课程编号： **课程性质：必修**

卫星导航算法与程序设计 1

成果总结报告

学院：测绘学院

专业：导航工程（智能导航实验班）

姓名：秦旗峰

学号：**2023302143029**

2025 年 3 月 31 日 至 2025 年 7 月 30 日

目录

1 程序设计概述

1.1 软件功能

本软件是基于 NovAtel OEM719、司南 K708、和芯星通 UB480 等 GNSS 板卡开发的卫星导航定位系统，主要实现 GPS/BDS 双系统的单点定位、测速及数据处理功能，支持实时数据采集与文件解析，具体功能如下：

- **数据读取与解码**：软件支持两种数据输入模式，即文件模式和实时数据流模式。在文件模式下，程序从二进制文件中读取数据；在实时模式下，通过串口（COM）或网络（TCP/IP）实时接收 GNSS 板卡数据，利用网络套接字从服务器接收数据。接收到的数据经过 DecodeNovOem7Dat 函数进行解码，解析出观测值数据和星历数据，分别存储在 EPOCHOBSDATA 和 GPSEPHREC 结构体中。
- **卫星位置与钟差计算**：通过 ComputeSatPVTAtSignalTrans 函数，结合解码得到的观测值和星历数据，计算卫星在信号发射时刻的位置、速度和钟差。该函数会根据卫星所属系统不同而调用不同的计算方法，如 CompGPSSatPVT 和 CompBDSSatPVT 分别用来计算 GPS 和 BDS 系统的卫星位置，并考虑了地球自转的影响。
- **单点定位解算（SPP）**：SPP 函数是本软件的核心算法。它利用计算出的卫星位置、对流层改正和无电离层组合观测值迭代计算出用户的位置和钟差。在计算过程中，会根据是否同时存在 GPS 和 BDS 卫星，采用不同的矩阵运算方法进行最小二乘解算，并计算定位误差和精度。
- **单点测速解算（SPV）**：SPV 函数用于计算用户的速度。根据卫星的位置、速度和观测值中的多普勒频移，通过矩阵运算求解用户的速度和速度误差。
- **坐标转换与结果输出**：将计算得到的用户位置从 XYZ 直角坐标系转换到 ENU 站心地平坐标系，并将定位结果（包括位置、钟差、PDOP、误差等）输出到控制台，同时将 ENU 坐标误差写入文件以备可视化处理。

1.2 接口设计

软件的接口主要通过函数调用的方式实现，各个模块之间通过结构体进行数

据传递。

• 数据通信接口

文件模式：在 main 函数中，用户选择文件模式后，程序使用 fopen_s 函数打开二进制文件，将文件数据读取到缓冲区 buff 中。

实时模式：通过 Socket 套接字连接 GNSS 板卡（IP：47.114.134.129，端口：7190），接收实时数据流。OpenSocket 函数创建网络套接字并连接到服务器，通过 recv 函数接收服务器发送的数据到缓冲区 buff。

•数据解码接口：DecodeNovOem7Dat 函数接收缓冲区 buff、数据长度 len、观测值结构体指针 obs、GPS 和 BDS 星历结构体数组指针 geph、beph 以及参考位置结构体指针 pos 作为输入。该函数对缓冲区中的二进制数据进行解码，将解码后的观测值和星历数据存储在相应的结构体中。

• 卫星位置计算接口：ComputeSatPVTAtSignalTrans 函数接收观测值结构体指针 Epk、GPS 和 BDS 星历结构体数组指针 Eph、BDSEph 以及用户位置数组 UserPos 作为输入。函数根据输入的观测值和星历数据，计算卫星在信号发射时刻的位置、速度和钟差，并更新观测值结构体中的 SatPVT。

• 单点定位解算接口：SPP 函数接收观测值结构体指针 Epoch、GPS 和 BDS 星历结构体数组指针 GPSEph、BDSEph 以及定位结果结构体指针 Res 作为输入。函数利用卫星的位置和观测值，通过迭代计算求解用户的位置和钟差，并将解算结果存储在定位结果结构体中。

• 单点测速解算接口：SPV 函数接收观测值结构体指针 Epoch 和定位结果结构体指针 Res 作为输入。函数根据卫星的位置、速度和观测值中的多普勒频移，计算用户的速度和速度误差，并更新定位结果结构体中的 Vel 和 SigmaVel 成员。

1.3 编程语言

本软件使用 C++语言编写，C++具有高效、灵活和面向对象的特点，适合处理复杂的计算和数据结构。编程环境为 Visual Studio2022，软件仅使用了标准库中的输入输出流、文件操作、数据计算和网络编程等头文件，提高了代码的可移植性和可维护性。

1.4 完成情况

软件已经实现了单点定位的基本功能，包括数据读取、解码、卫星位置计算、单点定位和解算、坐标转换等。支持文件或串口数据输入，定位精度达到米级。

未来，将进行 RTK 相对定位算法的编写以及将 PC 端程序移植到移动端。

现阶段，软件还存在一些可以改进的地方。例如，代码中缺乏详细的注释，对于一些复杂的计算和算法，理解起来较为困难，需要添加更多的注释；此外，软件的异常处理机制还不够完善，对于一些可能出现的错误（如文件打开失败、网络连接异常等），只是简单地输出错误信息，没有进行更完善的处理。

当然，这些问题已经在努力解决过程中，添加了许多基本的注释，对可能出现的错误也进行了容错处理，但难免会出现疏漏之处。

2 算法设计

2.1 坐标系统转换

2.1.1 常用坐标系介绍

（一）笛卡尔坐标(地心地固坐标系)

笛卡尔坐标系是现代大地测量与导航中常用的一种坐标系。该坐标系的坐标原点位于地球椭球的中心；Z 轴与地球椭球短轴(旋转轴)一致，指向北极(N);起始大地子午面与椭球赤道面的交线为 X 轴，指向起始子午线;Y 轴垂直于 X 轴和 Z 轴组成右手坐标系。空间某点的位置用三维直角坐标(X,Y,Z)来表示。

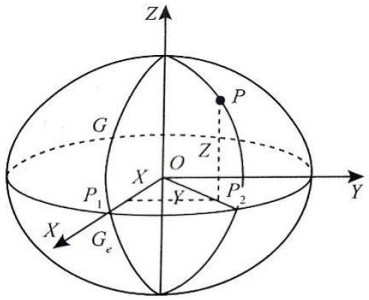

图 2-1 笛卡尔坐标系

（二）大地坐标系

地面上任意一点P'的大地坐标用三个坐标分量表示:大地纬度（latitude）B、大地经度（longitude）L和大地高（height）H。其中，大地纬度B是过P'点的椭球面法线与椭球赤道面之间的夹角，取值为$0°\sim90°$。北半球的大地纬度为正，自赤道向北量取，称为北纬；南半球的大地纬度为负，自赤道向南量取，称为南纬。大地经度L是过P'点的子午面与起始子午面间的夹角，取值$-180°\sim180°$。从起始子午面向东量取为正，称为东经，从起始子午面向西量取为负，称为西经。大地

高H是P'点沿发现至椭球面间的垂直距离$|P'P|$。

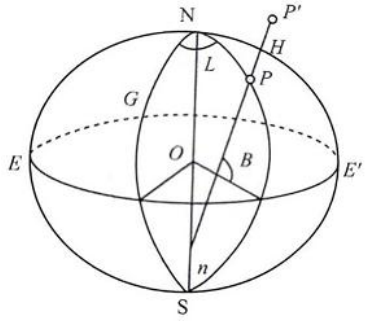

图 2-2 大地坐标系

（三）东北天测站地平坐标系

东北天测站地平坐标系，也叫站心坐标系，在卫星导航定位领域应用广泛。它以观测站为中心建立，坐标原点是测站所在位置点，X 轴指向正东（E），Y 轴指向正北（N），Z 轴与站心点的椭球法线重合且向上为正（U），与大地坐标系的高程方向一致，本质上属于地固坐标系。该坐标系主要用于计算卫星高度角和方位角，判断卫星位置，在测站坐标真值已知的情况下能够用于评估定位精度。

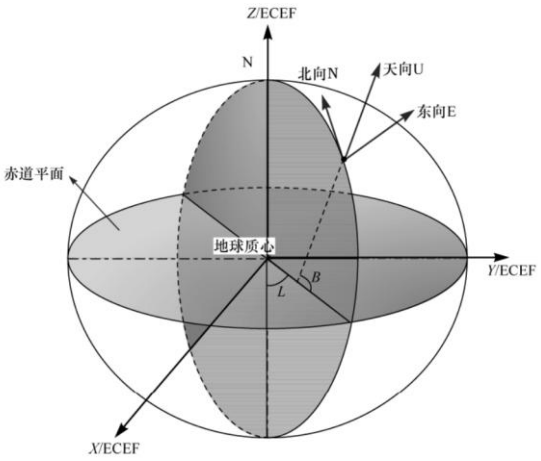

图 2-3 东北天测站地平坐标系

2.1.2 坐标系统转换

常量说明：

符号	说明
a	地球长半轴

b	地球短半轴
e	第一偏心率$e^2=\frac{a^2-b^2}{a^2}=2f-f^2$
f	扁率

（一）大地坐标到笛卡尔坐标的转换

对于一个已知大地坐标系的点$P(B,L,H)$，其笛卡尔坐标计算如下：

$$\begin{aligned} X&=(N+H)cosBcosL\\ Y&=(N+H)cosBsinL\\ Z&=[N(1-e^2)+H]sinB=\left[N\cdot\frac{b^2}{a^2}+H\right]sinB \end{aligned} \tag{1}$$

其中，N为卯酉圈半径：

$$N=\frac{a}{\sqrt{1-e^2\sin^2 B}} \tag{2}$$

（二）笛卡尔坐标到大地坐标的转换

若已知点P的笛卡尔坐标(X,Y,Z)，通过公式(1)的逆运算，不难得到大地坐标的计算公式：

$$\begin{aligned} tanB&=\frac{Z+Ne^2sinB}{\sqrt{X^2+Y^2}}\\ L&=arctan\frac{Y}{X}\\ H&=\frac{\sqrt{X^2+Y^2}}{cosB}-N \end{aligned} \tag{3}$$

纬度求解方程是一个超越方程，无法求得其解析解，一般使用迭代法求其根。这里，我们令$\Delta Z=Ne^2sinB$，并给予初值$\Delta Z_1=e^2Z$，带入纬度求解方程求得第一个B_1，并求出第二个ΔZ_2，再求出B_2……如此迭代，直到$|B_{n+1}-B_n|<\varepsilon$($\varepsilon$为一个极小值)时，$B_{n+1}$为纬度$B$较为精确的数值解。

