第五次上机作业

1.题目

(**上机题 3**) 设 $f(x)=\sin x,\ x\in[-\pi,\pi]$.

(1) 写出它在 $x_0=0$ 处次数超过 5 次的 Taylor 多项式 $P_n(x)$ $(n\leqslant 5)$, 分别做出 $f(x)$ 和 $P_n(x)$ 的对比图形, 观察对比图形, 给出你的结论并解释.

(2) 计算 $f(x)$ 在 $[-\pi,\pi]$ 上次数不超过 5 的最佳平方逼近多项式 $Q_n(x)$ $(n\leqslant 5)$, 分别做出 $f(x)$ 和 $Q_n(x)$ 的对比图形, 结合 (1) 给出你的结论并解释.

2.程序代码

代码一：f(x)=sinx,x∈[-Π，Π]，x 在 x0=0 处的 1-5 次 Taylor 多项式 Pn(x)，分别画出 1000 个点，进行函数图像的绘制，把 1000 个点对应的 x 和 y 输出到 txt 文件中，C++代码如下：

```cpp
#include <iostream>
#include <cmath>
#include <fstream>

const double PI = 3.14159265358979323846;

// 计算阶乘
double factorial(int n) {
    if (n == 0 || n == 1)
        return 1;
    else
        return n * factorial(n - 1);
}

// 计算泰勒展开的值
double taylorExpansion(double x, int n) {
    double result = 0;
    int n_tem;
    if (n==1)
        n_tem=0;
    else if (n==2)
        n_tem=0;
    else if (n==3)
        n_tem=1;
    else if (n==4)
        n_tem=1;
    else if (n==5)
        n_tem=2;
    else n_tem=2;
    for (int i = 0; i <= n_tem; ++i) {
        double numerator = std::pow(-1, i) * std::pow(x, 2*i+1);
        double denominator = factorial(2*i + 1);
        result += numerator / denominator;
    }
    return result;
}

int main() {
    int n;
    std::cout << "请输入展开的次数 n: ";
    std::cin >> n;

    std::ofstream outputFile("taylor_expansion5.txt");
    if (!outputFile.is_open()) {
        std::cout << "无法打开文件! " << std::endl;
        return 1;
    }

    // 绘制泰勒展开的图像
    int numPoints = 1000;  // 绘制的点数
    double stepSize = 2 * PI / numPoints;
    for (int i = 0; i < numPoints; ++i) {
        double x = -PI + i * stepSize;
        double y = taylorExpansion(x, n);
        outputFile << x << " " << y << std::endl;

    }

    outputFile.close();

    std::cout << "图像已保存到 taylor_expansion.txt" << std::endl;

    return 0;

}
```

代码二：代码一：f(x)=sinx,x∈[-Π，Π]，x 在 x0=0 处的 1-5 次最佳平方逼近多项式 Qn(x)，分别画出 1000 个点，进行函数图像的绘制，把 1000 个点对应的 x 和 y 输出到 txt 文件中，C++代码如下：

