# 绪论

## 研究背景和意义

## GNSS实时轨道的国内外服务现状

* + 1. 超快速轨道实时服务研究现状
    2. 实时滤波轨道服务研究现状

## 本文研究目标和研究内容

* + 1. 研究目标
    2. 研究内容

# 导航卫星精密轨道确定的基本原理

## 引言

正如前述所说，提供实时高精度的导航卫星轨道服务对当前实时高精度定位具有重要意义,而导航卫星精密轨道确定的核心任务，就是利用长时间连续弧段的高精度GNSS观测信息，确定导航卫星在该时间弧段内的空间位置信息。因此轨道确定中的其中一个常用的方法即为几何学定轨法(Bisnath et al. 1999；韩保民，2003) ，其核心原理与定位过程类似，利用高精度GNSS观测值直接交汇计算出目标体的动态位置。这类方法常常用于低轨卫星的轨道确定中，而对导航卫星，受限于地面观测几何构型，使用效果不足以满足当前对轨道精度的需求。

除利用高精度GNSS观测信息外，导航卫星自身的运动规律信息同样可以应用于定轨中。考虑到导航卫星在空间环境中主要受到地球的万有引力作用，因而其具有围绕地球的类椭圆的周期性的运动轨迹。若将卫星和地球均视为质点，此时导航卫星的运动可被简化为一个二体问题。根据天体力学原理，其运动轨迹可被六个轨道参数所确定，此时轨道确定问题转化为轨道参数的确定。然而在导航卫星实际的运动过程中，除主要的地球万有引力，还会受到到其他摄动力的影响（如地球非球形引力、地球潮汐引起的引力摄动、地球辐射等等），此时将难以构建类似二体问题中轨道参数的解析表达式描述卫星的运动轨迹。在这种情况下，假定受力模型已知，导航卫星的运动轨迹信息可以通过数值积分的得到。动力学定轨法（刘林，1992）正是结合了导航卫星的运动模型和GNSS几何观测信息进行轨道确定，其通过动力学方程来维系导航卫星连续弧段内的位置信息,其显著改善了仅使用GNSS观测信息定位所带来的误差，因此导航卫星动力学模型的构建是动力学定轨法中的一个关键部分。

接下来本章就针对对导航卫星动力学定轨法中所涉及的基础算法原理进行梳理和介绍。首先是有关导航卫星位置信息表述中常用的时空参考系统，然后就动力定轨法所涉及的GNSS观测模型以及导航卫星的运动模型进行基本阐述。

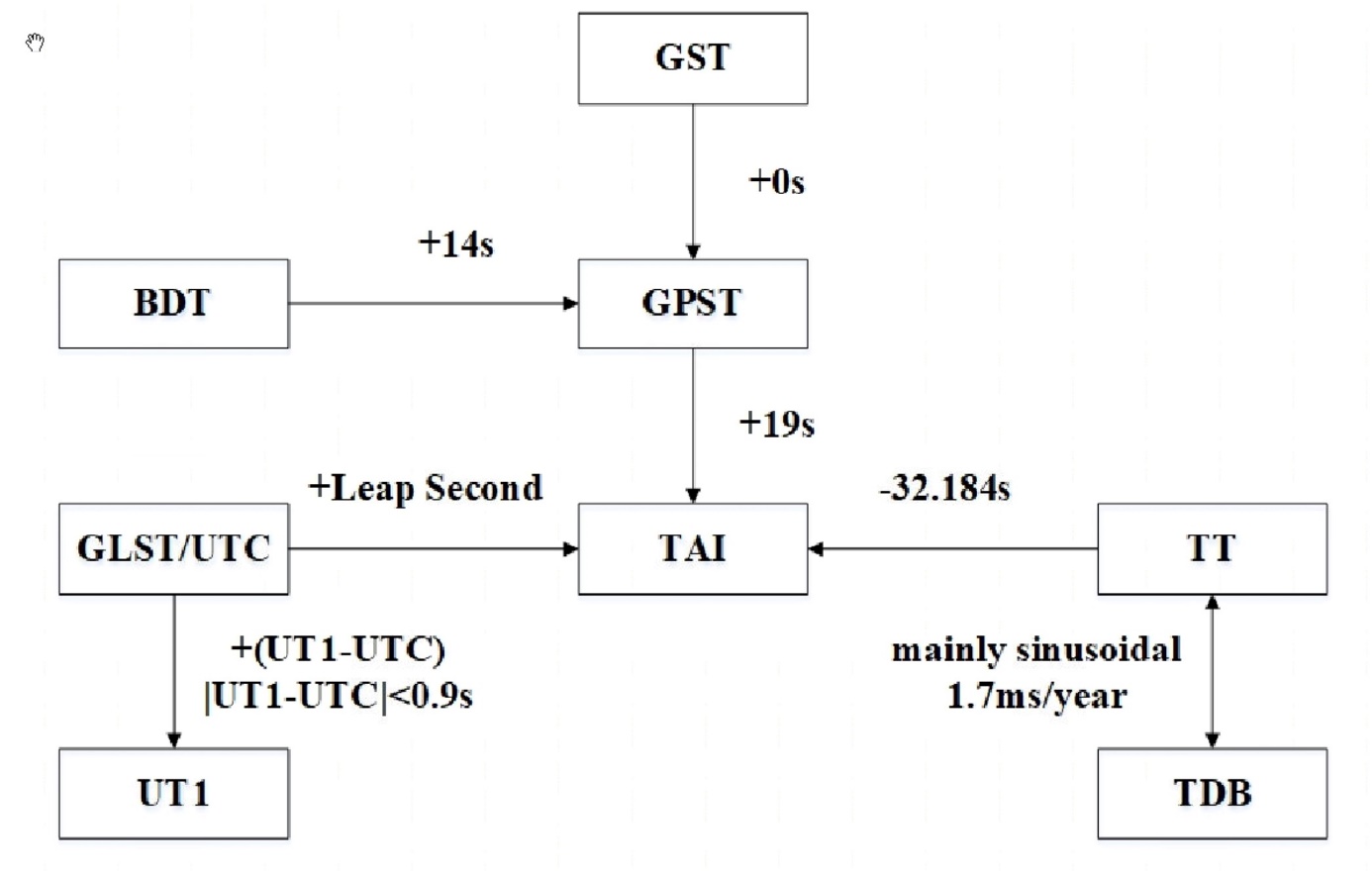
## 时空参考系统

* + 1. 时间系统

时间系统的参考基准是由起始历元和时间尺度两者所确定的。目前常用的时间系统根据其定义和用途，主要可以分为以下三种类别:

1. 以地球自转规律为基础所定义的世界时系统。常见的包括恒星时（Sidereal Time，ST），以及世界时（Universal Time,UT)。
2. 以天体运动方程为基础所定义的力学时系统。常见的包括地球时（Terrestrial Time，TT)和太阳质心动力学时（Barycentric Dynamical Time，TDB）。
3. 以原子运动规律为基础所定义的原子时系统。常见的包括国际原子时（temps atomique International，TAI），协调世界时（Barycentric Dynamical Time，UTC）。

尽管目前四大全球卫星导航系统定义了各自的时间系统作为参考基准，但它们都是属于原子时一类的时间系统。即它们的时间尺度相同，只是在定义起始历元上有所区别。图 2.1 给出了常见的时间系统之间的转换关系。



* + - * 1. 各常见时间系统转换示意图
    1. 坐标系统

坐标系统的参考基准是由坐标原点、坐标轴指向以及坐标轴尺度这三个方面所确定的。在导航卫星精密轨道确定过程中，常常涉及多个不同坐标系所描述的位置信息，通常需要归算至同一坐标参考系统中。因此明确各个坐标系统的定义以及它们之间的转化关系尤为重要。下面对定轨过程中涉及的常见坐标系统做一个介绍。

1. 地心固体坐标系统

地心固体坐标系统通常指的是一类具有跟随地球自转、与地球固连特性的坐标系统。由于该坐标系统的定义更为符合人们日常生活的直观感受，因此常常应用于许多科学应用活动中，如固定测站的坐标描述、地球重力场模型、地球潮汐模型等。由于地球本身极移等特性，很难在实际空间中找到地心固体坐标系统的参考标志，因此地心固体坐标系的定义是一种人为协议的结果。其中由国际地球自转服务（International Earth Rotation Service，IERS）所提出的国际地球参考系（International Terrestrial Reference System，ITRS）是目前国际上所公认采用的一种标准。几乎所有的地心固体坐标系的定义与ITRS保持或接近一致。而对于地心固体系统具体实现,则是通过维持一组参考点的位置和和速度完成。IERS就通过多种空间大地测量技术以及相应的全球参考站维护了一套国际地球参考框架（International Terrestrial Reference Frame，ITRF），也是目前最为完善、精度最高的地心固体坐标系的实现。除此之外，对各个导航卫星系统而言，由于其构建和使用的地心固体坐标系在定义或是实现坐标系的参考站上的不一致，也衍生出了不同的地心固体坐标系。GPS系统的广播星历即是以WGS84坐标系（World Geodetic System）为基础,类似的GLONASS系统使用的为PZ90坐标系以及Galileo使用的为GTRF坐标系。对于我国的北斗导航卫星系统而言，一开始使用的为2000中国大地坐标系（China Geodetic Coordinate System 2000，CGCS2000），而后从2017年12月开始，开始使用北斗坐标系统（BeiDou Coordinate System，BDCS）（魏子卿等， 2019）。

1. 惯性坐标系统

前述的地心固体坐标系统由于会随地球自转不断发生改变，因此难以在该坐标系统下构建导航卫星的动力学方程，而惯性坐标系统则更适用于导航卫星运动描述。在天文学中，天球参考系通常用来作为描述天体运动的惯性坐标系统。类似地，以地球质心为原点的地心赤道天球坐标系统常被用于导航卫星运动的描述中。在该坐标系统中，天极（即地理极点在天球上的投影）以及赤道面上的春分点被用于标定坐标轴的指向。但由于地球岁差和章动的影响，赤道天球坐标系统同样是随时间缓慢变化的，这对后续科学活动中的使用依然十分不方便。因此国际天文联合会（International Astronomical Union，IAU）提出了J2000.0的协议天球坐标系，其即对应的着TDB为2000年1月1日12时的赤道天球坐标系统。在导航卫星轨道确定中，由于地球参考站常常使用地心固体坐标系，因此常常涉及其与天球坐标系之间的转换，从前述两者的定义可知，通过对地球岁差、章动、极移、自转进行相应的分析，即可获得两者坐标系之间的旋转矩阵（李征航等，2010）。

1. 卫星轨道坐标系统

由于对卫星轨道精度以及力学模型进行评价时候，常常需要分析其在卫星瞬时速度方向及与地球质心连线方向上的分量大小，因此引入了对应的卫星轨道坐标系。该坐标系的三个方向分别由径向（Radial）、切向（Along-track）和法向（Cross-track）所构成，其具体定义可参照如下公式：



式中， 表示导航卫星的位置向量，表示导航卫星的速度向量，分别表示坐标系的径向、法向和切向。

1. 星固系

为了方便描述导航卫星上的如天线相位中心偏差、卫星天线相位缠绕等相关问题，因此引入星固坐标系。由于不同导航卫星天线制造与控制上的差异，具体星固系的定义也有所区别。这里以IGS定义的协议星固系为例：该坐标系统的原点定义在卫星质心，其中以太阳帆板旋转轴为Y轴，同时理论上天线所指地心方向为Z轴，最终X轴可由Y轴和Z轴所构成的右手坐标系唯一确定（Montenbruck 等，2015）。

## GNSS观测模型

这里参考下当时写IFCB论文里面的观测模型吧，相应的误差改正就简单的概况一下吧。

这里我们使用L1，L2来统一表示不同卫星系统的两个频率信号（如GPS的L1/L2，GAL的E1/E2，BDS的B1/B2）。则L1和L2频率上IF组合的GNSS伪距和相位的观测方程可以被表达为如下形式：



（这里记得改一下公式，将模糊度记为以整周为单位的表达）

## 导航卫星运动模型

* + 1. 动力学方程和状态转移
    2. 摄动力模型

## 参数估计方法

目前高精度 GNSS 数据处理的方法主要可以分为两大类，一类是最小二乘估计，一类是卡尔曼滤波估计。卡尔曼滤波通过递推的方式，利用参数的协方差信息 阵(实际上是包含了参数的先验信息以及之前所有历元的观测信息)以及当前历元 的新增观测数据，通过时间更新、状态更新两个步骤，给出了基于最小方差准则下 的参数最优估计值以及相应的协方差[21]。而最小二乘，则是根据相应观测量组建 观测方程，考虑观测量的随机模型，给出了基于最小残差平方和下的参数最优估计 值。虽然二者表现形式不同，其本质上是一样的。

* + 1. 最小二乘

参考下本科设计时候的思路写一下这部分的内容

* + 1. 卡尔曼滤波

参考之前整理的推导的卡尔曼滤波内容的部分下一下这部分的内容

# GNSS实时滤波精密轨道确定的研究与实现

## 引言

不论是提供事后或是实时的导航卫星轨道的位置服务，目前主流的GNSS精密轨道处理的方式仍是基于事后批处理的解算模型，其算法模型和处理流程随着多年来研究已经逐步趋于完善。相对的，基于实时观测数据，采用实时滤波解算方式进行轨道确定的处理模式在近几年仍在不断的研究和探索中。在事后批处理过程中，由于包含了所有观测数据信息，因此可以对观测数据进行统一的预处理，同时在解算过程中进行反复迭代计算和整体质量控制，以达到最优的数据处理效果。而实时滤波处理流程中往往需要通过处理实时观测数据流，实时给出当前已有观测数据信息下的最优轨道位置信息。因此实时滤波精密轨道确定在不论是在观测数据的预处理、参数估计方法、质量控制方法以及整体算法流程都与事后解算模式大不相同。如果直接应用事后解算模式的经验与方法，将难以取得理想的解算效果。

本章从实时滤波精密轨道确定的算法流程出发，分别对处理过程中的关键环节：参数估计方法、实时质量控制方法和实时模糊度固定方法，进行了相应的推导和实现，并通过实验对比分析验证了算法的正确性和有效性。最后，本文在目前已有的数据处理软件平台上GREAT（GNSS+ Research, Application and Teaching）的基础上，开发实现了具有GNSS实时滤波精密轨道确定的功能，并给出了该功能的整体结构组成及其算法处理流程。

## 基于平方根信息滤波的参数估计原理

在第二章的参数估计部分，我们介绍了常用的最小二乘算法和卡尔曼滤波估计算法。其中卡尔曼滤波估计算法更适合实时数据的处理，相对于最小二乘批处理需要存储所有的观测值信息，滤波估计算法无需存储历史时刻的观测数据信息，而是以待估参数的协方差矩阵信息进行存储。同时滤波算法在处理具有先验运动模型的最优估计问题也更为直观。但由于计算机中计算过程截断误差的存在，导致存在滤波因数值误差而发散的情况存在，因此引入了平方根滤波相关的算法（Dyer，1969），其核心原理通过采用原有滤波算法一半字节长度进行相关数据信息的存储，大大减少了数值计算误差，从而抑制了滤波发散的情况，具有更高的数值稳定性。这里，我们选用了实时滤波轨道处理中常用的平方根信息滤波（Square Root Information Filter,SRIF）作为参数估计的方法。由于广义最小二乘算法在测绘领域更为常用，因此本文从广义最小二乘算法角度出发，推导和梳理了SRIF算法过程。

* + 1. 量测更新算法

对于已经构建好线性化后的观测函数模型以及确定了待估参数的问题中，最小二乘算法所求解的核心问题为:在包含有多余观测量的情况下，求解出一组最优的待估参数，使得所有观测方程的残差平方和最小，即满足式。



式中， 为待估计的参数向量， 为观测函数模型化对在参数初始值上的线性化后的系数矩阵，为观测值与使用参数初值计算的观测函数初值之差得到的先验残差向量。分别为观测方程的数量与待估参数的维度。为观测方程的权矩阵，这里假定其为正定对称阵。在此基础上，除考虑观测信息外，还需要考虑到待估计参数具有先验信息。这里广义最小二乘算法的常见做法是构建基于先验的虚拟观测方程，和其他观测方程一同求解，即式可以被扩展成式。



式中，为单位阵,为待估参数的先验值。

这里为了求解式,可以等价求解其对应的正规方程(normal equations)，即。当观测系统存在病态观测的时候，也就是系数矩阵的条件数较大的时候，容易导致所求解的参数数值误差较大。为了避免这种情况，可以使用QR分解的方式进行求解。在使用QR分解算法之前，需要对上述广义最小二乘问题进行单位权规整化。对于前述的观测方程权矩阵和先验信息的权矩阵作三角化分解有， ，这里我们用下标表明权矩阵所在的时刻。因此,上述最小二乘问题可被等价转换为下面的表达：



式中, 分别表示规整化后的系数矩阵与先验残差向量。对系数矩阵进行QR分解有：



式中，为分解得到的正交矩阵。对式 左右两边同乘以，则可以得到相应的等价表达：



式中，为正规化后的验后观测残差向量。此时待估参数可以直接用由直接求解得到,该方程在SRIF中也也被称为信息方程。到可以发现此表达式与k时刻的规整化的先验信息虚拟观测方程类似，此时即为求解后参数的信息权矩阵,这里我们已经完成了SIRF量测更新的推导。在SRIF中，为待估参数的信息矩阵，因为其为原有的先验信息权矩阵三角化后的结果，在数值计算上具有更好的稳定性，其量测更新的核心原理即为式，其中分解得到即可直接作为量测更新后参数的信息矩阵。

* + 1. 时间更新算法

导航卫星精密轨道确定问题中包含着大量的随时间变化的动态参数，如轨道参数、钟差参数、对流层参数和模糊度参数等。如何在SRIF算法中随时间动态更新调整这些参数，也是实时滤波轨道处理中的一个关键步骤。尽管这些动态参数各自具有不同的特性，但考虑它们一般化的时间更新过程，都包含了参数增加、参数状态更新、参数消除这三个部分。接下来依次对这三个部分的具体算法流程进行相应的介绍。

考虑j时刻的信息方程为。其中，我们将参数中分为两个类别，其中表示过了时刻后需要消除的参数， 表示过了时刻还需要保留的参数。假定的信息矩阵为上三角阵（若不为上三角阵，可做一次QR分解后得到），此时信息方程可表示为如下形式：



对于下个历元新增加的参数这里我们使用进行表示。对于新增参数，考虑其具有的先验信息方程为。因此根据类似式的广义最小二乘原理，此时参数增加后的信息方程可以直接表示为如下形式：