（三）东北天测站地平坐标转换

若已知测站的大地坐标(B,L,H)，则转换矩阵定义为：

$$\boldsymbol{R}=\begin{bmatrix} -sinL & cosL & 0\\ -sinBcosL & -sinBsinL & cosB\\ cosBcosL & cosBsinL & sinB \end{bmatrix} \tag{4}$$

并可根据公式(1)求得测站的笛卡尔坐标$\boldsymbol{X_o}=[x_0,y_0,z_0]^T$。对于笛卡尔坐标为$\boldsymbol{X}=[x,y,z]^T$的点$P$，其东北天测站地平坐标$[dE,dN,dU]^T$计算公式为：

$$\begin{bmatrix} dE \\ dN \\ dU \end{bmatrix} = \boldsymbol{R}(\boldsymbol{X} - \boldsymbol{X_o}) \tag{5}$$

高度角$Elev$和方位角$Azim$为：

$$Elev = \text{atan}\left({dU}/{\sqrt{dN^2 + dE^2}}\right) \tag{6}$$

$$Azim = atan2(dE, dN) \tag{7}$$

2.2 时间系统转换

2.2.1 常用时间系统介绍

（一）通用计时法（日历表示法）

即公历时，用年、月、日、时、分、秒表示。

（二）儒略日（Julian Day）

是指从-4712 年 1 月 1 日（即公元前 4713 年 1 月 1 日）正午开始的天数。

（三）简化儒略日（MJD）

从儒略日中减去 2400000.5 天来得到，给出的是从 1858 年 11 月 17 日子夜开始的天数。

（四）GPS 时（GPS Time）

本软件使用的 GPS 时分为 GPST 和 BDST。GPST 以 1980 年 1 月 6 日子夜为起点，用周数和周内秒来表示。BDST 以 2006 年 1 月 1 日子夜为起点，用周数和周内秒来表示。BDST 与 GPST 相比，周相差 1356 周，周内秒相差-14 秒。

2.2.2 时间系统转换

（一）通用时到简化儒略日的转换

简化儒略日的计算可以分为整数天$Days$和小数天$FracDay$两个部分。记通用时的年、月、日、时、分、秒分别为Y、M、D、H、Min、S。小数部分由时分秒组成：

$$UT = H + \frac{Min}{60.0} + \frac{S}{3600.0} \tag{8}$$

$$FracDay = \frac{UT}{24.0} \tag{9}$$

儒略日的整数天由下式计算得到：

$$JD = INT(365.25 * y) + INT\big(30.6001 * (m + 1)\big) + D + \frac{UT}{24.0} + 1720981.5 \quad (10)$$

$$Days = INT(JD - 2400000.5) \quad (11)$$

其中，

$$y = \begin{cases} Y \quad , \quad if\ M > 2 \\ Y - 1 \ , else\ if\ M \le 2 \end{cases} \quad (12)$$

$$m = \begin{cases} M \quad , \quad if\ M > 2 \\ M + 12 \ , else\ if\ M \le 2 \end{cases} \quad (13)$$

（二）简化儒略日到通用时的转换

简化儒略日到通用时的转换定义了如下中间变量：JD、a、b、c、d和e。

$$JD = Days + 2400000.5 + FracDay \quad (14)$$

$$\begin{cases} a = IND(JD + 0.5) \\ b = a + 1537 \\ c = INT\left(\dfrac{b - 122.1}{365.25}\right) \\ d = INT(365.25 * c) \\ e = INT\left(\dfrac{b - d}{30.6001}\right) \end{cases} \quad (15)$$

$$\begin{cases} D = b - d - INT(30.6001 * e) \\ M = e - 1 - 12 * INT\left(\dfrac{e}{14.0}\right) \\ Y = c - 4715 - INT\left(\dfrac{7 + M}{10.0}\right) \\ H = INT(FracDay * 24) \\ Min = INT\big((FracDay * 24 - H) * 60\big) \\ S = (FracDay * 24 - H) * 3600.0 - Min * 60.0 \end{cases} \quad (16)$$

（三）简化儒略日到 GPS 时的转换

GPS 时由 GPS 周$Week$和 GPS 周内秒Sow组成，可以由简化儒略日的整数天$Days$和小数天$FracDay$按照下式计算得到：

$$\begin{cases} Week = \dfrac{INT(Days + FracDay - 44244)}{7.0} \\ Sow = (Days + FracDay - 44244 - Week * 7) * 86400 \end{cases} \quad (17)$$

（四）GPS 时到简化儒略日的转换

简化儒略日同样也能够由 GPS 时计算得到：

$$\begin{cases} Days = 44244 + Week * 7 + INT\left(\dfrac{Sow}{86400}\right) \\ FracDay = \dfrac{Sow}{86400.0} - INT\left(\dfrac{Sow}{86400}\right) * 1.0 \end{cases} \tag{18}$$

（五）通用时与 GPS 时的相互转换

利用以上四种时间转换，可以实现通用时与 GPS 时的互相转换。

2.3 信号发射时刻卫星位置计算

2.3.1 卫星位置与速度计算

计算卫星位置所涉及的常量说明：

符号	取值
π	圆周率3.1415926535898
c	光速$2.99792458 \times 10^8 m/s$
$\mu(GPS)$	GM $3.986005 \times 10^{14} m^3/s^2$
$\mu(BDS)$	GM $3.986004418 \times 10^{14} m^3/s^2$
$\dot{\Omega}_e(GPS)$	$7.2921151467 \times 10^{-5} rad/s$
$\dot{\Omega}_e(BDS)$	$7.2921150 \times 10^{-5} rad/s$

计算信号发射时刻卫星位置首先需要从数据流中解码出广播星历，利用广播星历计算卫星位置所需要的数据与说明在下表给出：

符号	说明
t_{oc}	卫星钟参考时刻
a_0	钟差
a_1	钟速
a_2	钟漂
C_{rs}	轨道半径正弦改正数
C_{rc}	轨道半径余弦改正数
$\sqrt{A}$	地球半长轴平方根
Δn	卫星角速度改正数
M_0	平近点角

C_{us}	升交点角距正弦改正项
C_{uc}	升交点角距余弦改正项
e	轨道偏心率
t_{oe}	星历参考时刻
C_{is}	轨道倾角正弦改正数
C_{ic}	轨道倾角余弦改正数
i_0	轨道倾角
$IDOT$	轨道倾角变化率
Ω_0	升交点赤经
$\dot{\Omega}$	升交点赤经变化率
ω	近地点角距

根据广播星历进行卫星位置计算可以按照如下步骤进行：

- **计算规划时间t_k**

对于信号发射时刻t（周内秒形式），结合星历参考时刻t_{oe}，计算出规划时间：

$$t_k = t - t_{oe} \tag{19}$$

- **计算卫星运动的平均角速度n**

根据广播星历给出参数$\sqrt{A}$计算参考时刻t_{oe}的平均角速度n_0：

$$n_0 = \frac{\sqrt{GM}}{\left(\sqrt{A}\right)^3} \tag{20}$$

然后根据角速度改正数Δn修正观测时刻卫星的平均角速度 n：

$$n = n_0 + \Delta n \tag{21}$$

- **计算观测瞬间卫星的平近点角M_k**

$$M_k = M_0 + n \times t_k \tag{22}$$

- **计算偏近点角E_k**

用弧度表示的开普勒方程为：

$$E_n = M_k + e \cdot sinE_{n-1} \tag{23}$$

此处的超越方程无法求得解析解，需要迭代解算偏近点角，初值设置为$E_0 = M_k$。直到$|E_n - E_{n-1}| < \varepsilon$($\varepsilon$为一个极小值)时，$E_n$是偏近点角$E_k$较为精确的数值解。

- **计算真近点角v_k和升交点角距Φ_k**

真近点角v_k：

$$v_k = atan2(\frac{\sqrt{1-e^2}sinE_k}{cosE_k - e}) \tag{24}$$

升交点角距Φ_k：

$$\Phi_k = \omega + v_k \tag{25}$$

• 计算二阶调和改正数

计算升交点角距的改正数δu_k：

$$\delta u_k = C_{us}sin2\Phi_k + C_{uc}cos2\Phi_k \tag{26}$$

计算轨道半径的改正数δr_k:

$$\delta r_k = C_{rs}sin2\Phi_k + C_{rc}cos2\Phi_k \tag{27}$$

计算轨道倾角改正数δi_k:

$$\delta i_k = C_{is}sin2\Phi_k + C_{ic}cos2\Phi_k \tag{28}$$

• 计算经过改正的升交点角距u_k

$$u_k = \Phi_k + \delta u_k \tag{29}$$

• 计算经过改正的轨道向径r_k

$$r_k = A(1 - ecosE_k) + \delta r_k \tag{30}$$

• 计算经过改正的轨道倾角i_k

$$i_k = i_0 + \delta i_k + IDOT * t_k \tag{31}$$

• 计算改正后的升交点经度

对于 GPS 卫星和北斗 IGSO 或 MEO 卫星，升交点赤经Ω_k计算公式为：

$$\Omega_k = \Omega_0 + (\dot{\Omega} - \dot{\Omega}_e)t_k - \dot{\Omega}_e * t_{oe} \tag{32}$$

而对于北斗的 GEO 卫星，计算公式为：

$$\Omega_k = \Omega_0 + \dot{\Omega} * t_k - \dot{\Omega}_e * t_{oe} \tag{33}$$

• 计算卫星在轨道平面中的位置

$$\begin{cases} x'_k = r_k cosu_k \\ y'_k = r_k sinu_k \end{cases} \tag{34}$$

• 计算卫星在空间中的位置

对于 GPS 卫星和北斗 IGSO 或 MEO 卫星，卫星的 ECEF 坐标可以由下面的公式计算得到：

$$\begin{cases} x_k = x_k' cos\Omega_{\mathrm{k}} - y_k' cosi_k sin\Omega_{\mathrm{k}} \\ y_k = x_k' sin\Omega_{\mathrm{k}} + y_k' cosi_k cos\Omega_{\mathrm{k}} \\ z_k = y_k' sini_k \end{cases} \tag{35}$$

GEO 卫星在式(35)的基础上可以由下面的公式计算得到：

$$\theta = -\frac{5}{180}\pi \tag{36}$$

$$p = \dot{\Omega}_e \cdot t_k \tag{37}$$

$$\boldsymbol{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{bmatrix} \tag{38}$$

$$\boldsymbol{R_z} = \begin{bmatrix} cosp & sinp & 0 \\ -sinp & cosp & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{39}$$

$$\begin{bmatrix} GEOX \\ GEOY \\ GEOZ \end{bmatrix} = \boldsymbol{R_z} \cdot \boldsymbol{R_x} \cdot \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} \tag{40}$$

