

**《GNSS原理与应用》**

**课间实习报告**

**学 院: 遥感信息工程学院**

**班 级: 20F10**

**学 号: 2020302131201**

**姓 名: 常耀文**

**指导教师: 付建红**

**2022年10月25日**

**目录**

[1. 实习概述 3](#_Toc117690781)

[1.1 卫星位置计算概述 3](#_Toc117690782)

[1.1.1广播星历计算卫星位置 3](#_Toc117690783)

[1.1.2精密星历计算卫星位置 6](#_Toc117690784)

[1.2 实习目的 7](#_Toc117690785)

[1.3 实习所使用的编程语言与头文件使用 8](#_Toc117690786)

[1.3.1编程语言 8](#_Toc117690787)

[1.3.1使用的头文件 8](#_Toc117690788)

[1.4 编程IDE 8](#_Toc117690789)

[1.5 实习数据 8](#_Toc117690790)

[2. 程序设计与编程实现 9](#_Toc117690791)

[2.1广播星历计算程序设计 9](#_Toc117690792)

[2.1.1设计函数与功能 9](#_Toc117690793)

[2.1.2广播星历程序计算结果展示 16](#_Toc117690794)

[2.2精密星历计算程序设计 17](#_Toc117690795)

[2.2.1设计函数与功能 17](#_Toc117690796)

[3. 实验结果分析 20](#_Toc117690797)

[3.1精密星历与广播星历较差分析 20](#_Toc117690798)

[3.2精密星历与广播星历较差代码 21](#_Toc117690799)

[4. 实习心得 21](#_Toc117690800)

# 实习概述

## 卫星位置计算概述

GPS技术中的一个重要内容就是精确确定卫星轨道。在GPS导航定位中，是将GPS在空中的三维位置作为已知值并广播给用户，用户根据GPS天线至卫星的距离才可以求得自己所处的位置。因此，精确的轨道信息是导航定位的基础，而对于更精密的定位应用，则需要更精确的卫星轨道。

在GPS定位过程中，计算所需时刻的GPS位置是通过接受到的GPS卫星星历等信息加工计算得到的。计算GPS卫星位置有两种方法：利用广播星历计算卫星位置。本次实习将对这两种方法展开研究。

### 1.1.1广播星历计算卫星位置

#### 1.1.1.1算法原理

根据广播星历中卫星电文提供的轨道参数按一定的公式可计算出观测瞬间 GPS 卫星在地固系的位置。通过卫星运动的公式与星历文件中提供的数据，解求卫星的位置坐标与位置信息。

#### 1.1.1.2算法流程

1）利用所获得的星历文件读取相关的参数,将参数存储，从而方便后续计算

2）计算卫星运行的平均角速度 n



**Fig1.角速度计算公式**

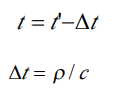
式中 µ 为 wGS84 坐标系中的地球引力常数， 且 µ =3.986005× 4 10 3 m / 2 s 。平均角速度 n。加上卫星电文给出的摄动改正数 ∆n ，便得到卫星运行的平均角速度 n:



**Fig2.利用摄动改正数计算平均角速度**

3）计算归化时间 tk

设 t’为接收机接收信号时刻，∆t 为信号传播 时间， ρ 为卫星至接收机间的距离，c 为光速，则 卫星发射信号的时刻 t 为



**Fig3.卫星发射信号时刻**

然后将观测时刻 t 归化到 GPS 时间系统：



**Fig4.将观测时刻归化到GPS系统**

式中 tk 称为相对于参考时刻toe的归化时间。 在计算tk时，应注意toe是由每星期历元(星期六/ 星期日子夜零点)开始计量的。当tk>302400s 时， 应 减 去 604800s; 当tk<-302400s 时，应加上604800s。

4） 计算观测时刻的卫星平近点角 M k 因为导航电文中已经给出参考时刻toe的平近点角M0，因此

****

**Fig5.平近点角Mk计算**

5）计算观测时刻的偏近点角 Ek

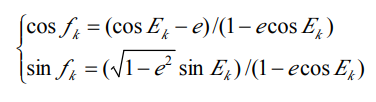
根据卫星电文给出的偏心率e和算得的Mk，利用开普勒方程：

****

**Fig5.偏近点角计算**

**进行迭代计算，解求偏近点角**

6）计算真近点角fk

****

****

**Fig6.真近点角计算**

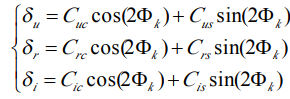
7）计算升角距角Φk

其中ω 为卫星星历中给出的近地点角距。



**Fig7.近地点角距计算**

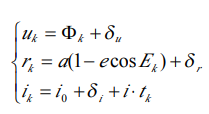
8）计算摄动改正项δ u，δ r，δ

****

**Fig8.摄动改正项计算**

Cus Cuc 为升交距的正余弦调和改正项振幅，Cis Cic 为轨道倾角的正余弦调和改正项振幅，Crs Crc 为轨道半径的正余弦调和改正项振幅。δ u，δ r，δ i 分别为升交距角的摄动量，卫星矢径的摄动量和轨道倾角的摄动量。