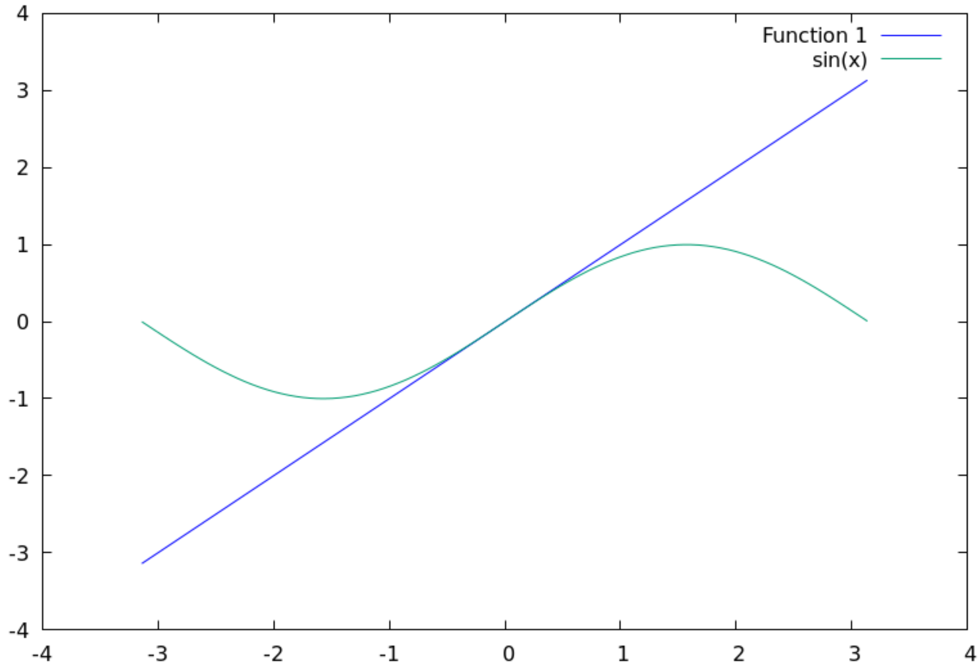
```cpp
#include <iostream>
#include <cmath>
#include <vector>

// 使用最小二乘法计算一次最佳平方逼近多项式的系数
void calculateApproximation(const std::vector<double>& x, const std::vector<double>& y, double&
    double sumX = 0.0, sumY = 0.0, sumXY = 0.0, sumXX = 0.0;

    // 计算各项累加和
    for (size_t i = 0; i < x.size(); ++i) {
        sumX += x[i];
        sumY += y[i];
        sumXY += x[i] * y[i];
        sumXX += x[i] * x[i];
    }
    // 计算系数
    double denominator = x.size() * sumXX - sumX * sumX;
    if (std::abs(denominator) > 1e-6) {
        a = (x.size() * sumXY - sumX * sumY) / denominator;
        b = (sumXX * sumY - sumX * sumXY) / denominator;
    } else {
        a = 0.0;
        b = 0.0;
    }
}

int main() {