• 计算卫星速度

卫星速度的计算公式是通过对卫星位置计算公式对信号发射时刻t求导获得的。对于非 GEO 卫星，其速度计算为：

$$\dot{E}_k = \frac{n}{1 - ecosE_k} \tag{41}$$

$$\dot{\Phi}_k = \frac{\sqrt{1-e^2}}{1 - ecosE_k}\dot{E}_k \tag{42}$$

$$\dot{u}_k = 2 * (C_{us} cos2\Phi_k - C_{uc} sin2\Phi_k)\dot{\Phi}_k + \dot{\Phi}_k \tag{43}$$

$$\dot{r}_k = Aesin(E_k)\dot{E}_k + 2 * (C_{rs} cos2\Phi_k - C_{rc} sin2\Phi_k)\dot{\Phi}_k \tag{44}$$

$$\dot{I}_k = IDOT + 2 * (C_{is} cos2\Phi_k - C_{ic} sin2\Phi_k)\dot{\Phi}_k \tag{45}$$

$$\dot{\Omega}_k = \dot{\Omega} - \dot{\Omega}_e \tag{46}$$

$$\dot{\boldsymbol{R}} = \begin{bmatrix} cos\Omega_k & -sin\Omega_k cosi_k & -(x_k' sin\Omega_k + y_k' cos\Omega_k cosi_k) & y_k' sin\Omega_k sini_k \\ sin\Omega_k & cos\Omega_k cosi_k & x_k' cos\Omega_k - y_k' sin\Omega_k cosi_k & -y_k' cos\Omega_k sini_k \\ 0 & sini_k & 0 & y_k' cosi_k \end{bmatrix} \tag{47}$$

$$\begin{cases} \dot{x_k'} = \dot{r}_k cosu_k - r_k \dot{u}_k sinu_k \\ \dot{y_k'} = \dot{r}_k sinu_k + r_k \dot{u}_k cosu_k \end{cases} \tag{48}$$

由式(41)~(48)可得非 GEO 卫星的速度为：

$$\begin{bmatrix} \dot{x}_k \\ \dot{y}_k \\ \dot{z}_k \end{bmatrix} = \dot{\boldsymbol{R}} \begin{bmatrix} \dot{x_k'} \\ \dot{x_k'} \\ \dot{\Omega}_k \\ \dot{I}_k \end{bmatrix} \tag{49}$$

而对于北斗的 GEO 卫星，其速度计算可按照下式进行：

$$\dot{\Omega}_k = \dot{\Omega} \tag{50}$$

对式(40)求导，可速度得：

$$\frac{d}{dt}\begin{bmatrix} GEOX \\ GEOY \\ GEOZ \end{bmatrix} = \boldsymbol{R_z R_x}\frac{d}{dt}\begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} + \dot{\boldsymbol{R}}_{\boldsymbol{z}}\boldsymbol{R_x}\begin{bmatrix} z_k \\ y_k \\ z_k \end{bmatrix} \tag{51}$$

2.3.2 卫星钟差与钟速计算（含相对论效应）

若卫星伪距观测值ρ已知，信号接收时刻为t_{rec}，我们可计算出粗略的信号发射时刻的钟面时t_{sv}：

$$t_{sv} = t_{rec} - \frac{\rho}{c} \tag{52}$$

记卫星钟差为Δt_{sv}，信号发射时刻t与钟面时有如下关系：

$$t = t_{sv} - \Delta t_{sv} \tag{53}$$

Δt_{sv}由下式给出：

$$\Delta t_{sv} = a_0 + a_1(t - t_{oc}) + a_2(t - t_{oc})^2 + \Delta t_r \tag{54}$$

式中，t可忽略精度，用t_{sv}代替。根据星历参数，由此可计算出一个Δt_{sv}时间偏移，将Δt_{sv}带回式(53)可计算出第一个信号发射时刻t，将t代入式(54)，计算出新的Δt_{sv}，从而计算出更加精确的信号发射时刻t和卫星钟钟差Δt_{sv}。Δt_r是相对论矫正项：

$$\Delta t_r = Fe\sqrt{A}sinE_k \tag{55}$$

其中，e和$\sqrt{A}$由星历参数得到，E_k为卫星轨道偏近点角，由星历参数计算得到。F是相对论影响系数，$F = \frac{-2\sqrt{\mu}}{c^2}$。

卫星钟速可以通过对钟差求导得到：

$$\frac{d(\Delta t_{sv})}{dt} = a_1 + 2a_2(t - t_{oc}) + \Delta \dot{t}_r \tag{56}$$

$$\Delta \dot{t}_r = Fe\sqrt{A}\cos(E_k)\,\dot{E}_k \tag{57}$$

2.3.3 地球自转改正

通过上式计算得到信号发射时刻的卫星位置$\boldsymbol{X_0} = [x_k, y_k, z_k]^T$和卫星速度$\dot{\boldsymbol{X}}_{\boldsymbol{0}} = [\dot{x}_k, \dot{y}_k, \dot{z}_k]^T$，经过相对论改正后的位置和速度分别为：$\boldsymbol{X}$和$\dot{\boldsymbol{X}}$：

$$\boldsymbol{X} = \boldsymbol{R_z}(\dot{\Omega}_e \Delta t)\boldsymbol{X_0} \tag{58}$$

$$\dot{\boldsymbol{X}} = \boldsymbol{R_z}(\dot{\Omega}_e \Delta t)\dot{\boldsymbol{X}}_{\boldsymbol{0}} \tag{59}$$

其中，$\dot{\Omega}_e$为地球自转角速度；$\boldsymbol{R_z}(\dot{\Omega}_e \Delta t)$为绕$z$轴的旋转矩阵：

$$\boldsymbol{R_z}(\alpha) = \begin{bmatrix} cos\alpha & sin\alpha & 0 \\ -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{60}$$

Δt为信号传播时间，在这里需要通过 SPP 解算出用户坐标或者设定初值与卫星位置计算集合距离后进行计算：

$$\Delta t = \frac{\sqrt{(x^s - x_r)^2 + (y^s - y_r)^2 + (z^s - z_r)^2}}{c} \tag{61}$$

式中，(x^s, y^s, z^s)和(x_r, y_r, z_r)分别是卫星和用户坐标。

2.4 误差改正与粗差、周跳探测

2.4.1 对流层延迟误差改正

对流层延迟误差是卫星信号穿过离地 50 千米以下非电离大气层时，因传播速度变化导致的路径延迟现象。该误差由干分量（90%）和湿分量（10%）构成，其中湿分量因水汽分布不确定性成为主要修正难点。在天顶方向误差约 2.4 米，仰角 5 度时可达 25 米。

本软件对对流层延迟的修正使用霍普菲尔德（Hopefiled）经验模型，标准气象元素由下表给出：

符号	说明与取值
H_0	参考面大地高 $0m$
T_0	参考面干温 $15℃ + 273.16$
p_0	参考面气压 $1013.25mbar$
RH_0	参考面相对湿度 50%

使用 Hopefiled 模型进行对流层延迟计算公式如下：

$$
\begin{gathered}
RH = RH_0 \cdot \exp\left(-0.0006396(H - H_0)\right) \\
p = p_0 \cdot \left(1 - 0.0000226 \cdot (H - H_0)\right)^{5.225} \\
T = T_0 - 0.0065(H - H_0) \\
e = RH \cdot \exp(-37.2465 + 0.213166T - 0.000256908T^2) \\
h_w = 11000m \\
h_d = 40136 + 148.72(T_0 - 273.16)\ [m] \\
K_w = 155.2 \times 10^{-7} \cdot \frac{4810}{T^2} e(h_w - H) \\
K_d = 155.2 \times 10^{-7} \cdot \frac{p}{T}(h_d - H) \\
\Delta_{Trop} = \Delta_d + \Delta_w = \frac{K_d}{sin\sqrt{E^2 + (2.5°)^2}} + \frac{K_w}{sin\sqrt{E^2 + (1.5°)^2}}
\end{gathered} \tag{62}
$$

其中，H为测站高度，T、p和RH分别为测站上的干温、气压和相对湿度。E为卫星相对于测站的高度角，单位为角度。

使用 Hopefiled 模型对对流层改正对干分量的吸收效果较好，能够达到 80%左右，而湿分量较难吸收，但总体上能够达到 SPP 的定位精度要求。

2.4.2 电离层延迟误差改正

电离层位于地球大气层上部，大约在距离地面 60 公里到 1000 公里的高度。该区域的气体分子（主要是氮气和氧气）受到太阳紫外线辐射和 X 射线等高能辐射的轰击，发生电离，产生大量的自由电子和正离子，从而对经过的电磁波产生折射效应。

电离层属于色散介质，对不同频率的信号产生不同的延迟影响，大小与信号频率的平方成反比。对于单频信号，可使用经验模型进行改正。常用的电离层经验模型为 Klobuchar 模型，此模型参数在广播星历中播发。

本软件使用双频信号，因此采用消电离层（IF）线性组合对误差进行改正。对于信号频率为f_1和f_2，伪距观测值为P_1和P_2，以米为单位的载波相位观测值为L_1和L_2，具体表达式为：

$$
P_i = \rho_r^s + dt_r - dt^s + d_{r,f_i} - d_{f_i}^s + d\rho_r^s + I_{r,f_i}^s + T_r^s + e_{r,f_i}^s \tag{63}
$$

$$
L_i = \rho_r^s + dt_r - dt^s + \lambda_{f_i} \cdot \left(N_{r,f_i}^s + \delta_{r,f_i} - \delta_{f_i}^s\right) + d\rho_r^s - I_{r,f_i}^s + T_r^s + \varepsilon_{r,f_i}^s \tag{64}
$$

其中ρ_r^s为真实卫地距，dt_r和dt^s分别为接收机钟差和卫星钟差，d_{r,f_i}和$d_{f_i}^s$分别为接收机端硬件延迟和卫星端硬件延迟。在载波相位表达式中：λ_{f_i}为载波波长，δ_{r,f_i}和$\delta_{f_i}^s$分别为接收机端硬件延迟和卫星端硬件延迟，N_{r,f_i}^s为整周未知数。$d\rho_r^s$

为卫星轨道误差，I^s_{r,f_i}是电离层延迟误差，T^s_r为对流层延迟误差，e^s_{r,f_i}为其他误差$(i = 1,2)$。

相位（以米为单位）的 IF 组合：

$$L_{IF} = \frac{1}{f_1^2 - f_2^2}(f_1^2 L_1 - f_2^2 L_2) \tag{65}$$

伪距 IF 组合：

$$P_{IF} = \frac{1}{f_1^2 - f_2^2}(f_1^2 P_1 - f_2^2 P_2) \tag{66}$$

使用以上线性组合能够更好消除电离层误差，定位结果更加准确。本软件使用P_{IF}作为解算时使用的伪距观测值。

2.4.3 伪距粗差与相位周跳探测

进行伪距粗差与相位周跳探测时，一般思路是构建组合观测值对变化缓慢或不变化的部分进行观测。本软件使用 GF（geometry-free）组合观测值和 MW（Melbourne-Wubbena）组合观测值。