9）计算经摄动改正的升交距角uk，卫星矢径rk和轨道倾角ik



**Fig9.** **升交距角、卫星矢径和轨道倾角计算**

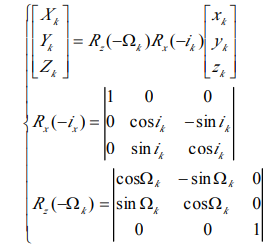
10）计算观测时刻的升交点精度Ωk

****

**Fig10.** **升交角点精度计算**

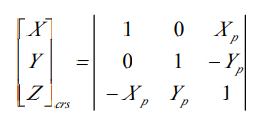
其中 Ω0 = Ωoe −GASTw,Ω0 ,Ω,toe的值可以从卫星电文中提取,ωe为地球自转的角速度，GASTw为格林尼治恒星时。

11）计算卫星在 WGS—84 坐标系的位置,将卫星在轨道平面坐标系的坐标（xk , yk ,zk） 坐标转换，即可算得观测时刻卫星在 WGS—84 坐标系的坐标

****

**Fig11.将坐标转换到WGS-84坐标系**

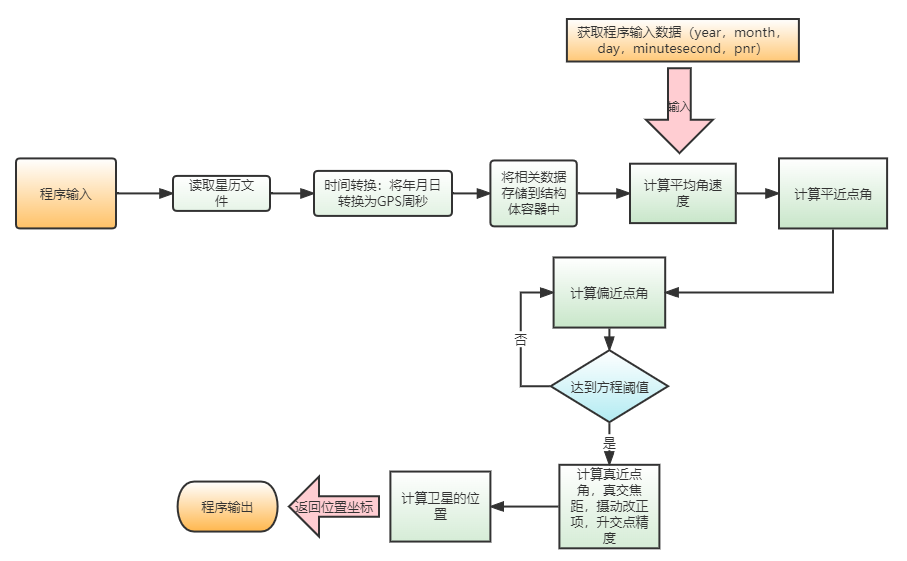
将坐标旋转矩阵带入

****

**Fig12.利用坐标旋转矩阵转换到坐标**

其中 X p Y p ,为极移改正。

#### 1.1.1.3算法设计流程框图

****

**Fig13.广播星历程序设计框图**

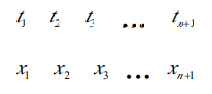
### 1.1.2精密星历计算卫星位置

#### 1.1.2.1算法原理

利用插值的方法与精密星历的文件，可以通过插值方法，拟合出在一个时间段中任意时刻的卫星坐标。

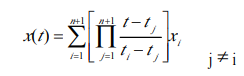
#### 1.1.2.2算法流程

根据精密星历提供的数据可以借助插值公式，内插出任意时刻(例如观测历元)的卫星坐标。目前比较常用的是拉格朗日插值公式。设在时间轴的 n+1 个节点上的卫星坐标(x分量)为



