计 算 方 法

实验一 Lagrange插值

|  |  |
| --- | --- |
| 姓名 | 郭茁宁 |
| 学号 | 1183710109 |
| 院系 | 计算机科学与技术学院 |
| 专业 | 软件工程 |

哈尔滨工业大学

# 实验报告一

# 题目（摘要）

利用拉格朗日插值多项式求的近似值

输 入：个数据点，；插值点

输 出：在插值点的近似值

* 问题1：拉格朗日插值多项式的次数越大越好吗？
* 问题2：插值区间越小越好吗？
* 问题3：在区间考虑拉格朗日插值问题，为了使得插值误差较小，应如何选取插值节点？
* 问题4：考虑拉格朗日插值问题，内插比外推更可靠吗？

# 前言（目的和意义）

目的：利用拉格朗日插值多项式求的近似值

意义：学习根据实际问题建立的数学模型，针对数学模型的特点确定适当的计算方法，编制出计算机能够执行的计算程序，输入计算机，进行调试，完成运算等数值计算的过程。不只会套用教科书中的标准程序进行数值计算，独立地将学过的数值算法编制成计算机程序，灵活应用已经掌握的算法求解综合性较大的课题。理解数值计算程序结构化的思想，提高编程能力，加深对“计算方法”课程内容的理解和掌握，进一步奠定从事数值计算工作的基础。具体可以利用所掌握的“高级语言”顺利地编制出计算机程序，上机实习，完成实验环节的教学要求。不简单地套用现成的标准程序完成实验题目，把重点放在对算法的理解、程序的优化设计、上机调试和计算结果分析上，达到实验课的目的。

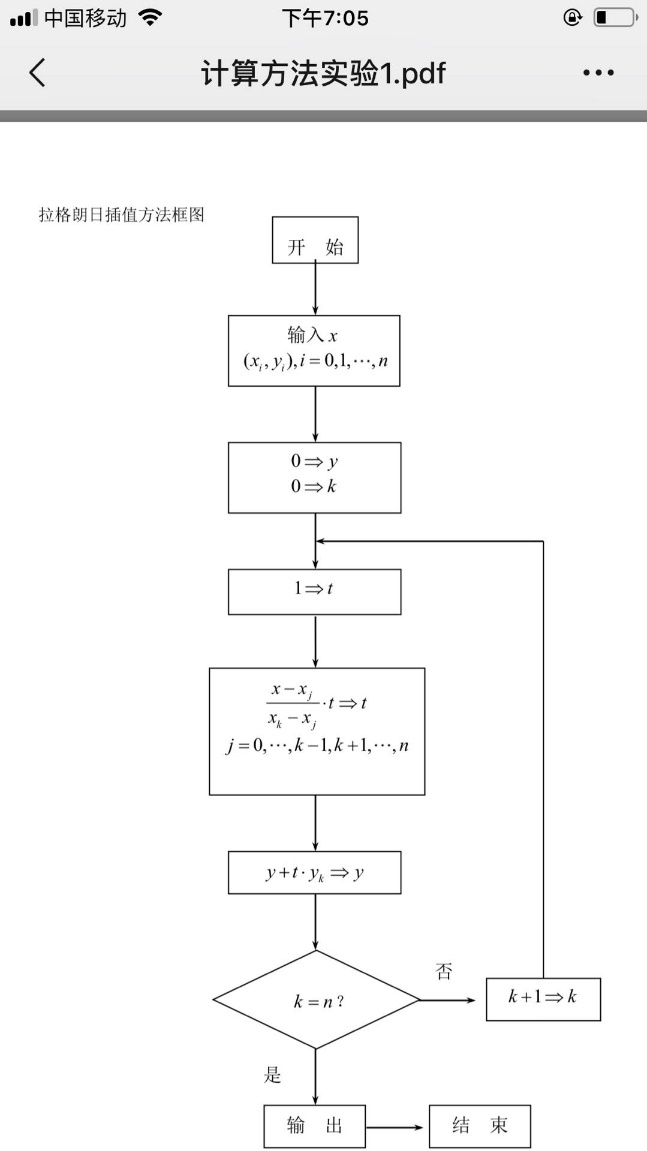
# 数学原理

给定平面上个不同的数据点，，，；则满足条件

的次拉格朗日插值多项式

是存在唯一的。若，且函数充分光滑，则当时，有误差估计式

# 程序设计流程



核心代码：

    double x, y = 0.0;

    scanf("%lf", &x);

    double a[N + 1], b[N + 1];

    int n = 0;

    while (scanf("%lf%lf", &a[n], &b[n]) >= 2) n++;

    n--;

    for (int k = 0; k <= n; k++) {

        double l = 1.0;

        for (int j = 0; j <= n; j++) {

            if (j != k) l \*= (x - a[j]) / (a[k] - a[j]);

        }

        y += l \* b[k];

    }

printf("x = %.3lf\ny = %.3lf", x, y);

测试框架（批量分析）：

#include <cmath>

#include <cstdio>

#define N1 3   // n amount

#define N2 4   // x amount

#define N3 20  // n max

int Ns[N1] = {5, 10, 20};

double x[N2] = {-0.95, -0.05, 0.05, 0.95};

double l = -1.0;

double r = 1.0;

double X(int *k*, int *n*) {

    double h = (r - l) / n;

    return l + k \* h;