考虑到和可以构建如下的状态变化方程：



式中， 为线性化后的状态转移矩阵，为状态转移方程的信息权矩阵吗，其三角化分解后的结果为，为状态转移方程中的过程噪声，其大小一般与状态转移的时间间隔相关。这里可以发现，本质上完成参数的状态更新过程，即等价于将式当作观测方程，对式的信息方程完成一次量测更新。因此对参数进行状态更新依然可以基于QR分解完成。这里我们仿照式构造如下的最小二乘模型，并对系数矩阵进行QR分解，可以得到如式的形式：

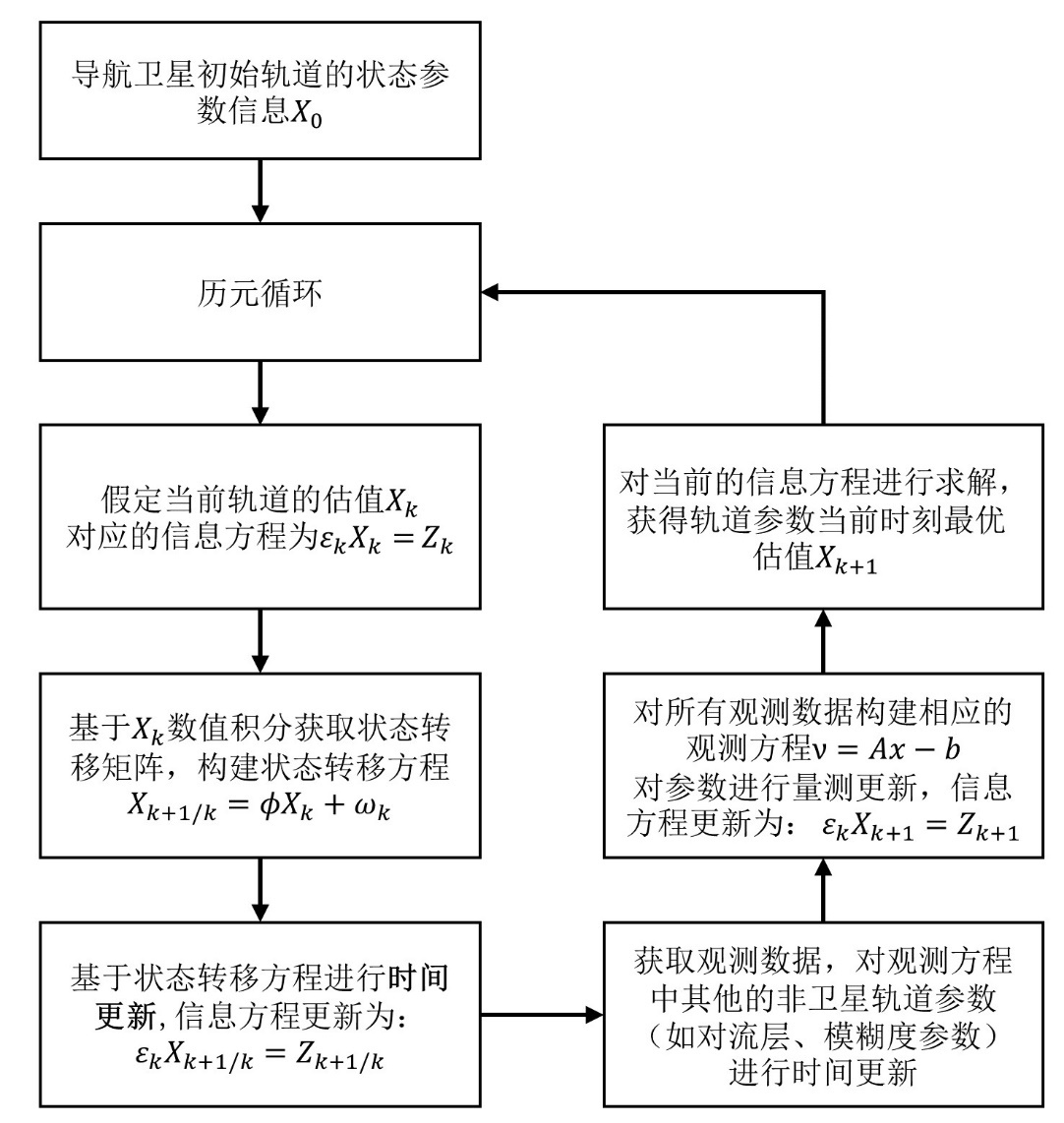


式中，表示了j+1时刻下的信息方程的信息矩阵，为对系数矩阵进行QR分解得到的正交矩阵。这样便推导得到了参数状态过更新后的信息方程。最后，可以发现由于信息矩阵为上三角矩阵，对于后续不再需要的参数，可以很自然地将其对应的信息方程中的所在行删除，方程剩余的部分依然满足信息方程的结构。同时，由于SRIF算法中的时间更新和量测更新是随时间不断地迭代进行，因此删除过时参数可以大大避免因为这些参数导致的无意义的计算资源的消耗。

至此，已经推导得到了基于QR分解实现时间更新基本原理。对于参数增加和消除，由于信息方程的上三角化的特殊结构，可以通过简单的矩阵变换得到。对于参数状态更新的部分，其核心关键就在于构建式的状态转移方程以及式的信息方程的时间更新。

* + 1. 基于SRIF的实时滤波轨道处理流程

前面推导了平方根信息滤波的量测更新和时间更新的基本原理,图 3.1 给出了基于平方根信息滤波的实时滤波轨道的基本处理流程。



* + - * 1. 基于SRIF算法的实时精密轨道确定处理流程示意图

首先,这里将实时滤波轨道中的时间更新分成了轨道参数与其他参数两个部分。其中，对于轨道参数的更新，我们通常需要通过数值积分获取前后历元的状态转移矩阵，从而构建相应的状态转移方程。但考虑到数值积分过程较为耗时，这里通常不需要在每个历元进行积分计算，而是使用之前数值积分的计算结果作一个近似处理。对于状态转移方程的构建,本文采用随机游走的函数模型(Random Walk,RW)来进行描述。假定导航卫星的速度参数和动力学模型参数的误差为零均值白噪声(White Noise,WN)模型（这里我们忽略了动力学模型参数误差对速度误差造成的影响），因此对于前后时间间隔为的轨道参数状态转移方程，其协方差矩阵（其值和信息权矩阵为互逆关系）为下式所示。



式中，和分别表示了表示导航卫星速度误差和导航卫星动力学模型噪声的白噪声方差，具体实现则采用了经验值和（参考文献xxxx）。由于在平方根滤波中，需要对方程进行单位权标准化，因此我们通常使用其三角化后的结果,即如下式所示：



而对于非轨道参数的更新，在无电离层模型中，主要需要考虑钟差参数、对流层参数、系统间偏差参数和模糊度参数的状态更新。对于钟差参数和模糊度参数，通常直接采用WN模型进行估计，因此在时间更新过程中只需要考虑参数增加和参数消除的步骤；而对于系统间偏差参数和对流层参数则采用分段常数的模型进行估计。其中系统间偏差考虑其随时间变换较为稳定，可以按每天分段，而对流层参数则相对更新更为频繁，可以按每两个小时进行分段。

完成所有参数的时间更新后，根据当前历元的观测数据构建相应的观测方程，即可进行SRIF的量测更新。通常，我们不直接估计待估参数的绝对量，而是估计相对于初值的改正量。尽管量测更新后已经可以得到待估参数的最优估计值，但并不能直接将其更新为新的初值直接进行滤波后续的更新，因为更新前后待估参数的所估计的改正量相对的初值已经不一致了，其参数含义已经发生了改变。对于所选取初值较为准确或者线性化误差影响小的参数（如钟差、模糊度参数），可以选择继续沿用之前的初值进行滤波的后续处理。而对于线性化误差影响较大的参数（如轨道参数），则应当对这些参数做一次时间更新，其中，状态方程为如下式所示：



式中，表示t时刻的待估参数，表示初值更新后t+1时刻的待估参数，表示这个状态方程的信息权。由于本质上两个参数仅仅是初值不一致，其状态方程的噪声应为0，即信息权为，因此无法按一般方法直接进行时间更新。这里可以通过对原有参数做一次线性变换以此等价实现信息权为的时间更新。这里考虑具有如下一般形式的状态方程：



考虑对于参数具有如下的信息方程：



将式带入式即可以对信息方程完成如下的更新：



式中也是给出对于间更新过程中的状态方程设定极小噪声的方法。由于计算机数值误差的原因，设置过大的信息权阵依然会引起SRIF滤波器的发散现象，采用上述的转换，可以实现接近的更新。

## GNSS实时数据精化

上一节中我们阐述了SRIF滤波器在精密轨道确定中的使用方法，其中SRIF量测更新过程成立的前提是假定了参与更新的观测方程的观测噪声满足高斯正态分布的随机模型。因此若要确保SRIF滤波器能够得到正确的估计结果，需要尽可能的保证输入的是“干净”的观测值，否则其中的“异常值”将会导致滤波器收敛速度减缓甚至发散。由于GNSS信号传播过程会受到外界环境因素（如多路径效应、大气环境不稳定、接发硬件因素）等影响，导致大量GNSS观测数据中包含异常观测值的概率并不低。特别地，对于GNSS观测值中的相位观测值,由于本身的特性，还存在着整周跳变（周跳）的问题，这严重影响着观测值得精度水平。因而需要对GNSS异常值进行有效地探测和剔除。在精密轨道确定事后批处理的模式中，其对GNSS异常值的处理，通常会在参数估计前对观测数据进行整体预处理，以及在参数估计后利用观测方程的验后残差进行相应的探测。类似地，实时滤波轨道中同样可以在SRIF量测更新前后，分别对GNSS数据质量进行相应的检测。接下来依次对这两部分所采用的算法模型和处理策略进行介绍。

* + 1. 实时数据质量检测算法

考虑到GNSS载波相位观测值具有较高地量测精度，其很大程度决定着数据处理的最终精度，因此有效地探测其本身的粗差和周跳对GNSS数据处理至关重要。Turboedit方法（Blewit，1990）是目前被广泛应用的载波相位观测值质量检测算法，实时滤波轨道处理中同样使用了该方法对载波相位观测值进行检测。其基本思想是通过构造Melbourne-Wubbena(MW)双频组合观测值和无几何距离（Geometry-Free，GF)组合观测值，设定相应的阈值，对连续时间内的组合观测值序列进行分析，从而筛选出包含周跳或粗差的观测值。同时对于GNSS实时数据，由于其处理具有不可逆性，因此不能直接对整个弧段内的观测值序列直接进行分析，需要在原有基础上进行改进，以适应实时处理过程（张小红等，2010）。

具体的算法流程如下。这里我们使用表示观测值所采用的频率，分别表示相位和伪距观测值。因此GF组合观测值可以被表达为如下形式:



式中，表示电离层延迟在两个频率观测上的差值,表示模糊度，分别表示对应频率的波长。可以看到不随几何距离发生改变，且在模糊度未发生周跳的情况下，其仅随电离层延迟之差的变化而改变。则GF观测值的前后历元差值可以表达为：



其中电离层延迟随时间变化缓慢，接近于0，因此可以作为判断模糊度周跳判断的依据。在模糊度没有发生周跳的情况下，通常在厘米级别波动。考虑到精密轨道确定中的时间采样间隔一般相对较长（300s为例），可以选取经验阈值为15cm。在实际处理中，低高度角的观测数据由于多路径效应等原因其观测噪声相对更大，应适当放松判断阈值。本文中所采用的阈值模型具体如下所示：



式中，表示观测数据的卫星高度角。GF组合观测值判断周跳成功的前提是在电离层活动平静，延迟变化缓慢，同时对于特殊的双频周跳组合也存在探测盲区。因此还需要借助MW组合观测值进行进一步的判断。MW组合观测值可以表达为如下形式：



式中，为MW组合观测值，也称为宽巷模糊度。可以看到其值为两个频率上的模糊度的直接差值，在模糊度没有发生跳变的情况下，其前后历元MW组合观测值的差值应表现出零均值白噪声的特性。考虑到MW组合观测值的计算包含了伪距观测值，因此其中还包含了较大的伪距观测噪声，直接使用前后历元的比较结果受噪声影响较大，容易造成周跳的误判，因此常用一段时间的观测值对其进行相应的平滑。考虑到实时数据无法像事后处理存储所有的观测数据，这里需要采用滑动窗口的方式进行相应的计算，具体如下所示：



式中，表示了滑动窗口的大小，分别表示了MW组合观测值序列在滑动窗口内的均值和标准差。接着，可以构造如下的检验量：



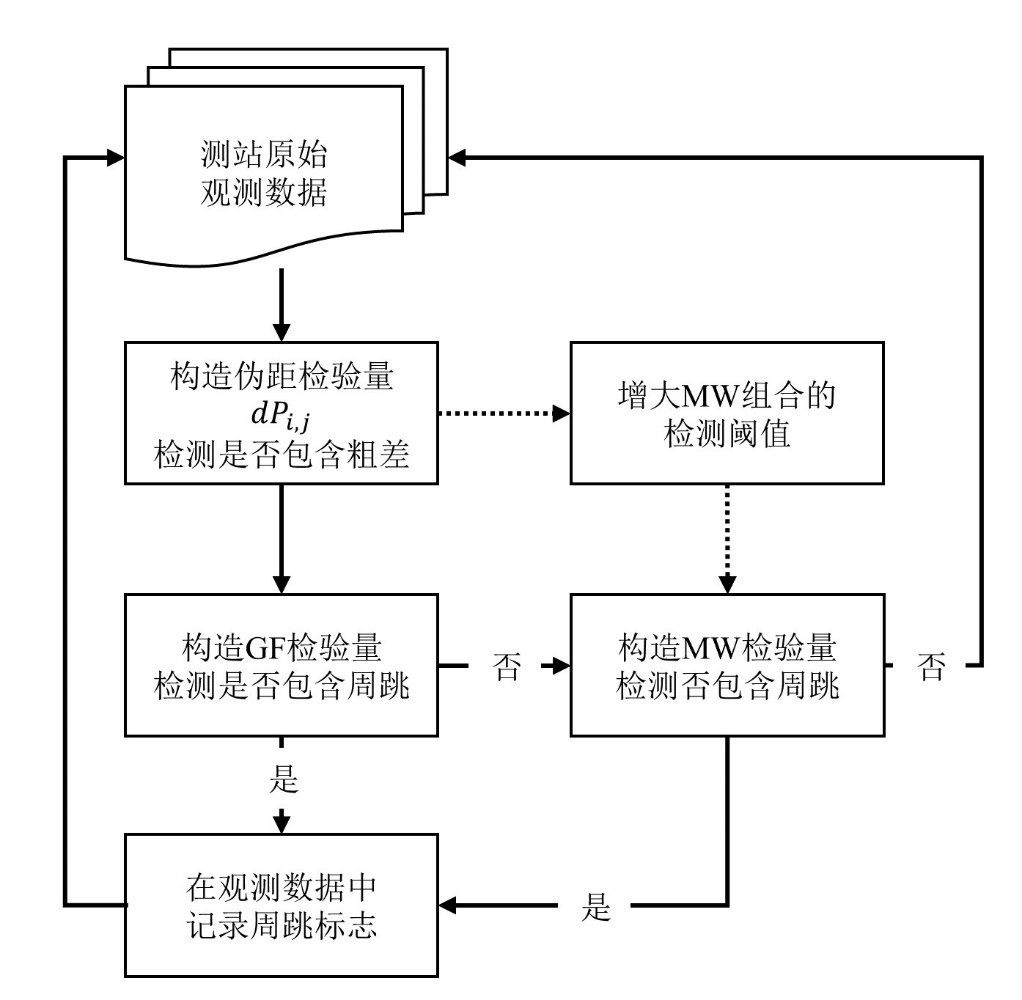
式中，为MW组合周跳判断的阈值，这里选用的为经验值，类似GF组合，同样采用了高度角阈值模型。尽管MW组合观测值同样存在的一定的探测盲区，但其与GF组合观测值结合可以互相消除各自的探测盲区，因此上述两种方法的同时使用基本可以探测出所有周跳情况。

对于GNSS伪距观测值，尽管精度相较相位观测值更低，但可以为GNSS方程提供钟差基准，同时对滤波器的收敛起主要作用。另外考虑到MW组合观测值中包含了伪距观测值，因此包含粗差的伪距观测值也会导致组合量阈值超限，进而造成一些相位观测值数据的浪费。因此这里对伪距观测量进行一个粗略的质量检测，若探测为粗差则相应地放大MW组合观测值的判断阈值。具体的算法原理如下式\eqref{eq:preprocess\_code}所示：



式中，分别表示伪距观测值中所采用的频率，分别表示卫星端和接收机端的硬件延迟在不同频率或信号通道上的差值。表示不同频率的电离层延迟之差，表示观测噪声之差。为相应的判断阈值。考虑对同个频率不同信号通道上的观测值，其电离层延迟可以被消除，因此相同频率不同信号通道的判断阈值小于不同频率的判断阈值。

综上所述，实时滤波轨道处理中实时数据处理检测的算法主要包括了伪距粗差探测和载波相位周跳探测，其中周跳探测部分则主要采用了结合GF组合观测值和MW组合观测进行联合判断的方式。整体的算法流程如下图 3.2 所示。



* + - * 1. GNSS实时数据预处理算法流程图
    1. 实时质量控制算法

前一节当中所述的实时数据质量检测算法适用于参数估计前的一个初步质量控制，通常所设定的判断阈值相对较大，只能用于探测一些“大粗差”，而想要进行更为细致的质量控制，需要采用基于假设检验的数据质量探测方法,其中DIA质量控制方法是目前最为常用的方法。其基本思想是针对参数估计后的观测方程延后残差构造检验量（通常为单位权中误差），通过卡方假设检验判断是否存在粗差。如果存在，则需要定位观测值中的粗差并剔除，并重新进行参数估计和假设检验，直至没有粗差为止。

基于平方根信息滤波的DIA质量控制方法与常见的最小二乘和卡尔曼滤波等最优估计方法中所进行质量控制的方法具有一定的区别。相较于需要根据参数的估计值重新计算观测方程的验后残差，SRIF的量测更新后可以直接得到标准化的验后残差，同时在重新计算剔除粗差观测后的验后残差也更为简便。接下来将基于SRIF的质量控制算法分为探测粗差、确定粗差、剔除粗差三个部分，依次介绍。