    std::vector<double> x; // x坐标值
    std::vector<double> y; // sin(x)的实际值

    // 在[-π, π]范围内生成1000个点的x坐标值，并计算对应的sin(x)值
    for (int i = 0; i < 1000; ++i) {
        double xi = -M_PI + i * 2.0 * M_PI / 1000.0;
        x.push_back(xi);
        y.push_back(std::sin(xi));
    }

    double a, b;
    calculateApproximation(x, y, a, b);

    std::cout << "一次最佳平方逼近多项式: y = " << a << "x + " << b << std::endl;

    return 0;
}
```

代码三：将输出的 txt 文件中对应的坐标点和相应的 sin(x)值进行比较，输出 png 图片。

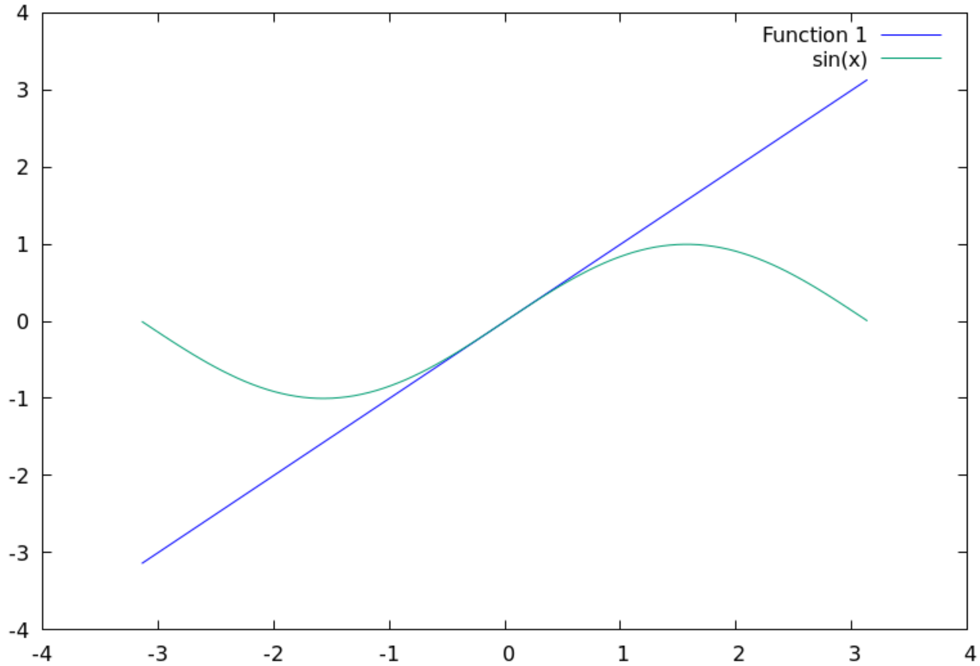
```
set terminal pngcairo size 800,600
set output 'chaifen5.png'
set title 'n=5'
set xlabel 'x'
set ylabel 'y'
plot 'data5.txt' with lines lt 1 lc rgb 'blue' title "chaifen_5",\
     sin(x) with lines title 'sin(x)'
```