GF 组合，即消几何距离的组合观测值，又称为“电离层残差组合”。组合观测值L_{GF}和P_{GF}计算如下：

$$L_{GF} = L_1 - L_2 = N^s_{r,f_1}\lambda_{f_1} - N^s_{r,f_1}\lambda_{f_1} + A\frac{f_1^2 - f_2^2}{f_1^2 f_2^2} \tag{67}$$

$$P_{GF} = P_1 - P_2 = -A\frac{f_1^2 - f_2^2}{f_1^2 f_2^2} \tag{68}$$

其中，A为电离层延迟系数，$A = 40.3 \cdot TEC$，TEC是信号传播路径上单位截面积柱体内的自由电子总数。

由于载波相位观测值噪声比伪距观测值噪声更小，因此常用式(67)作为 GF 组合观测值计算。GF 组合观测值与接收机至卫星的几何距离无关，消除了轨道误差、接收机钟差、卫星钟差和对流层延迟误差的影响，但是其整周模糊度失去了整数特性。

MW 组合观测值是不同类型不同频率观测值间的线性组合。

$$L_{MW} = \frac{1}{f_1 - f_2}(f_1 L_1 - f_2 L_2) - \frac{1}{f_1 + f_2}(f_1 P_1 - f_2 P_2) = \lambda_{MW}\left(N^s_{r,f_1} - N^s_{r,f_2}\right) \tag{69}$$

其中，$\lambda_{MW} = \frac{c}{f_1 - f_2} = 0.86m$，为 MW 组合观测值的波长。

MW 组合观测值消除了电离层、对流层、接收机和卫星钟差以及测站卫星几何距离等误差影响，仅受观测噪声和多路径效应影响；该组合观测值的计算结果只包含宽巷模糊度参数，常用于整周模糊度的确定和周跳探测。但由于 MW 组合观测值计算过程引入了伪距观测值，观测噪声较大。

已知t_{k-1}时刻的$L_{GF}(t_{k-1})$和平滑值$\bar{L}_{MW}(t_{k-1})$，当前时刻t_k可计算得到$L_{GF}(t_k)$和$L_{MW}(t_k)$，按下式进行粗差探测：

$$|L_{GF}(t_k) - L_{GF}(t_{k-1})| < \delta_{GF} \tag{70}$$

$$|L_{MW}(t_k) - \bar{L}_{MW}(t_{k-1})| < \delta_{MW} \tag{71}$$

δ_{GF}和δ_{MW}分别是 GF 和 MW 组合的检测阈值，一般设置为：$\delta_{GF} = 0.05m$，$\delta_{MW} = 3m$。若式(70)和(71)均满足，说明观测值未出现伪距粗差和相位周跳，则更新计算 MW 组合的平滑值：

$$\bar{L}_{MW}(t_k) = \frac{(k-1) \times \bar{L}_{MW}(t_{k-1}) + L_{MW}(t_k)}{k} \tag{72}$$

并将观测值设置为可用。否则将观测值设置为不可用，并重置L_{GF}和L_{MW}平滑值。

2.5 单点定位（SPP）与测速（SPV）

2.5.1 单点定位算法

单点定位算法中使用的卫星信号频率在下表给出：

符号	取值
f_{L_1}	$1575.42 \times 10^6 Hz$
f_{L_2}	$1227.60 \times 10^6 Hz$
f_{B_1}	$1561.098 \times 10^6 Hz$
f_{B_2}	$1268.520 \times 10^6 Hz$

本软件单点定位算法采用最小二乘解算。式(66)为无电离层组合的伪距观测值，对于不同系统的卫星j，将P_{IF}展开：

$$P_{G,IF}^j = \rho_G^j + c \cdot \delta t_G - c \cdot \delta t_G^j + T^j + Tgd_{G,IF}^j + \varepsilon_{G,IF}^j \tag{73}$$

$$P_{C,IF}^j = \rho_C^j + c \cdot \delta t_C - c \cdot \delta t_C^j + T^j + Tgd_{C,IF}^j + \varepsilon_{C,IF}^j \tag{74}$$

其中，下角标G和C分别代表 GPS 系统和 BDS 系统，δt_G，δt_C表示接收机对 GPS 系统和对 BDS 系统的钟差；δt_G^j，δt_C^j表示 GPS 系统和 BDS 系统的卫星j的钟差；T^j是卫星j的对流层延迟；$\varepsilon_{G,IF}^j$和$\varepsilon_{C,IF}^j$是其他未建模的噪声；$Tgd_{G,IF}^j$和$Tgd_{C,IF}^j$是 GPS 系统和 BDS 系统的卫星j的无电离层组合硬件延迟：

$$\begin{aligned} Tgd_{G,IF}^j &= 0 \\ Tgd_{C,IF}^j &= \frac{f_{B_1}^2}{f_{B_1}^2 - f_{B_2}^2} Tgd_{C,B_1}^j \end{aligned} \tag{75}$$

式中，Tgd_{C,B_1}^j是 B1I 信号的星上设备时延差$Tgd_{C,B_1}^j = c \cdot Tgd_1$，$Tgd_1$是解码星历消息时获取到的参数。

设置用户初始位置为$\boldsymbol{X}_R^0$，卫星位置为$\boldsymbol{X}_i^j$。对观测方程泰勒级数展开线性化，保留零阶项和一阶项：

$$P_{i,IF}^j = \rho_i^{j0} - c \cdot \delta t_i^j + T_i + Tgd_{i,IF}^j + l_i^j x_R + m_i^j y_R + n_i^j z_R + \delta t_i + v_{i,IF}^j \tag{76}$$

其中，$v_{i,IF}^j$是误差项，ρ_i^{j0}是代入用户初始位置$\boldsymbol{X}_R^0$，与卫星计算得到的几何距离，δt_i是以米为单位的接收机钟差；l_i^j、m_i^j和n_i^j是空间余弦，具体表达式为：

$$\begin{cases} l_i^j = \dfrac{x_R^0 - x_i^j}{\rho_i^{j0}} \\ m_i^j = \dfrac{y_R^0 - y_i^j}{\rho_i^{j0}} \\ n_i^j = \dfrac{z_R^0 - z_i^j}{\rho_i^{j0}} \end{cases} \tag{77}$$

令$w_i^j = P_{i,IF}^j - (\rho_i^{j0} - c \cdot \delta t_i^j + T_i + Tgd_{i,IF}^j)$，可得：

$$w_i^j = l_i^j x_R + m_i^j y_R + n_i^j z_R + \delta t_i + v_{i,IF}^j \tag{78}$$

假定在某历元对a颗 GPS 卫星和b颗 BDS 卫星进行了观测，化为最小二乘标准式有：

$$\boldsymbol{w} = \boldsymbol{B}\boldsymbol{x} + \boldsymbol{v} \tag{79}$$

其中：

$$\boldsymbol{w} = \left[w_G^1, \dots, w_G^a, w_C^1, \dots, w_C^b\right]^T$$
$$\boldsymbol{B} = \begin{bmatrix} l_G^1 & m_G^1 & n_G^1 & 1 & 0 \\ \vdots & \vdots & \vdots & 1 & 0 \\ l_G^a & m_G^a & n_G^a & 1 & 0 \\ l_C^1 & m_C^a & n_C^a & 0 & 1 \\ \vdots & \vdots & \vdots & 0 & 1 \\ l_C^b & m_C^a & n_C^a & 0 & 1 \end{bmatrix} \tag{80}$$
$$\boldsymbol{x} = [x_R, y_R, z_R, \delta t_G, \delta t_C]$$

我们假设所有观测为等精度观测，将观测值权阵$\boldsymbol{P}$置为单位阵，由最小二乘法，可解得：

$$\boldsymbol{x} = (\boldsymbol{B^T P B})^{-1} \boldsymbol{B^T P w} \tag{81}$$

由于我们对非线性化观测方程采取了线性化，存在线性化误差，因此将式(80)的结果作为下一次最小二乘的初值，直到$\left|\left|\boldsymbol{x}_n - \boldsymbol{x}_{n-1}\right|\right|_2 < \varepsilon$（$\varepsilon$为一个极小值），将$\boldsymbol{x}_n$作为最后的定位结果。

验后单位权中误差$\hat{\sigma}_0$常用于评定观测值的绝对精度，计算公式如下：

$$\hat{\sigma}_0 = \sqrt{\frac{\boldsymbol{v}^T \boldsymbol{P v}}{a + b - 5}} \tag{82}$$

协因数矩阵:

$$\boldsymbol{Q_{\hat{X}\hat{X}}} = (\boldsymbol{B}^T \boldsymbol{P B})^{-1} \tag{83}$$

可展开成下式：

$$\boldsymbol{Q_{\hat{X}\hat{X}}} = \begin{bmatrix} q_{\hat{x}\hat{x}} & q_{\hat{x}\hat{y}} & q_{\hat{x}\hat{z}} & q_{\hat{x}\widehat{\delta t_G}} & q_{\hat{x}\widehat{\delta t_C}} \\ q_{\hat{y}\hat{x}} & q_{\hat{y}\hat{y}} & q_{\hat{y}\hat{z}} & q_{\hat{y}\widehat{\delta t_G}} & q_{\hat{y}\widehat{\delta t_C}} \\ q_{\hat{z}\hat{x}} & q_{\hat{z}\hat{y}} & q_{\hat{z}\hat{z}} & q_{\hat{z}\widehat{\delta t_G}} & q_{\hat{z}\widehat{\delta t_C}} \\ q_{\widehat{\delta t_G}\hat{x}} & q_{\widehat{\delta t_G}\hat{y}} & q_{\widehat{\delta t_G}\hat{z}} & q_{\widehat{\delta t_G}\widehat{\delta t_G}} & q_{\widehat{\delta t_G}\widehat{\delta t_C}} \\ q_{\widehat{\delta t_C}\hat{x}} & q_{\widehat{\delta t_C}\hat{y}} & q_{\widehat{\delta t_C}\hat{z}} & q_{\widehat{\delta t_C}\widehat{\delta t_G}} & q_{\widehat{\delta t_C}\widehat{\delta t_C}} \end{bmatrix} \tag{84}$$

如果在某个历元 GPS 卫星数为 0，则协因数矩阵第四行第四列所有元素均为 0，若 BDS 卫星数为 0，则协因数第五行第五列均为 0，可以将全零行和全零列删除，矩阵变为4×4。