首先是探测粗差。以式中的SRIF量测更新方程为例，其等价表达如下：



式中，用于表示参数的先验信息和观测方程标准化的先验残差。分别表示对的敏感矩阵，分别表示量测更新后参数信息和标准化的验后残差。利用可以构造如下的检验量：



式中，为卡方检验，为置信度, 为观测方程数。除了通过检验量判断，还可以判断验后残差是否大于相应阈值。

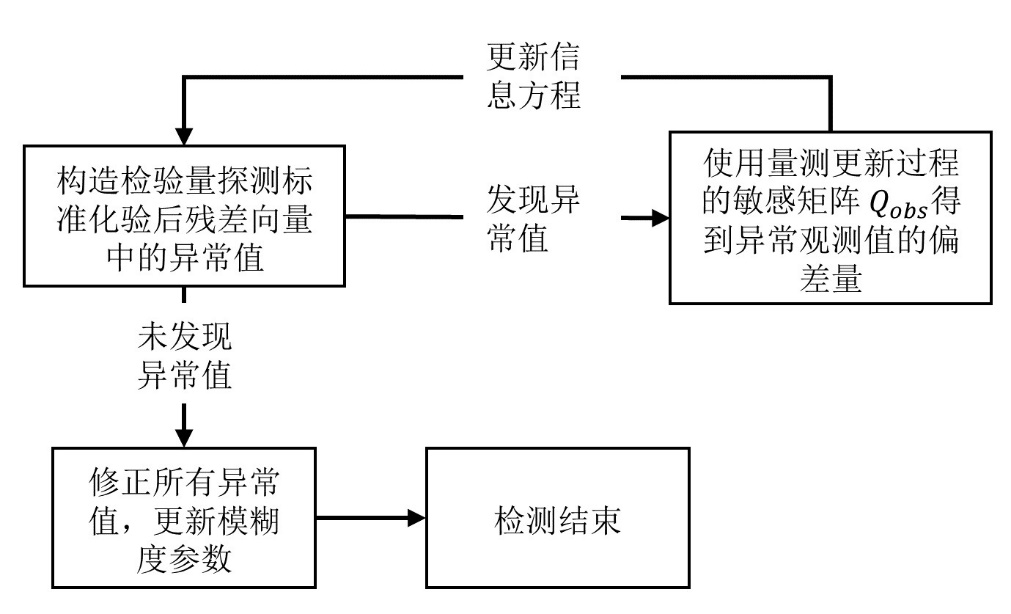
接着是定位粗差。如果通过上面的步骤确认了观测值中存在相应的粗差。则可以对进行相应的排序，从大到小依次对各个观测值进行判断。假定当前观测值存在粗差，一般而言可以通过舍弃观测值，或者新增该观测值的偏差参数来重新计算进行上个步骤的检验量。考虑到敏感矩阵不随观测值发生改变，因此偏差参数可以直接表达为如下形式：



得到偏差参数的估值后，即可以得到新的标准化验后残差。此时可以根据上一节中探测粗差的步骤重新进行判断，以此不断迭代探测出观测值中所有的异常值。

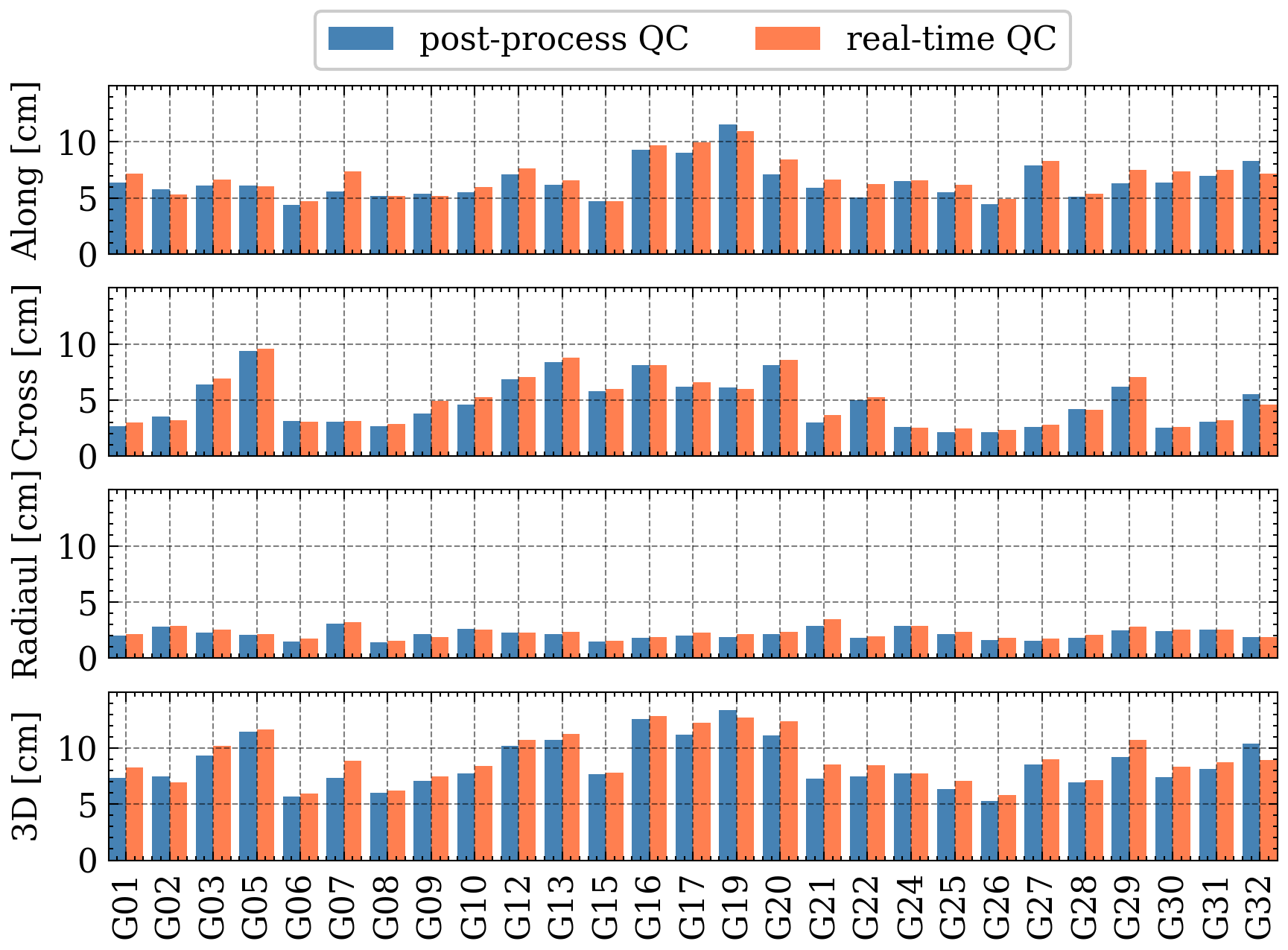
最后是剔除粗差。根据式可以统一求出所有异常观测值的偏差参数，从而获得粗差剔除后的信息方程。同时在探测出的异常观测值中，对于伪距可以统计测站出现的频率，依次剔除部分数据观测较差的测站；对于相位观测值，一般将其认定发生了周跳，因此需要对相应的模糊度参数进行SRIF时间更新。

综上所述，基于平方根滤波算法的质量控制算法可以快速构造检验量并利用敏感矩阵对异常观测值的偏差量进行确定，其整体的算法流程如下图 3.3 所示。



* + - * 1. 基于SRIF的质量控制算法流程图
    1. 实验结果和分析

为了进一步验证前述实时数据质量检测算法的有效性，这里采用与事后质量控制算法进行对比的方式进行测试。考虑到事后质量控制算法流程通常需要几次迭代处理，因此基本可以认为事后处理最终得到的观测数据中不包含观测粗差和周跳（仍然可能包含极少的粗差和小周跳）。将其作为输入数据进行仿实时滤波轨道确定，解算的结果即作为参考基准，以此对比验证实时数据质量检测算法的效果。



* + - * 1. GPS卫星仿实时滤波轨道的Real-Time QC和Post-Process QC的解算值与COD事后轨道产品比较RMS平均值的统计图

具体实验方案如下：选用了2021年年积日为126-135的10天的事后观测数据，对GPS系统进行仿实时滤波轨道确定。定轨弧段为48h，采用浮点解策略，其余处理策略以及测站分布与5.3章中实验方案一致。这里用“Post-Process QC”表示定轨过程中使用事后质量控制算法迭代得到的数据文件作为输入文件，以及使用“Real-Time QC”表示定轨过程中采用原始数据并通过实时数据质量检测算法进行质量控制。这里认为实时滤波轨道在24小时后完全收敛，因此这里分别将每颗卫星收敛后的轨道与COD分析中心事后轨道进行法向、切向、径向以及三维方向上的比较，图 3.4 给出了每颗卫星两种实验方案道轨道比较RMS在这一处理时段的平均值的统计结果。

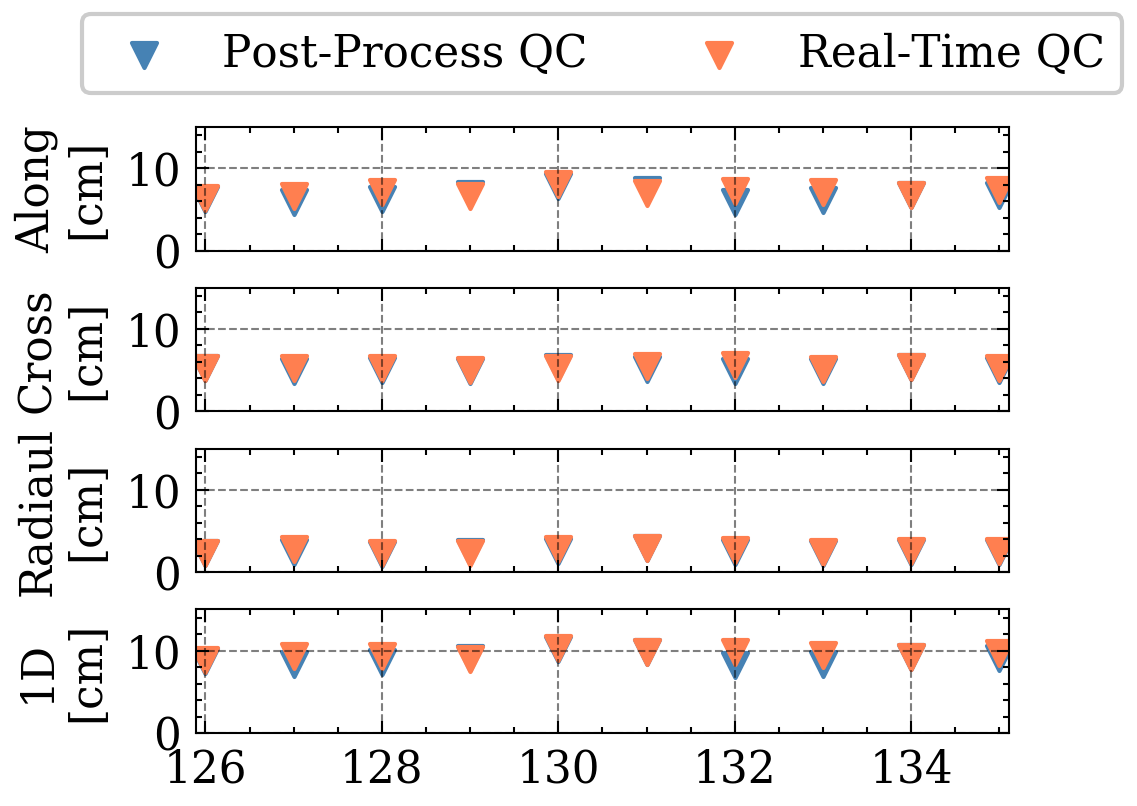
从上图可以看到，在径向方向上，实时数据质量检测算法与事后质量控制算法结果基本接近。而在法向和切向上，几乎对于所有卫星而言，实时处理策略相比事后策略精度略差，差值基本在2~4mm之间。表3.1进一步统计了所有卫星的轨道比较的平均RMS，结果表明，Post-Process QC的轨道精度略优于Real-Time QC的结果。Post-Process的轨道三维平均RMS为8.6cm，Real-Time QC的轨道三维平均RMS为9.1cm，两者相差5mm，其中法向和径向上相差2mm，切向方向相差5mm。总体来说，两者处理策略的轨道结果精度基本相当，精度相差在1cm以内，能够验证实时数据质量检测算法的有效性和正确性。同时表中还给出了已有的基于SRIF的GPS浮点解的研究结果（参考文献，Dai,2016）。可以发现，Real-Time QC相较于Dai的实验结果，在法向和切向上精度稍低，分别差了8mm和2mm，在径向方向上则更优，相差7mm。总体而言，两者轨道精度相差仍在1cm以内，处于一个相当的精度水平，侧面验证了SRIF算法及实时质量控制算法的正确性。两者仍存在差异则可能在于所使用的处理策略和观测数据不一致（如测站列表和处理的时间段）所导致。

GPS仿实时滤波不同解算策略下浮点解与COD事后轨道产品比较RMS平均结果统计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Along(cm) | Cross(cm) | Radial(cm) | 3D(cm) |
| Post-Process QC | 6.4 | 4.7 | 2.1 | 8.6 |
| Real-Time QC | 6.9 | 4.9 | 2.3 | 9.1 |
| Dai,2016 | 6.1 | 4.7 | 3.0 | 8.3 |

这里进一步考虑Post-Process QC和Real-Time QC两者实验结果的差异，尽管两者核心算法原理一致，但具体的处理策略上不一致。由于实时数据处理的不可逆性，导致在进行周跳探测过程中，实时数据质量检测算法只能通过处理当前已有的时间窗口做前向探测，而相对的，事后处理策略则可以通过整个弧段的数据进行更为精细的判断。因此使用实时处理算法更容易造成一些因观测噪声导致的周跳误判，影响模糊度的求解，最终导致轨道精度的降低。从上述分析的原因可知，假定在数据质量相对良好的情况下，两者算法理论上应该具有相同的解算结果。

为了验证上述观点，这里给出了所有卫星的轨道比较平均RMS时序图，如图 3.5 所示。可以看到，仅在部分天（如132和133）的情况下，两者处理策略存在明显差异，其余绝大部分天下的结果基本相当。说明在观测数据正常情况下，两者处理策略处理结果为一致，在部分天的观测数据包含一些特殊情况噪声下，实时处理策略会由于上述原因造成周跳误判导致精度略微下降。



* + - * 1. 平均RMS时序图

（TODO：给出两者质量控制对轨道收敛时间的影响）

## 实时双差模糊度固定算法

在上节曾提到，需要对GNSS载波相位观测值进行精细的质量控制的一个重要原因就是在于其质量好坏很大程度决定了数据处理的最终精度水平。除了对相位数据作质量检测外，另一个充分利用载波相位观测值的方式就是考虑到其固有的整周特性。但在一般的数据处理过程中，由于伪距观测量引入的钟差基准噪声，接收机端和发射端的伪距和相位的硬件延迟偏差等原因，导致模糊度参数通常是以浮点的方式进行求解（参考文献，XXLi）。为了进一步提升数据处理精度，目前对模糊度固定算法已经有了很多相关内容的研究，这些方法大致可以分为以下三种思路：通过构建双差模糊度进行固定（常被应用于相对定位和GNSS网解中）；通过构建星间单差并采用相位偏差小数产品（全称，UPD）进行改正（常用于PPP的解算过程中）；通过采用整数钟或者Carrier-Range的方式对非差模糊度直接固定。本质上这三种的模糊度固定算法的原理是等价的，这里采用了固定双差模糊度的方法来实现实时滤波轨道确定中的实时模糊度固定。

接下来我们首先阐述双差模糊度固定算法的基本原理，然后介绍其在基于SRIF的实时滤波定轨中的应用。最后进行了实验验证和结果分析。

* + 1. 实时双差模糊度固定算法

在第二章中，式给出了L1和L2频的IF组合的GNSS观测方程。这里对直接进行解算会导致模糊度参数包含了硬件延迟的小数部分，从而使得求解的结果为实数而非整数，如下式所示：



式中，表示IF组合模糊度的整周部分，和分别表示IF组合中接收机端和卫星端的UPD，其分别由各自的伪距端和相位端的硬件延迟的小数部分构成。考虑到UPD在同个测站或卫星上是一致的，且其随时间变化较为稳定（Blewitt,1989），因此这里通过构建双差模糊度，可以将接收机端和卫星端的UPD消除，进而获得其整周部分，具体的表达式如下所示：



式中，表示了测站对和卫星对构建成的双差模糊度。由于小数部分已经被消除，这里双差模糊度应具有整周特性。但考虑到无电离层组合为IF模糊度引入了更大的观测噪声，与此同时IF模糊度本身具有的波长相对IF组合观测值的观测噪声较为接近，使得难以对上述构建的双差模糊度进行直接固定。因此这里常用做法是对IF组合模糊度重新表达为宽巷（Wide-lane,WL）模糊度和窄巷（Narrow-Lane,NL）模糊度的组合：



式中，表示WL模糊度，表示NL模糊度。WL模糊度由于其为和上的模糊度差值，其波长约为86cm，相对观测噪声（约为1cm）较长，因此容易对WL模糊度进行直接固定。在此之后计算出NL浮点模糊度并进行相应固定，由于此时NL模糊度波长为11cm左右，相较于原有的IF组合模糊度，已经更为容易进行固定。

根据MW组合，WL模糊度可以被表示为如下形式：



这里得到的为非差WL模糊度，类似非差IF组合模糊度，其中包含了接收机端和卫星端UPD的影响（即和），因此其求解结果为浮点数而非整数。同样地，通过构建双差WL模糊度可以将接收端和卫星端共同的UPD消去，使其变为整数。具体的表达形式如下：



式中，为双差WL模糊度，这里可以采用概率误差函数检验的方式对其进行固定。由于MW组合中引入伪距的观测噪声，因此直接使用单历元MW组合构建的双差WL模糊度容易因观测噪声导致固定效果不佳。因此这里可以通过对模糊度参数所在时间弧段内的所有历元观测值求取WL模糊度均值进行噪声平滑。考虑到实时滤波轨道处理中，无法像事后处理事先预知模糊度参数的处理弧段，因此在实时过程中需要不断的更新WL模糊度的均值和标准差，这里给出其相应的递推求解过程：



式中，表示当前历元在所处WL模糊度时间弧段内的历元数，和分别表示噪声平滑后的WL模糊度均值和方差。因此可以进一步得到双差WL模糊度的均值和方差：



式中，为经过平滑得到双差模糊度，其应当十分接近某个整数，这里选用采用概率密度函数检验方法（Dong and Bock,1989）进行进一步的的判断。构造如下的概率函数：



式中，为双差WL模糊度的标准差，为取整函数，为互补误差函数，为双差WL模糊度最接近的整数。这里计算得到的为能否被固定的概率，通过采用一定的判断阈值，将高于判断阈值的双差WL模糊度固定为。联合固定后的双差WL模糊度与原始的浮点双差IF组合模糊度，则双差NL模糊度及其方差可以被表达为如下形式：