}

double Y(double *x*) { return pow(2.718281828459, x); }

int main() {

    for (int i = 0; i < N2; i++) printf("\tx=%.2lf", x[i]);

    printf("\n");

    for (int i = 0; i < N1; i++) {

        double a[N3 + 1], b[N3 + 1];

        int n = Ns[i];

        for (int k = 0; k <= n; k++) {

            a[k] = X(k, n);  // x

            b[k] = Y(a[k]);  // y

        }

        printf("n=%d", n);

        for (int p = 0; p < N2; p++) {

            double y = 0.0;

            for (int k = 0; k <= n; k++) {

                double l = 1.0;

                for (int j = 0; j <= n; j++) {

                    if (j != k) l \*= (x[p] - a[j]) / (a[k] - a[j]);

                }

                y += l \* b[k];

            }

            printf("\t%.6lf", y);

        }

        printf("\n");

    }

    printf("Actual");

    for (int p = 0; p < N2; p++) printf("\t%.6lf", Y(x[p]));

    return 0;

}

# 实验结果、结论与讨论

## 问题1

* 拉格朗日插值多项式的次数越大越好吗？

（1）设，，考虑等距节点的拉格朗日插值多项式，即将区间进行等分，记，，，构造，利用拉格朗日插值多项式作为的近似值。分别取，，，同时计算在，，，，处的函数值。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

（2）设，，考虑等距节点的拉格朗日插值多项式，即将区间进行等分，记，，，构造，利用拉格朗日插值多项式作为的近似值。分别取，，，同时计算在，，，处的函数值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

结论：

拉格朗日插值多项式的次数不是越多越好。在上述实验中或的匹配效果也不错。根据定义，插值点可以在节点处与实际函数匹配，但不能保证在节点之间逼近实际函数，插值次数越高，插值结果越偏离原函数的现象称为多项式摆动Runge现象。

## 问题2

* 插值区间越小越好吗？

（1）设，，考虑等距节点的拉格朗日插值多项式，即将区间进行等分，记，，，构造，利用拉格朗日插值多项式作为的近似值。分别取，，，同时计算在，，，处的函数值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

（2）设，，考虑等距节点的拉格朗日插值多项式，即将区间进行等分，记，，，构造，利用拉格朗日插值多项式作为的近似值。分别取，，，同时计算在，，，处的函数值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

结论：

在分段段数相同的情况下，插值区间越大，误差越大；原因是在较大的区间里，相较于更小的空间变化更大，因此越小的区间函数摆动较小、误差较小。

## 问题3

* 在区间考虑拉格朗日插值问题，为了使得插值误差较小，应如何选取插值节点？

（1）设，，考虑非等距节点的拉格朗日插值多项式，记，，构造，利用拉格朗日插值多项式作为的近似值。分别取，，，同时计算在，，，处的函数值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

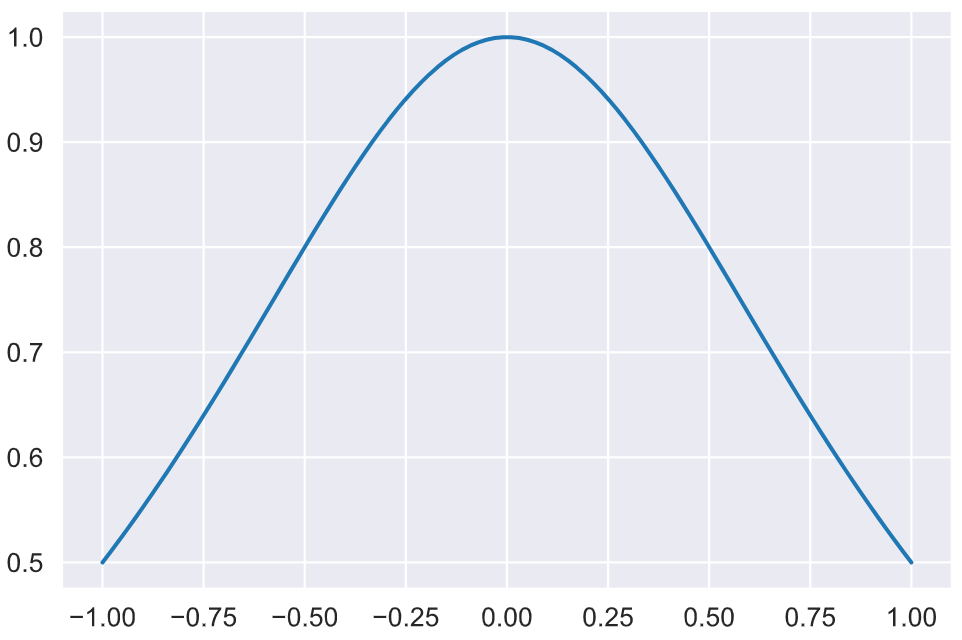
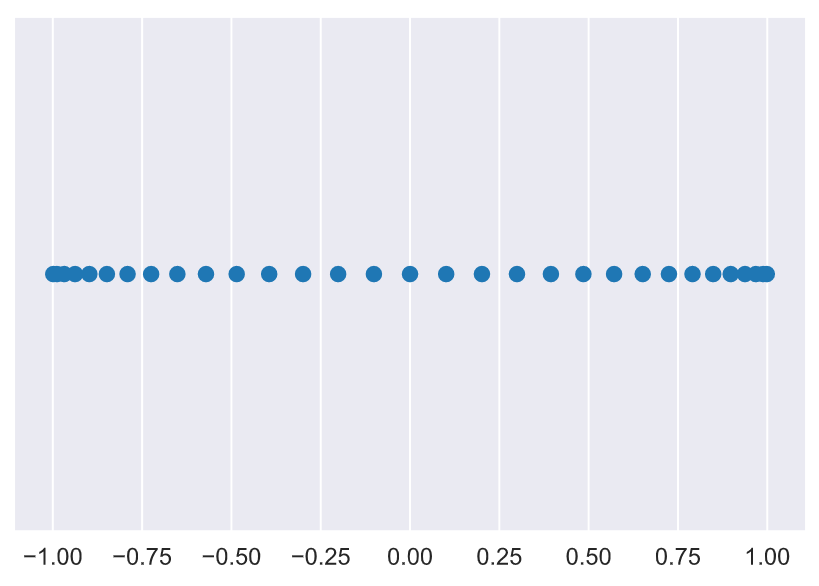
（2）设，，考虑非等距节点的拉格朗日插值多项式，记，，构造，利用拉格朗日插值多项式作为的近似值。分别取，，，同时计算在，，，处的函数值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

