****

**实验报告**

**课程名称： 数值分析第二次上机报告**

**实验项目： 曲线拟合与函数插值和数值积分**

**专业班级： 软件工程1808班**

**姓 名： 赵祎泽 学 号： 181203817**

**实验室号： 715-1 实验组号：**

**实验时间： 2020.11.24 批阅时间：**

**指导教师： 张琪 成 绩：**

**沈阳工业大学实验报告**

（适用计算机程序设计类）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **一、实验目的：**  1. 学会编写曲线拟合与函数插值方面的程序  2. 学会编写数值积分方面的程序  **二、实验内容：**  **曲线拟合与函数插值**  1. 设 ，x ∈[−5,5] 在[−5,5]内取 n +1个等距节点 (k = 0,1, 2,…, n)  时的插值多项式Ln(x)并在同一张图上画出f (x)和所有Ln( x)的图形。  2. 已知某天在各个整点时刻的温度如下表所示，使用最小二乘法，确定这一天的气温变化规律。   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 时间t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | 温度 y(t) | 14 | 13 | 13 | 13 | 13 | 14 | 15 | 17 | 19 | | 时间t | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | | 温度 y(t) | 21 | 22 | 24 | 27 | 30 | 31 | 30 | 28 | 26 | | 时间t | 18 | 19 | 20 | 21 | 22 | 23 | 24 |  |  | | 温度 y(t) | 24 | 23 | 21 | 19 | 17 | 16 | 15 |  |  |   分别采用下列函数进行拟合，分析误差，并作图比较效果。  （1）二次多项式；  （2）三次多项式；  （3）形如 的函数，其中 a,b,c 为待定常数。  **数值积分**   1. 对于积分 ，应用下列方法进行计算：   （1）分别采用n = 50,100,200的复化梯形公式；  （2）采用两点、三点高斯-勒让德求积公式。  **三、实验方案（程序设计说明）**  使用C语言或C++函数编写程序  **四、实验步骤或程序（经调试后正确的源程序）**  见附件A  **五、程序运行结果**  曲线拟合与函数插值   1. 拉格朗日插值函数图像 2. 数据拟合结果 3. 二次多项式      1. 三次多项式     其图像为：   1. 形如 的函数，其中 a, b, c 为待定常数。   数值积分   1. (1) n=20     n=50    n=100    (2)两点高斯-勒让德    三点高斯-勒让德    **六、出现的问题及解决方法** |

**附件A 沈阳工业大学实验报告**

（适用计算机程序设计类）

专业班级： 软件工程1808班 学号： 181203817 姓名： 赵祎泽

**实验步骤或程序**：

**曲线拟合与函数插值**

1. **拉格朗日插值**

#include <iostream>

using namespace std;

//预先定义插值节点的个数为1000个，根据控制台输入的个数num从而确定插值节点的个数

const int N=1000;

double f1(double x) {

return 1.0 / (1 + x \* x);

}

double Lagrange(int n,double X) {

double yResult = 0.0;

double x0[N], x1[N], x2[N];

//LValue[N]存放的是每次求解的插值基函数的通项

double LValue[N];

//循环变量k,m

int k, m;

//插值基函数中的上下累乘temp1,temp2

double temp1, temp2;

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

x0[i] = -5 + 10.0 \* i / n;

}

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

x1[i] = f1(x0[i]);

}

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

temp1 = 1.0;

temp2 = 1.0;

for (m = 0; m < n; m++) {

if (m == k) {

continue;

}

temp1 \*= (X - x0[m]);

temp2 \*= (x0[k] - x0[m]);

}

LValue[k] = temp1 / temp2;

}

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

yResult += x1[i] \* LValue[i];

}

return yResult;

}

int main(void) {

int num;

cout << "输入插值节点的个数(小于" << N << "个): ";

cin >> num;

double X;

cout<<"\n--请输入待求解的插值节点的X值--\n";

cin>>X;

double Res = Lagrange(num,X);

cout << "\n--插值结果为: " << Res << endl;

return 0;

}

1. **数据拟合**

#include <iostream>

#include <cstdlib>

#include <cmath>

using namespace std;

void polyfit(int n,const double \*x,const double \*y, int poly\_n, double p[]);

void gauss\_solve(int n,double A[],double x[],double b[]);

/\*==================polyfit(n,x,y,poly\_n,a)==================\*/

/\*=======拟合y=a0+a1\*x+a2\*x^2+……+apoly\_n\*x^poly\_n========\*/

/\*=====n是数据个数 xy 是数据值 poly\_n 是多项式的项数=======\*/

/\*===返回a0,a1,a2,……a[poly\_n]，系数比项数多一（常数项）=====\*/

void polyfit(int n,const double x[],const double y[],int poly\_n,double p[]) {

int i, j;

double \*tempx, \*tempy, \*sumxx, \*sumxy, \*ata;

tempx = (double \*) calloc(n, sizeof(double));

sumxx = (double \*) calloc((poly\_n \* 2 + 1), sizeof(double));

tempy = (double \*) calloc(n, sizeof(double));

sumxy = (double \*) calloc((poly\_n + 1), sizeof(double));

ata = (double \*) calloc((poly\_n + 1) \* (poly\_n + 1), sizeof(double));

for (i = 0; i < n; i++) {

tempx[i] = 1;

tempy[i] = y[i];