3. 运行结果

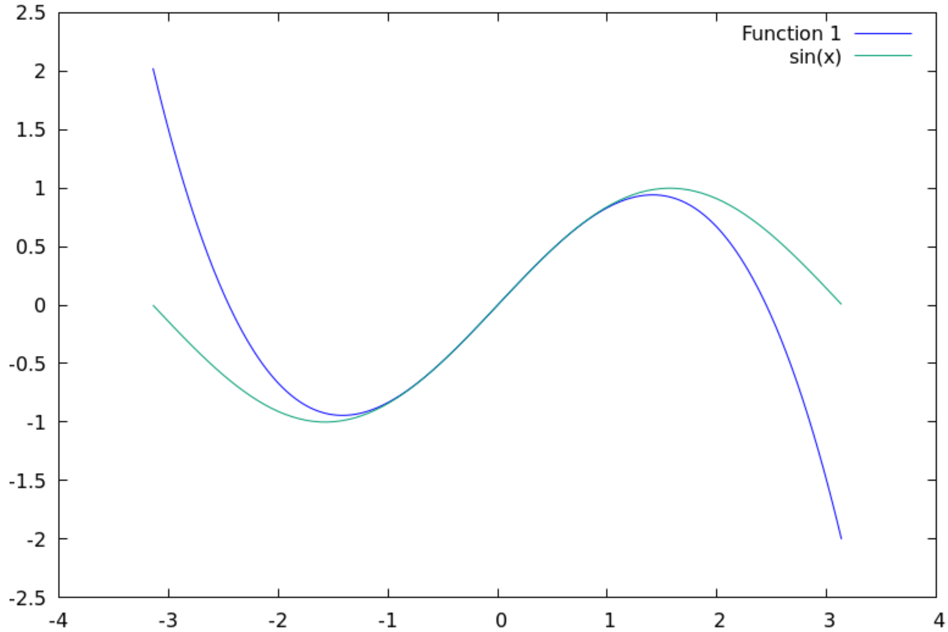
Taylor 的 1-5 次展开与原图像的比较图片如下：



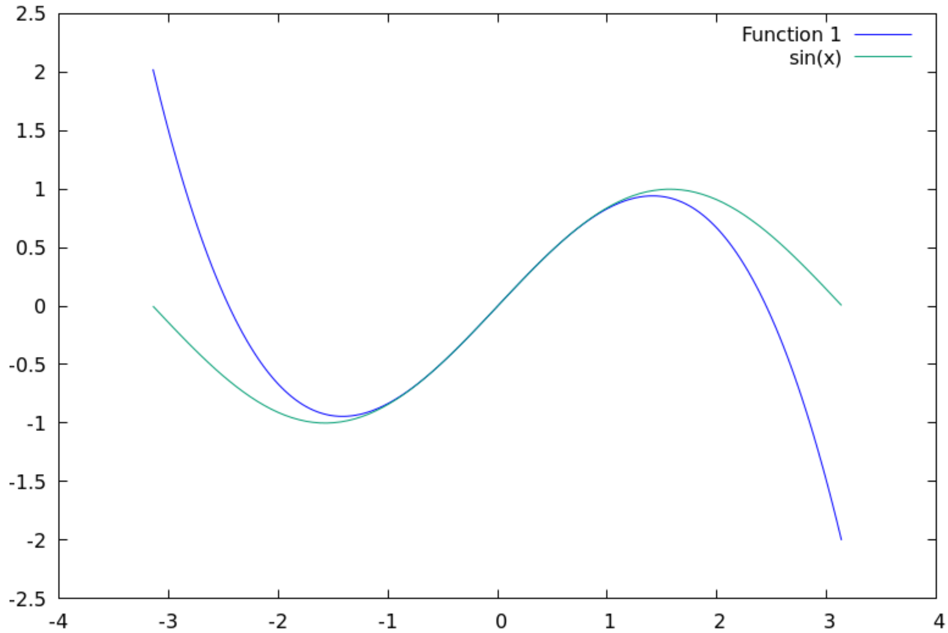
1 次



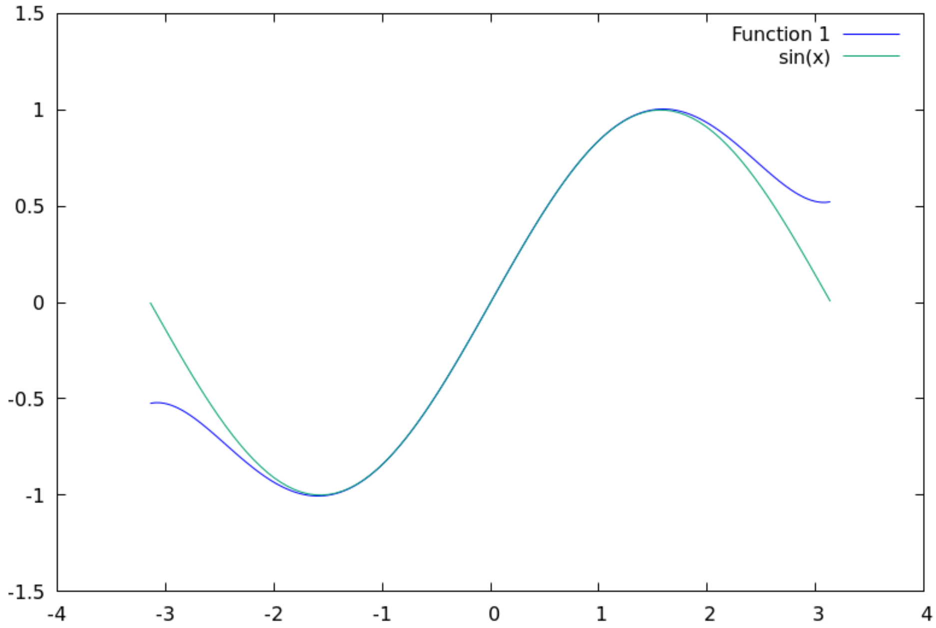