**Fig14.精密星历卫星坐标与时间对应关系**

则在任意时刻 t 的函数值的插：

****

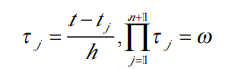
**Fig15.插值公式**

对于等间距的形式：

****

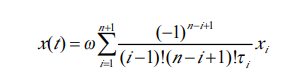
**Fig16.等间距插值公式**

令

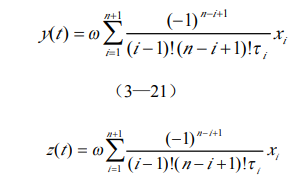


**Fig17.计算关系**

改写计算公式为：

****

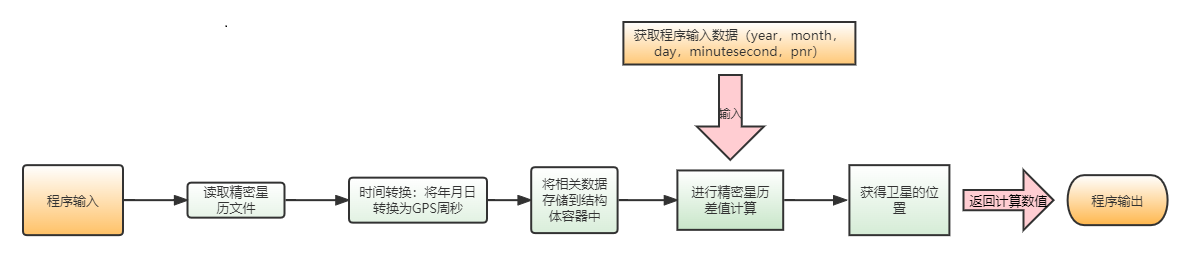
同理，对于卫星坐标的 y，z 分量，分别有



**Fig18.计算y，z坐标**

通过上述的计算流程与方式，就可以获得卫星在任何时刻的精密星历卫星坐标

#### 1.1.2.3算法设计流程图



**Fig19.精密星历计算卫星坐标值程序设计框图**

## 实习目的

实践是检验真理的唯一标准，为了更加清晰地了解有关GNSS卫星位置计算原理，采用了实践编程的手段进行知识的深化与理解，从而实践实现真理。

## 实习所使用的编程语言与头文件使用

### 1.3.1编程语言

由于本次实习要求使用C或C++编程语言进行实现相关的代码的内容，因此采用的主要编程语言是C与C++

### 1.3.1使用的头文件

编程使用了包含一般C++中常见的头文件，如iostream等头文件的使用。

#include<iostream>

#include<string>

#include<fstream>

**Fig20.程序使用的头文件**

## 编程IDE

为了实现编程任务，本次实习采用的编译器是Microsoft Visual Studio 2022，Microsoft Visual Studio（简称VS）是美国[微软公司](https://baike.baidu.com/item/%E5%BE%AE%E8%BD%AF%E5%85%AC%E5%8F%B8?fromModule=lemma_inlink)的[开发工具](https://baike.baidu.com/item/%E5%BC%80%E5%8F%91%E5%B7%A5%E5%85%B7/10464557?fromModule=lemma_inlink)包系列产品。VS是一个基本[完整](https://baike.baidu.com/item/%E5%AE%8C%E6%95%B4/32785?fromModule=lemma_inlink)的开发工具集，它包括了整个[软件生命周期](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F/861455?fromModule=lemma_inlink)中所需要的大部分工具，如[UML](https://baike.baidu.com/item/UML/446747?fromModule=lemma_inlink)工具、代码[管控](https://baike.baidu.com/item/%E7%AE%A1%E6%8E%A7/24264804?fromModule=lemma_inlink)工具、[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83/298524?fromModule=lemma_inlink)(IDE)等等。所写的[目标代码](https://baike.baidu.com/item/%E7%9B%AE%E6%A0%87%E4%BB%A3%E7%A0%81/9407934?fromModule=lemma_inlink)适用于微软支持的所有平台，包括[Microsoft Windows](https://baike.baidu.com/item/Microsoft%20Windows?fromModule=lemma_inlink)、[Windows Mobile](https://baike.baidu.com/item/Windows%20Mobile/749381?fromModule=lemma_inlink)、[Windows CE](https://baike.baidu.com/item/Windows%20CE?fromModule=lemma_inlink)、[.NET Framework](https://baike.baidu.com/item/.NET%20Framework?fromModule=lemma_inlink)、[.NET Compact Framework](https://baike.baidu.com/item/.NET%20Compact%20Framework/1926409?fromModule=lemma_inlink)和Microsoft [Silverlight](https://baike.baidu.com/item/Silverlight?fromModule=lemma_inlink) 及[Windows Phone](https://baike.baidu.com/item/Windows%20Phone/9227600?fromModule=lemma_inlink)。

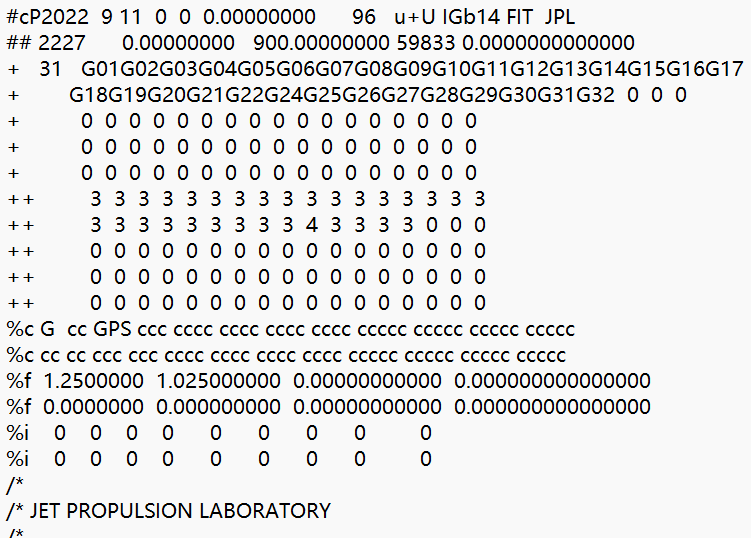
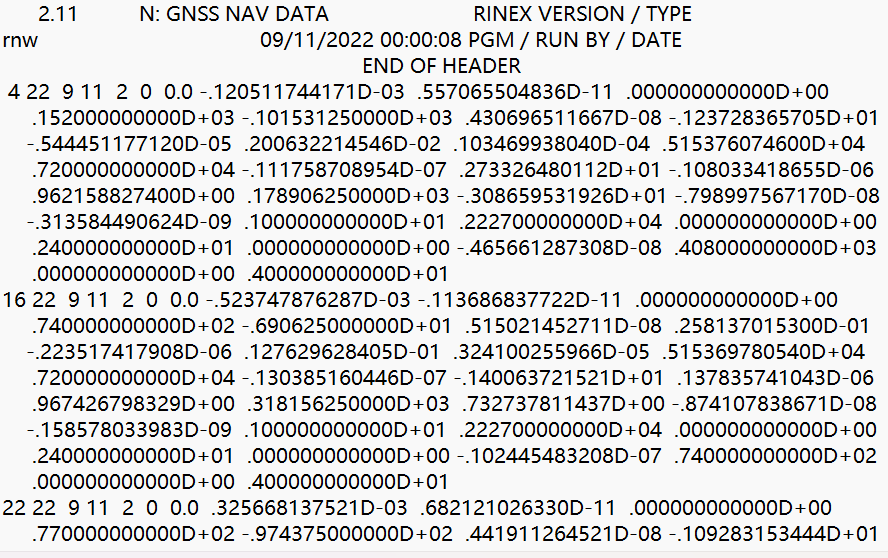