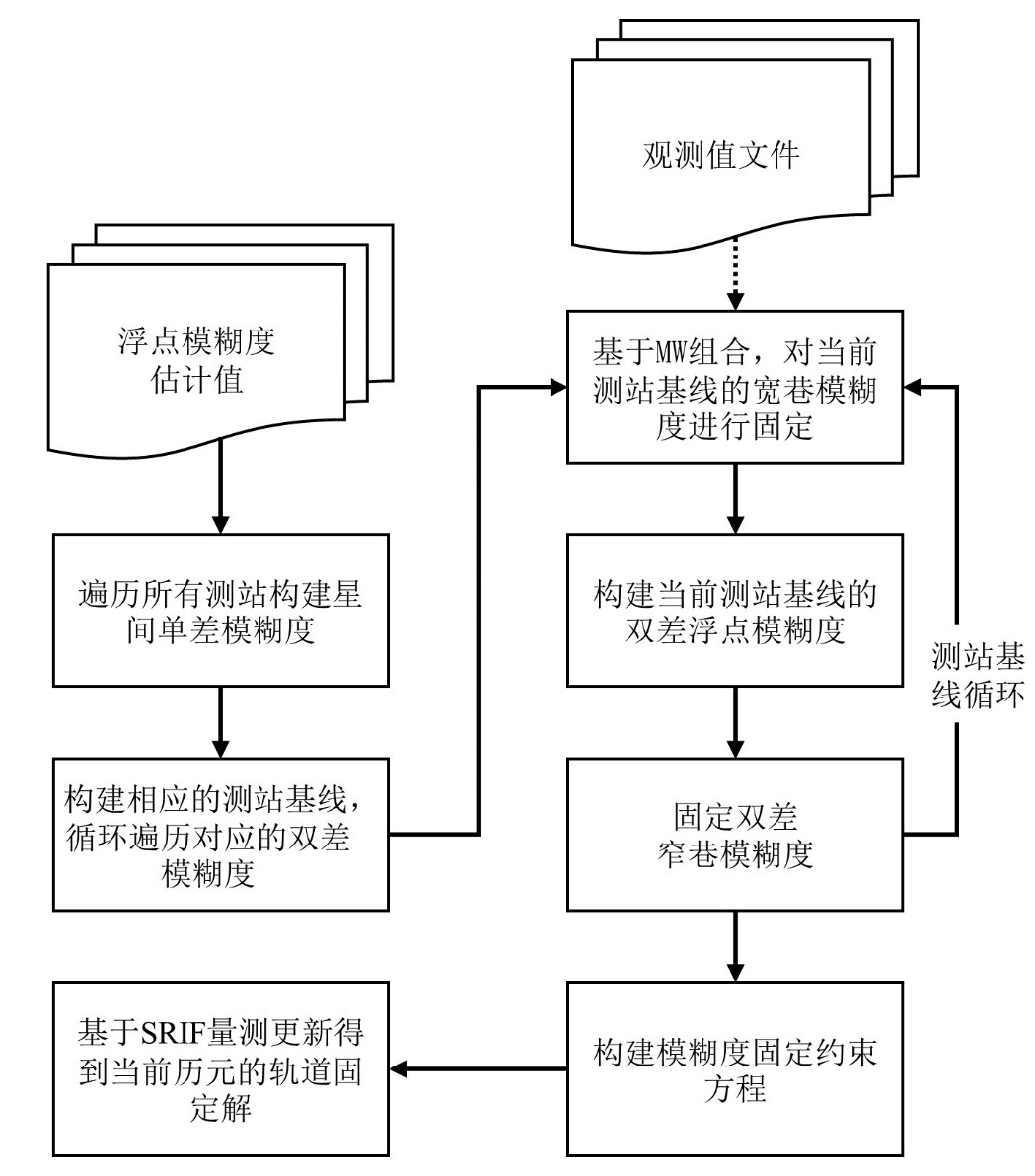
式中，为双差IF模糊度浮点解的方差。此时，同样采用概率密度函数检验方法对做模糊度固定判断，由此获得固定后的双差NL模糊度，最终得到双差IF组合模糊度的整数形式。考虑到实时滤波轨道处理中SRIF滤波器维护的是非差IF组合模糊度参数，需要将回带至式构建得到模糊度固定约束方程，如下式所示：



式中，表示SRIF滤波器中维护的非差模糊度参数。将式作为虚拟观测方程进行SRIF量测更新即可实现对双差模糊度的固定。为约束方程所选择的先验权，这里跟具体的处理策略选择有关。对于选取了独立双差模糊度基线的方案（Ge et al, 2005; Blewitt et al., 2008），通常按照约束方程先验精度为0.001mm进行定权；Bertiger等人（2010）则采用了对所有双差模糊度进行约束以避免部分模糊度固定错误的影响，此时则按照先验精度为1mm进行定权。下一节中针对这两种固定策略进行了实验比较和分析，这里不再过多阐述。

另外，在实时滤波轨道处理中，参与固定的部分模糊度参数可能因观测数据弧段较短导致固定效果较差甚至出现固定错误的情况。在事后轨道处理中，可以通过反复迭处理等方式发现模糊度固定错误的弧段进行改正，然而在实时数据处理过程中由于其实时性和处理的不可逆性导致了难以实现类似的处理策略。因此这里在实时滤波轨道处理中维护的仍是基于浮点解的SRIF滤波器，模糊度固定约束的量测更新只针对当前历元浮点解的SRIF滤波器进行，从而避免模糊度固定错误对后续解算造成的影响。

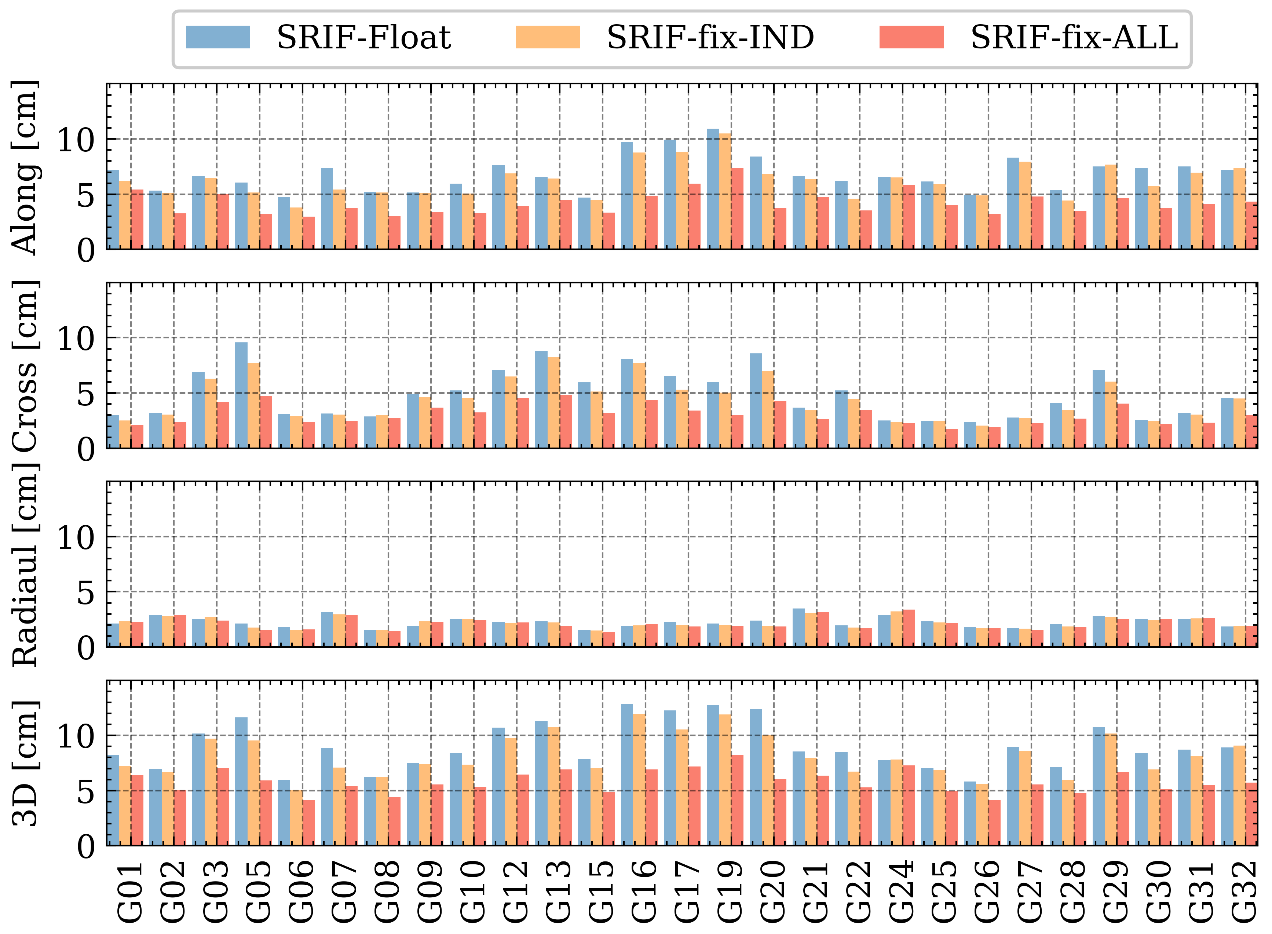
综上所述，图 3.6 给出了双差模糊度固定算法在实时滤波轨道中应用的流程图。根据SRIF滤波器中的浮点模糊度估计值，循环遍历所有构建的基线，依次进行WL模糊度和NL模糊度的固定，最终将双差模糊度固定约束方程临时加至SRIF滤波器中即可得到当前历元的轨道固定解。



* + - * 1. 基于SRIF的实时滤波轨道的模糊度固定处理流程
    1. 实验结果和分析

为了进一步验证基于上述原理实现的双差模糊度固定算法在实时滤波轨道处理中的有效性，以及分析不同模糊度约束策略的固定效果，本文采用不同处理策略进行了仿实时滤波轨道确定的实验，分析对比了滤波收敛后的实时轨道结果。这里首先给出实验设计方案，然后接着依次展示了不同导航卫星系统的轨道分析结果。

具体实验方案如下：本文选择了2021年年积日为126-135的10天事后观测数据进行了三系统（GPS、GAL和BDS）的仿实时滤波轨道模糊度固定解的实验。考虑到GLONASS的频分多址特性，暂不考虑对其进行模糊度固定。实验过程共采用了三种不同的处理策略，分别是：实时滤波轨道浮点解，即原始的基于SRIF的实时滤波轨道解算，不作任何约束，后续统一用“SRIF-Float”表示该处理策略；在实时滤波轨道浮点解基础上对每个历元采用模糊度固定紧约束方案，即在每个历元过程中首先按照基线距离选取独立测站基线，在此基础上进一步选取独立双差模糊度进行固定，同时对模糊度固定约束方程给定0.01mm得先验精度（即的先验权），后续统一用“SRIF-Fix-IND”表示该处理策略；在实时滤波轨道浮点解基础上对每个历元采用模糊度固定松约束的方案，即不再选取独立的双差模糊度，而是将所有能够固定的双差模糊度作为约束更新至SRIF滤波器中，同时对模糊度固定约束方程给定1mm得先验精度（即得先验权），后续统一使用“SRIF-Fix-ALL”表示该处理策略。三种方案除模糊度固定处理部分有区别外，其他处理策略均与5.2节种的实验方案相同，其中，一般认为实时滤波轨道在24h后已经完全收敛，因此开始添加模糊度固定约束的时间段均为滤波解算24h后。后续轨道分析的统计结果也均是针对开始进行模糊度固定后的时间弧段。



* + - * 1. GPS卫星仿实时滤波轨道确定浮点解和固定解的轨道结果与COD事后产品轨道比较RMS平均结果统计图

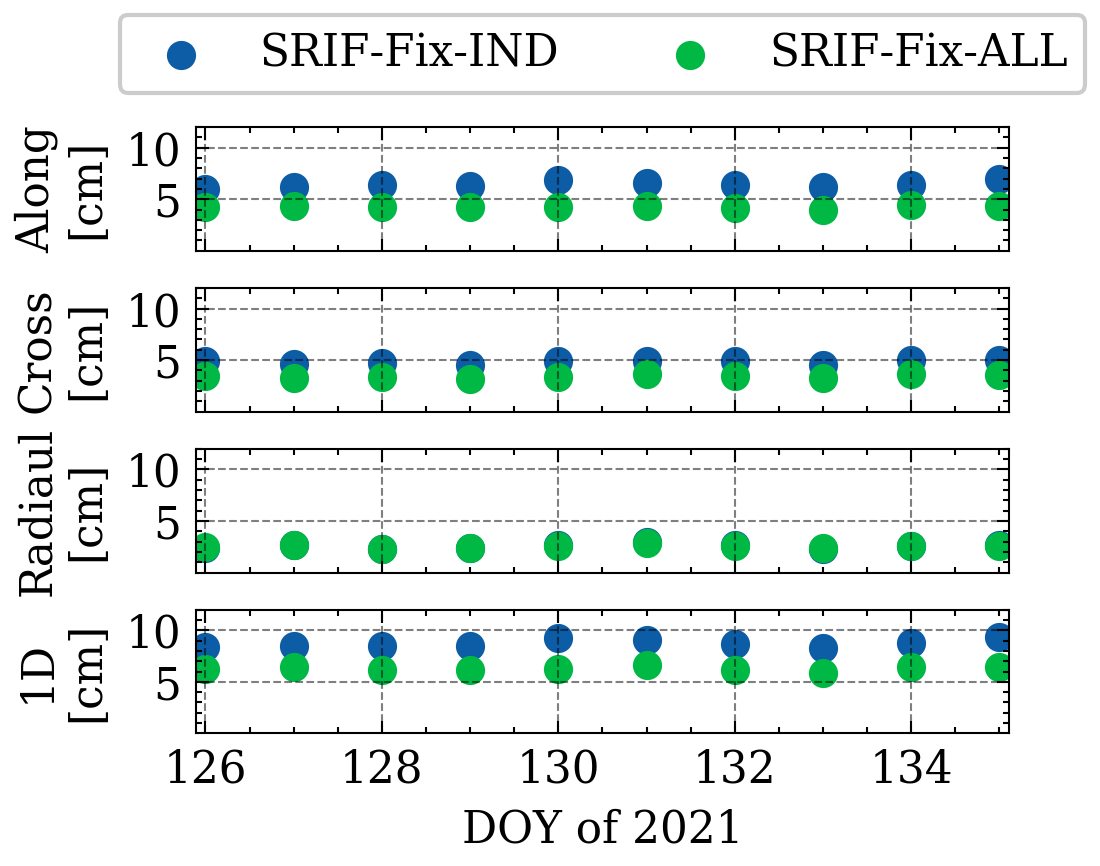
图 3.7 给出了GPS系统仿实时滤波轨道浮点解和固定解的轨道结果与COD产品轨道比较的平均RMS的统计结果。首先可以看到对于所有卫星不同处理策略的解算结果而言，其在切向、法向、径向上的平均RMS值基本上优于10cm，9cm，和5cm。相较于浮点解的结果，可以看到SRIF-Fix-IND和SIRF-Fix-ALL两者固定解方案对轨道精度均有所改善。其中两者改善效果都呈现了在切向和法向方向上改善更为明显，而在径向方向上则改善微弱的趋势。这结果是符合GNSS轨道确定的算法原理的，由于地面GNSS观测值几何构型受限，因此GNSS观测值在轨道径向方向提供了较多的观测，在浮点解求解过程中就已经能获得较高的精度，模糊度固定对其改善效果微弱，而对于切向和法向方向而言，其在浮点解中包含了较多的观测误差，模糊度固定更多在对径向的垂直方向上提供了改善信息，因此在切向和法向上的改善更为显著（参考文献，Lauricheses et al. 2013）。除此之外，SRIF-Fix-ALL的改善程度要明显优于SRIF-Fix-IND方案，前者在法向和切向上的改善量级约为2-3cm，而后者在切向和方向上的改善量级约1cm左右。在SRIF-Fix-ALL的方案中，所有GPS卫星经过固定后的轨道结果在切向、法向和径向上均优于7cm，4cm和3cm。为了进一步量化两者固定解方案对轨道的改善程度，这里统计了所有卫星轨道比较的平均RMS值，如下表3.2所示。

GPS仿实时滤波轨道确定不同处理策略下轨道结果与COD比较平均RMS结果统计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Along[cm] | Cross[cm] | Radial[cm] | 3D[cm] |
| SRIF-Float | 6.9 | 4.9 | 2.3 | 9.1 |
| SRIF-Fix-IND | 6.2 | 4.4 | 2.2 | 8.2 |
| SRIF-Fix-ALL | 4.2 | 3.1 | 2.2 | 5.8 |

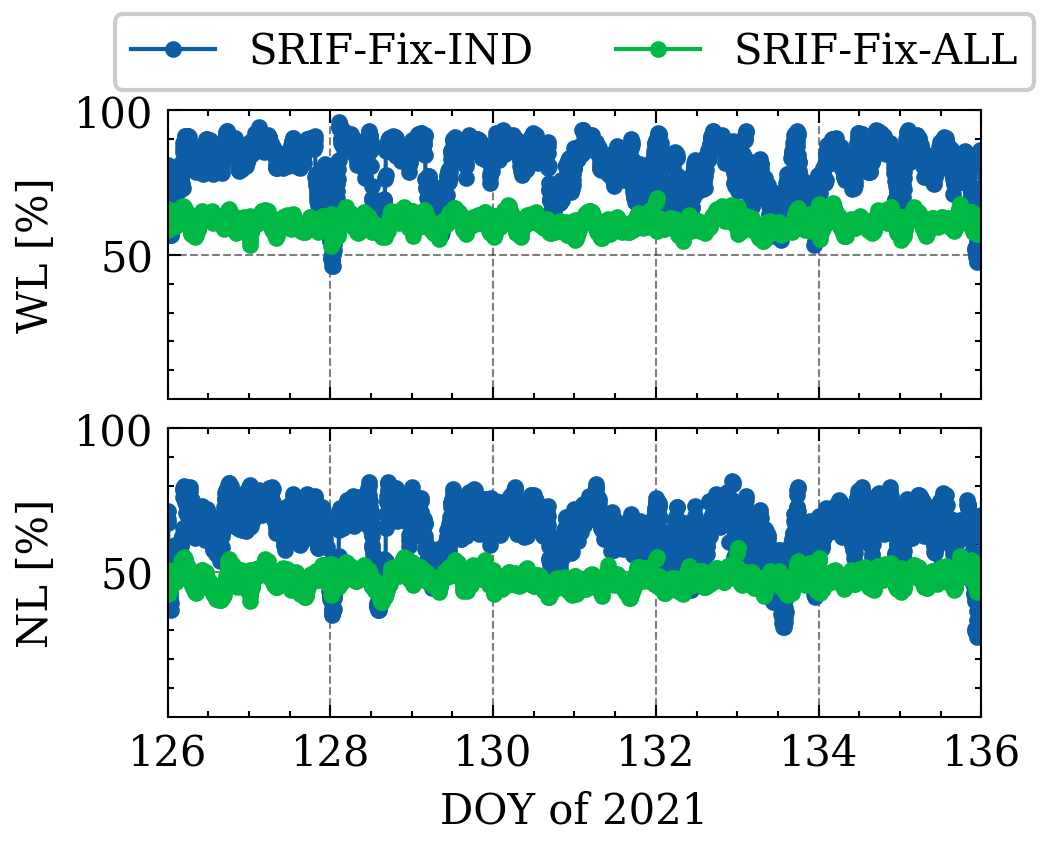
从表中可以看到，对于SRIF-Fix-IND固定解的解决方案，轨道精度从原有浮点解的（6.9cm，4.9cm，2.3cm）改善至（6.2cm，4.4cm，2.2cm），分别在切向、法向和径向上改善了10.1%，10.2%和4.3%。三维方向精度则从原有的9.1cm改善至8.2cm，提升了9.9%，改善量级在1cm左右。相较而言，SRIF-Fix-ALL固定解的方案则是有了更为显著的改善。其在切向、法相和径向上精度为4.2cm，3.1cm和2.2cm，相较于浮点解分别改善了39.1%，36.7%和4.3%，在三维方向的精度也提升至了5.8cm，改善了36.3%，改善量级在3cm左右。该固定解结果也略优于目前的研究结果（Dai et al.2019, Li et al. 2019）。上述轨道三个方向上的改善程度也印证了前述的固定解方案在切向和法向的改善效果要显著于在径向方向上的改善的结论。对于SRIF-Fix-IND和SRIF-Fix-ALL两者方案而言，改善差距主要体现在切向和法向上。考虑到两者处理策略的区别主要在于待模糊度独立基线的选取以及模糊度固定约束上，而在实时滤波轨道处理中，实时模糊度求解的质量相较于事后求解质量更差（理论上是单向平滑和双向平滑的差距），由此可能进一步导致了模糊度固定错误率较高，从而导致两者固定解方案结果出现差异。对于SRIF-Fix-ALL方案而言，由于添加了所有模糊度的松约束方程，能够较好地抵消错误固定的模糊度的影响，因此呈现了更好地改善结果。