2 次



3 次



4 次



5 次

最佳平方逼近多项式的 1-5 次展开与原图像的比较图片如下：

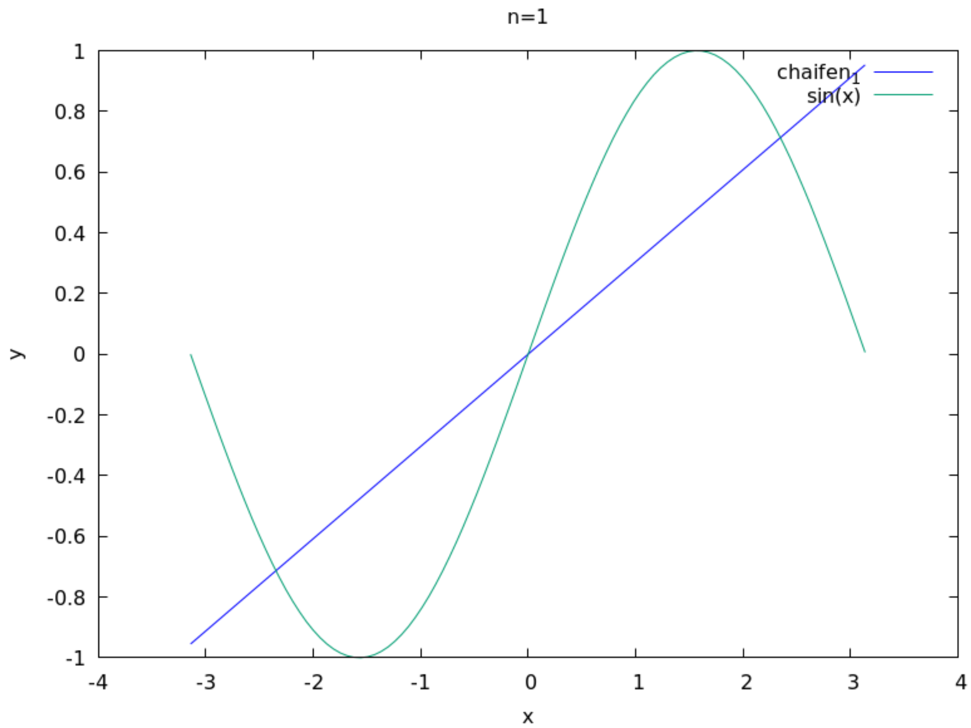
1 次：y=0.3039636x

2 次：y=0.3039636x

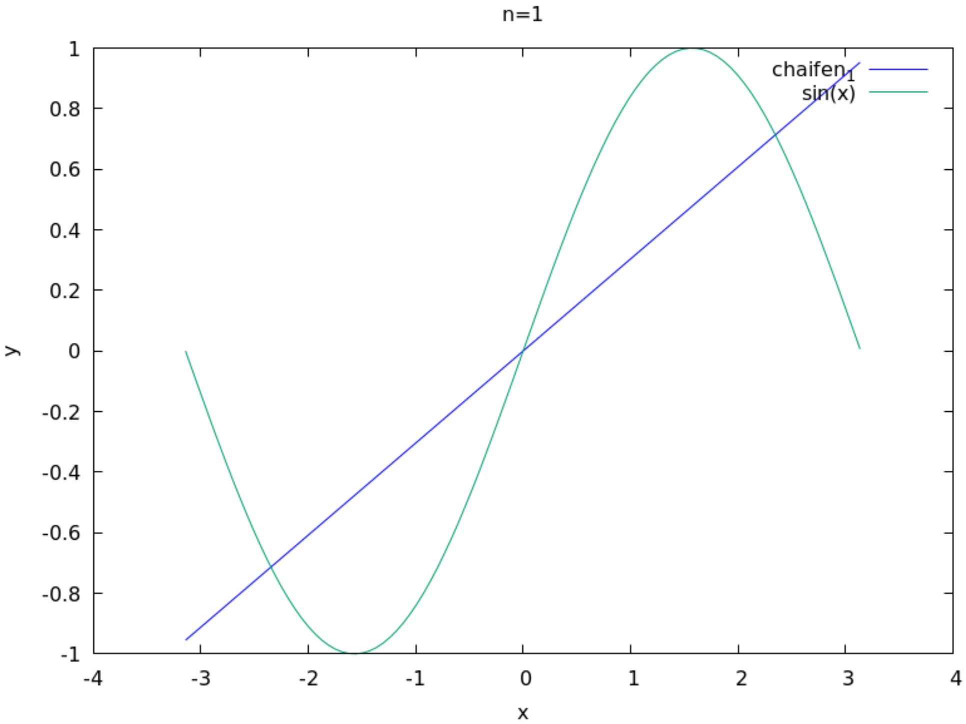
