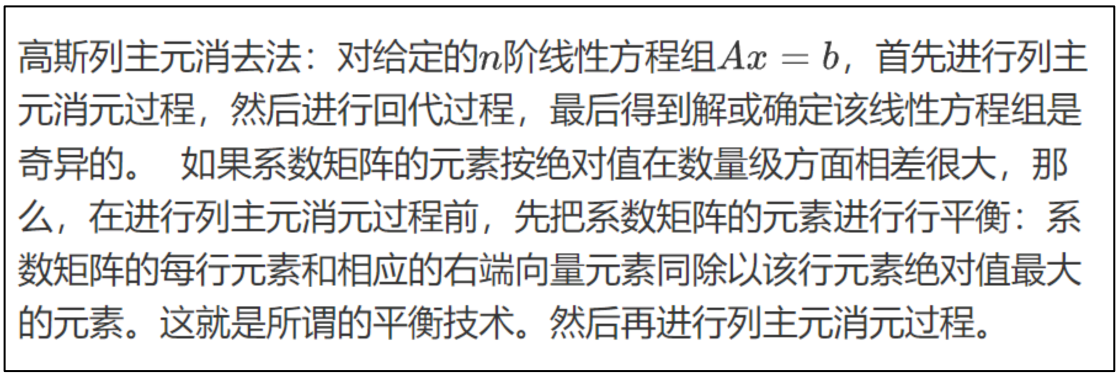
2. 实验 2 中两个方程理论解应当是一样的，而实际计算出来有所差异，且(1)中所得近似解精度比(2)中精度更高。这是因为(1)和(2)中的两个函数呈平方关系，而取平方运算能将小数级别的误差变得更小，所以当在程序中反映出同样误差的情况下，实际误差(2)中应当更大。

实验报告四

第一部分：问题分析 *（描述并总结出实验题目）*

实验题目要求利用高斯列主元消去法求四个线性方程组的近似解，目的在于巩固学生对于高斯列主元消去法的使用，熟悉高斯列主元消去法的步骤，体会高斯列主元消去法的思想。

第二部分：数学原理



高斯列主元消去法：对给定的n阶线性方程组$Ax=b$，首先进行列主元消元过程，然后进行回代过程，最后得到解或确定该线性方程组是奇异的。 如果系数矩阵的元素按绝对值在数量级方面相差很大，那么，在进行列主元消元过程前，先把系数矩阵的元素进行行平衡：系数矩阵的每行元素和相应的右端向量元素同除以该行元素绝对值最大的元素。这就是所谓的平衡技术。然后再进行列主元消元过程。