结论：

两个函数的插值效果都很好，以第一个函数为例。

函数和函数图像如下：

可以看出在变化较快（斜率较大）的区间内，取值较多，因此插值效果较好。第二个函数同理。

## 问题4

（1）设，关于以，，为节点的拉格朗日插值多项式，利用拉格朗日插值多项式作为的近似值。同时计算在，，，处的函数值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

（2）设，关于以，，为节点的拉格朗日插值多项式，利用拉格朗日插值多项式作为的近似值。同时计算在，，，处的函数值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

（3）设，关于以，，为节点的拉格朗日插值多项式，利用拉格朗日插值多项式作为的近似值。同时计算在，，，处的函数值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

（4）设，关于以，，为节点的拉格朗日插值多项式，利用拉格朗日插值多项式作为的近似值。同时计算在，，，处的函数值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

结论：

由对比可知，内插时插值收敛于实际函数值，超出内插范围插值会发散，且距离插值区间越远外推误差越大；不同取值时外推的插值也差别巨大，说明外推具有极大不确定性。

计 算 方 法

实验二 Romberg积分法

|  |  |
| --- | --- |
| 姓名 | 郭茁宁 |
| 学号 | 1183710109 |
| 院系 | 计算机科学与技术学院 |
| 专业 | 软件工程 |

哈尔滨工业大学

# 实验报告二

# 题目（摘要）

利用龙贝格(Romberg)积分法计算积分

输入：

输出： 龙贝格数表

# 前言（目的和意义）

目的：利用龙贝格(Romberg)积分法计算积分

意义：学习根据实际问题建立的数学模型，针对数学模型的特点确定适当的计算方法，编制出计算机能够执行的计算程序，输入计算机，进行调试，完成运算等数值计算的过程。不只会套用教科书中的标准程序进行数值计算，独立地将学过的数值算法编制成计算机程序，灵活应用已经掌握的算法求解综合性较大的课题。理解数值计算程序结构化的思想，提高编程能力，加深对“计算方法”课程内容的理解和掌握，进一步奠定从事数值计算工作的基础。具体可以利用所掌握的“高级语言”顺利地编制出计算机程序，上机实习，完成实验环节的教学要求。不简单地套用现成的标准程序完成实验题目，把重点放在对算法的理解、程序的优化设计、上机调试和计算结果分析上，达到实验课的目的。

# 数学原理

利用复化梯形求积公式、复化辛普生求积公式、复化柯特斯求积公式的误差估计式计算积分。记，，，其计算公式：

一般地，利用龙贝格算法计算积分，要输出所谓的数表

# 程序设计流程

1．准备初值，计算



且（为等份次数）

2．按梯形公式的递推关系，计算



3．按龙贝格公式计算加速值



4．精度控制。对给定的精度，若



则终止计算，并取作为所求结果；否则，重复2~4步，直到满足精度为止。

核心代码：

#include <cmath>

#include <cstdio>

#include <iostream>

using namespace std;

#define N 100

int n;

long double a, b, e, T[N][N] = {{0.0}};

long double f(long double *x*) { return x \* x \* exp(x); }

long double x(int *i*, long double *h*) { return a + h \* i; }

int main() {

    scanf("%llf%llf%llf%d", &a, &b, &e, &n);

    int k = 0;

    for (; k < n; k++) {

        long double h = (b - a) / pow(2, k), sum = 0.0;

        for (int i = 1; i <= pow(2, k) - 1; i++) sum += f(x(i, h));

        T[k][0] = 0.5 \* h \* (f(a) + 2 \* sum + f(b));

        for (int m = 1; m <= k; m++)

            T[k][m] =

                (pow(4, m) \* T[k][m - 1] - T[k - 1][m - 1]) / (pow(4, m) - 1);

        if (k > 0) {

            if (fabs(T[k][0] - T[k][k]) <= e) break;

            cout << T[k][0] << '\t' << T[k][k] << endl;

        }

    }

    cout << T[k - 1][0] << '\n' << T[k - 1][k - 1] << endl;

    return 0;