**Fig21.Microsoft Visual Studio 2022**

## 实习数据

本次实习所采用的数据是jpl22270.sp3精密星历文件与wdc62540.22n广播星历文件。

利用之前计算的流程文件就可以利用相关的方法解决有关的问题。



**Fig17.精密星历文件内容 Fig18.广播星历文件内容**

# 程序设计与编程实现

## 2.1广播星历计算程序设计

算法流程图在广播星历算法流程中已经有了设计，在这里不过多赘述，下面将介绍具体实现代码与相关功能

### 2.1.1设计函数与功能

首先设置读取广播星历文件的基本格式，在这里设置一个结构体实现：

//基本广播星历块

struct EPHEMERISBLOCK

//每小时一个卫星对应一个基本星历块

{

//PRN号

int PRN;

double a0, a1, a2;//时间改正数

//六个轨道参数

double IODE, Crs, Deltan, M0;// ORBIT - 1

double Cuc, e, Cus, SqrtA;// ORBIT - 2

double Toe, Cic, OMEGA, Cis;// ORBIT - 3

double i0, Crc, omega, OMEGAdot;// ORBIT - 4

double IDOT, GpsWeekNumber, L2C, L2P;// ORBIT - 5

double SatAccuracy, SatHealth, TGD, IODC;// ORBIT - 6

};

为了进行GPS周秒与年月日时分计算，设置了GPSTIME进行分析与计算

struct GPSTIME

{

int weekno;

double weekSecond;

};

设置卫星位置的结构体，输出想要的卫星位置坐标

struct Position

{

double X;

double Y;

double Z;

};//卫星位置

由于文件中数据存储的方式并不是科学计数法，需要将文件中的D字符替换为e字符，从而可以进行相关的实验计算，为此设置字符串替代函数

string& replace\_all(string& src, const string& old\_value, const string& new\_value) {

// 每次重新定位起始位置，防止上轮替换后的字符串形成新的old\_value

for (string::size\_type pos(0); pos != string::npos; pos += new\_value.length()) {

if ((pos = src.find(old\_value, pos)) != string::npos) {

src.replace(pos, old\_value.length(), new\_value);

}

else break;

}

return src;

}

**读取广播星历文件是本次实习的难点，在这里设计了读取函数，采用动态分配内存的方法将结果存储在了动态分配的结构体为类型的数组中，然后利用该结构体数组进行广播星历计算。**

//读广播星历文件，数据存储于上面定义的指针中

//参数：strEpheNam表示广播星历文件的完整路径

// EphemerisBlockNum 返回读取到的星历块个数

int ReadBrodcastEphemeris(string strEpheNam, int& EphemerisBlockNum, EPHEMERISBLOCK\* m\_pGpsEphemeris)

{

int HeadLineNum = 0;

int WeekNo;

double WeekSecond;

//打开文件

fstream pfEph;

pfEph.open(strEpheNam, ios::in);

if (!pfEph.is\_open()) return 0;

//读入头文件

string strLine;

while (1)

{

getline(pfEph, strLine);

HeadLineNum++;

if (strLine.find("END OF HEADER") != string::npos)

break;

}

//计算星历块数

int AllNum = 0;

while (!pfEph.eof())

{

getline(pfEph, strLine);//这个位置指针没有复位

AllNum++;

}

//临时读入星历块

EphemerisBlockNum = (AllNum + 1) / 8;

GPSTIME \* pGpsTime = new GPSTIME[EphemerisBlockNum];

if (!pGpsTime) return 0;

//将文件指针调整到数据位置

pfEph.clear();

pfEph.seekg(0, ios::beg);

for (int i = 0; i < HeadLineNum; i++)

getline(pfEph, strLine);

//定义读取的参数

int mPrn;//卫星号PRNo

int year, month, day, hour, minute;//卫星钟参考时刻

double msecond;

double a0, a1, a2;//卫星钟飘参数

double IODE, Crs, DeltN, M0;//数据星历发布时间，在轨道径向方向上周期改正正弦的振幅

double Cuc, e, Cus, sqrtA;//轨道延迹方向上周期改正余弦振幅 、扁心率、轨道延迹方向上周期改正正弦振幅 、长半轴平方根

double Toe, Cic, OMEGA, Cis;//星历参考时刻、轨道倾角周期改正余弦项振幅、参考时刻升交点赤径主项、轨道倾角周期改正正弦项振幅

double i0, Crc, omega, OMEGADOT;//参考时间轨道倾角、在轨道径向方向上周期改正余余弦的振幅、近地点角距、升交点赤径在赤道平面中的长期变化

double IDOT, L2C, GPSWeek, L2P;////轨道倾角变化率、gps周

double AccuracyofSat, HealthofSat, TGD, IODC;//卫星精度、卫星健康、电离层群迟改正数(读取文件可见为0，所以不予读取)

for (int i = 0; i < EphemerisBlockNum; i++)

{

//读取卫星PRN号，星历参考时间

getline(pfEph, strLine);

replace\_all(strLine, "D", "e");

sscanf\_s(strLine.c\_str(), "%d %d %d %d %d %d %lf %lf %lf %lf", &mPrn, &year, &month, &day, &hour, &minute, &msecond, &a0, &a1, &a2);

year += 2000;

