数值分析课程设计论文

**论文完成时间： 2018 年 12月 9 日**

目录

[数值分析课程设计论文 1](file:///G:\CppCode\数值分析\秦九韶算法课程设计论文-最终版(1).docx#_Toc532566125)

[摘要 3](#_Toc532566126)

[关键词 3](#_Toc532566127)

[引言 4](#_Toc532566128)

[背景介绍 4](#_Toc532566129)

[研究意义 4](#_Toc532566130)

[问题的提出 4](#_Toc532566131)

[1 算法理论 4](#_Toc532566132)

[1.1 多项式拆分 4](#_Toc532566133)

[1.2 算法原理 5](#_Toc532566134)

[1.3设计算法流程图 6](#_Toc532566135)

[2设计程序 7](#_Toc532566136)

[2.1 Qt介绍 7](#_Toc532566137)

[2.2 界面设计 7](#_Toc532566138)

[3 测试效果 8](#_Toc532566139)

[3.1实验的结果 8](#_Toc532566140)

[3.2实验结果的分析 9](#_Toc532566141)

[4秦九韶算法与初始算法比较 9](#_Toc532566142)

[4.1 测试数据 9](#_Toc532566143)

[4.2运行结果 9](#_Toc532566144)

[5 总结 10](#_Toc532566145)

[参考文献 10](#_Toc532566146)

[附录 11](#_Toc532566147)

# 摘要

本论文针对数学中的多项式求值的问题进行了相应的研究，通过在计算机上设计秦九韶算法，并设计相应的简单程序，通过C++软件来运行，可以解决数学上多项式求值的问题。运用秦九韶算法可以方便快速的解决多项式求值问题，进而可以方便解决代数方程求根问题。因此本次论文研究的秦九韶算法是极具有研究意义的。秦九韶算法是我国宋代的一位数学家秦九韶提出来的，该算法领先世界五六个世纪。是我国古代劳动人民智慧的结晶，是我国伟大国库中的瑰宝。直至今天该算法仍是世界上多项式求值的最先进的方法。

# 关键词

多项式求值；秦九韶算法；多项式的分解；算法优化

# 引言

背景介绍

秦九韶算法是中国古代伟大的数学家、中世纪的数学泰斗---[秦九韶](https://baike.so.com/doc/6032027-6245028.html)的算法理论之一。（秦九韶（约公元1202年－1261年），字道古，南宋末年人，出生于[鲁郡](https://baike.so.com/doc/4822725-5039335.html" \t "https://baike.so.com/doc/_blank)（今山东曲阜一带人））。秦九韶[算法](https://baike.so.com/doc/2758411.html)是中国南宋[时期](https://baike.so.com/doc/4750992.html)的数学家秦九韶提出的一种多项式[简化](https://baike.so.com/doc/5991735.html)算法。秦九韶算法的特点是通过一次式的多次计算，逐步得出高次多项式的值，即对一个n次多项式，只需做n次乘法和加法即可。即秦九韶算法是一种将一元n次多项式的求值问题转化为n个一次式的算法。其大大简化了计算过程，即使在现代，利用计算机解决多项式的求值问题时，秦九韶算法依然是最优的算法。在西方被称作霍纳算法，是以英国数学家霍纳命名的。

研究意义

[秦九韶](https://baike.so.com/doc/6032027-6245028.html)算法其最大的意义在于将求n次多项式的值转化为求n个一次多项式的值。在人工计算时，利用秦九韶算法和其中的系数表可以大幅简化运算；对于计算机程序算法而言，加法比乘法的计算效率要高很多，因此该算法仍有极大的意义，对于计算机来说，做一次乘法运算所用的时间比作一次加法运算要长得多，所以此算法极大地缩短了CPU运算时间。

问题的提出

如果多项式f(x)是多项式，则f(x)=0称为代数方程，这里由于多项式的特殊性，可以针对其特点提供更为有效的算法即秦九韶算法。通过秦九韶算法可以有效快速的求解出多项式的值，进而求出代数方程的根。