3 次：y=-0.0933877x^3+0.85698333

4 次：y=-0.0933877x^3+0.85698333

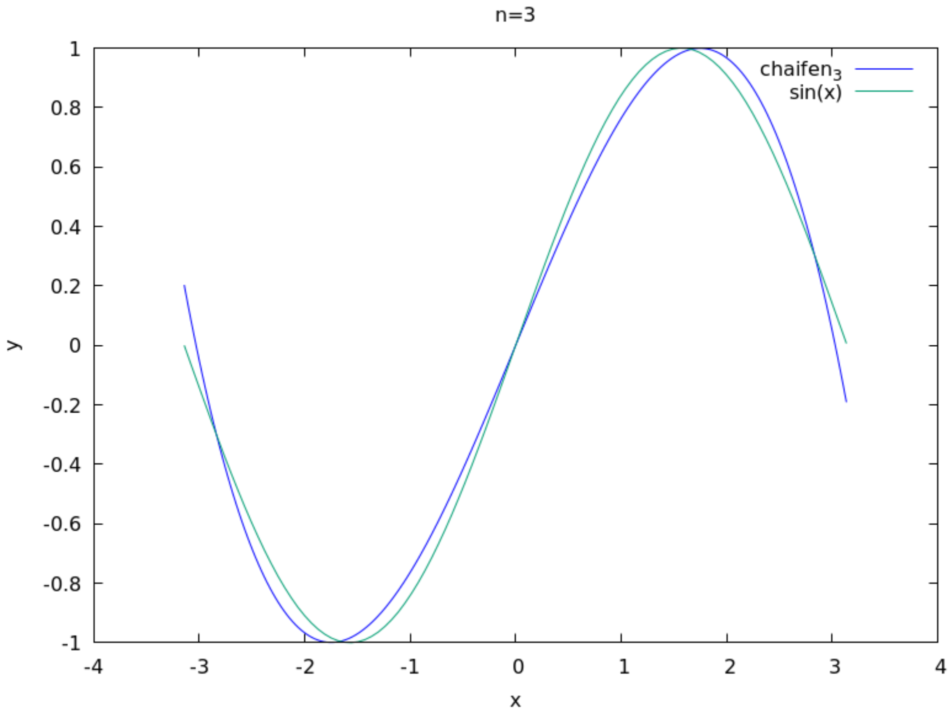
5 次：y= 0.00564312x^5- 0.15527141x^3+ 0.98786214x