根据协因数矩阵，精度衰减因子$PDOP$可以由下式计算得到：

$$PDOP = \sqrt{q_{\hat{x}\hat{x}} + q_{\hat{y}\hat{y}} + q_{\hat{z}\hat{z}}} \tag{85}$$

2.5.2 单点测速算法

单点测速（SPV）利用计算得到的卫星速度、用户位置、钟速和多普勒频移观测值进行计算。多普勒频移观测方程为：

$$D_i^j = \dot{\rho}_R^j - c \cdot \delta\dot{t}^j + \delta\dot{t}_i + \dot{d}_{i,trop}^j + \dot{d}_{i,ion}^j + v^j \tag{86}$$

其中，$\dot{\rho}_R^j = \frac{(x^j - x_R)(\dot{x}^j - \dot{x}_R) + (y^j - y_R)(\dot{y}^j - \dot{y}_R) + (z^j - z_R)(\dot{z}^j - \dot{z}_R)}{\rho_R^j}$，$\rho_R^j$是卫星到接收机的欧氏距离。$\delta\dot{t}^j$是卫星钟差变化率，即卫星钟速，在计算卫星信号发射时刻钟差、钟速处已经求得。$\delta\dot{t}_i$是接收机钟差变化率，单位为m/s。$\dot{d}_{i,trop}^j$和$\dot{d}_{i,ion}^j$为对流层延迟变化率和电离层延迟变化率，单位均为m/s。v^j是误差项。

在本软件中，认为对流层延迟变化率和电离层延迟变化率很小，视其为 0，不需要使用无电离层组合，只使用D_1观测值。且接收机的 GPS 和 BDS 钟差变化率相同，共用一个钟速参数$\delta\dot{t}$。该观测方程为线性观测方程，整理可得：

$$w^j = l^j \dot{x}_R + m^j \dot{y}_R + n^j \dot{z}_R + \delta\dot{t} + v^j \tag{87}$$

其中，$w^j = D_1^j - (\dot{\rho}_R^j - c \cdot \delta\dot{t}^j)$；$l^j = \frac{x_R - x^j}{\rho_R^j}$，$m^j = \frac{y_R - y^j}{\rho_R^j}$，$n^j = \frac{z_R - z^j}{\rho_R^j}$。

假设在某个历元共对s颗卫星进行了观测，化为最小二乘标准式有：

$$\boldsymbol{w} = \boldsymbol{B}\boldsymbol{x} + \boldsymbol{v} \tag{88}$$

$$\boldsymbol{x} = [\dot{x}_R, \dot{y}_R, \dot{z}_R, \delta\dot{t}]^T \tag{89}$$

$$\boldsymbol{w} = [w^1, w^2, \cdots, w^s]^T \tag{90}$$

$$\boldsymbol{B} = \begin{bmatrix} l^1 & m^1 & n^1 & 1 \\ l^2 & m^2 & n^2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ l^s & m^s & n^s & 1 \end{bmatrix} \tag{91}$$

假设所有观测为等精度，将观测值权阵$\boldsymbol{P}$置为单位阵，可解得：

$$\boldsymbol{x} = (\boldsymbol{B}^{\boldsymbol{T}}\boldsymbol{P}\boldsymbol{B})^{-1}\boldsymbol{B}^{\boldsymbol{T}}\boldsymbol{P}\boldsymbol{w} \tag{92}$$

验后单位权中误差$\hat{\sigma}_0$为：

$$\hat{\sigma}_0 = \sqrt{\frac{\boldsymbol{v}^T\boldsymbol{P}\boldsymbol{v}}{s-4}} \tag{93}$$

协因数矩阵:

$$\boldsymbol{Q}_{\hat{\boldsymbol{X}}\hat{\boldsymbol{X}}} = (\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{B})^{-1} \tag{94}$$

3 程序设计

3.1 程序设计方案

基于 C++编程的面向对象特点和模块化编程思想，本标准单点定位程序按照：坐标系统与时间系统转换模块、数据读取与解码模块、粗差与周跳探测模块、信号发射时刻卫星位置计算模块和标准单点定位与测速模块以及矩阵运算模块等六大模块进行编写。

坐标系统与时间系统转换模块包含了常用坐标系统与时间系统的相互转换；数据读取与解码模块包含数据文件或实时数据流读取功能、二进制文件转化功能；粗差与周跳探测模块包括对流层误差计算、组合观测值计算和粗差探测功能；信号发射时刻卫星位置计算模块包括广播星历解算、卫星钟差计算和地球自转改正功能；标准单点定位与测速模块结合所有模块功能，利用最小二乘解算出用户坐标和速度；矩阵运算模块则包括了大部分矩阵运算功能，包括矩阵的加减乘、矩阵求逆、矩阵转置和矩阵删除特定列或行等功能。

各个模块之间关系如下图：

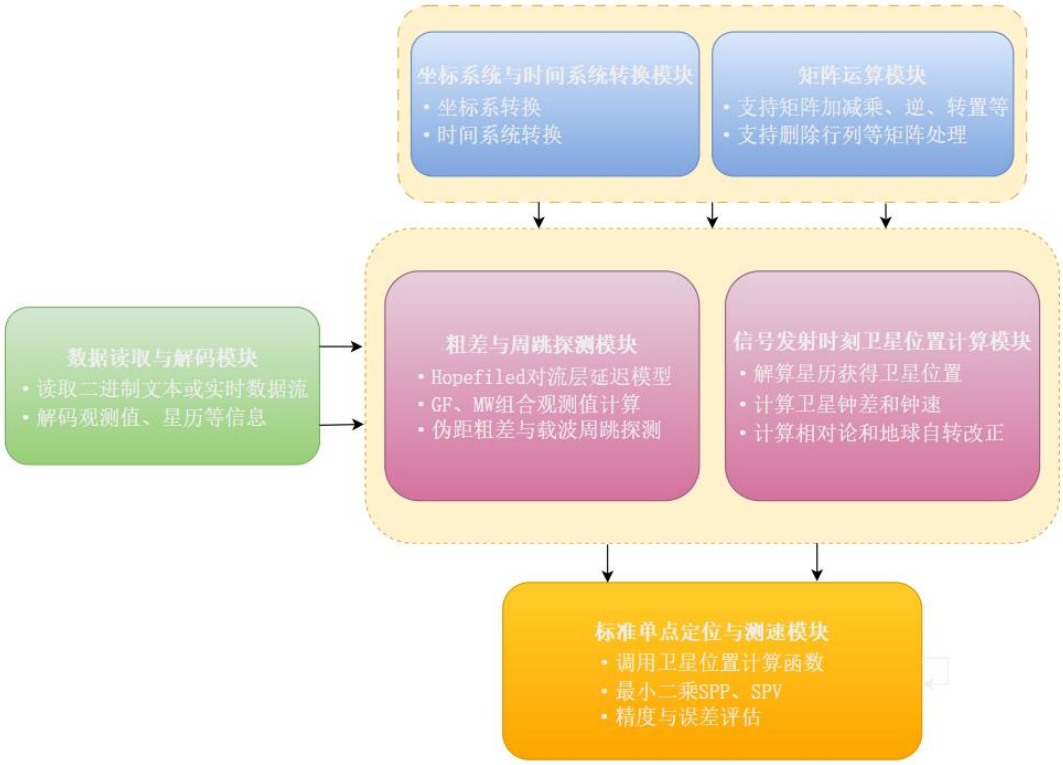

图 3-1 模块关系图

坐标系统与时间系统转换模块和矩阵运算模块为全局工具模块，为整个程序提供统一的时间基准和坐标系统支持以及提供计算能力。关于时间处理、坐标计算的模块（如卫星位置计算和定位解算）都会调用该模块进行必要的系统转换。矩阵运算模块作为通用数学支持组件，提供矩阵加减乘、求逆、转置、删除行列等一系列线性代数运算功能，用于定位解算、误差分析等多个模块，是整个系统高效运行的重要基础。

数据读取与解码模块是程序的数据入口，负责从二进制格式的文本文件或实时流中读取 GNSS 观测数据，并对其进行解析，提取出观测值和广播星历等关键信息。该模块输出的数据将直接传递给下游模块使用。

粗差与周跳探测模块对观测数据进行质量控制，主要执行粗差和周跳检测、组合观测值计算（如 MW 和 GF 组合）、对流层延迟计算等任务，对于检测不通过的数据设置为不可用，不参与后续解算。

同时，信号发射时刻卫星位置计算模块根据解码得到的星历信息，结合时间系统转换模块提供的时间信息，计算卫星在信号发射时的精确空间位置。同时，还会完成卫星钟差修正、地球自转改正等操作。

程序核心是标准单点定位与测速模块，将综合使用上述所有模块的结果，构建观测方程，调用矩阵运算模块进行最小二乘求解，迭代计算用户的三维坐标和速度信息。

本软件设计的六大模块之间形成了一条清晰的数据处理流程：从数据输入、预处理、模型构建到最终解算输出，各模块职责明确、协同配合。

3.2 程序模块设计

3.2.1 坐标系统与时间系统转换模块

（一）坐标系统转换模块包括如下图六个函数：

```
void BLHToXYZ(const GEOCOOR& geo, XYZ& xyz,const double R,const double E);   //大地坐标系转换为地心地固坐标系
void XYZToBLH(const XYZ& xyz, GEOCOOR& geo, const double R, const double E);   //地心地固坐标系转换为大地坐标系
void ToENU(const XYZ& origin_xyz, const XYZ& target_xyz,  ENUCOOR& enu, const double R, const double E);   //地心地固坐标系转换为测站地平坐标系
void ToENU(const GEOCOOR& origin_geo, const XYZ& target_xyz, ENUCOOR& enu, const double R, const double E);   //大地坐标系转换为测站地平坐标系
void CompSatElAz(const double Xr[], const double Xs[], double* Elev, double* Azim, const double R, const double E); //卫星高度角方位角计算函数
void CompEnudPos(const double X0[], const double Xr[], double denu[], const double R, const double E);  //定位误差计算函数
```

图 3-2 坐标转换函数

```
void BLHToXYZ(const GEOCOOR& geo, XYZ& xyz,const double R,const double E);   //大地坐标系转换为地心地固坐标系
```

函数与参数说明：

地心地固坐标系与大地坐标系之间的转换函数，无返回值

大地坐标系(BLH)是指经度、纬度和高程的坐标系统，地心地固坐标系(XYZ)是指以地球中心为原点的三维直角坐标系统。

* @param geo 大地坐标系坐标

* @param xyz 地心地固坐标系坐标

* @param R 地球半径

* @param E 地球偏心率

```
void XYZToBLH(const XYZ& xyz, GEOCOOR& geo, const double R, const double E);   //地心地固坐标系转换为大地坐标系
```