WeekNo = Calendar2GpsTime(year, month, day, hour, minute, msecond, WeekSecond);

pGpsTime[i].weekno = WeekNo;

pGpsTime[i].weekSecond = WeekSecond;

getline(pfEph, strLine);

replace\_all(strLine, "D", "e");

sscanf\_s(strLine.c\_str(), "%lf %lf %lf %lf", &IODE, &Crs, &DeltN, &M0);

//读 Cuc,e,Cus,sqrtA

getline(pfEph, strLine);

replace\_all(strLine, "D", "e");

sscanf\_s(strLine.c\_str(), "%lf %lf %lf %lf", &Cuc, &e, &Cus, &sqrtA);

//Toe,Cic,OMEGA,Cis;

getline(pfEph, strLine);

replace\_all(strLine, "D", "e");

sscanf\_s(strLine.c\_str(), "%lf %lf %lf %lf", &Toe, &Cic, &OMEGA, &Cis);

//i0,Crc,w,OMEGADOT

getline(pfEph, strLine);

replace\_all(strLine, "D", "e");

sscanf\_s(strLine.c\_str(), "%lf %lf %lf %lf", &i0, &Crc, &omega, &OMEGADOT);

//IDOT,L2Cod,GPSWeek,L2PCod

getline(pfEph, strLine);

replace\_all(strLine, "D", "e");

sscanf\_s(strLine.c\_str(), "%lf %lf %lf %lf", &IDOT, &L2C, &GPSWeek, &L2P);

//AccuracyofSat,HealthofSat,TGD,IODC

getline(pfEph, strLine);

replace\_all(strLine, "D", "e");

sscanf\_s(strLine.c\_str(), "%lf %lf %lf %lf", &AccuracyofSat, &HealthofSat,&TGD, &IODC);

//

getline(pfEph, strLine);//把最后一个读取掉

//赋值

m\_pGpsEphemeris[i].PRN = mPrn;

m\_pGpsEphemeris[i].a0 = a0;

m\_pGpsEphemeris[i].a1 = a1;

m\_pGpsEphemeris[i].a2 = a2;

//&IODE, &Crs, &DeltN, &M0

m\_pGpsEphemeris[i].IODE = IODE;

m\_pGpsEphemeris[i].Crs = Crs;

m\_pGpsEphemeris[i].Deltan = DeltN;

m\_pGpsEphemeris[i].M0 = M0;

//&Cuc, &e, &Cus, &sqrtA

m\_pGpsEphemeris[i].Cuc = Cuc;

m\_pGpsEphemeris[i].e = e;

m\_pGpsEphemeris[i].Cus = Cus;

m\_pGpsEphemeris[i].SqrtA = sqrtA;

//Toe,Cic,OMEGA,Cis;

m\_pGpsEphemeris[i].Toe = Toe;

m\_pGpsEphemeris[i].Cic = Cic;

m\_pGpsEphemeris[i].OMEGA = OMEGA;

m\_pGpsEphemeris[i].Cis = Cis;

//i0,Crc,omega,OMEGADOT

m\_pGpsEphemeris[i].i0 = i0;

m\_pGpsEphemeris[i].Crc = Crc;

m\_pGpsEphemeris[i].omega = omega;

m\_pGpsEphemeris[i].OMEGAdot = OMEGADOT;

//iDOT,L2Cod,GPSWeek,L2PCod

m\_pGpsEphemeris[i].IDOT = IDOT;

m\_pGpsEphemeris[i].L2C = L2C;

m\_pGpsEphemeris[i].L2P = L2P;

m\_pGpsEphemeris[i].GpsWeekNumber = GPSWeek;

//AccuracyofSat,HealthofSat,TGD,IODC

m\_pGpsEphemeris[i].SatAccuracy = AccuracyofSat;

m\_pGpsEphemeris[i].SatHealth = HealthofSat;

m\_pGpsEphemeris[i].TGD = TGD;

m\_pGpsEphemeris[i].IODC = IODC;

}

pfEph.close();

if (pGpsTime) { delete[]pGpsTime; pGpsTime = NULL; }

return 1;

}

将年月日转换为GPS周秒函数

//从年 月 日 转换为GPS 周秒

int Calendar2GpsTime(int nYear, int nMounth, int nDay, int nHour, int nMinute, double dSecond, double& WeekSecond)

{

int DayofMonth = 0;

int DayofYear = 0;

int weekno = 0;

int dayofweek;

int m;

if (nYear < 1980 || nMounth < 1 || nMounth > 12 || nDay < 1 || nDay > 31) return -1;//如果时间不合法，函数返回-1

//计算从1980年到当前的前一年的天数

for (m = 1980; m < nYear; m++)

{

if ((m % 4 == 0 && m % 100 != 0) || (m % 400 == 0))

{

DayofYear += 366;

}

else

DayofYear += 365;

}

//计算当前一年内从元月到当前前一月的天数

for (m = 1; m < nMounth; m++)

{

if (m == 1 || m == 3 || m == 5 || m == 7 || m == 8 || m == 10 || m == 12)

DayofMonth += 31;

else if (m == 4 || m == 6 || m == 9 || m == 11)

DayofMonth += 30;

else if (m == 2)

{

if ((nYear % 4 == 0 && nYear % 100 != 0) || (nYear % 400 == 0))

DayofMonth += 29;

else

DayofMonth += 28;

}

}

DayofMonth = DayofMonth + nDay - 6;//加上当月天数/减去1980年元月的6日

weekno = (DayofYear + DayofMonth) / 7;//计算GPS周

dayofweek = (DayofYear + DayofMonth) % 7;