}

# 实验结果、结论与讨论

1. ，

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

1. ，

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

1. ，

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

1. ，

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

## 思考题：输入的参数有什么意义？

输入的参数越大，在有限区间上分段越小，由复化梯形公式的误差得，计算精度越高。

计 算 方 法

实验三 四阶Runge—Kutta方法

|  |  |
| --- | --- |
| 姓名 | 郭茁宁 |
| 学号 | 1183710109 |
| 院系 | 计算机科学与技术学院 |
| 专业 | 软件工程 |

哈尔滨工业大学

# 实验报告三

# 题目（摘要）

利用四阶龙格—库塔(Runge—Kutta)方法求解微分方程初值问题

输 入：

输 出：初值问题的数值解，。

# 前言（目的和意义）

目的：利用四阶龙格—库塔(Runge—Kutta)方法求解微分方程初值问题

意义：学习根据实际问题建立的数学模型，针对数学模型的特点确定适当的计算方法，编制出计算机能够执行的计算程序，输入计算机，进行调试，完成运算等数值计算的过程。不只会套用教科书中的标准程序进行数值计算，独立地将学过的数值算法编制成计算机程序，灵活应用已经掌握的算法求解综合性较大的课题。理解数值计算程序结构化的思想，提高编程能力，加深对“计算方法”课程内容的理解和掌握，进一步奠定从事数值计算工作的基础。具体可以利用所掌握的“高级语言”顺利地编制出计算机程序，上机实习，完成实验环节的教学要求。不简单地套用现成的标准程序完成实验题目，把重点放在对算法的理解、程序的优化设计、上机调试和计算结果分析上，达到实验课的目的。

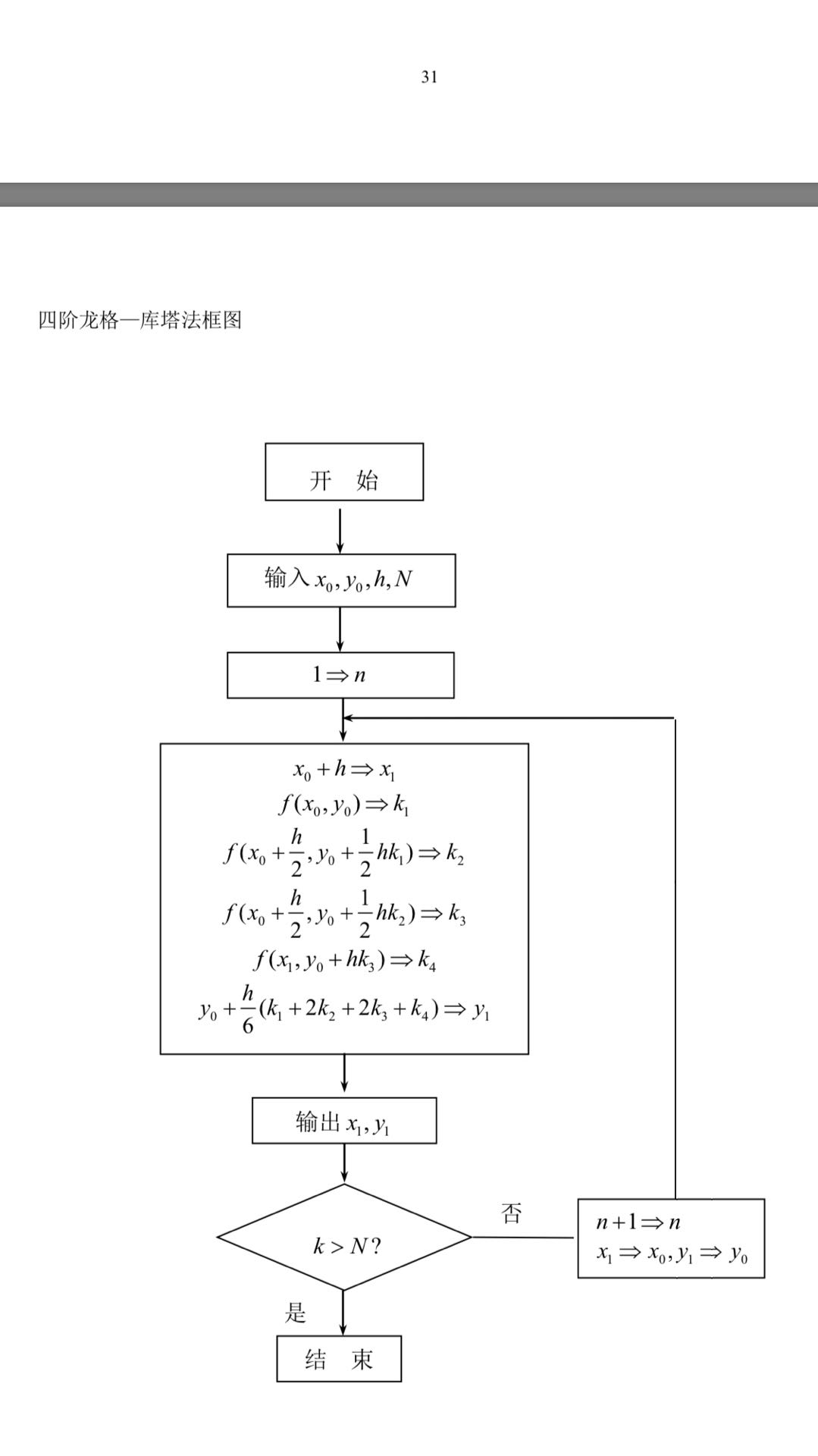
# 数学原理

给定常微分方程初值问题

记，，利用四阶龙格—库塔方法

可逐次求出微分方程初值问题的数值解，。

# 程序设计流程



#include <cmath>

#include <cstdio>

#include <iostream>

using namespace std;