函数与参数说明：

地心地固坐标系转换为大地坐标系，无返回值

此函数将地心地固坐标系（XYZ）转换为大地坐标系（BLH）。需要通过迭代计算纬度和高度，直到达到所需的精度。

* @param xyz 地心地固坐标系坐标

* @param geo 大地坐标系坐标

* @param R 地球半径

* @param E 地球偏心率

```
void ToENU(const XYZ& origin_xyz, const XYZ& target_xyz,  ENUCOOR& enu, const double R, const double E);   //地心地固坐标系转换为测站地平坐标系
void ToENU(const GEOCOOR& origin_geo, const XYZ& target_xyz, ENUCOOR& enu, const double R, const double E);   //大地坐标系转换为测站地平坐标系
```

此处重载了 ToENU 函数，使其在不同情况下能够更加方便地调用。

函数与参数说明：

东北天测站地平坐标系转换函数，无返回值此函数将大地坐标系（BLH）或地心地固坐标系（XYZ）转换为测站地平坐标系（ENU）。

* @param origin_xyz 测站地心地固坐标系坐标

* @param origin_geo 测站大地坐标系坐标

* @param target_xyz 目标地心地固坐标系坐标

* @param enu 目标点的测站地平坐标系坐标

* @param R 地球半径

* @param E 地球偏心率

```
void CompSatElAz(const double Xr[], const double Xs[], double* Elev, double* Azim, const double R, const double E); //卫星高度角方位角计算函数
```

函数与参数说明：

卫星高度角和方位角计算函数，无返回值

此函数计算卫星相对于测站的高度角和方位角。

* @param Xr 测站地心地固坐标系坐标

* @param Xs 卫星地心地固坐标系坐标

* @param Elev 输出的高度角

* @param Azim 输出的方位角

* @param R 地球半径

* @param E 地球偏心率

```
void CompEnudPos(const double X0[], const double Xr[], double denu[], const double R, const double E);  //定位误差计算函数
```

函数与参数说明：

定位误差计算函数，无返回值

此函数计算测站定位误差，通过比较真实位置和估计位置的差异来计算误差。

* @param X0 真实位置坐标

* @param Xr 估计位置坐标

* @param denu 输出的定位误差（东、北、天）

* @param R 地球半径

* @param E 地球偏心率

（二）时间系统转换模块包含以下函数

```
void MjdTimeToCommonTime(const MJDTIME& mjd, COMMONTIME& common);   //将简化儒略日转换为通用时
void MjdTimeToGPSTime(const MJDTIME& mjd, GPSTIME& gps);   //将简化儒略日转换为GPS时间
void GPSTimeToMjdTime(const GPSTIME& gps, MJDTIME& mjd);   //将GPS时间转换为简化儒略日
void GPSTimeToCommonTime(const GPSTIME& gps, COMMONTIME& common);   //将GPS时间转换为通用时
void CommonTimetoGPSTime(const COMMONTIME& common, GPSTIME& gps);   //将通用时转换为GPS时间
void CommonTimetoMjdTime(const COMMONTIME& common, MJDTIME& mjd);   //将通用时转换为简化儒略日
```

图 3-3 时间转换函数

```
void MjdTimeToCommonTime(const MJDTIME& mjd, COMMONTIME& common);   //将简化儒略日转换为通用时
```

函数与参数说明：

简化儒略日转化为通用时函数，无返回值

此函数将简化儒略日转化为通用时。

* @param mjd 简化儒略日

* @param common 通用时

```
void MjdTimeToGPSTime(const MJDTIME& mjd, GPSTIME& gps);   //将简化儒略日转换为GPS时间
```

函数与参数说明：

简化儒略日转化为 GPS 时函数，无返回值

此函数将简化儒略日转化为 GPS 时间

* @param mjd 简化儒略日

* @param gps GPS 时间

```
void GPSTimeToMjdTime(const GPSTIME& gps, MJDTIME& mjd);   //将GPS时间转换为简化儒略日
```

函数与参数说明：

GPS 时转化为简化儒略日函数，无返回值

此函数将 GPS 时间转化为简化儒略日

* @param gps GPS 时间

* @param mjd 简化儒略日

```
void GPSTimeToCommonTime(const GPSTIME& gps, COMMONTIME& common);   //将GPS时间转换为通用时
```

函数与参数说明：

GPS 时转化为通用时函数，无返回值

此函数将 GPS 时间转化为通用时。

* @param gps GPS 时间

* @param common 通用时

```
void CommonTimetoGPSTime(const COMMONTIME& common, GPSTIME& gps);   //将通用时转换为GPS时间
```

函数与参数说明：

通用时转化为 GPS 时函数，无返回值

此函数将通用时转化为 GPS 时间。

* @param common 通用时

* @param gps GPS 时间

```
void CommonTimetoMjdTime(const COMMONTIME& common, MJDTIME& mjd);   //将通用时转换为简化儒略日
```

函数与参数说明：

通用时转化为简化儒略日函数，无返回值

此函数将通用时转化为简化儒略日。

* @param common 通用时

* @param mjd 简化儒略日

3.2.2 数据读取与解码模块

本软件实时数据流来源于 NovAtel OEM719 板卡的串行通信，通过网络套接字连接网络端口实现数据接收，同时也支持二进制文件解析。此模块使用的函数主要分为二进制读取函数，以及解码函数。

```
unsigned int UI4(unsigned char* p);    //读取4字节无符号整数
int I4(unsigned char* p);    //读取4字节有符号整数
short I2(unsigned char* p);    //读取2字节有符号整数
float F4(unsigned char* p);    //读取4字节浮点数
double D8(unsigned char* p);    //读取8字节双精度浮点数
unsigned short UI2(unsigned char* p);    //读取2字节无符号整数
unsigned int Crc32(unsigned char* buff, int len);    //计算CRC32校验
void DecodeRange(unsigned char* data, int len, EPOCHOBSDATA* obs);    //解码观测数据
void DecodeGpsEph(unsigned char* data, int len, GPSEPHREC geph[]);    //解码GPS星历数据
void DecodeBdsEph(unsigned char* data, int len, GPSEPHREC beph[]);    //解码北斗星历数据
void DecodePsrPos(unsigned char* data, int len, POSRES* pos);    //解码定位结果数据
int DecodeNovOem7Dat(unsigned char buff[], int& len, EPOCHOBSDATA* obs, GPSEPHREC geph[], GPSEPHREC beph[], POSRES* pos);    //解码NovAtel OEM7数据
```

图 3-4 数据读取与解码函数

（一）数据读取函数

数据读取函数包括读取 2 字节无（有）符号短整型、4 字节无（有）符号整型、4 字节单精度浮点型、8 字节双精度浮点型等函数。传入参数为数据指针，返回值为相应读取到的数值。

（二）CRC 校验程序

CRC 校验码是数据通信领域中最常用的一种差错校验码，用于验证数据在通信过程中是否存在错误，可靠性很高。其原理是以一定规则根据原始数据产生一个新的二进制序列即 CRC 校验码，将其附加在原数据二进制序列后一同播发给用户，接收端接收数据后，把原数据的二进制序列按相同规则产生二进制序列

与附加的 CRC 二进制序列进行比较，相同表示传递的数据没有问题，不相同表示传递的数据出现错误与误差。

```
unsigned int Crc32(unsigned char* buff, int len)
```

```
unsigned int Crc32(unsigned char* buff, int len)
{
    int i, j;
    unsigned int crc = 0;    //初始化CRC值为0
    for (i = 0; i < len; i++)
    {
        crc ^= buff[i];    //将当前字节与CRC值进行异或操作
        for (j = 0; j < 8; j++)    //对每个字节的8位进行处理
        {
            if (crc & 1)    //如果CRC的最低位为1
                crc = (crc >> 1) ^ POLYCRC32;    // 右移1位并与多项式进行异或
            else
                crc >>= 1;    // 如果最低位为0，则仅右移1位
        }
    }
    return crc;    //返回计算得到的CRC值
}
```

图 3-5 CRC 校验函数

函数与参数说明：

计算 CRC 校验码，返回计算得到的结果，与解码结果对比，判断数据在传输过程中是否出现错误问题。

* @param buff 输入数据缓冲区

* @param len 输入数据长度

* @return 返回计算得到的 CRC32 校验码

（三）解码函数

本软件需要解码的消息类型为：观测值、GPS/BDS 星历、定位结果。

```
void DecodeRange(unsigned char* data, int len, EPOCHOBSDATA* obs)
```

函数与参数说明：

解码观测值数据，无返回值。解码之后的观测值将按照读取顺序放在 obs 中。

* @param data 输入的字节数据缓冲区

* @param len 输入数据的长度

* @param obs 输出的观测数据结构体

在解码观测值时需要注意观测值可用性状态以及解码获得的观测值单位。

```
void DecodeBdsEph(unsigned char* data, int len, GPSEPHREC beph[])
```

函数与参数说明：

解码北斗卫星历元数据，无返回值。解码后星历将按照卫星 PRN 号存放在 beph 中，使用时也可以用 PRN 作为索引。

* @param data 输入的字节数据缓冲区

* @param len 输入数据的长度

* @param beph 输出的北斗卫星历元数据数组

```
void DecodeGpsEph(unsigned char* data, int len, GPSEPHREC geph[])
```

函数与参数说明：

解码 GPS 卫星历元数据，无返回值。同样，解码得到的 GPS 广播星历将按照 PRN 号按顺序存放。

* @param data 输入的字节数据缓冲区

* @param len 输入数据的长度

* @param geph 输出的 GPS 卫星历元数据数组

```
void DecodePsrPos(unsigned char* data, int len, POSRES* pos)
```

函数与数据说明：

解码定位结果数据，无返回值。解码得到的结果将用作参考真值计算误差。

* @param data 输入的字节数据缓冲区

* @param len 输入数据的长度

* @param pos 输出的定位结果结构体

```
int DecodeNovOem7Dat(unsigned char buff[], int& len, EPOCHOBSDATA* obs, GPSEPHREC geph[], GPSEPHREC beph[], POSRES* pos)
```

函数与参数说明：

解码 NovAtel OEM7 格式的数据。

* @param buff 输入的字节数据缓冲区

* @param len 输入数据的长度

* @param obs 输出的观测数据结构体

* @param geph 输出的 GPS 卫星历元数据数组

* @param beph 输出的北斗卫星历元数据数组

* @param pos 输出的定位结果结构体

* @return 返回解码状态，0 表示成功，非 0 表示失败

解码函数的编写需要按照 OME7 文件进行对照解码，相关文档将在附录给出。

3.2.3 粗差与周跳探测模块

粗差与周跳探测位于解码模块之后，对观测值进行操作，判断其可用性。该模块还有计算对流层延迟误差函数，对流层延迟的计算需要使用到卫星的高度角。和测站的大地高，需要解算星历和定位之后计算。

```
double Hopfield(const double H, const double Elev);     //Hopefield电离层模型
void DetectOutlier(EPOCHOBSDATA* Obs);     //粗差与周跳探测
```