//计算GPS 周秒时间

WeekSecond = dayofweek \* 86400 + nHour \* 3600 + nMinute \* 60 + dSecond;

return weekno;

}

计算卫星位置函数，按照课本与老师教授的流程一步步操作即可实现，特别需要注意的是在迭代设计时，设置阈值为1.0e-12，可近似看为0。

//计算卫星位置

Position GetPosition(int year, int month, int day, int hour, int minute, double second, int PRN, int EphemerisBlockNum, EPHEMERISBLOCK\* m\_pGpsEphemeris) {

//首先找到卫星在数组中的位置

double WeekSecond;

if (Calendar2GpsTime(year, month, day, hour, minute, second, WeekSecond) == -1) cout << "Time Error!" << endl;

int t = 0;

for (; t < EphemerisBlockNum; t++) {

if (m\_pGpsEphemeris[t].PRN == PRN) break;

}

if (t == EphemerisBlockNum) {

cout << "No Related Satellite!";

exit(-1);

}

EPHEMERISBLOCK\* m = m\_pGpsEphemeris + t;//m为卫星数据指针

const double GM = 3.986004415E14;//地球引力常数

const double omegaE = 7.292115147E-5;//地球自转角速度

double n = sqrt(GM) / pow(m->SqrtA, 3) + m->Deltan;//卫星星历平均角速度

double M = m->M0 + n \* (WeekSecond - m->Toe);//平近点角

//计算偏近点角

double E0 = 0;

double E = M + m->e \* sin(E0);

while (abs(E - E0) >= 1e-12) {

E0 = E;

E = M + m->e \* sin(E);

}

//计算真近点角（注意判断象限）

double f;

double deno = (cos(E) - m->e);//分母

double numer = sqrt(1.0 - pow(m->e, 2)) \* sin(E);//分子

f = atan(numer / deno);

if (deno >= 0 && numer >= 0) {

f = atan(numer / deno);

}

else if (numer >= 0 && deno < 0) {

f = PI + atan(numer / deno);

}

else if (numer < 0 && deno < 0) {

f = PI + atan(numer / deno);

}

else {

f = 2 \* PI + atan(numer / deno);

}

double u0 = f + m->omega;//升交距角

double r0 = m->SqrtA \* m->SqrtA \* (1 - m->e \* cos(E));//卫星向径

//计算摄动改正项

double deltu = m->Cuc \* cos(2 \* u0) + m->Cus \* sin(2 \* u0);

double deltr = m->Crc \* cos(2 \* u0) + m->Crs \* sin(2 \* u0);

double delti = m->Cic \* cos(2 \* u0) + m->Cis \* sin(2 \* u0);

//计算改正后的升交距角、卫星向径和轨道倾角

double u = u0 + deltu;

double r = r0 + deltr;

double i = m->i0 + m->IDOT \* (WeekSecond - m->Toe) + delti;

//卫星在轨道坐标系中位置

double x = r \* cos(u);

double y = r \* sin(u);

//计算升交点经度

double L = m->OMEGA + m->OMEGAdot \* (WeekSecond - m->Toe) - WeekSecond \* omegaE;

//计算卫星在瞬时地球坐标系下的坐标

Position Final;

Final.X = x \* cos(L) - y \* cos(i) \* sin(L);

Final.Y = x \* sin(L) + y \* cos(i) \* cos(L);

Final.Z = y \* sin(i);

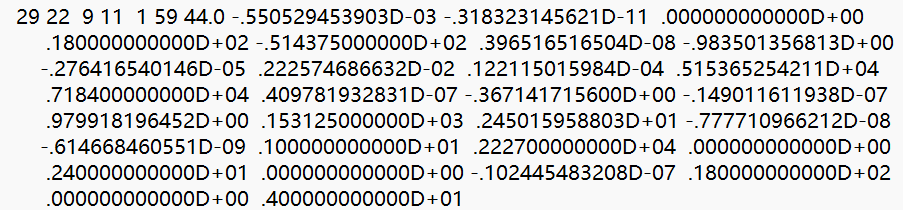
return Final;

}

最后在主程序中输入想要时刻的某颗卫星的pnr号，即可利用上述函数得到该卫星在此刻的位置。

### 2.1.2广播星历程序计算结果展示

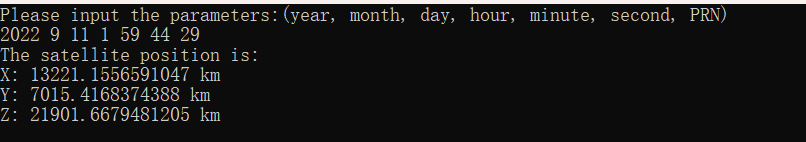
通过对文件进行分析，**本次文件中共有302块星历块**，而为了验证程序的正确性，随机选择了文件中的一块数据：



**Fig22.随机选择的数据块**

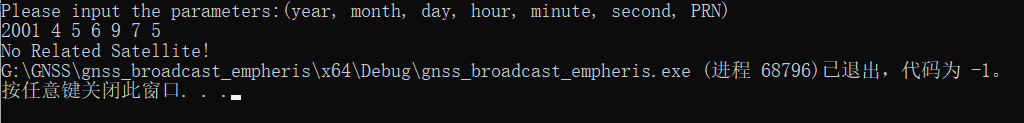
从文件可以得到该卫星pnr是29，时间是2022年9月11日1时59分44秒

在程序执行中输入上述数值，可以发现结果：

****

**Fig23.程序可行性验证**

为了保护程序，如果输入的数据并不符合要求，或者文件没有打开，会返回相应的提示，比如输入卫星的pnr号错误，会返回以下结果：



**Fig24.程序保护计算**

当输入的数据不符合预期时，输出了NO related Satellite！

**通过上述的程序运行结果可以证明程序是完全可以运行的，证明了广播星历计算的可执行性！**

## 2.2精密星历计算程序设计

算法流程图在精密星历算法流程中已经有了设计，在这里不过多赘述，下面将介绍具体实现代码与相关功能

### 2.2.1设计函数与功能

读取精密星历文件函数

int readpreccisonEphemeris(string strfilename,int& EpheremisBlockNum, Coordinates\*&m\_pGpsEphemeris) {

ifstream fin(strfilename);

if (!fin.is\_open()) {

return 0;