这里为进一步分析SRIF-Fix-IND和SRIF-Fix-ALL两者方案之间的差异，图 3.8 给出了2021年126-135天内所有卫星轨道结果与COD产品轨道比较平均RMS统计结果的时序图。可以看到，相较于SRIF-Fix-IND方案，SRIF-Fix-ALL方案的轨道精度在处理时段内都呈现了稳定的改善效果。其中在切向上改善最为明显，法向其次，径向则基于没有改变，与之前的结果一致。



* + - * 1. GPS仿实时滤波固定解轨道结果与COD产品轨道比较平均RMS的时序图

图 3.9 则给出了处理弧段内两者固定策略每个处理历元的WL和NL模糊度固定率的时序图。表3.3给出了两者处理策略的固定率均值的统计结果。可以看到，不论是WL还是NL模糊度，SRIF-Fix-IND方案都呈现出明显高于SRIF-Fix-ALL方案的固定率，其中SRIF-Fix-IND方案的WL和NL模糊度的固定率均值80.8%和65.1%，而SRIF-Fix-ALL方案的固定率为61.1%和47.8%。除此之外，可以看到SRIF-Fix-IND方案中WL和NL模糊度固定率都变动的较为剧烈而SRIF-Fix-ALL方案中则表现得相对平稳。尽管SRIF-Fix-IND表现出了更高的模糊度固定率，但其对轨道精度的改善程度反而更小。这里分析导致该现象的可能原因，首先考虑到两者固定解方案尽管理论上具有等价的固定效果，甚至对于SRIF-Fix-IND上应该表现略微更优（因为SRIF-Fix-ALL方案选择了对所有的模糊度固定约束，其中必然是会包含有部分错误固定的模糊度），但前提是SRIF-Fix-IND方案所固定的独立模糊度中不应当包含有错误固定的模糊度，即独立基线或者是独立模糊度集的选取算法对SRIF-Fix-IND方案结果具有较大的影响（Ge et al.2006）。然而SRIF-Fix-IND方案所采用独立集选取方法仍是沿用事后处理相同的算法，并不适用于实时处理。对于实时滤波轨道处理中，由于每个历元的模糊度信息是实时动态变化的，常规的算法容易受限，这里的一个表现就是SRIF-Fix-IND方案的模糊度固定率随时间的变化程度较大，容易出现固定率“断崖式”下跌的现象。这里考虑到对实时模糊度选取上的复杂性，同时SRIF-Fix-ALL方案本身对错误固定模糊度就具有更好地鲁棒性，算法实现上也相较容易，因此相比SRIF-Fix-IND方案，其在实时滤波轨道处理中的会表现出更好的稳定性和改善效果。

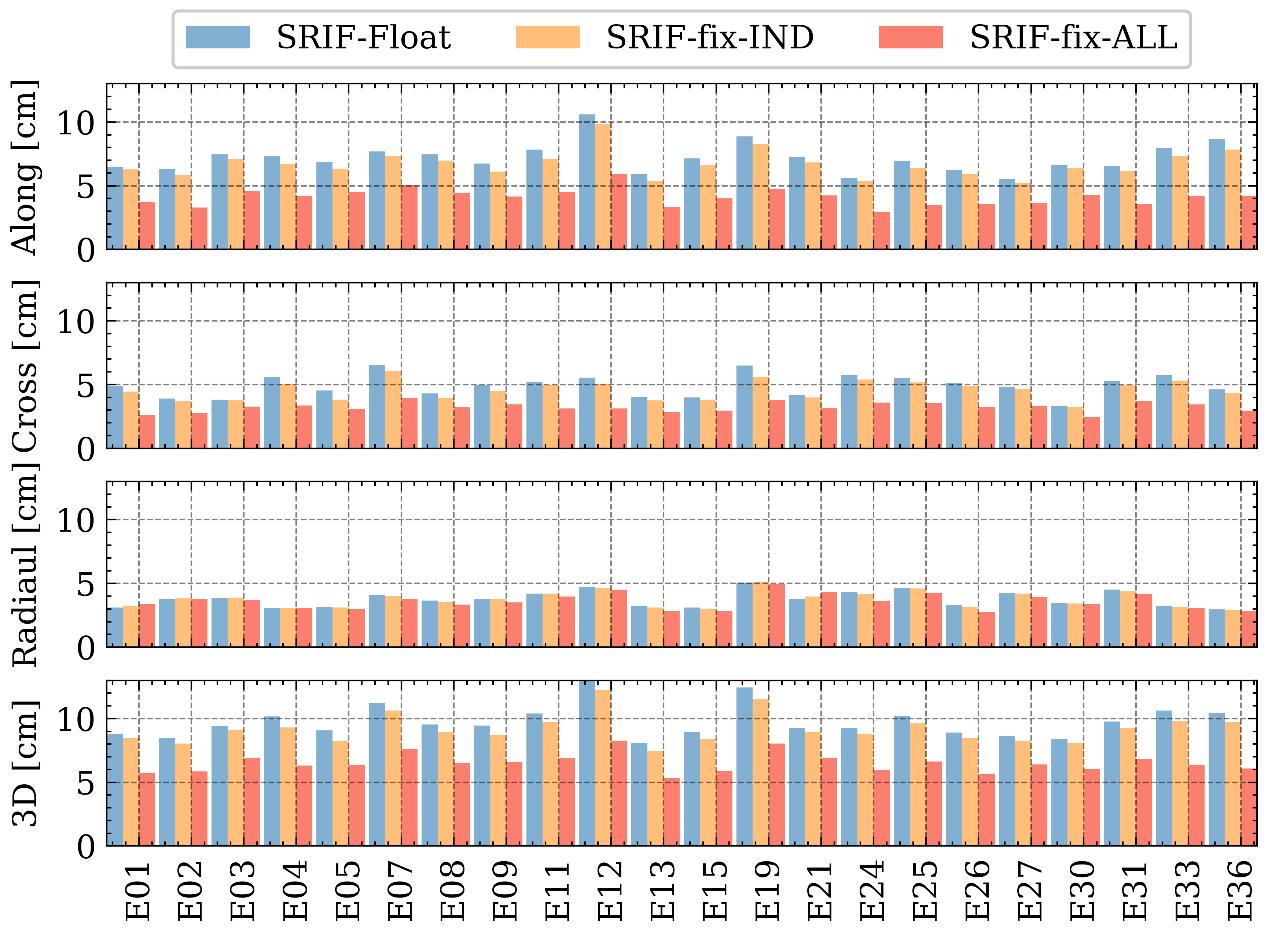


* + - * 1. GPS仿实时滤波轨道固定解中WL和NL模糊度固定率时序图

GPS仿实时滤波轨道固定解中模糊度固定率均值统计表

|  |  |  |
| --- | --- | --- |
| Item | WL [%] | NL [%] |
| SRIF-Fix-IND | 80.8 | 65.1 |
| SRIF-Fix-ALL | 61.1 | 47.8 |

上述主要分析了模糊度固定算法对GPS系统的影响。接下来分别给出模糊度固定算法在GAL和BDS系统上的实验结果。图 3.10 给出了GAL系统仿实时滤波轨道各卫星在处理弧段内轨道结果与COD产品轨道比较的RMS统计图。首先可以看到，对于绝大多数卫星而言，其浮点解轨道精度在切向，法向和径向上的精度分别优于8cm，6cm和4cm。两种固定解方案的轨道精度相较浮点解均有所改善，主要体现在切向和法向上，径向上则改善微弱。类似上述GPS的实验结果，SRIF-Fix-ALL的固定解的提升精度要明显优于SRIF-Fix-IND方案，其中SRIF-Fix-IND在切向和法向上相对浮点解的改善量级基本在1cm以内，而SRIF-Fix-ALL的方案对切向和法向上基本在2-3cm内。表3.4进一步给出了处理弧段内所有卫星的平均RMS统计结果。可以看到，SRIF-Fix-IND的实验方案将GAL轨道精度从原有浮点解的（7.2cm，4.9cm，3.8cm）提升至（6.7cm，4.6cm，3.7cm），分别在切向、法向和径向上改善了6.9%，6.1%和2.6%，总体精度改善较为微弱。而采用SRIF-Fix-ALL的处理策略，GAL系统固定后的实时轨道精度在切向，法向和径向上分别能够达到4.1cm，3.2cm和3.6cm，相较于浮点解分别改善了43.1% ，34.7%和5.3% 。轨道三维精度也从原有的9.8cm改善至6.5cm，提升了33.7%，整体轨道精度改善显著，改善幅度与GPS实验结果相当。

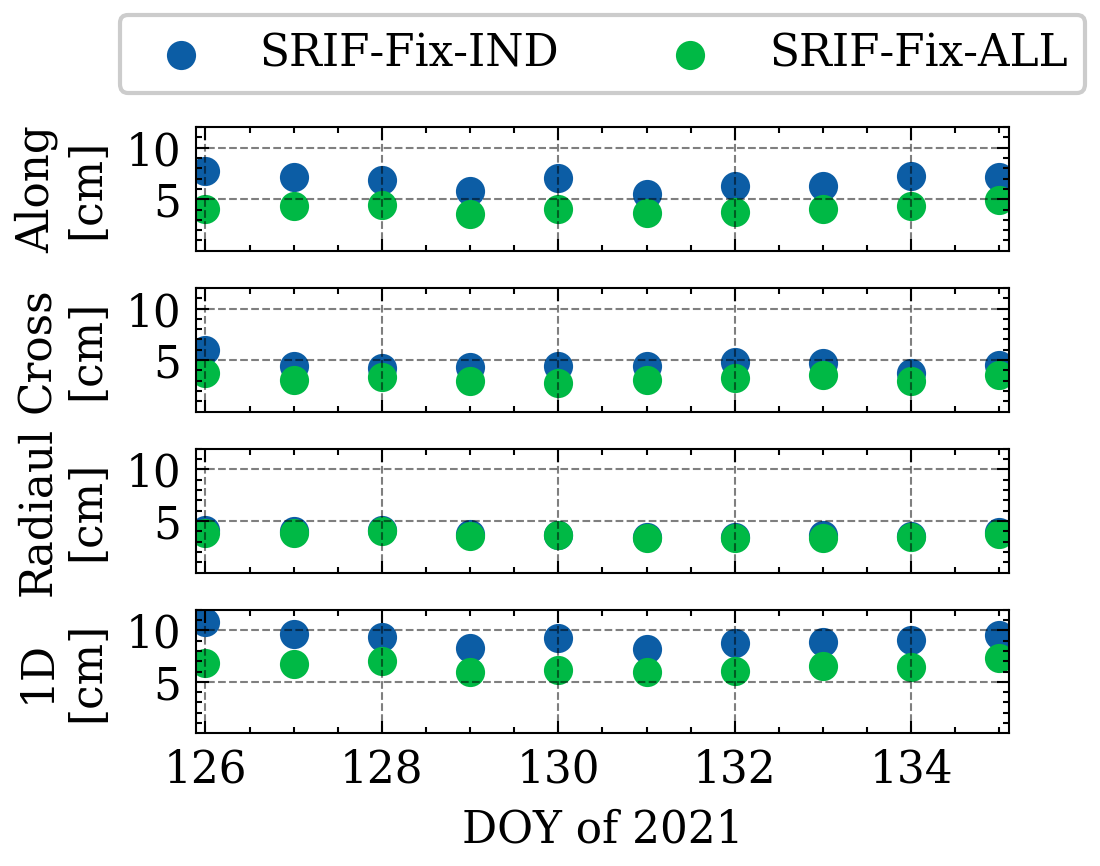


* + - * 1. GAL卫星仿实时滤波轨道确定浮点解和固定解与COD事后产品轨道比较RMS统计图

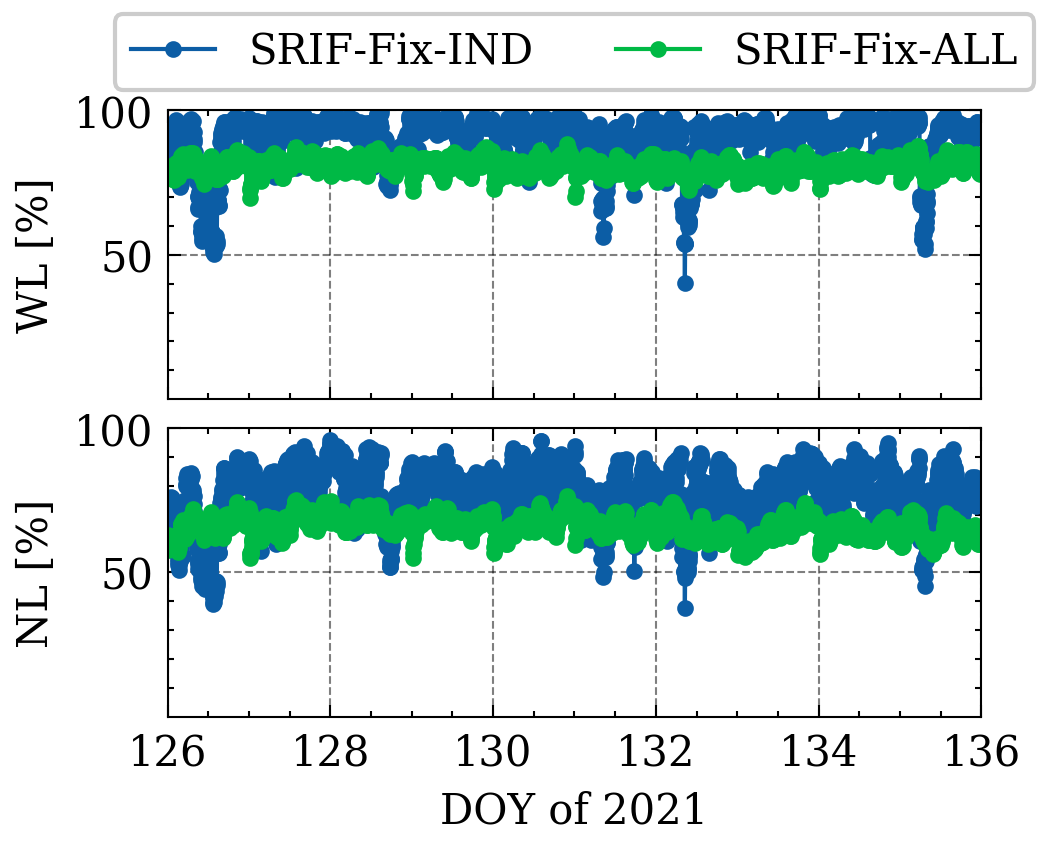
GAL仿实时滤波轨道确定不同处理策略下轨道结果与COD比较平均RMS结果统计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Along[cm] | Cross[cm] | Radial[cm] | 3D[cm] |
| SRIF-Float | 7.2 | 4.9 | 3.8 | 9.8 |
| SRIF-Fix-IND | 6.7 | 4.6 | 3.7 | 9.2 |
| SRIF-Fix-ALL | 4.1 | 3.2 | 3.6 | 6.5 |

为进一步对比两种固定解方案对GAL仿实时轨轨道结果的影响。图 3.11 给出了处理弧段内GAL卫星轨道与COD轨道比较平均RMS的时序图。类似GPS实验结果，相对于SRIF-Fix-IND的实验方案，SRIF-Fix-ALL的轨道精度在处理弧段内的有着稳定的提升。图 3.12 则给出了两者实验方案在处理弧段内每个历元的WL和NL模糊度固定率的时序图，表3.5给出了模糊度固定率平均值的统计信息。可以看出，GAL实验结果也同样呈现出了SRIF-Fix-IND的WL和NL模糊度固定率均高于SRIF-Fix-ALL方案的现象，同时SRIF-Fix-IND的模糊度固定率随时间变化幅度，而SRIF-Fix-ALL的模糊度固定率则保持较为稳定的状态，原因与前述GPS的实验分析类似，这里不再赘述。其中，SRIF-Fix-IND的WL和NL模糊度固定率能够达到90.4%和80.9%，而SRIF-Fix-ALL的WL和NL模糊度固定率为77.4%和66.0%。两者方案的模糊度固定率均显著高于了GPS的模糊度固定率。