图 3-6 粗差与周跳探测模块

```
double Hopfield(const double H, const double Elev)
```

函数与参数说明：

Hopefield 对流层模型，使用时需要注意传入参数的单位。

* @param H 高度（单位：米）

* @param Elev 卫星仰角（单位：度）

* @return 返回对流层延迟（单位：米）

```
void DetectOutlier(EPOCHOBSDATA* Obs)
```

函数与参数说明：

粗差与周跳探测，无返回值，直接对 Obs 内容进行操作。

* @param Obs 指向 EPOCHOBSDATA 结构体的指针，包含卫星观测数据

3.2.4 信号发射时刻卫星位置计算模块

信号发射时刻卫星位置计算模块包括星历解算、卫星钟差计算和地球自转改正。星历解算在解码之后进行，卫星钟差计算需要已知观测值后进行，地球自转改正需要获得用户坐标才能计算。该模块主要包括以下函数：

```
bool CompSatClkOff(const int Prn, const GNSSsys Sys, const GPSTIME* t, GPSEPHREC* GPSEph, GPSEPHREC* BDSEph, SATPVT* Mid);   //计算卫星钟差
int CompGPSSatPVT(const int Prn, const GPSTIME* t, const GPSEPHREC* GPSEph, SATPVT* Mid);    //计算GPS卫星位置、速度和钟差
int CompBDSSatPVT(const int Prn, const GPSTIME* t, const GPSEPHREC* BDSEph, SATPVT* Mid);    //计算BDS卫星位置、速度和钟差
int EarthRotate(double UserPos[3], const GPSEPHREC* Eph, SATPVT* Mid);    //地球自转改正
```

图 3-7 卫星位置计算模块函数

```
bool CompSatClkOff(const int Prn, const GNSSsys Sys, const GPSTIME* t, GPSEPHREC* GPSEph, GPSEPHREC* BDSEph, SATPVT* Mid);   //计算卫星钟差
```

函数与参数说明：

该函数用于计算卫星钟差和钟速。该函数利用传入的信号发射时刻t判断星历可用性，若可用性通过，则计算卫星的钟差和钟速并将其保存在Mid中。

* @param Prn 卫星编号

* @param Sys 卫星系统

* @param t 信号发射时刻

* @param GPSEph GPS 卫星星历

* @param BDSEph BDS 卫星星历

* @param Mid 卫星 PVT 结构体指针

* @return true:计算成功; false:计算失败

```
int CompGPSSatPVT(const int Prn, const GPSTIME* t, const GPSEPHREC* GPSEph, SATPVT* Mid);    //计算GPS卫星位置、速度和钟差
```

函数与参数说明：

计算 GPS 卫星位置、速度和钟差。按照星历解算方式计算卫星位置，同时在函数添加了相对论效应修正 CompSatClkOff 函数计算的钟差和钟速。

* @param Prn 卫星编号

* @param t 信号发射时刻

* @param GPSEph GPS 卫星星历

* @param Mid 卫星 PVT 结构体指针

* @return 0:计算成功; 非 0:计算失败

```
int CompBDSSatPVT(const int Prn, const GPSTIME* t, const GPSEPHREC* BDSEph, SATPVT* Mid);    //计算BDS卫星位置、速度和钟差
```

函数与参数说明：

计算 BDS 卫星位置、速度和钟差。该函数大体与 GPS 卫星星历解算相似，需要注意的是 BDS 系统中 GEO 卫星的位置和速度与其他卫星存在不同。

* @param Prn 卫星编号

* @param t 信号发射时刻

* @param BDSEph BDS 卫星星历

* @param Mid 卫星 PVT 结构体指针

* @return 0:计算成功; 非 0:计算失败

```
int EarthRotate(double UserPos[3], const GPSEPHREC* Eph, SATPVT* Mid);    //地球自转改正
```

函数与参数说明：

地球自转改正函数，该函数需要传入用户的定位坐标，对广播星历结算结果做修正。

* @param UserPos 用户位置（ECEF 坐标）

* @param Eph 卫星星历

* @param Mid 卫星 PVT 结构体指针

3.2.5 标准单点定位与测速模块

单点定位与测速是软件的核心，在该模块中 ComputeSatPVTAtSignalTrans 函数调用了信号发射时刻卫星位置计算模块中的 CompSatClkOff 函数、CompGPSSatPVT 和 CompBDSSatPVT 函数，计算信号发射时刻并计算卫星钟差与钟速、卫星位置。模块中的 SPP 函数负责调用 ComputeSatPVTAtSignalTrans 不断更新出更精确的卫星坐标和对流层延迟，在此基础上计算出更精确的接收机坐标。SPV 函数则是在进行了 SPP 之后利用多普勒观测值对用户速度进行求解。

```
void ComputeSatPVTAtSignalTrans(EPOCHOBSDATA* Epk, GPSEPHREC* Eph, GPSEPHREC* BDSEph, double UserPos[3]);    //计算信号发射时刻卫星位置和速度
bool SPP(EPOCHOBSDATA* Epoch, GPSEPHREC* GPSEph, GPSEPHREC* BDSEph, POSRES* Res);    // 单点定位函数
void SPV(EPOCHOBSDATA* Epoch, POSRES* Res);    // 单点测速函数
```

图 3-8 单点定位与测速模块函数

```
void ComputeSatPVTAtSignalTrans(EPOCHOBSDATA* Epk, GPSEPHREC* Eph, GPSEPHREC* BDSEph, double UserPos[3]);    //计算信号发射时刻卫星位置和速度
```

函数与参数说明：

计算卫星信号发射时刻位置，无返回值。该函数用于计算卫星信号发射时刻位置，首先迭代计算出信号发射时刻，再调用星历解算函数进行计算，并做地球自转改正。在这个函数中需要计算出：信号发射时刻、卫星钟差与钟速、经过地球自转改正的卫星位置与速度、对流层延迟。

* @param Epk 观测数据结构体指针

* @param Eph GPS 历元数据结构体指针

* @param BDSEph BDS 历元数据结构体指针

* @param UserPos 用户位置

```
bool SPP(EPOCHOBSDATA* Epoch, GPSEPHREC* GPSEph, GPSEPHREC* BDSEph, POSRES* Res);    // 单点定位函数
```

函数与参数说明：

单点定位计算函数。该函数迭代调用 ComputeSatPVTAtSignalTrans 函数和最小二乘对用户位置进行解算。

* @param Epoch 观测数据结构体指针

* @param GPSEph GPS 历元数据结构体指针

* @param BDSEph BDS 历元数据结构体指针

* @param Res 位置解算结果结构体指针

* @return true 定位成功, false 失败

```
void SPV(EPOCHOBSDATA* Epoch, POSRES* Res);    // 单点测速函数
```

函数与参数说明：

卫星速度计算函数，无返回值。在进行了 SPP 的基础上利用多普勒观测值对速度进行解算。

* @param Epoch 观测数据结构体指针

* @param Res 位置解算结果结构体指针

3.2.6 矩阵计算模块

矩阵计算模块涵盖了大部分矩阵运算的功能：

```
void PrintMatrix(int rows, int cols, const double* matrix, int width = 10, int precision = 4);    // 打印矩阵
void MatrixMultiply(int m1, int n1, int m2, int n2, const double M1[], const double M2[], double M3[]);    // 矩阵乘法
void MatrixAdd(int m1, int n1, const double M1[], const double M2[], double M3[]);    // 矩阵加法
void MatrixTranspose(int rows, int cols, const double* matrix, double* transposed);    // 矩阵转置
bool MatrixInverse(int n, const double* matrix, double* inverse);    // 矩阵求逆
bool MatrixInverse(int m, int n, double* matrix, double* inverse);    // 矩阵分块求逆
void MatrixExtr(int rows, int cols, int a[], int a_len,  /* 被删除的行数组及其长度*/int b[], int b_len, double m[], double n[]);    // 矩阵行列式
void MatrixSub(int m1, int n1, const double M1[], const double M2[], double M3[]);    // 矩阵减法
```

图 3-9 矩阵运算模块函数

矩阵采用一维数组+行列数表示，表达方式较为简便，可移植性高。

4 结果分析

4.1 SPP 定位结果分析

本次 SPP 定位测试持续约 9h，ENU 方向上误差如图所示：

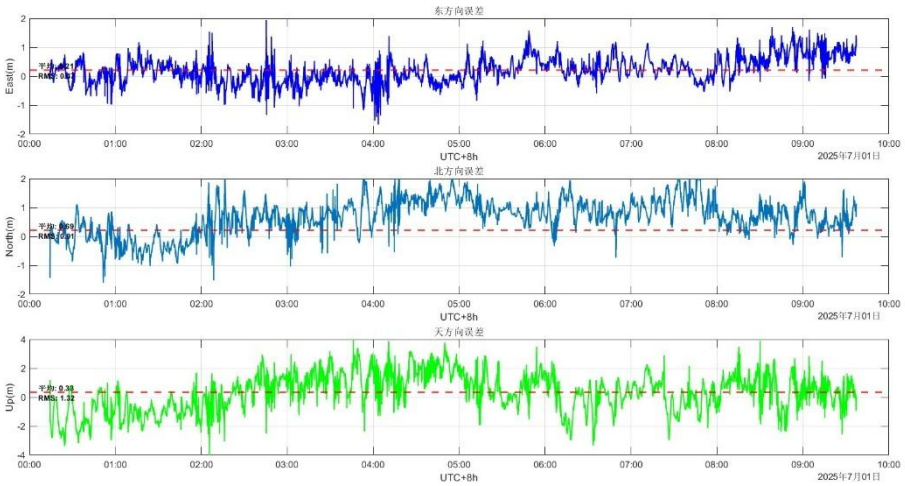

图 4-1 ENU 三方向误差时序图

北方向上平均误差为$mean_N = 0.69m$，均方根值$rms_N = 0.91m$；天方向上平均误差为$mean_U = 0.33m$，均方根值$rms_U = 1.32m$。