int n;

double a, b, fa;

double f(double *x*, double *y*) { return -y \* y; }

double f\_(double *x*) { return 1.0 / (x + 1.0); }

int main() {

    scanf("%lf%lf%lf%d", &a, &b, &fa, &n);

    double x = a, y = fa, h = (b - a) / n;

    for (int i = 1; i <= n; i++) {

        double k1 = h \* f(x, y);

        double k2 = h \* f(x + h / 2, y + k1 / 2);

        double k3 = h \* f(x + h / 2, y + k2 / 2);

        double k4 = h \* f(x + h, y + k3);

        x += h;

        y += 1.0 / 6.0 \* (k1 + 2 \* k2 + 2 \* k3 + k4);

        printf("%.2lf\t%lf\t%.2lf\n", x, y, fabs(f\_(x) - y));

    }

    return 0;

}

# 实验结果、结论与讨论

（1）

准确解：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | | |  | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

（2）

准确解：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | | |  | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

## 思考题：对实验1，数值解和解析解相同吗？为什么？试加以说明。

实验一中(1)数值解和解析解相同，(2)数值解和解析解稍有不同，因为四阶Runge-Kutta方法是以小段的线性算法来近似获得微分方程的数值解，(1)的准确解是1阶的，(2)的准确解是无限阶的，因此对于(1)数值解和解析解相同。

计 算 方 法

实验四 Newton迭代法

|  |  |
| --- | --- |
| 姓名 | 郭茁宁 |
| 学号 | 1183710109 |
| 院系 | 计算机科学与技术学院 |
| 专业 | 软件工程 |

哈尔滨工业大学

# 实验报告四

# 题目（摘要）

利用牛顿迭代法求的根

输 入：初值，精度，最大迭代次数

输 出：方程根的近似值或计算失败标志

# 前言（目的和意义）

目的：利用牛顿迭代法求的根

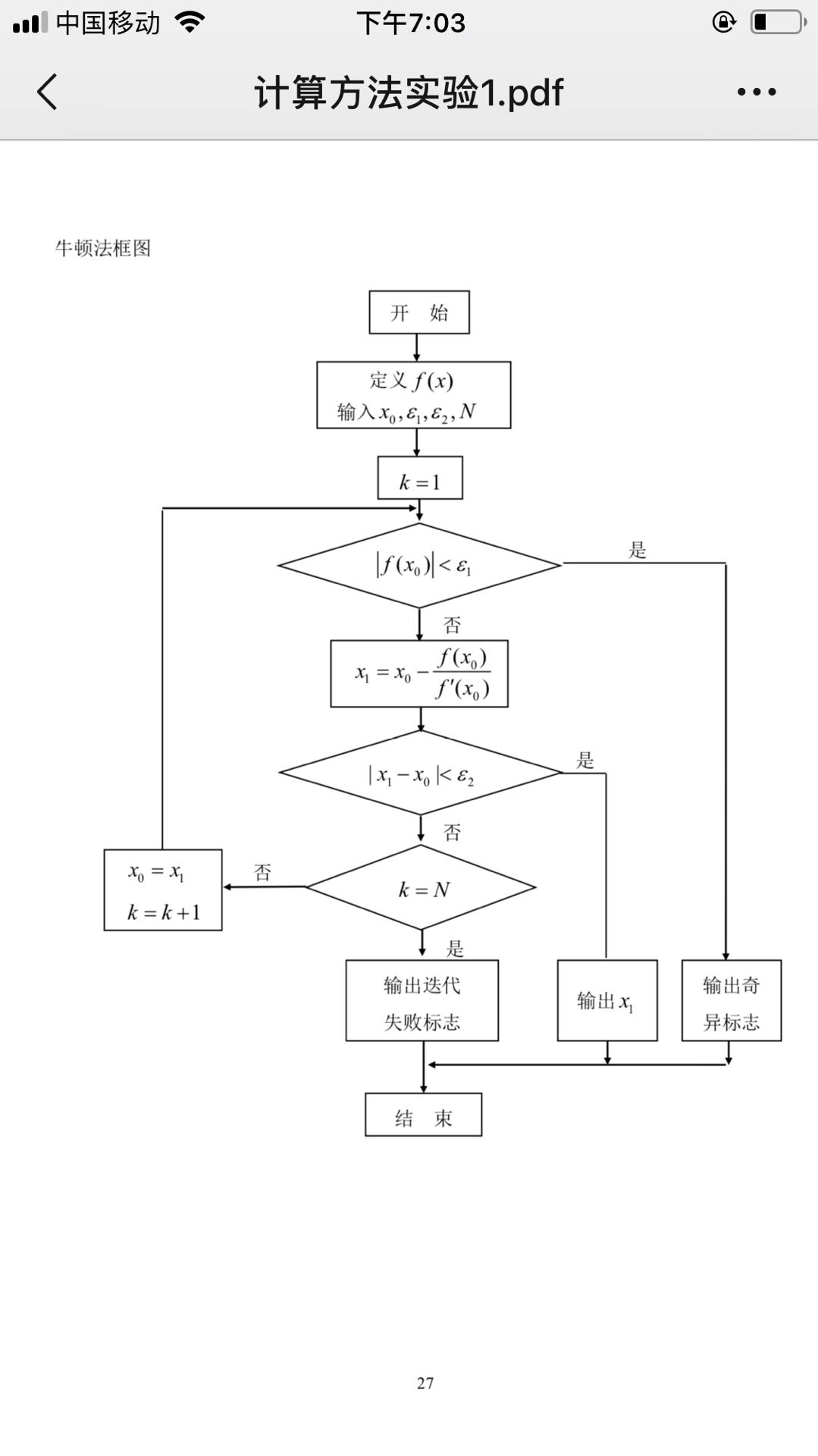
意义：学习根据实际问题建立的数学模型，针对数学模型的特点确定适当的计算方法，编制出计算机能够执行的计算程序，输入计算机，进行调试，完成运算等数值计算的过程。不只会套用教科书中的标准程序进行数值计算，独立地将学过的数值算法编制成计算机程序，灵活应用已经掌握的算法求解综合性较大的课题。理解数值计算程序结构化的思想，提高编程能力，加深对“计算方法”课程内容的理解和掌握，进一步奠定从事数值计算工作的基础。具体可以利用所掌握的“高级语言”顺利地编制出计算机程序，上机实习，完成实验环节的教学要求。不简单地套用现成的标准程序完成实验题目，把重点放在对算法的理解、程序的优化设计、上机调试和计算结果分析上，达到实验课的目的。