* + - * 1. GAL仿实时滤波固定解轨道结果与COD产品轨道比较平均RMS的时序图

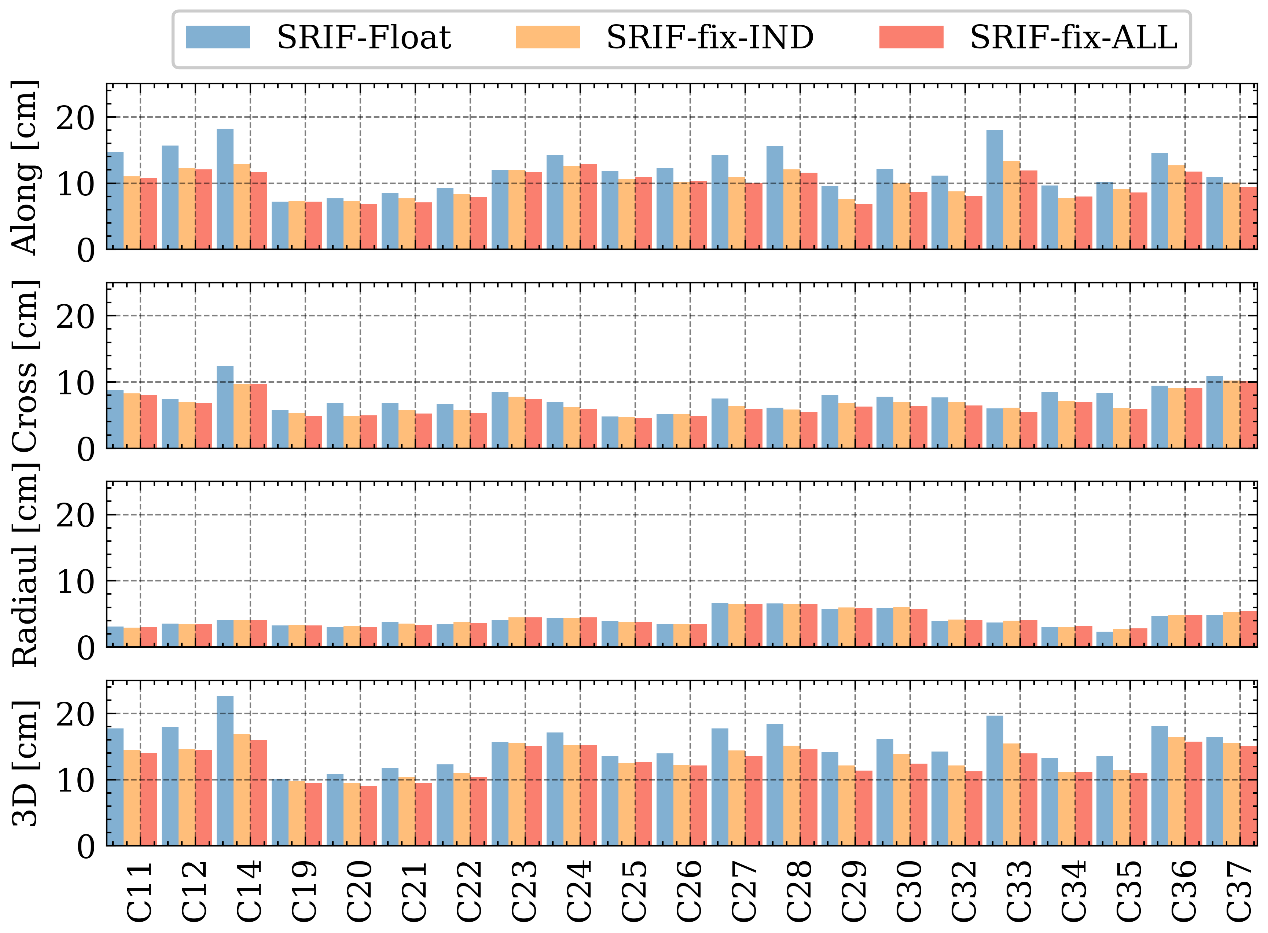


* + - * 1. GAL仿实时滤波轨道固定解中WL和NL模糊度固定率时序图

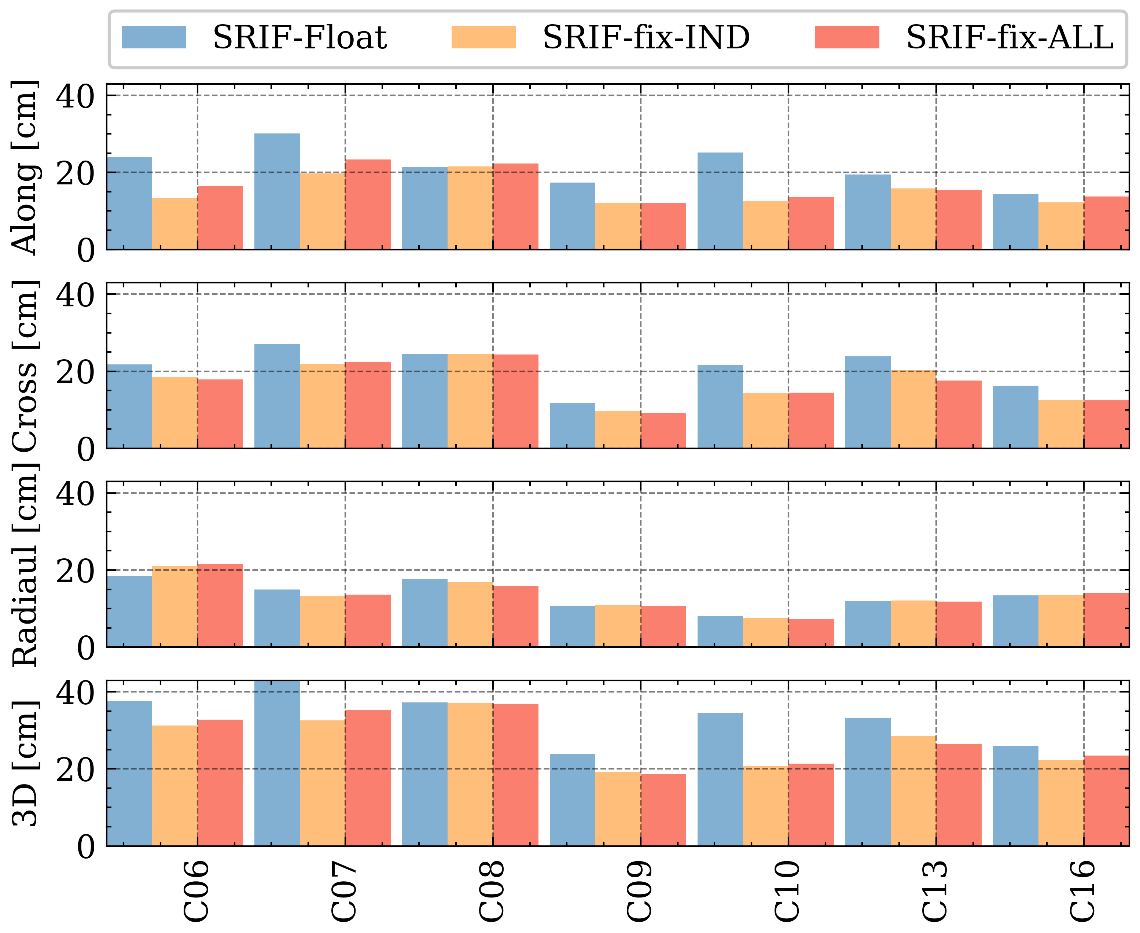
GAL仿实时滤波轨道固定解中模糊度固定率均值统计表

|  |  |  |
| --- | --- | --- |
| Item | WL [%] | NL [%] |
| SRIF-Fix-IND | 90.4 | 80.9 |
| SRIF-Fix-ALL | 77.4 | 66.0 |

对于北斗系统的实时滤波轨道确定，由于这里暂不考虑GEO卫星的解算，因此模糊度固定算法仅作用在MEO和IGSO卫星上，图 3.13 和图 3.14 分别给出了BDS MEO和IGSO卫星的仿实时滤波轨道与COD产品轨道比较的RMS统计图。对于BDS-MEO卫星而言，其浮点解在切向、法向和径向上分别基本优于 18cm，10cm和6cm。两种固定解方案对BDS MEO卫星轨道精度均有所改善，与GPS和GAL卫星不同，这里两种方案对浮点解轨道改善程度接近，在切向上改善量级约为1-2cm，在法向上改善量级为1cm以内，对径向基本都没有改善。对BDS IGSO卫星而言，其浮点解轨道在切向、法向和径向上基本优于30cm，25cm和20cm。类似地，两种固定解方案对BDS IGSO卫星浮点解轨道改善程度基本相当，甚至在部分卫星上（C06、C07和C08）上SRIF-Fix-ALL的方案的改善程度会略低于SRIF-Fix-IND方案。表3.6分别针对BDS MEO和IGSO卫星统计了仿实时滤波轨道与COD产品轨道比较的平均RMS值。可以看到，SRIF-Fix-IND方案下BDS MEO卫星轨道精度能够达到（10.2cm，6.8cm和4.3cm），相较于浮点解在切向和法向上分别提升了17.1%和10.5%，轨道三维精度为13.3cm，提升了14.2%。而采用SRIF-Fix-ALL方案的BDS MEO卫星轨道精度为（9.7cm，6.5cm和4.2cm），相较于浮点解在切向和法向上分别提升了21.1%和14.5%，轨道三维精度为12.8cm，提升了17.4%。与前面的结论相同，两者固定解方案对BDS MEO卫星的轨道精度改善程度相当，且SRIF-Fix-ALL的方案会略优于SRIF-Fix-IND方案。对于BDS IGSO卫星而言，两者固定解方案同样对于径向上的轨道精度均没有改善。在切向和法向上，使用SRIF-Fix-IND方案轨道精度提升了29.5%和17.1%，使用SRIF-Fix-DD方案轨道精度提升了23.0%和19.5%。使用两者方案最终的轨道三维精度为27.4cm和27.8cm，在同一精度水平。



* + - * 1. BDS MEO卫星仿实时滤波轨道浮点解和固定解与COD事后产品轨道比较RMS统计图

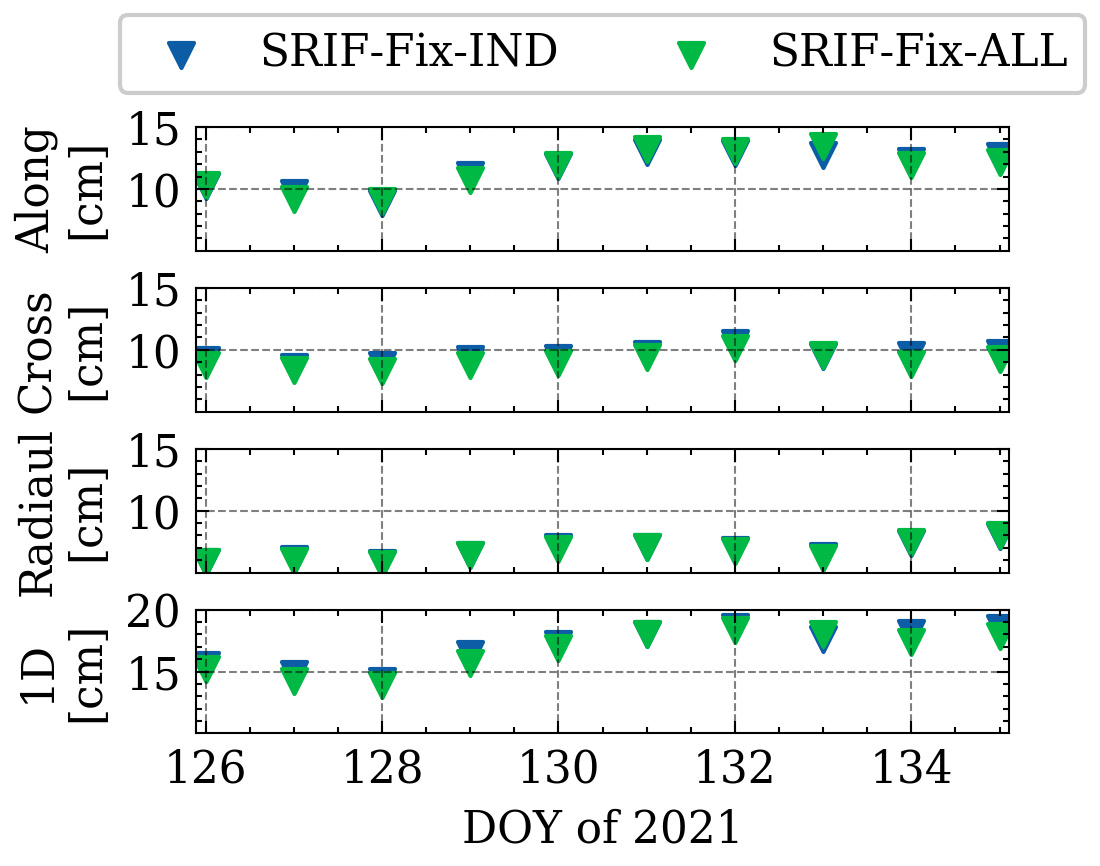


* + - * 1. BDS IGSO卫星仿实时滤波轨道浮点解和固定解与COD事后产品轨道比较RMS统计图

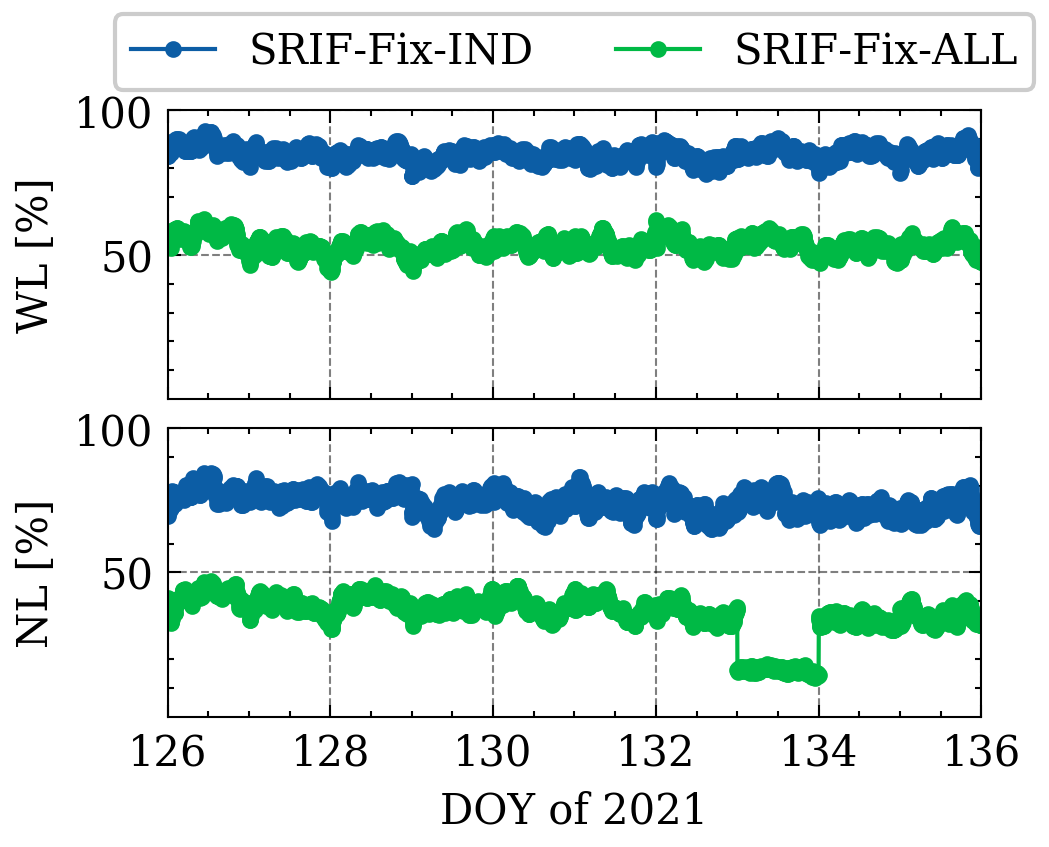
BDS仿实时滤波轨道浮点解和固定解与COD事后产品轨道比较平均RMS统计表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Item | | Along[cm] | Cross[cm] | Radial[cm] | 3D[cm] |
|  | SRIF-Float | 12.3 | 7.6 | 4.2 | 15.5 |
| BDS-MEO | SRIF-Fix-IND | 10.2 | 6.8 | 4.3 | 13.3 |
|  | SRIF-Fix-ALL | 9.7 | 6.5 | 4.2 | 12.8 |
|  | SRIF-Float | 21.7 | 21.0 | 13.6 | 33.7 |
| BDS-IGSO | SRIF-Fix-IND | 15.3 | 17.4 | 13.6 | 27.4 |
|  | SRIF-Fix-ALL | 16.7 | 16.9 | 13.6 | 27.8 |

相较于GPS和GAL的实验结果而言，模糊度固定算法对BDS轨道的精度提升程度较小，且两种模糊度固定方案没有呈现明显的区别。这里主要可能的原因在于，对于BDS实时滤波轨道确定中所采用的力学模型尚不完善，导致轨道参数吸收了力学模型模型中带来的误差，估计精度受限，这里仅从BDS浮点解的轨道结果就可以看出其与GPS和GAL卫星尚存在一定差距。进而导致轨道参数对模糊度固定带来的改善并没有那么敏感。因此这两种模糊度固定算法对轨道改善的差异就被进一步缩小了。图 3.15 给出了BDS仿实时滤波轨道解于COD产品轨道比较的平均RMS时序图。可以看到两者固定解方案的轨道精度随时间变化处于基本相当的趋势，大部分时间下SRIF-Fix-ALL的方案相较于SRIF-Fix-ALL仍处在精度略优的趋势。图 3.16 则进一步给出BDS仿实时滤波轨道中WL和NL模糊度固定率的时序图，表3.7则给出了BDS模糊度固定率的平均值结果。SRIF-Fix-IND的固定率依然显著高于SRIF-Fix-ALL方案的固定率，前者的WL和NL模糊度固定率能达到85.1% 和74.3%，后者则分别为53.5%和35.6%。其结果与GPS系统类似。但与GPS和GAL实验结果不同，BDS的SRIF-Fix-IND方案下的模糊度固定率随时间变化变化较为稳定，这也侧面反映了该处理时段内的观测数据质量相对较好，进而缩小了两者固定解方案的差异。同时可以发现，对于SRIF-Fix-ALL的方案，年积日133天内的NL模糊度固定率出现了下降的跳变，进而影响了固定解轨道精度，从而导致了轨道精度一定程度上的降低，这里从图 3.15 也可以看出在年积日为133天下SRIF-Fix-ALL的轨道精度要低于SRIF-Fix-IND的轨道精度。



* + - * 1. BDS仿实时滤波轨道固定解与COD产品轨道比较平均RMS时序图



* + - * 1. BDS仿实时滤波轨道固定解中WL和NL模糊度固定率时序图

BDS仿实时滤波轨道固定解中模糊度固定率平均值统计表

|  |  |  |
| --- | --- | --- |
| Item | WL [%] | NL [%] |
| SRIF-Fix-IND | 85.1 | 74.3 |
| SRIF-Fix-ALL | 53.5 | 35.6 |