根据对误差时序图的统计与分析结果，标准单点定位（Standard Point Positioning, SPP）在本地东-北-天（East-North-Up, ENU）坐标系下的误差呈现出方向性差异，表现出较为典型的定位精度分布规律。具体而言，东向（E 方向）的平均误差为$mean_E = 0.21m$，均方根误差（RMS）为$rms_E = 0.47m$，说明该方向的定位误差波动较小，系统偏差和随机误差均处于较低水平，观测几何条件良好，是误差最小、最稳定的方向。这与 GNSS 卫星在东西方向上具有更优观测几何结构、DOP 值较小密切相关。

相比之下，北向（N 方向）的误差明显偏大，平均误差达到$mean_N = 0.69m$，均方根误差为$rms_N = 0.91m$，表明该方向可能存在较大的系统偏差，同时随机误差幅度也较高。造成该现象的原因可能是卫星轨道分布在地球赤道面附近，使得在中低纬度地区，南北方向的几何约束相对较弱。

三个方向上最为突出的是天向（U 方向）的误差，其平均误差虽为$mean_U = 0.33m$，但均方根误差却高达$rms_U = 1.32m$，是三个方向中波动最大的方向。这

种垂直方向误差较大的现象是 GNSS 定位中的普遍问题，其根本原因在于：GNSS 系统中可见卫星主要分布于高空，仰角较大，缺乏来自低仰角的卫星支撑，导致垂直方向的观测几何结构弱，垂直 DOP 值（VDOP）高，误差更容易在解算过程中被放大。同时，对流层延迟作为一种高度相关的系统误差，对 U 方向的影响尤为显著。

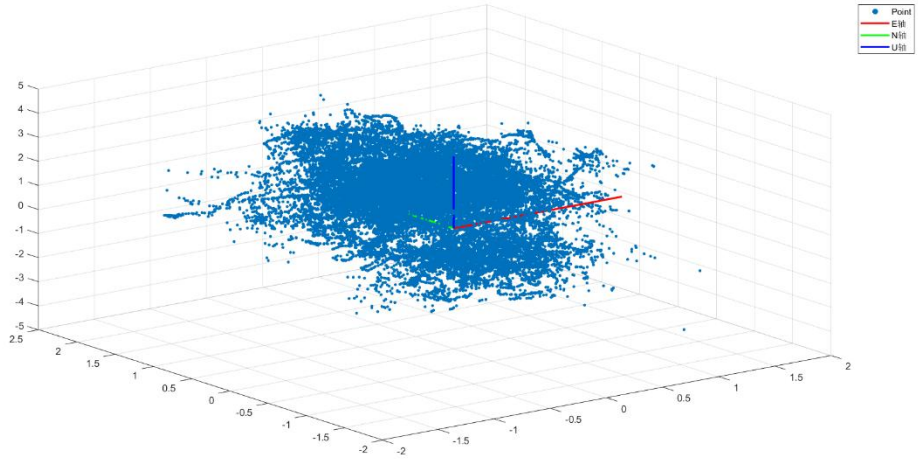

图 4-2 定位结果三位分布图

ENU 三方向误差的统计结果反映出 SPP 定位结果的误差特征，即水平方向误差相对较小且稳定，尤其是 E 方向精度较好，而垂直方向误差显著偏大，精度和稳定性均较差。

4.2 粗差与周跳探测分析

为了进行伪距粗差与相位周跳探测，在观测时段内选取了 5 颗 GPS 卫星（G03、G04、G09、G27、G30），和 6 颗 BDS 卫星（GEO：C02、C03；IGSO：C07、C08；MEO：C30、C36）分别计算它们的 GF 和 MW 组合观测值并绘制出时序图，根据公式(67)和(69)，GF 组合观测值受到整周模糊度与总电子含量的影响，MW 仅受到整周模糊度影响。在未发生粗差和整周跳变以及电离层平稳性假设条件下，GF 和 MW 组合观测值应该保持稳定。

下图绘制了 11 颗卫星的 GF 和 MW 组合观测值时序图，由于 MW 组合观测值采用的是平滑值，因此表现上更加平稳，总体而言，在大部分时段，组合观测值表现平稳，符合规律。

通过对上图的观察，C07 号卫星在7:00~8:00出显著振荡后信号消失，G03、G04、G09、G30 和 C36 均在急剧波动后信号消失。考虑到卫星在进出视域时高

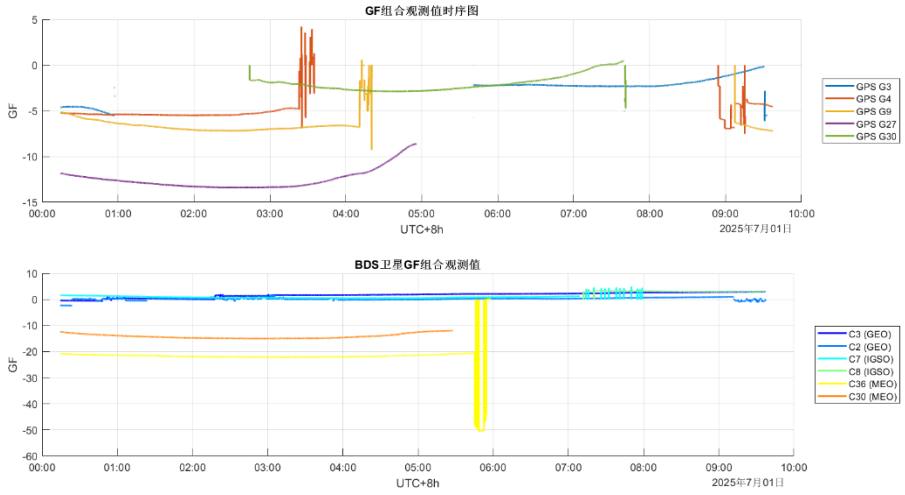

图 4-3 GF 组合观测值时序图

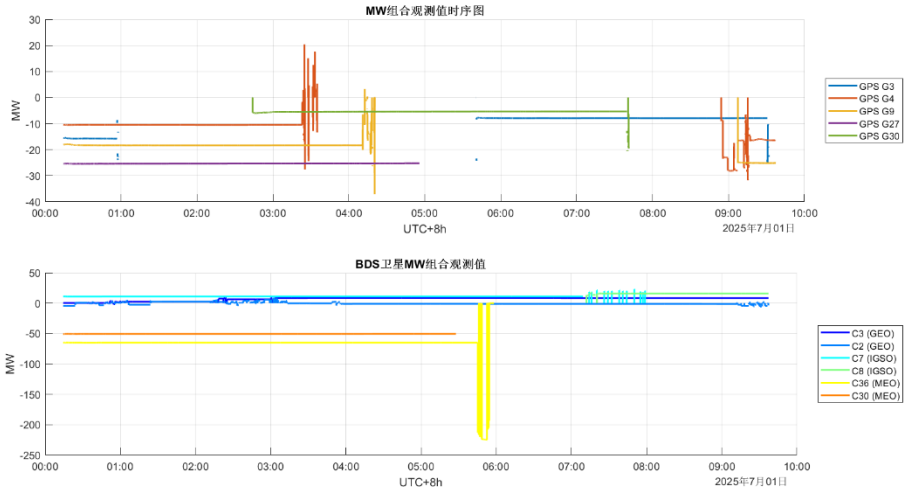

图 4-4 MW 组合观测值时序图

度角较低，受到大气延迟影响上升和信号几何强度下降的影响，信号质量下降导致信号失索和周跳发生。

5 体会与心得

在本学期的卫星导航算法课程中受益匪浅。本学期，我们实现了从 OME7 板卡中实时获取观测值和广播星历等数据，进行导航定位测速（PVT）解算，并输出定位结果、进行精度分析和可视化处理。

很早就从学长和其他老师那里听说这门课是一门十分经典有价值的课，加上这学期的亲身体会，我对这门课的经典之处和价值有了更加具体和深刻的认识。首先，此课程实现了对前置专业课的复习与巩固，是从理论到实践的跨越。本课程要求对导航学、卫星导航原理和最优估计等课程有较强理解，熟练运用卫星导航知识和最优化理论。在课上，老师对相关知识的讲解和重温加深了我们对卫导原理的记忆和理解，也在时刻提醒我不断复习原理，提升专业素养。

除了加深对专业知识的理解与熟练程度，更为重要的是，我从这门课的学习到了面向对象的编程和模块化编程的能力以及编程素养。为了实现程序编写，老师在课上深入阐释了 C++编程的模块化思想，我印象最深的一句话就是“一个函数只做一件事”，这句话听起来简单，但是实现却不是那么容易。我以往编程都是在一个主函数实现所有事情，函数功能界限不明确，一个函数身兼多职，这样往往导致程序维护与 debug 繁琐与困难。在理解了老师的编程思想后，我尽量多写一些小函数去完成某种特定的功能，主函数只负责调用，这样调试与找错便能逐函数进行，目标明确的同时提高效率。

对于变量的定义，曾经的我只会使用 int 和 double 类型，从没有考虑过内存分配和截断误差。在这门课上，我学会了去考虑变量的类型，例如我常常把年份的类型定义为 int，但其实 unsigned short 就够了，因为年份没有负数，且目前的年份还没有达到$2^{16}-1$；再比如学习简化儒略日的存储方式之前我曾很少考虑过把一个很大数按照整数部分和小数部分分开存储从而提高精度。

老师说，不要那么在意结果。一开始我不是那么认同，因为编程不就是为了得到正确的结果吗？现在看来，我那时还是多么幼稚。一个程序的结果是由程序的每一个函数与每一项功能共同作用的，只在意结果是无法高效完成目标的，过程才是最重要的。写一个函数，就测试一下，力求每一步走踏实，每一步不犯错，结果自然水到渠成。

老师总是跟我们说，要提高做事的能力。按照我的理解，这个能力应该是指做事的经验。我每敲出一行代码，每调试一个函数，每找出自己的一个错误，都是经验的增加与能力的提升。现在，我能很轻易的找出其他同学程序的错误并修改正确，这不仅仅是因为我也犯过同样的错误，更是因为我调试方法与经验的丰富与对常出现错误地方的敏感。学习这一门课最大的收获在于编程能力的提升与编程思想的升华，很多课的编程任务我都坚定使用老师在课上传授的方式，认真贯彻老师的思想。这也的确让我面对一个新的编程任务时更加得心应手，从容不迫。

老师总是批评我们沉默，不肯与老师交流，我也知道老师想要和我们沟通来探究我们不明白的点。其实我更倾向找老师一对一沟通，这样问的更加具体，也能让我有足够时间整理思绪和逻辑。我想这也是很多不爱在课堂上发问同学的境遇。

这是我第一次完成代码总行数接近 2000 行的项目，对此我颇有成就感。很多小函数能在其他课上重复使用，我也能很自豪地说这是算法课上学来的。

这学期真的很感谢王老师指点迷津，和老师交流很多收获很大。感谢王老师，期待下学期的算法课程。