# 数学原理

求非线性方程的根，牛顿迭代法计算公式

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，。如果，，，那么，对充分小的，当时，由牛顿迭代法计算出的收敛于，且收敛速度是2阶的；如果，，，那么，对充分小的，当时，由牛顿迭代法计算出的收敛于，且收敛速度是1阶的；

# 程序设计流程



#include <cmath>

#include <cstdio>

#include <iostream>

using namespace std;

double x, e1, e2;

int n;

double f(double *x*) { return cos(x) - x; }

double df(double *x*) { return -sin(x) - 1; }

int main() {

    scanf("%lf%lf%lf%d", &x, &e1, &e2, &n);

    for (int i = 1; i <= n; i++) {

        double F = f(x), DF = df(x);

        if (fabs(F) < e1) {

            printf("%lf", x);

            return 0;

        }

        if (fabs(DF) < e2) {

            printf("Failed");

            return 0;

        }

        double x1 = x - F / DF;

        double tol = fabs(x - x1);

        if (tol < e1) {

            printf("%lf", x1);

            return 0;

        }

        x = x1;

    }

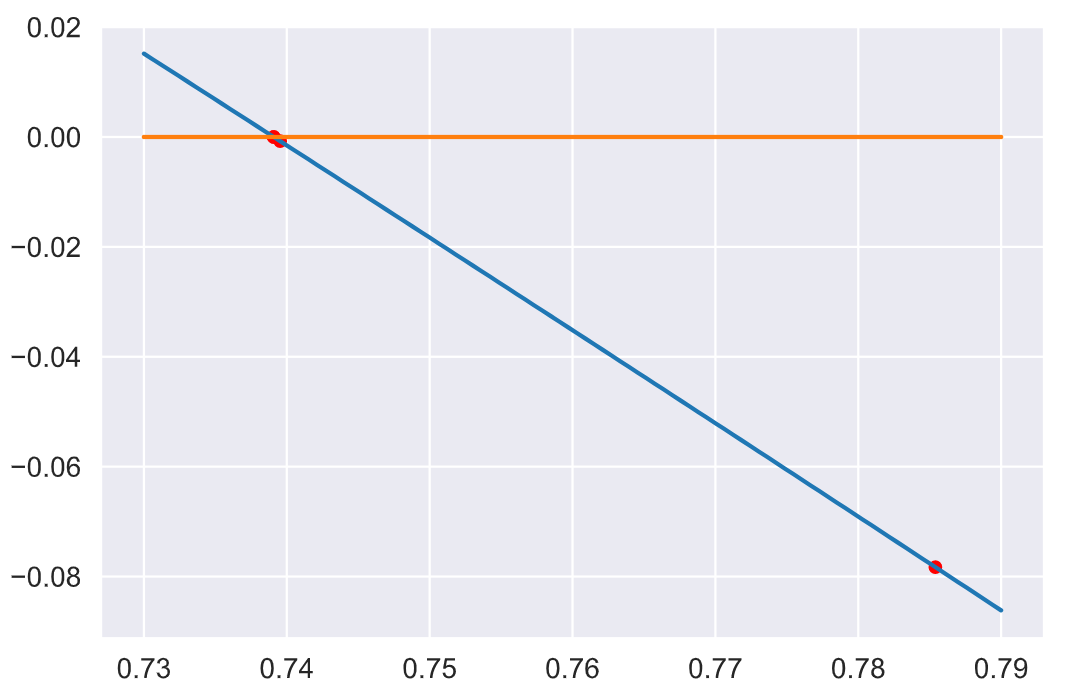
    printf("Failed");

    return 0;