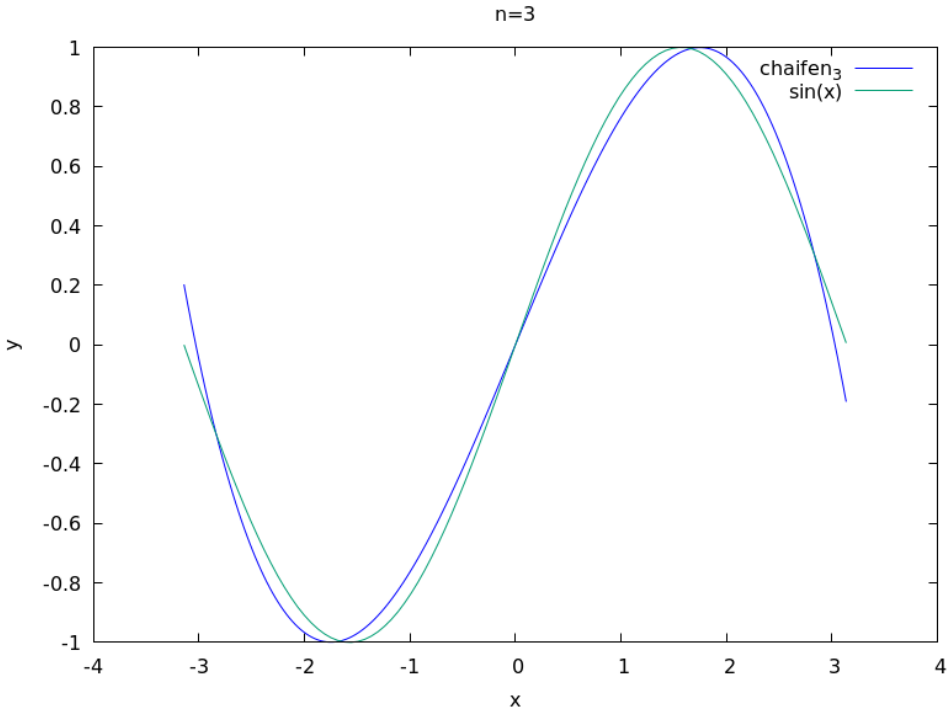
1 次



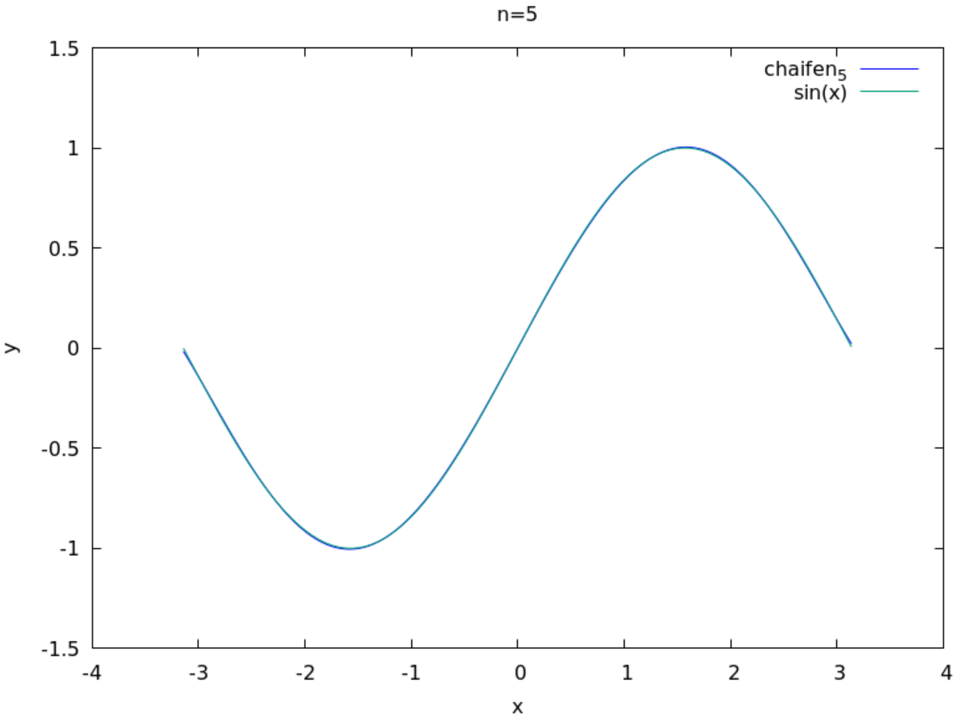
2 次



3 次



4 次



5 次

4.结果分析与上机体会

4.1结果分析：

通过观察结果，可以发现在 n=1 时，在 x∈[-Π,Π]范围内泰勒展开后的范围为[-Π,Π]而最佳平方逼近多项式的范围为[-0.954928，0.954928]，平方逼近多项式的范围与真实范围更加接近，对于高次展开，可以观察到平方逼近多项式展开在 n=5 时，多项式的图像几乎与 sin（x）重合了，证明拟合效果很好，而泰勒展开在定义域两端的拟合效果不是很好，在定义域中间拟合效果较好。而对于泰勒展开和平方逼近多项式展开，它们在 n=1 和 n=2 的效果相同，在 n=3 和 n=4 的效果相同，原因是 sinx 为奇函数，本题的范围是[-Π,Π]，所以造成了以上的结果。

4.2上机体会：

本次上机实验采用了C++编写，深刻理解了泰勒展开和平方逼近多项式的含义，编译C++的时候

会使用GSL库，而GSL在windows操作系统下使用较为复杂，因此本实验在ubuntu系统中运行，学习了GDB、gnuplot等CMAKE的知识。