第三部分：程序设计流程

主实现类：

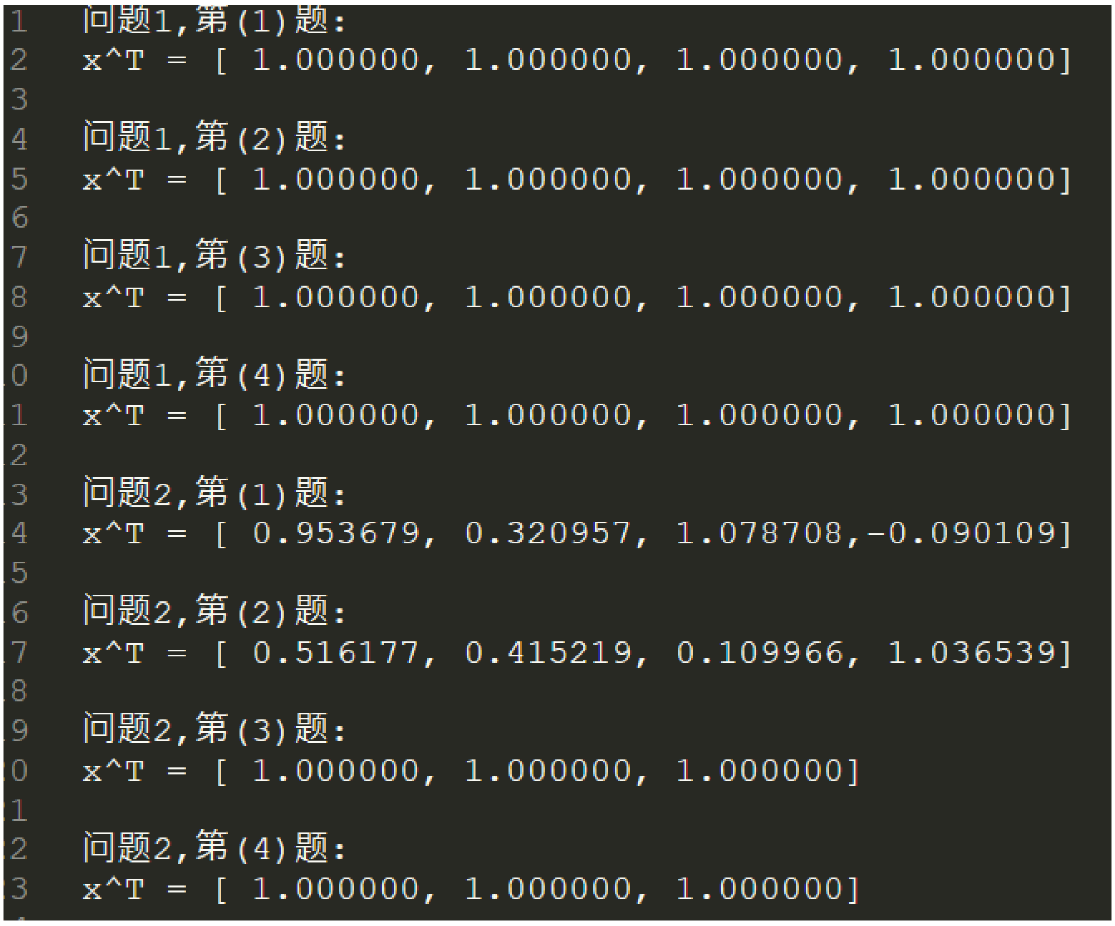
```
/**
 * @author Kosmischer
 **/
public class Calculator {

    private static int series = 0;
    private final int n;
    private final double[][] a;
    private final double[] b;

    public Calculator(int n, double[][] a, double[] b){
        this.n = n;
        this.a = a;
        this.b = b;
        series++;
    }

    public void calculate(){
        double[][] m = new double[n][n];
        for(int k=0; k<n-1; k++){
            int p = 0;
            double maxTemp = 0;
            for(int j=k; j<n; j++){
                if(Math.abs(a[j][k]) > maxTemp){
                    maxTemp = Math.abs(a[j][k]);
                }
            }
            for(p=k; p<n; p++){
                if(Math.abs(a[p][k])==maxTemp){
                    if(a[p][k]==0){
                        System.out.println("Singular Matrix !");
                        return;
                    }
                    if(p != k){
                        for(int ii=0; ii<n; ii++){
                            double temp = a[p][ii];
                            a[p][ii] = a[k][ii];
                            a[k][ii] = temp;
                        }
                        double temp = b[p];
                        b[p] = b[k];
                        b[k] = temp;
                    }
                    break;
                }
            }


            for(int i=k+1; i<n; i++){
                m[i][k] = a[i][k] / a[k][k];
                for(int j=k+1; j<n; j++){
                    a[i][j] = a[i][j] - a[k][j]*m[i][k];

                }
                b[i] = b[i] - b[k]*m[i][k];
            }
        }

        if(a[n-1][n-1] == 0){
            System.out.println("Singular Matrix");
            return;
        }

        double[] x = new double[n];
        x[n-1] = b[n-1]/a[n-1][n-1];

        for(int k=n-2; k>=0; k--){
            double Sigma = 0;
            for(int j=k+1; j<n; j++){

                Sigma += a[k][j]*x[j];
            }

            x[k] = (b[k]-Sigma) / a[k][k];
        }

        if(series < 5){
            System.out.println("问题 1,第("+series+")题:");
        }
        else {
            System.out.println("问题 2,第("+(series-4)+")题:");
        }
        System.out.print("x^T = [");
        for(int i=0; i<n; i++){
```

```
            System.out.printf("%9.6f,",x[i]);
        }
        System.out.println("]\n");

    }
}
```