}

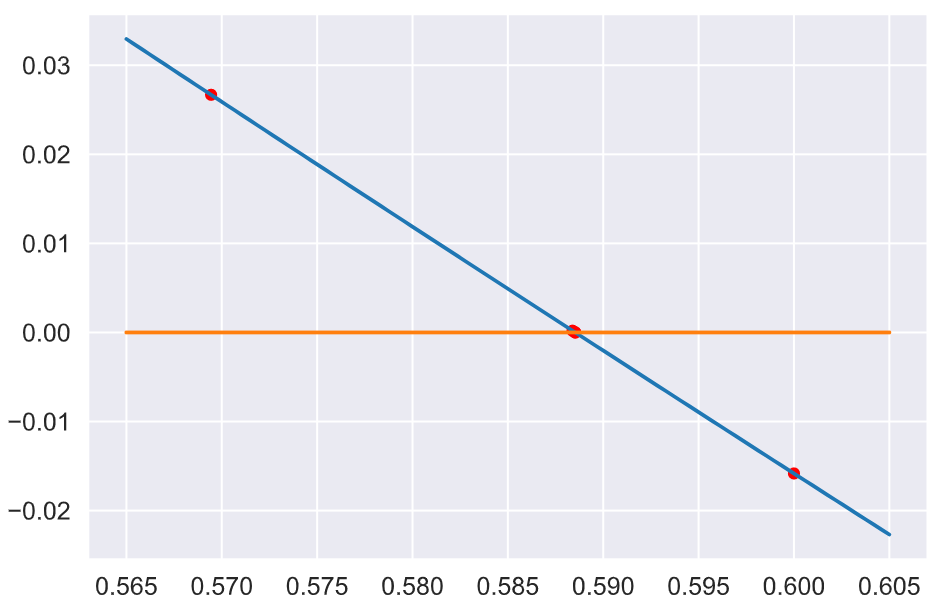
# 实验结果、结论与讨论

1. ，，，，



解得

1. ，，，，



解得

## 思考题：对实验1 确定初值的原则是什么？实际计算中应如何解决？

在的根附近任取一点作为初始近似根。

由于Newton法具有局部收敛性，所以在实际问题中，当提供了接近于根的初始值位置即可使用，其可确保收敛。当初值难以确定时，迭代序列就不一定收敛，要通过如二分法等算法确定近似值，再将该近似值作为初值，保证收敛性。

计 算 方 法

实验五 Gauss列主元消去法

|  |  |
| --- | --- |
| 姓名 | 郭茁宁 |
| 学号 | 1183710109 |
| 院系 | 计算机科学与技术学院 |
| 专业 | 软件工程 |

哈尔滨工业大学

# 实验报告五

# 题目（摘要）

利用Gauss列主元消去法、显式相对Gauss列主元消去法、隐式相对Gauss列主元消去法求解线性方程组。

输 入：；， ，

输 出：线性方程组的近似解，

# 前言（目的和意义）

目的： 利用Gauss列主元消去法、显式相对Gauss列主元消去法、隐式相对Gauss列主元消去法求解线性方程组。

意义：学习根据实际问题建立的数学模型，针对数学模型的特点确定适当的计算方法，编制出计算机能够执行的计算程序，输入计算机，进行调试，完成运算等数值计算的过程。不只会套用教科书中的标准程序进行数值计算，独立地将学过的数值算法编制成计算机程序，灵活应用已经掌握的算法求解综合性较大的课题。理解数值计算程序结构化的思想，提高编程能力，加深对“计算方法”课程内容的理解和掌握，进一步奠定从事数值计算工作的基础。具体可以利用所掌握的“高级语言”顺利地编制出计算机程序，上机实习，完成实验环节的教学要求。不简单地套用现成的标准程序完成实验题目，把重点放在对算法的理解、程序的优化设计、上机调试和计算结果分析上，达到实验课的目的。

# 数学原理

高斯（Gauss）列主元消去法：对给定的阶线性方程组，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。

如果真正进行运算去确定相对主元，则称为显式相对Gauss列主元消去法；如果不进行运算，也能确定相对主元，则称为隐式相对Gauss列主元消去法。

显式相对Gauss列主元消去法：对给定的阶线性方程组，首先进行列主元消元过程，在消元过程中利用显式平衡技术，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。

隐式相对Gauss列主元消去法：对给定的阶线性方程组，首先进行列主元消元过程，在消元过程中利用隐式平衡技术，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。