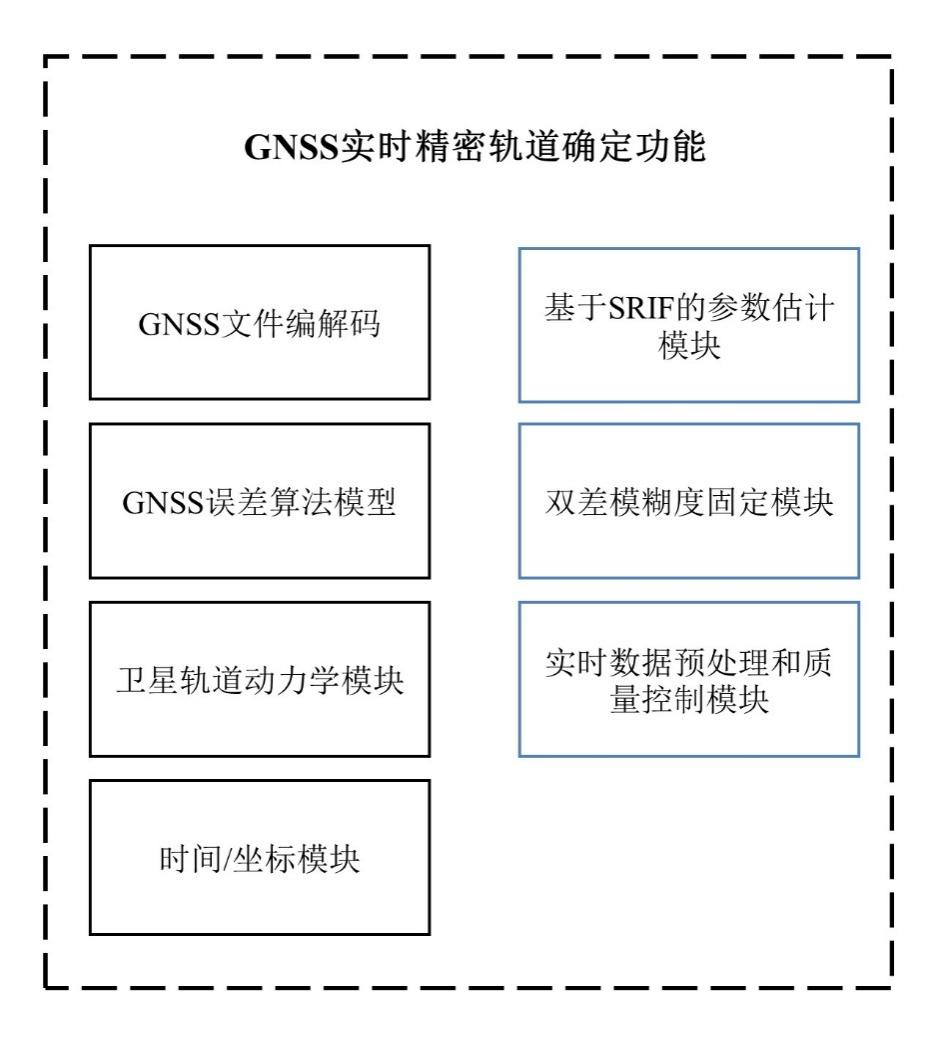
综上所述，前述的双差模糊度固定算法能够有效地提升实时滤波轨道的精度，同时实验进一步验证了采用SRIF-Fix-ALL的固定解方案更适用于实时轨道处理中，特别对于GPS和GAL系统，其固定解结果对轨道精度提升基本在30-40%之间。

## 实时滤波精密轨道处理软件平台

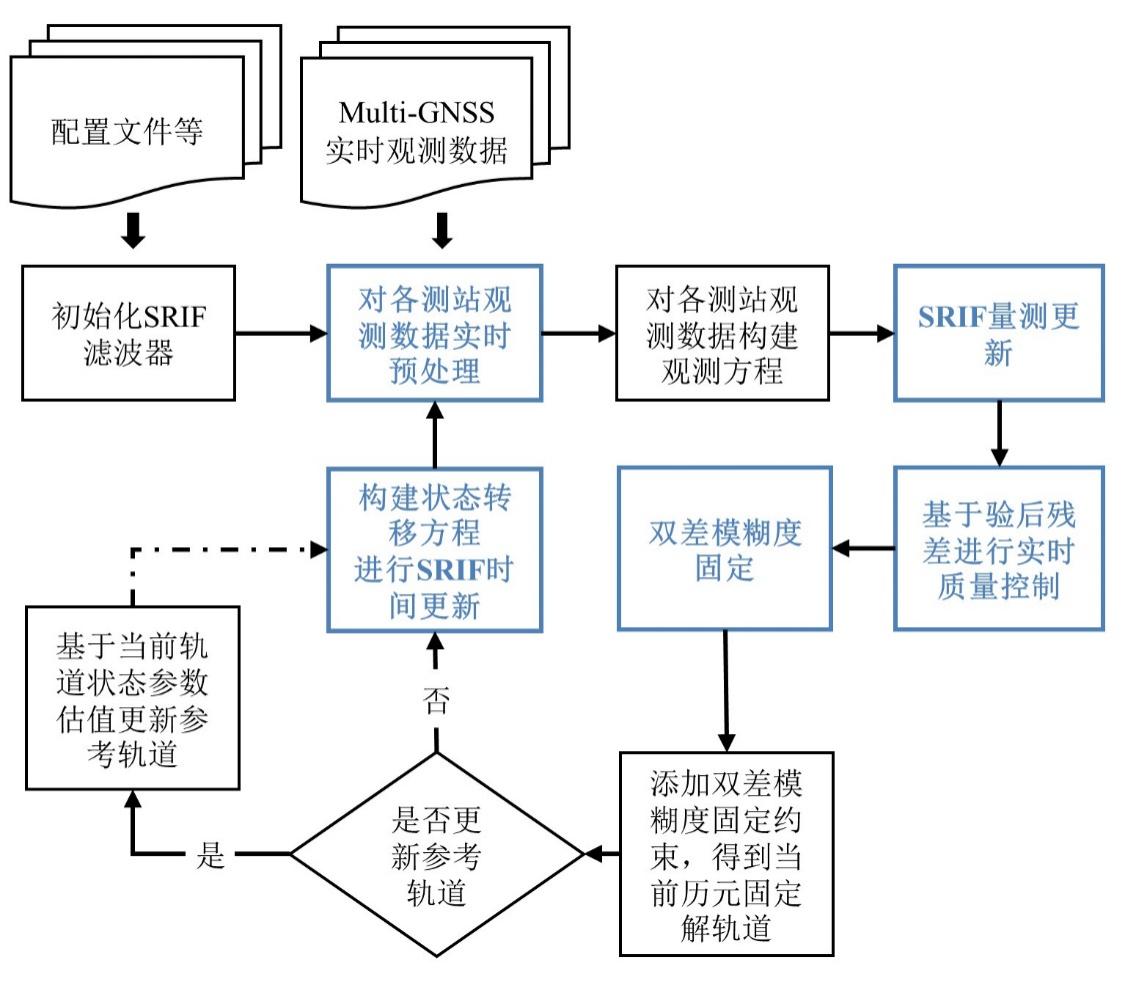
前面依次针对GNSS实时滤波精密定轨中的参数估计、数据质量精化以及模糊度固定算法进行了推导和实验验证。本人则在GREAT（GNSS+ Research, Application and Teaching）软件平台基础上完整实现了上述实时滤波轨道的相关算法，并开发完成了基于平方根信息滤波的GNSS实时精密轨道确定的功能。GREAT软件是由武汉大学测绘学院开发和设计，其主要用于大地测量、导航定位领域内的科学工程研究和教学工作（参考文献，GREAT-UPD）。接下来对其中完成的实时滤波轨道功能模块和算法流程进行简单介绍。

首先该功能模块上同样保持了原有GREAT软件平台的特性，采用C++进行开发，具有跨平台、高效率、易配置和扩展性强等特点。由于原有GREAT软件仅具有基础的GNSS数据处理功能以及传统的基于事后处理模式的精密轨道确定功能，因此这里针对实时精密轨道确定算法，将其中的核心功能抽象成了以下三个模块：基于SRIF的参数估计模块，实时数据质量精化模块和实时双差模糊度固定模块。在此基础上，构成了整个实时滤波轨道处理功能。图 3.17 给出了实时滤波轨道功能中的模块构成，图中右边蓝色框内为重点新增的功能模块。

图 3.18 则给出了实时滤波轨道功能的整体算法流程图。图中蓝色边框内的流程对应了上述新增的算法功能模块，也是软件开发过程中重点完成的部分，其中的算法原理和详细流程在本章前面均已进行相关阐述。



* + - * 1. 基于SRIF的GNSS实时精密定轨功能模块组成图



* + - * 1. 基于SRIF的GNSS实时精密定轨算法流程图

## 本章小结

本章针对实时滤波轨道确定处理中的关键环节进行了梳理和推导验证。首先是参数估计方法，这里选用了数值稳定性更高的平方根信息滤波作为参数估计的算法，并对其中的时间更新算法和量测更新算法进行了推导和实现，给出了其在实时轨道处理中的应用方法和处理流程。接下来分别给出了实时数据质量预处理算法和基于SRIF的实时质量控制算法的相关原理，并通过与事后质量控制方法对比测试的方式验证了实时数据精化算法的有效性，实验结果表明，采用实时数据质量检测算法的GPS仿实时滤波浮点解在三维方向上轨道精度能够达到9.1cm。然后重点阐述了实时滤波轨道处理中双差模糊度固定算法的原理和应用，并通过对多个系统的仿实时滤波轨道实验验证了模糊度固定算法能够有效提升轨道精度。同时对比测试了不同模糊度固定方案的改善效果，实验结果表明，采用对所有双差模糊度做松约束固定的方式改善效果最好，更适用于实时滤波轨道处理中，其中对GPS、GAL、BDS IGSO和MEO卫星，三维方向上的轨道精度分别提升了36.3%，33.7%，17.4%和17.5%。最后在GREAT软件平台上开发完成了上述关键环境的算法模块，并给出了软件功能模块组成和整体的算法流程图。

# GNSS实时滤波精密轨道确定中的效率优化

## 引言

提供GNSS实时精密轨道服务的一个重要前提就是需要保证轨道的实时性。对于精密定轨的事后处理模式，通常是通过对当前处理弧段的轨道进行预报外推，从而使用预报轨道作为实时产品提供给用户使用，因此为了保障实时需求，事后处理模式通常可以灵活地根据轨道处理效率预报特定的时间以提供稳定的实时产品。与此不同，在实时滤波精密轨道确定中，需要对实时数据进行处理解算完成后直接播发实时轨道信息，因此其解算时间在根本上决定了实时产品的播发时延和可用性。另一方面，随着近年来各GNSS系统和地面测站的不断建设完善，可用的GNSS观测数据也呈现了快速增长的趋势，不断增加了GNSS精密定轨处理过程的计算负担。这些因素都表明着具备实时高效的处理算法是实时滤波轨道提供实时服务的一个关键部分。

因此本章将首先介绍高性能计算中常见的优化方法，并给出基于SRIF的实时滤波轨道处理流程算法的整体优化思路。其次分别给出针对实时滤波定轨处理流程中的观测方程构建、卫星轨道积分、SRIF滤波量测/时间更新、实时双差模糊度固定给出了具体的优化原理和方法，同时通过实验对比分析了各部分优化算法的有效性。

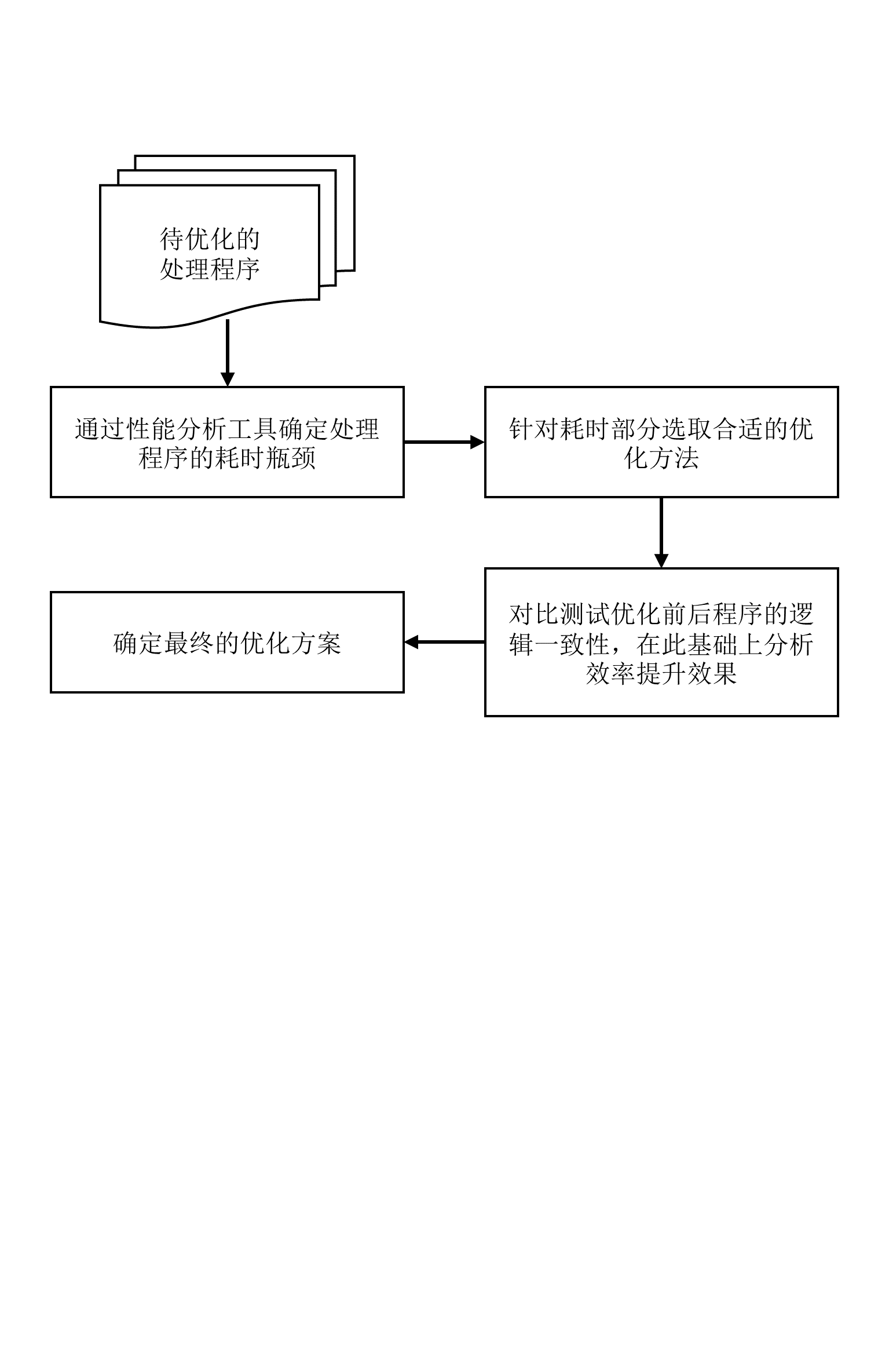
## 实时滤波轨道的优化方法

在本文的3.5节中已经提到了本文的实时滤波轨道功能模块主要基于GREAT软件平台进行开发。其编写语言为C++，同时主要运行环境为常见的计算服务器。因此对实时滤波轨道程序进行效率优化可以遵循一般的程序优化方法。本小节首先给出了程序设计优化的基本原理和常用方法。然后分析了基于SRIF的实时滤波轨道确定的算法流程，对其中的关键环节给出了相应的优化策略。

* + 1. 程序设计优化的一般方法

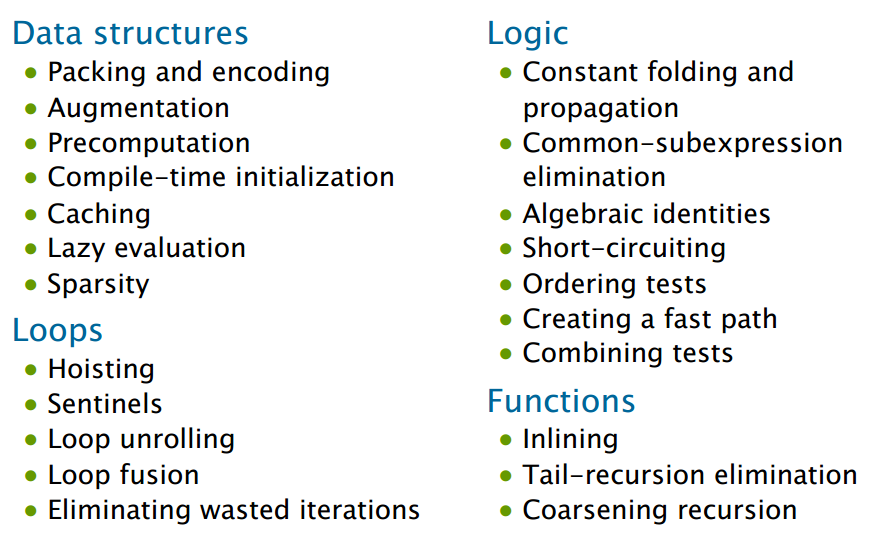
尽管根据摩尔定律（Moore’s Law）处理器芯片的性能将会每隔一段时间就发生翻倍的变化，但仅依仅赖硬件更新对程序效率提升是有限的。尤其是频繁更新硬件在人力和物力上通常是难以接受和实现的，同时也无法根本上解决对于类似GNSS观测数据将随时间快速增长的情况。另一方面，目前大部分的软件程序本身并没有充分发挥现代处理器硬件的性能，因此从软件层面上对程序设计进行优化以提高处理效率也是目前更为常用的做法。

在对相关处理程序进行优化之前，分析其处理瓶颈是一个重要前提。针对不同类型以及不同运行平台上的程序，目前已有很多的程序性能分析工具（如Java的JVMTI接口，Inte Vtune软件，GNU性能分析工具gprof等）。在此基础上，需要找出处理程序中的运行时间的瓶颈部分，进一步选择合适的优化方法，通过不断对比测试确保程序逻辑不变的前提下，分析处理效率的提升效果，从而确定最终的优化方案。下图xxx给出上述的程序设计优化的一般思路。



* + - * 1. 程序优化的一般过程

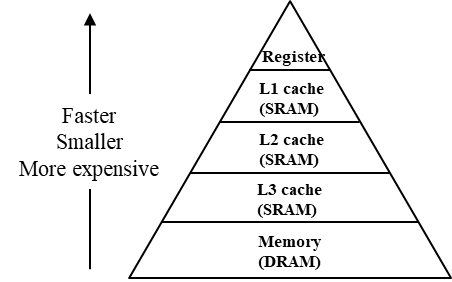
对于目前常用的软件优化方法，根据其特点，大致可以分为以下四类：首先是从算法设计层面进行改善。这里需要结合处理程序所解决的问题，优化处理算法的时间复杂度，从而减少求解问题所需的工作量。一个常见的例子就是选用合理的排序算法，对于大量无序的数据情况下，快速排序（Quick Sort）算法通常具有比冒泡排序算法更小的时间复杂度。其次是软件编写中的一些通用的优化技巧。Jon Bentley等总结了一系列与程序语言、程序运行平台无关的优化方法被称为Bently’ Rules（参考书籍，Writing Efficient Programs），被广泛应用。随着时间也发展出了New Bently’ Rules等准则（如图 4.2 所示）。