}

for (i = 0; i < 2 \* poly\_n + 1; i++) {

for (sumxx[i] = 0, j = 0; j < n; j++) {

sumxx[i] += tempx[j];

tempx[j] \*= x[j];

}

}

for (i = 0; i < poly\_n + 1; i++) {

for (sumxy[i] = 0, j = 0; j < n; j++) {

sumxy[i] += tempy[j];

tempy[j] \*= x[j];

}

}

for (i = 0; i < poly\_n + 1; i++) {

for (j = 0; j < poly\_n + 1; j++) {

ata[i \* (poly\_n + 1) + j] = sumxx[i + j];

}

}

gauss\_solve(poly\_n + 1, ata, p, sumxy);

}

/\*============================================================

高斯消元法计算得到n次多项式的系数

n: 系数的个数

ata: 线性矩阵

sumxy: 线性方程组的Y值

p: 返回拟合的结果

============================================================\*/

void gauss\_solve(int n,double A[],double x[],double b[]) {

int i, j, k, r;

double max;

for (k = 0; k < n - 1; k++) {

max = fabs(A[k \* n + k]);

r = k;

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

if (max < fabs(A[i \* n + i])) {

max = fabs(A[i \* n + i]);

r = i;

}

}

if (r != k) {

for (i = 0; i < n; i++) //change array:A[k]&A[r]

{

max = A[k \* n + i];

A[k \* n + i] = A[r \* n + i];

A[r \* n + i] = max;

}

max = b[k]; //change array:b[k]&b[r]

b[k] = b[r];

b[r] = max;

}

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

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

A[i \* n + j] -= A[i \* n + k] \* A[k \* n + j] / A[k \* n + k];

b[i] -= A[i \* n + k] \* b[k] / A[k \* n + k];

}

}

for (i = n - 1; i >= 0; x[i] /= A[i \* n + i], i--) {

for (j = i + 1, x[i] = b[i]; j < n; j++)

x[i] -= A[i \* n + j] \* x[j];

}

}

int main(void) {

int i, sizenum;

double P[25];

cout << "请输入维度" << endl;

int dimension = 3;

cin >> dimension; // 拟合数据的维数

// 要拟合的数据

const double xx[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24};

const double yy[] = {14, 13, 13, 13, 13, 14, 15, 17, 19, 21, 22, 24, 27, 30, 31, 30, 28, 26, 24, 23, 21, 19, 17, 16, 15};

sizenum = sizeof(xx) / sizeof(xx[0]);

polyfit(sizenum, xx, yy, dimension, P);

cout << "拟合系数, 按升序排列如下:" << endl;

for (i = 0; i < dimension + 1; i++) {

cout << "P[" << i << "] = " << P[i] << endl;

}

}

**数值积分**

1. **(1) 复化梯形公式**

#include <iostream>

#include <cmath>

using namespace std;

double f(double x) {

return exp(-(x \* x / 2));

}

double computeT(double a, double b, int n) {

double h = (b - a) / n, T = 0;

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

T += f(a + i \* h); //1 <= k <= n - 1

return h \* (f(a) + 2 \* T + f(b)) / 2;

}

int main(void) {

int n; // 区间等分数

cout << "请输入区间等分数" << endl;

cin >> n;

double a , b; // 上下限

cout << "请输入积分上下限a b" << endl;

cin >> a >> b;

cout << "结果是 = " << computeT(a, b, n) << endl;

return 0;

}

**(2) 两点高斯-勒让德**

#include <iostream>

#include <cmath>

using namespace std;

double function1(double x) {

return exp(-(pow(x, 2) / 2));

}

double function2(double a,double b) {

return (b - a) / 2 \*

(function1((a + b) / 2 - (b - a) / (2 \* sqrt(3))) + function1((a + b) / 2 + (b - a) / (2 \* sqrt(3))));

}

int main(void) {

double a, b;

cout << "Please input a b" << endl;

cin >> a >> b;

cout << function2(a, b) << endl;

return 0;

}

**三点高斯-勒让德**

#include <iostream>

#include <cmath>

using namespace std;

double function1(double t,double a,double b) {

double x = (b - a) / 2 \* t + (a + b) / 2;

return exp(-(pow(x, 2) / 2));

}

double function2() {

double a, b;

cout << "Please input a b" << endl;

cin >> a >> b;

return (b - a) / 2 \* (5.0 / 9 \* function1(-sqrt(15) / 5, a, b) + 8.0 / 9 \* function1(0, a, b) +

5.0 / 9 \* function1(sqrt(15) / 5, a, b));

}

int main(void) {

cout << function2() << endl;

return 0;

}