Gauss列主元消去法

1对，做1.1—1.3，消元过程

1.1 寻找最小的正整数，和。如果，输出奇异标志，停机；

1.2 如果，那么交换两行；

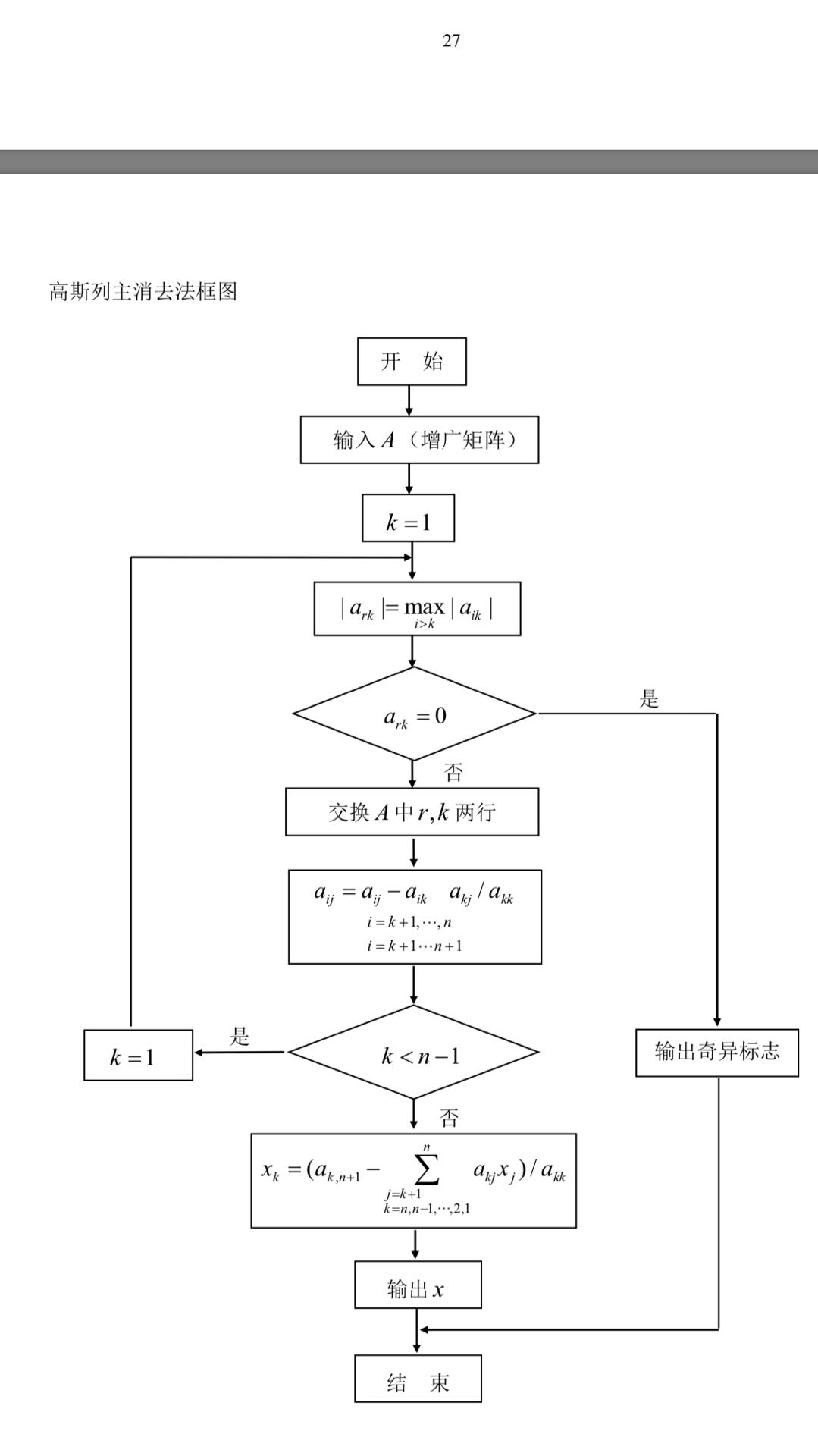
1.3 对，记，计算

2. 如果输出奇异标志，停机；

3. 置 ，回代过程

4. 对，置

# 程序设计流程



#include <cmath>

#include <cstdio>

#include <iostream>

using namespace std;

#define N 10

int n;

double a[N][N], b[N], x[N];

int main() {

    scanf("%d", &n);

    for (int i = 1; i <= n; i++)

        for (int j = 1; j <= n; j++) scanf("%lf", &a[i][j]);

    for (int i = 1; i <= n; i++) scanf("%lf", &b[i]);

    for (int k = 1; k < n; k++) {

        int p = k;

        double maxabs = fabs(a[k][k]);

        for (int j = k + 1; j <= n; j++)

            if (fabs(a[j][k]) - maxabs > 0) {

                p = j;

                maxabs = fabs(a[j][k]);

            }

        if (a[p][k] == 0) {

            printf("Singular");

            return 0;

        }

        if (p != k) {

            double tmp;

            for (int j = 1; j <= n; j++) {

                tmp = a[p][j];

                a[p][j] = a[k][j];

                a[k][j] = tmp;

            }

            tmp = b[p];

            b[p] = b[k];

            b[k] = tmp;

        }

        for (int i = k + 1; i <= n; i++) {

            double m\_ik = a[i][k] / a[k][k];

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

            b[i] -= b[k] \* m\_ik;

        }

    }

    if (a[n][n] == 0) {

        printf("Singular");

        return 0;

    }

    x[n] = b[n] / a[n][n];

    for (int k = n - 1; k >= 1; k--) {

        double sigma = 0.0;

        for (int j = k + 1; j <= n; j++) sigma += a[k][j] \* x[j];

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

    }

    for (int i = 1; i <= n; i++) printf("%lf\t", x[i]);

    return 0;

}

# 实验结果、结论与讨论

(1)

解得

准确结果为

(2)

解得

准确结果为

(3)

解得

准确结果为

(4)

解得

准确结果为

## 思考题：计算实验1、实验2的各个题目说明：对什么类型的线性方程组三种方法是一致的？

在各主元不是非常小、且数量级差别不大的时候，三种方法结果一致。