}

//数出有多少个时段的数据

EpheremisBlockNum = 0;

char buf[1024];

int size = sizeof(buf);

for (int i = 0; i < 22; i++) {//读取前二十行文件

fin.getline(buf, size);

}

while (fin.getline(buf, size)) {

if (buf[0] == '\*') {

++EpheremisBlockNum;

}

}

fin.clear();

fin.seekg(0, ios::beg);

//申请内存

m\_pGpsEphemeris = new Coordinates[EpheremisBlockNum];

for (int i = 0; i < 22; i++) {

fin.getline(buf, size);

}

//读文件

//定义计时器

int i = -1;

int PRN;

double x, y, z;

int nYear;

int nMounth;

int nDay;

int nHour;

int nMinute;

double dSecond;

while (fin.getline(buf, size)) {

if (buf[0] == 'P' && buf[1] == 'G') {

sscanf\_s(buf + 2, "%d %lf %lf %lf", &PRN, &x, &y, &z);

Position pt;

x \*= 1000;

y \*= 1000;

z \*= 1000;

pt.X = x;

pt.Y= y;

pt.Z = x;

pt.PRN = PRN;

m\_pGpsEphemeris[i].coordinate.push\_back(pt);

}

else if (buf[0] == '\*') {

++i;

sscanf\_s(buf + 1, "%d %d %d %d %d %lf", &nYear, &nMounth, &nDay, &nHour, &nMinute, &dSecond);

m\_pGpsEphemeris[i].time.nYear = nYear;

m\_pGpsEphemeris[i].time.nMounth = nMounth;

m\_pGpsEphemeris[i].time.nDay = nDay;

m\_pGpsEphemeris[i].time.nHour = nHour;

m\_pGpsEphemeris[i].time.nMinute = nMinute;

m\_pGpsEphemeris[i].time.dSecond = dSecond;

}

}

fin.close();

return 1;

}

精密星历文件有着22行头文件，因此在读取相关数据时要先取出头文件，在遇到\*后存储时间，并且存储数据快的值，即每个点的x，y，z坐标。

由于插值函数使用C++较为复杂，在这里选择的python实现拉格朗日插值的功能，由于老师没有要求本部分的提交，在这里不予展示。

精密星历文件与广播星历求精度函数

int num;

EPHEMERISBLOCK\* m\_pGpsEphemeris = NULL;

string ph = "wdc62540.22n";

Ephemeris2Coordinate(2022, 9, 11, 2, 0, 0,ph,coordinate, num);

int number;

Coordinates\* data = nullptr;

string sh = "jpl22270.sp3";

readpreccisonEphemeris(sh, number, data);

system("pause");

Time time(2022, 9, 11, 2, 0, 0);

Coordinates output;

for (int i = 0; i < number; ++i)

{

if (data[i].time == time)

{

output = data[i];

break;

}

}

for (int i = 0; i < num; ++i)

{

for (int j = 0; j < output.coordinate.size(); ++j)

{

if (coordinate[i].PRN == output.coordinate[j].PRN)

{

double delta\_x = abs(coordinate[i].X - output.coordinate[j].X),

delta\_y = abs(coordinate[i].Y - output.coordinate[j].Y),

delta\_z = abs(coordinate[i].Z- output.coordinate[j].Z);

cout.setf(ios::fixed);

cout << "广播星历：" << coordinate[i] << "精密星历：" << output.coordinate[j] << fixed << setprecision(3)

<< "delta\_x:\t" << delta\_x << "\tdelta\_y:\t" << delta\_y << "\tdelta\_z:\t" << delta\_z <<

"\tdelta:\t" << sqrt(pow(delta\_x, 2) + pow(delta\_y, 2) + pow(delta\_z, 2)) << endl << endl;

cout.unsetf(ios::fixed);

}

}

}

delete[] coordinate;

coordinate = nullptr;

delete[] data;

data = nullptr;

在这里选取2022年9月11日2时0分0秒的全部卫星坐标数据求取均方差，由于解算的不同，最后对数据分析后可以得出误差均值在8.524m，因为用时间跨度较大的数据去解算可能存在偏差，因此在处理较大偏差之后，再进行均值计算。