# 1 算法理论

1.1 多项式拆分

把一个n次多项式



改写成如下形式：



求多项式的值时，首先计算最内层括号内一次多项式的值，即 ：



这样，求n次多项式f(x)的值就转化为求n个一次多项式的值。

* 1. 算法原理

秦九韶算法与直接计算相比,大大节省了乘法的次数,使计算量减少,并且逻辑结构简单。

原始算法需要计算机单独对x求乘法做幂的计算，从而耗费了比较多的计算时间。

具体来说对于一个n次的[多项式函数](https://baike.so.com/doc/6906559-7128403.html" \t "https://baike.so.com/doc/_blank)，

用原始方法（用重复乘法计算幂，再把各项相加）计算出结果最多需要



次加法



次乘法。

原因是n次多项式，从最高项开始计算，依次为



根据等差数列求和公式:



得出共需要



次乘法。

而累加n个项需要n个加法。

若用秦九韶的方法计算幂则需要



次加法



次乘法。

原因是根据拆分出来的



显而易见共需要n次乘法。

如果计算中的[数值数据](https://baike.so.com/doc/1770422-1872235.html" \t "https://baike.so.com/doc/_blank)是以字节方式储存的，

那么原始方法约需要占用的字节的倍空间。

而使用秦九韶算法时，

至多只需作次加法和次乘法，最多需要占用字节的倍空间。

所以秦九韶算法比原始算法的计算效率要高很多。

1.3设计算法流程图

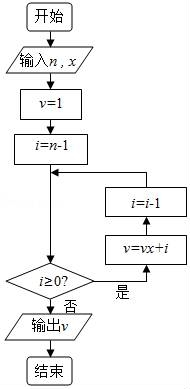


图1 理论算法图

# 2设计程序

我们选择了Qt作为我们的设计软件，因为之前学过C++，能给我们提供一个良好的技术支持。并且Qt集成了多个类库，方便我们进行各种开发

2.1 Qt介绍

Qt是一个1991年由Qt Company开发的跨平台C++图形用户界面应用程序开发框架。它既可以开发GUI程序，也可用于开发非GUI程序，比如控制台工具和服务器。Qt是面向对象的框架，使用特殊的代码生成扩展（称为元对象编译器(Meta Object Compiler, moc)）以及一些宏，Qt很容易扩展，并且允许真正地组件编程。2008年，Qt Company科技被诺基亚公司收购，Qt也因此成为诺基亚旗下的编程语言工具。2012年，Qt被Digia收购。2014年4月，跨平台集成开发环境Qt Creator 3.1.0正式发布，实现了对于iOS的完全支持，新增WinRT、Beautifier等插件，废弃了无Python接口的GDB调试支持，集成了基于Clang的C/C++代码模块，并对Android支持做出了调整，至此实现了全面支持iOS、Android、WP,它提供给应用程序开发者建立艺术级的图形用户界面所需的所有功能。基本上，Qt 同 X Window 上的 Motif，Openwin，GTK 等图形界 面库和 Windows 平台上的 MFC，OWL，VCL，ATL 是同类型的东西。

2.2 界面设计

界面分为输入框，输出框，还有帮助手册和提示框。

输入框:

让用户进行数据的输入

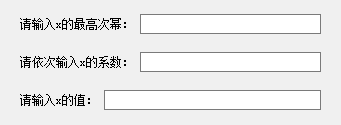


图2界面图

输出框:

软件能展示多项式，以及使用秦九韶算法分解后的多项式展示，并且能求解出值，还能比较原始算法与秦九韶算法的效率。

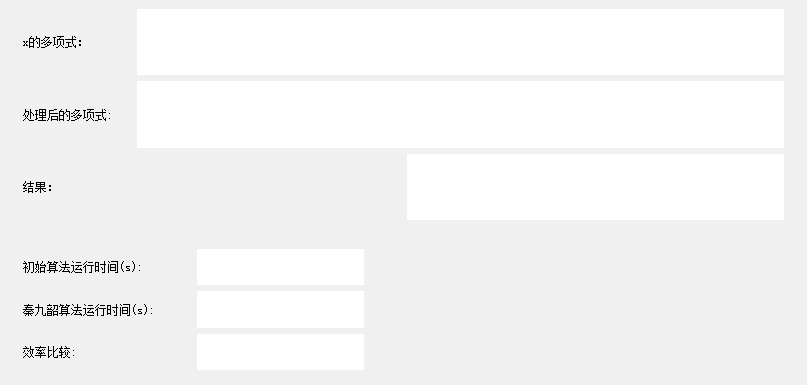


图3输出框图

帮助手册:

让用户熟悉软件使用

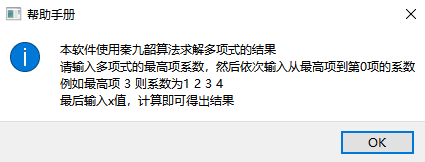


图4帮助手册图

错误提示框:

提示用户按规定的格式输入数据。

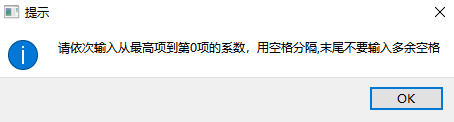


图5提示框图

# 测试效果

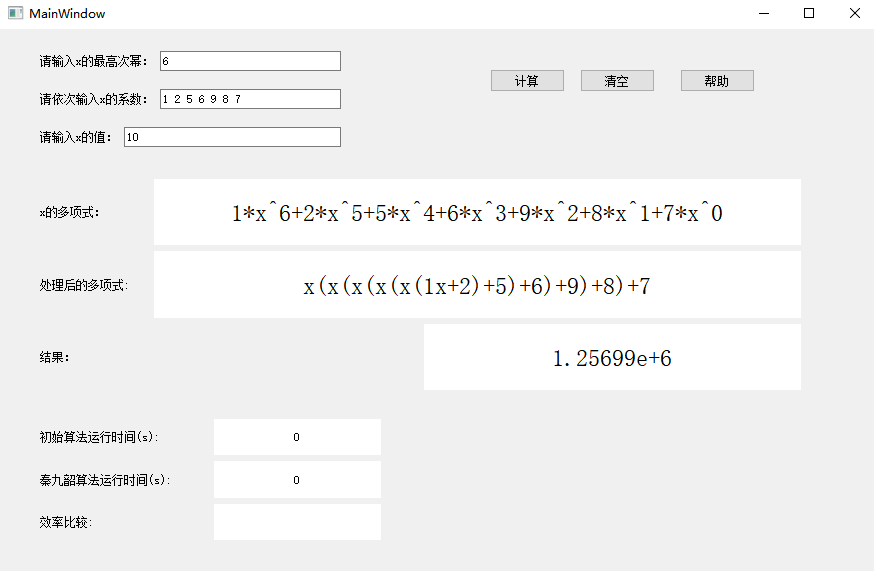
3.1实验的结果

图6实验结果图

注:由于数据量很小，所以两种算法的运行时间接近,结果为0

3.2实验结果的分析

该软件能对用户输入的多项式进行还原和拆分，并且能通过秦九韶算法来求解多项式的值，直观的显示了结果和过程。并且拥有两种算法比较速率的提示，让读者的感受到秦九韶算法的优化效果。