测试类：

```
/**
 * @author Kosmischer
 * */
public class Main {
    public static void main(String[] args) {
        double[][] a1 = {{0.4096, 0.1234, 0.3678, 0.2943},
                        {0.2246, 0.3872, 0.4015, 0.1129},
                        {0.3645, 0.1920, 0.3781, 0.0643},
                        {0.1784, 0.4002, 0.2786, 0.3927}};
        double[]  b1 =  {1.1951, 1.1262, 0.9989, 1.2499};
        Calculator calculator1 = new Calculator(4,a1,b1);
        calculator1.calculate();

        double[][] a2 = {{136.01, 90.860, 0.0000, 0.0000},
                        {90.860, 98.810,-67.590, 0.0000},
                        {0.0000,-67.590, 132.01, 46.260},
                        {0.0000, 0.0000, 46.260, 177.17}};
        double[]  b2 =  {226.87, 122.08, 110.68, 223.43};
        Calculator calculator2 = new Calculator(4,a2,b2);
        calculator2.calculate();

        double[][] a3 = {{1.0/1, 1.0/2, 1.0/3, 1.0/4},
                        {1.0/2, 1.0/3, 1.0/4, 1.0/5},
                        {1.0/3, 1.0/4, 1.0/5, 1.0/6},
                        {1.0/4, 1.0/5, 1.0/6, 1.0/7}};
        double[]  b3 =  {25.0/12, 77.0/60, 57.0/60, 319.0/420};
        Calculator calculator3 = new Calculator(4,a3,b3);
        calculator3.calculate();

        double[][] a4 = {{10,  7,  8,  7},
                        { 7,  5,  6,  5},
                        { 8,  6,  10, 9},
                        { 7,  5,  9, 10}};
        double[]  b4 =  {32, 23, 33, 31};
        Calculator calculator4 = new Calculator(4,a4,b4);
        calculator4.calculate();

        double[][] a5 = {{ 197,  305, -206, -804},
                         {46.8, 71.3,-47.4, 52.0},
                         {88.6, 76.4,-10.8,  802},
                         {1.45, 5.90, 6.13, 36.5}};
        double[]   b5 =  {136, 11.7, 25.1, 6.60};
        Calculator calculator5 = new Calculator(4,a5,b5);
        calculator5.calculate();

        double[][] a6 = {{0.5398, 0.7161,-0.5554,-0.2982},
                         {0.5257, 0.6924, 0.3565,-0.6255},
                         {0.6465,-0.8187,-0.1872, 0.1291},
                         {0.5814, 0.9400,-0.7779,-0.4042}};
        double[]   b6 =  {0.2058, -0.0503, 0.1070, 0.1859};
        Calculator calculator6 = new Calculator(4,a6,b6);
        calculator6.calculate();

        double[][] a7 = {{10,  1,  2},
                         { 1, 10,  2},
                         { 1,  1,  5}};
        double[]   b7 =  {13, 13, 7};
        Calculator calculator7 = new Calculator(3,a7,b7);
        calculator7.calculate();

        double[][] a8 = {{ 4, -2, -4},
                         {-2, 17, 10},
                         {-4, 10,  9}};
        double[]   b8 =  {-2, 25, 15};
        Calculator calculator8 = new Calculator(3,a8,b8);
        calculator8.calculate();
    }
}
```

第四部分：实验结果、结论与讨论



```
问题1,第(1)题:
x^T = [ 1.000000, 1.000000, 1.000000, 1.000000]

问题1,第(2)题:
x^T = [ 1.000000, 1.000000, 1.000000, 1.000000]

问题1,第(3)题:
x^T = [ 1.000000, 1.000000, 1.000000, 1.000000]

问题1,第(4)题:
x^T = [ 1.000000, 1.000000, 1.000000, 1.000000]

问题2,第(1)题:
x^T = [ 0.953679, 0.320957, 1.078708,-0.090109]

问题2,第(2)题:
x^T = [ 0.516177, 0.415219, 0.109966, 1.036539]

问题2,第(3)题:
x^T = [ 1.000000, 1.000000, 1.000000]

问题2,第(4)题:
x^T = [ 1.000000, 1.000000, 1.000000]
```