# 实验结果分析

## 3.1精密星历与广播星历较差分析

在广播星历文件中，将所有的在9月11日两时的卫星位置利用广播星历计算，用于与精密星历进行误差分析。

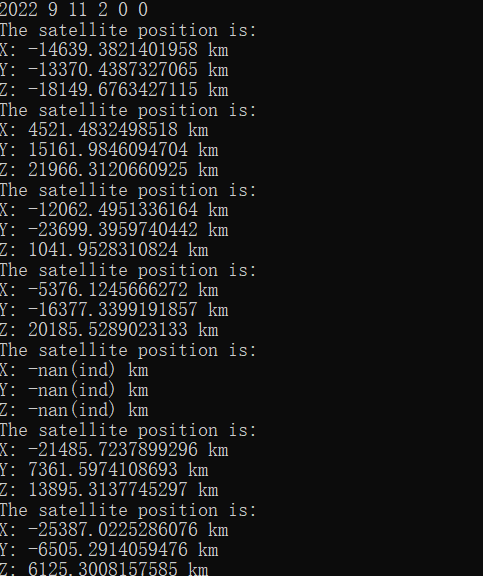
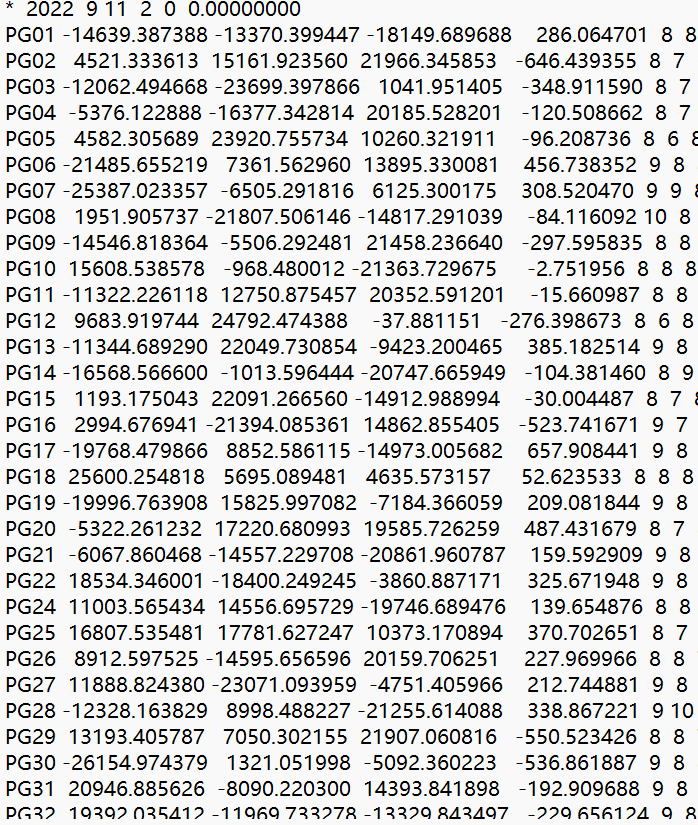
 

Fig25.广播星历输出坐标文件 Fig26.精密星历坐标文件

利用python读取分析后，进行计算可以发现，在本时间点的各卫星较差之和为

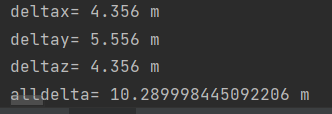


Fig27.精密星历与广播星历较差

通过函数，可以分析任意时刻的较差，可以发现，时间跨度较大的广播星历与精密星历解算时可能存在较大的较差。

## 3.2精密星历与广播星历较差代码

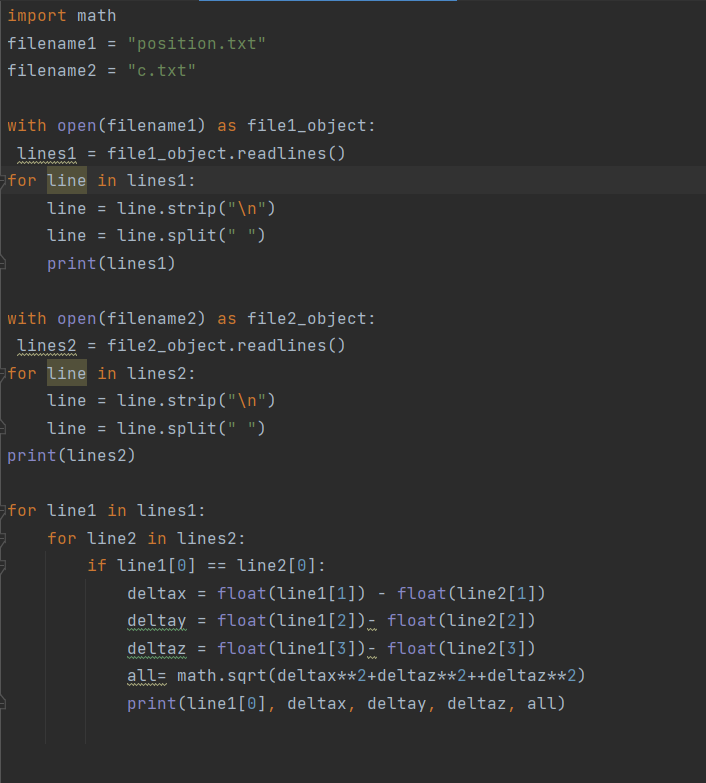


Fig28.精密星历与广播星历较差分析代码

# 实习心得

本次GNSS实习使用面向对象语言C++进行程序设计，编写了一个完整的广播星历读取与计算函数对实验数据进行计算，输出输入时刻卫星的位置。深入理解了卫星定位的原理，通过自我编程加强动手能力的培养，对实验结果的分析，增强综合运用所学知识解决实际问题的能力。并在实习过程中有不断试错、不断探究的过程，是一次完整而有收获的实习，并且在实习过程中，及时使用了老师教授的知识，丰满了GNSS的知识体系，对于接下来的课程学习具有重要的意义。

综上，本次实习兼顾了理论与实践，做到了实践体现真知，对于我有极大的裨益。