# 4秦九韶算法与初始算法比较

4.1 测试数据

我们选取当x最高次幂为150000次，并且x的系数等于它当前的x次幂,依次递增，即



X的值为100。

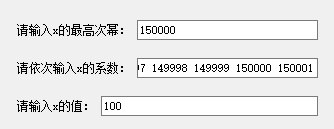


图7界面图

4.2运行结果

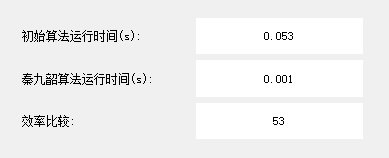


图8实验结果图

可见当x的次幂足够大时，秦九韶算法将达到初始运行算法的50倍以上，这对大数据处理运行速度的提高是立竿见影的。

4.3**两种算法计算速度比较分析**

秦九韶算法与直接计算相比,大大节省了乘法的次数,使计算量减少,并且逻辑结构简单

秦九韶算法减少做乘法的次数,在计算机上也就加快了计算的速度。而且秦九韶算法避免对自变量x单独做幂的计算,而是与系数一起逐次增长幂次,从而可提高计算。原始算法需要计算机单独对x求乘法做幂的计算，从而耗费了比较多的计算时间。具体来说就是秦九韶算法对于一个n次的[多项式函数](https://baike.so.com/doc/6906559-7128403.html" \t "https://baike.so.com/doc/_blank)，用原始方法（用重复乘法计算幂，再把各项相加）计算出结果最多需要n次加法和[n\*(n+1)]/2次乘法。若用x迭代的方法计算幂则需要n次加法和2n+1次乘法。如果计算中的[数值数据](https://baike.so.com/doc/1770422-1872235.html" \t "https://baike.so.com/doc/_blank)是以字节方式储存的，那么原始方法约需要x占用的字节的2n倍空间。而使用秦九韶算法时，至多只需作n次加法和n次乘法，最多需要x占用的字节的n倍空间。所以秦九韶算法比原始算法的计算效率要高很多。

# 5 总结

本次论文研究，通过秦九韶算法思想在计算机设计编写简单的程序来实现多项式的求值。可以提高计算的效率，简化复杂的计算过程,也通过秦九韶算法与原始的算法的比较，体现了秦九韶算法的优越性，也体现了我国古代人民的智慧是无穷的。

# 参考文献

[1]. 董付国. 秦九韶算法思想在RSA密码算法中的应用研究[J]. 计算机工程与应用, 2008,44(28):65-66

[2]. 徐士良. 数值分析算法描述与习题解答[M]. 机械工业出版社, 2003.

[3]. 吴文俊. (1987). 秦九韶与《数书九章》. 北京师范大学出版社.

[4]. 人民教育出版社教学资源分社.《高中教育教材·数学（人教版\_选修5）》．人民教育出版社，2014年.

# 附录

**Mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include<model.h>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~***MainWindow***();

void **clearUIAll**();

void **clearUIShow**();

void **setRealStr**();

void **setNewStr**();

void **originalCal**();

private slots:

void **on\_btn\_do\_clicked**();

void **on\_btn\_clear\_clicked**();

void **on\_btn\_help\_clicked**();

private:

Ui::MainWindow \*ui;

model\* Model;

};

#endif // MAINWINDOW\_H

**Model.h**

#ifndef MODEL\_H

#define MODEL\_H

#include<QVector>

#include<QStringList>

class model

{

public:

model();

void **SetX**(double x);

void **SetNum**(int num);

void **setList\_n**(QStringList strL);

double **cal**();

void **clearNum**();

QVector<double> **getList**();

private:

//val for x

double x;

int num;

QVector<double> list\_n;

};

#endif // MODEL\_H

**Mainwindow.cpp中最主要的计算代码:**

void MainWindow::**on\_btn\_do\_clicked**()

{

Model->clearNum();

clearUIShow();

QStringList strL=this->ui->le\_string->text().split(" ");

int num=this->ui->le\_mostN->text().toInt();

double X=this->ui->le\_x->text().toDouble();

Model->setList\_n(strL);

Model->SetNum(num);

Model->SetX(X);

QVector<double> t\_list=Model->getList();

if(num==0&&X==0) return;

if(num!=t\_list.size()-1)

{

QMessageBox::information(this,"提示","请依次输入从最高项到第0项的系数，用空格分隔,"

"末尾不要输入多余空格 ");

return ;

}

//显示多项式

setRealStr();

//显示操作后的多项式

setNewStr();

clock\_t time\_start=clock();

//计算结果 调用秦九韶算法

this->ui->lbl\_ans->setText(QString::number(Model->cal()));

clock\_t time\_end=clock();

double time\_ans=(double)(time\_end-time\_start)/1000.0;

this->ui->lbl\_newTime->setText(QString::number(time\_ans));

originalCal();

double rTime=ui->lbl\_realTime->text().toDouble();

double nTime=ui->lbl\_newTime->text().toDouble();

if(nTime!=0)

{

this->ui->lbl\_effiTime->setText(QString::number(rTime/nTime));

}

}

**Model.cpp中求出结果的代码:**

double model::**cal**()

{

double ans=this->list\_n[0];

for(int i=1;i<list\_n.size();i++)

{

ans=ans\*this->x+list\_n[i];

}

return ans;

}