* + - * 1. MIT 6.172上提出的New Bently Rules

接着是和程序运行的处理器架构相关的优化技术。现代处理器的一大特点即是具有支持直接操作矢量数的硬件结构，即SIMD（Single instruction stream，multiple-data stream）技术。其本质上是通过一条CPU指令完成了多个数据的相同操作，从而实现了空间上的并行性。除处理器架构上的发展，目前存储器的实现也通常采用了分级多层次的结构。器主要思想是通过在高速运行的处理器设备和容量大但处理慢的存储设备中插入一个容量更小处理更快的设备作为数据缓存（如图 4.3 所示）。因此充分利用该层次结构可以改善程序中数据访问过程的时间和空间上的局部性，从而缩短运行时间。运用前述这两种技术的常见例子就是当前不断发展的高性能矩阵库，如Eigen、OpenBlas和Intel MKL等。这些矩阵库为用户提供了具有高效运算的矩阵处理接口，在此基础上，将原有处理程序中的数据运算操作替换为相应的矩阵操作，也是一种目前常见的程序设计优化方法。（参考文献。）

最后的一类优化方法则是从改善程序并行性上出发。这类方法的基本思路均是通过将原有程序处理问题分解为多个可以并行求解的子问题，然后在不同处理单元上实现时间上的并行求解，这类方法往往能大幅度改善程序的运行效率，也是本文所重点采用的优化方法之一。根据应用场景和实现方法的不同，大致主要有以下几类并行优化方式：针对多处理器/集群的并行框架，如OPENMP/MPI；针对GPU平台的并行框架如CUDA、OpenACC等；针对CPU和GPU的异构并行计算框架，如OpenCL；分布式计算框架如MapReduce等。



* + - * 1. 现代存储器的层次结构示意图
    1. 基于SRIF的实时定轨中的效率优化分析

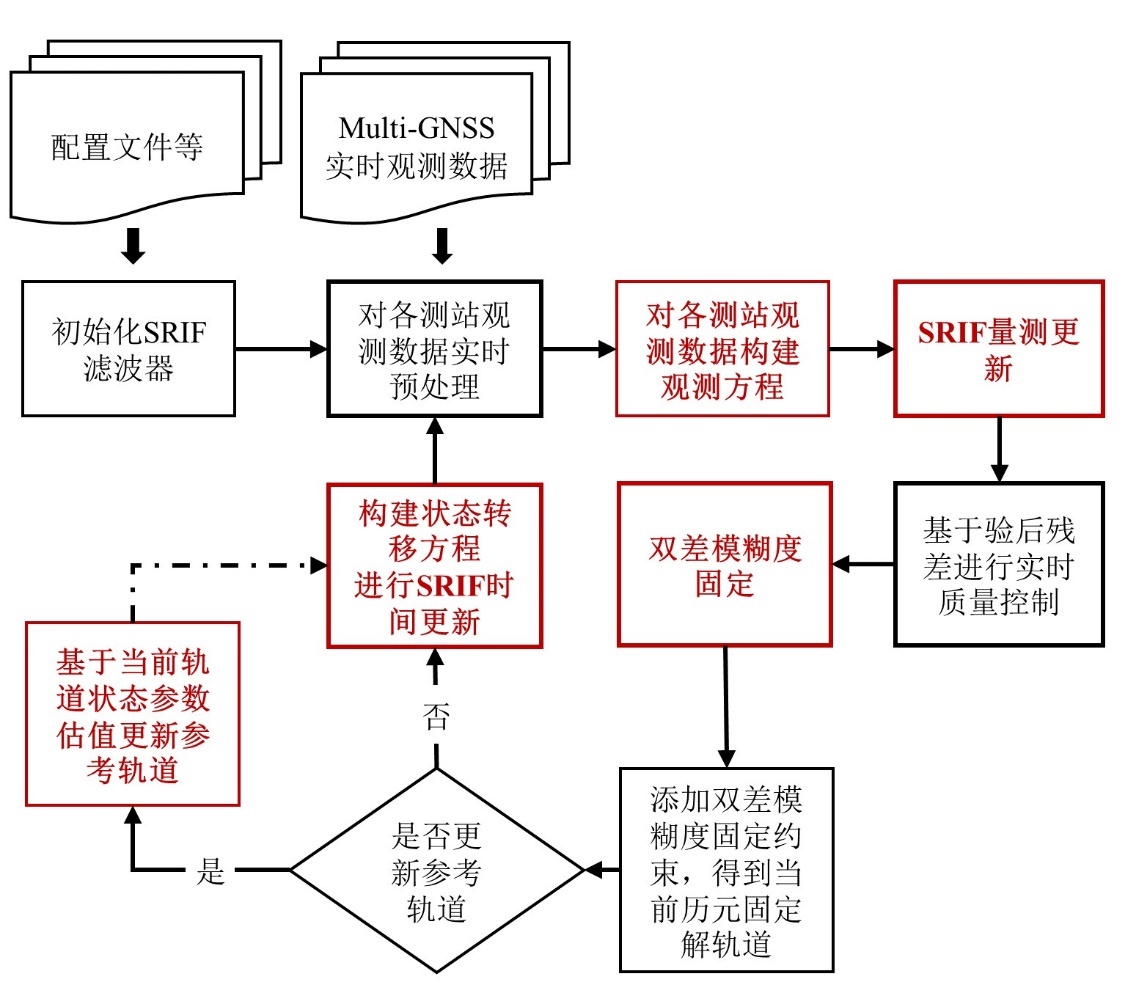
根据前述的程序设计优化思路，这里首先对基于SRIF的实时滤波轨道处理算法流程进行了性能分析，统计了各部分的时间耗时占比。具体实验采用了105个测站的事后观测数据，并对GPS、GAL和BDS进行三系统仿实时滤波轨道解算，对多天的运算结果取历元均值，最后的统计结果如图 4.4 所示。

（饼状图：各部分时间流程图占比图）

* + - * 1. 基于SRIF的实时滤波轨道的处理流程各部分耗时占比统计图

可以看到，其中SRIF的量测更新和时间更新占据了绝大多数计算耗时，主要因为其中包含了较多的矩阵计算，这里可以考虑采用前述的高性能矩阵库的方式对其进行优化。其次分别是卫星轨道积分和观测方程的构建。这两部分的计算耗时是与测站和卫星数量具有正相关的关系，因此也是需要进行优化地一个关键部分。特别地，考虑到对于卫星轨道积分中，各个卫星的计算过程是相互独立，对于各个测站的观测方程的构建也是具有相互独立的性质，因此这里非常适合采用并行计算的方法对这两部分算法流程进行相应优化。除此之外，实时双差模糊度固定算法的耗时占比也是不可忽视的一部分，且实时模糊度固定算法直接与模糊度参数数量相关，而模糊度参数的数量同时取决于测站数和卫星数，因此模糊度固定算法在计算耗时上具有较大的增长趋势，同样需要进行相应优化。类似地，不同测站基线间的双差模糊度固定过程为相互独立，可以考虑采用并行算法进行相关优化，同时对于采用了选取独立基线进行模糊度固定紧约束的方案，双差模糊度独立性检验也是模糊度固定算法中一个主要耗时部分，需要对检测算法进行相应改善。有关这些部分的具有优化方法在后续章节会进行详细阐述，因此这里仅做了整体的简要概括。

综上所述，图 4.5 给出了基于SRIF的实时滤波轨道处理中的优化部分（采用红色边框进行标注），表4.1则给出了这些优化部分的前后策略对比示意图。



* + - * 1. 基于SRI的实时滤波轨道处理的整体算法流程图（红色框图为主要优化部分）

实时滤波定轨处理中各部分优化前后策略对比示意图

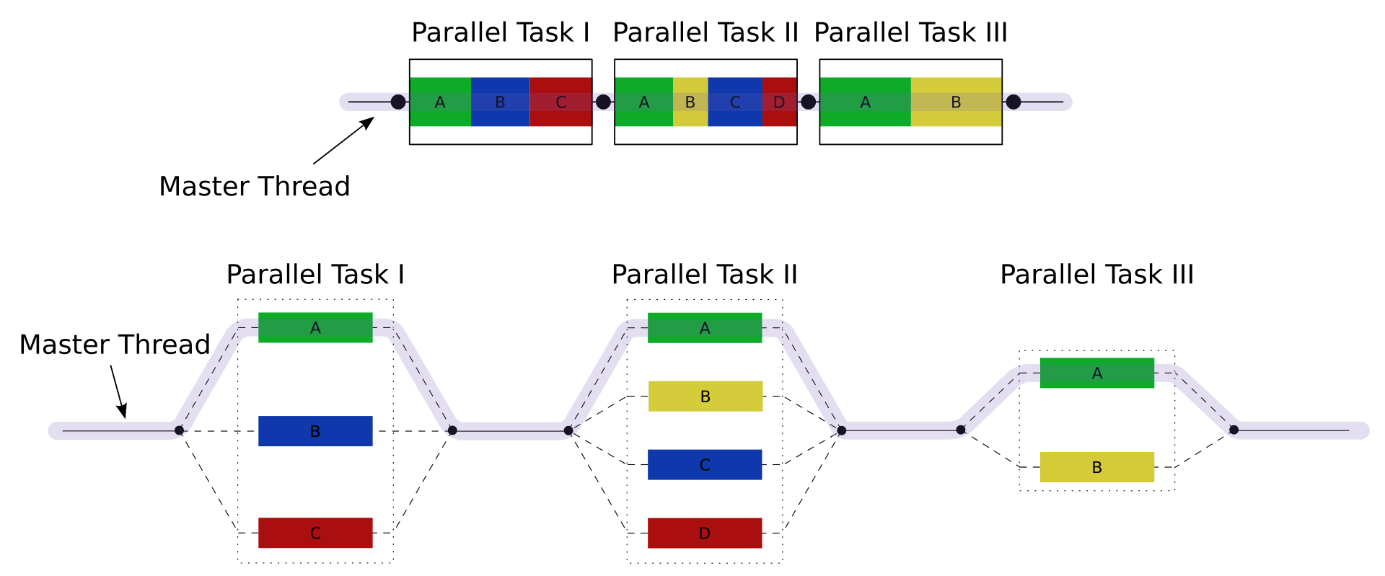
|  |  |  |
| --- | --- | --- |
| Item | 旧策略 | 新策略 |
| SRIF参数估计模块 | 基于C-Lapack的实现 | 基于Eigen的实现 |
| 观测方程构建 | 串行处理 | 基于测站间的并行处理 |
| 卫星轨道积分 | 串行处理 | 基于卫星间的并行处理 |
| 实时双差模糊度固定 | 串行处理 | 基于基线间的并行处理  独立性检测算法改善 |

## 基于OpenMP的多线程并行

本节首先简要介绍了OpenMP的特点和使用方法。接着基于OpenMP的框架，研究了基于测站间并行的观测方程构建以及基于卫星间并行的轨道积分的处理方法，在GREAT平台上实现了两者的并行解算算法。最后通过与原有的串行算法的对比实验进一步分析了该并行算法对处理效率的提升效果，验证了该算法的有效性。

* + 1. OpenMP简介

前述的多种并行计算方法中，OpenMP框架提供了一套跨平台的，支持Fortran、C/C++语言的多线程并行接口。OpenMP主要针对的运行平台是基于共享内存式的多处理器计算平台，其本质上是基于不同系统平台上底层的系统线程库，通过开辟多个新的线程，将原有的串行循环的程序代码分解到不同的线程中进行执行（如图 4.6 所示），最后统一等待所有线程处理完成。但与一般系统线程库不同，OpenMP提供的接口实际为程序编译的指导性注释（Compiler Directive），而非直接提供编程语言上的接口，由此通过编译器自动将对应的程序片段并行化。OpenMP提供的高层抽象接口避免了程序编写者直接与底层的系统线程库直接交互，规避了许多繁琐的细节，尤其对于并行代码实现中常见的难题如线程粒度划分以及线程间负载均衡的问题，OpenMP则是在框架层面上就已经负责实现。从而基于OpenMP框架可以帮助程序编写者更注重于并行算法上的设计，而非具体的实现细节，降低了并行算法实现的难度与复杂度，提高了开发效率。另一方面，基于OpenMP编写的程序具有高移植性，在不支持OpenMP的运行环境内也能让程序以原有的串行模型运行。因此OpenMP框架被广泛应用在了针对多处理器的单机并行算法实现中。



* + - * 1. OpenMP并行原理示意图

使用OpenMP框架实现并行算法的整体流程的基本思路如下所示：首先需要分析原有的串行程序，确定或者构造一个可以进行并行的代码区域（通常为一个循环），在代码区域前加入如下表4.2所示的预编译指令（这里特别针对C++语言）。其中directive和clause都是与具体的并行策略相关的设置。具体地，OpenMP框架将在该代码区域执行前生成一系列子线程，对于代码区域中任务划分及线程分配的问题，则共同取决于前述的并行设置，运行平台等因素。除此之外，由于OpenMP的并行过程涉及共享内存（如全局变量）并发读写问题，因此需要谨慎考虑并行区域中共享数据的读写冲突，在必要的地方添加OpenMP提供的同步互斥或通信的接口，以确保不会更改算法的原有逻辑。最后，需要采用支持OpenMP指令的编译器将对应程序编译为可执行程序，以测试并行算法的提升效果。

OpenMP中的预编译指令

|  |
| --- |
| Algorithm. 基于OpenMP的并行算法 |
| #pragma omp <directive> [clause[[,] clause] ...] |

尽管基于OpenMP框架实现并行算法的流程并不复杂，但依然存在着一些需要注意的问题。首先是对于并行开发常见的因竞争条件以及同步错误导致的程序问题将变得难以调试，因为具体的并行代码均是有OpenMP框架生成，无法直接对应原有的代码流程。其次对于待并行区域代码并没有良好的异常处理机制，从而导致对突发异常的原因排查较为困难。除此之外，由于OpenMP提供了较为高层的并行抽象，因此将难以实现将线程绑定到指定处理器或是其他线程级上更为细粒度的操作。

本文后续基于OpenMP框架实现的并行算法将不再阐述具体的代码实现细节，均默认是在建立在OpenMP框架正确使用的基础上，重点给出算法并行化思路以及效率提升的效果。

* + 1. 测站间并行构建观测方程

对于实时滤波轨道处理中，在进行SRIF时间更新前，需要对所有测站的观测数据构造对应的观测方程。对于单个测站卫星对，其观测方程构建的基本流程如下：首先分别获取测站位置并迭代计算卫星位置初值；接着对传播过程中涉及的误差采用模型进行改正，从而获得观测残差；最后计算该组观测方程中所涉及参数的系数以及对应的权重，即完成方程的构建。可以看到对于不同测站卫星对间的观测方程构建而言，其之间是不存在前后依赖的关系，可以直接应用OpenMP框架将这部分代码流程直接改为并行结构而无需对整体代码结构做过多调整。在传统观测方程构建的串行处理程序中，需要先对测站列表进行循环遍历，接着遍历该测站上的所有卫星观测数据，方能完成全部观测方程的构建，在程序实现上本质为一个嵌套的二重循环。对于嵌套循环，如果直接对每个循环区域直接采用OpenMP预编译指令并行化，将难以得到理想的优化效果。因为如此编译得到的程序将会在每个循环之前都创建指定的线程数，因此总共所创建的线程数将是循环嵌套数的指数倍。过多的线程一方面将导致创建线程的时间开销大大增加，另一方面则是容易使得线程数远大于常用处理器中的物理核心数，进而使得系统在切换线程间的时间开销增加。综上原因，对嵌套循环的并行改造需要进一步分析内外的循环次数即耗时对比。对于精密轨道确定中的观测方程构建的循环而言，存在如下关系：外层循环的测站数>>内层循环的卫星数≈现代计算处理器的常见核心数（16-32）。理想情况下，无论是对内外层循环进行并行化，都将获得等价的并行模型。然而对于内层循环并行化的情况，其增加了线程创建的开销（且随着外层循环次数的增加而增多），因此对外层循环应用OpenMP并行化将能获得更好的改善效果，图xx给出了并行前后的流程示意图。

(并行化前后对比流程图)

* + - * 1. 基于OpenMP并行优化前后的观测方程构建算法流程图

尽管对于不同测站间的观测方程的构建是不存在相互依赖的，但是流程中存在因对公共数据读写产生的竞态条件。从上图可以看到，在不同测站-卫星对间的并行构建过程中存在许多可能的竞态条件：如同时对星历数据、误差模型等全局数据进行读取；在构建完成后，向存储结果的全局数据同时写入等。严格来说，仅包含读取操作并不会引发竞态条件，因此这里仅需要对写入操作的部分采用OpenMP提供的同步原语（即线程锁）进行限制即可。

* + 1. 卫星间并行轨道积分

在实时滤波轨道处理中，卫星轨道积分通常出现在需要更新作为初值的参考轨道的过程中。对于单颗卫星的轨道积分，其基本流程如下所示：获取当前时刻卫星轨道的状态参数，采用龙格库塔积分获取短步长下的一系列卫星轨道状态作为后续阿达姆斯积分的启动点，最后积分求得所需弧段上其他时刻的卫星轨道状态。可以看到，类似不同测站-卫星对间的观测方程构建，不同卫星间的轨道积分也不存在相互依赖的关系，十分适合采用并行优化的方法。串行的轨道积分处理程序中，通常需要依次遍历所有卫星完成积分过程，因此只需要通过OpenMP的预编译指令将卫星循环的代码域并行化即可。优化前后的轨道积分流程对比如下图xxx所示：

（并行化前后对比流程图）

* + - * 1. 基于OpenMP并行优化前后的卫星轨道积分算法流程图

类似地，需要对轨道积分中可能存在的竞态条件进行同步限制，相对观测方程构建的过程，其所需要需要的同步操作更少，整体代码改动较少。

* + 1. 实验结果和分析

为了进一步验证上述基于OpenMP实现的并行算法的有效性，这里通过统计优化前后的多系统仿实时滤波轨道确定的计算耗时进行了对比实验分析。实验过程中所采用的测站数、卫星数和线程数由于实验方案的需要会进行选择和调整，除此之外，其余所使用的定轨处理策略与5.2节中一致。同时，本次实验的运行平台为武汉大学测绘学院的计算服务器系统，型号为EYPZ xxx，处理器为AMD xxx，具有32个处理核心以及xxxHz的运行频率。后续程序性能分析的相关实验均是采用相同的运行平台，将不再赘述。

在给出实验分析之前这里我们首先需要给出对于衡量程序并行化性能的一个指标，即加速比（speedup），其具体的定义如下式所示：

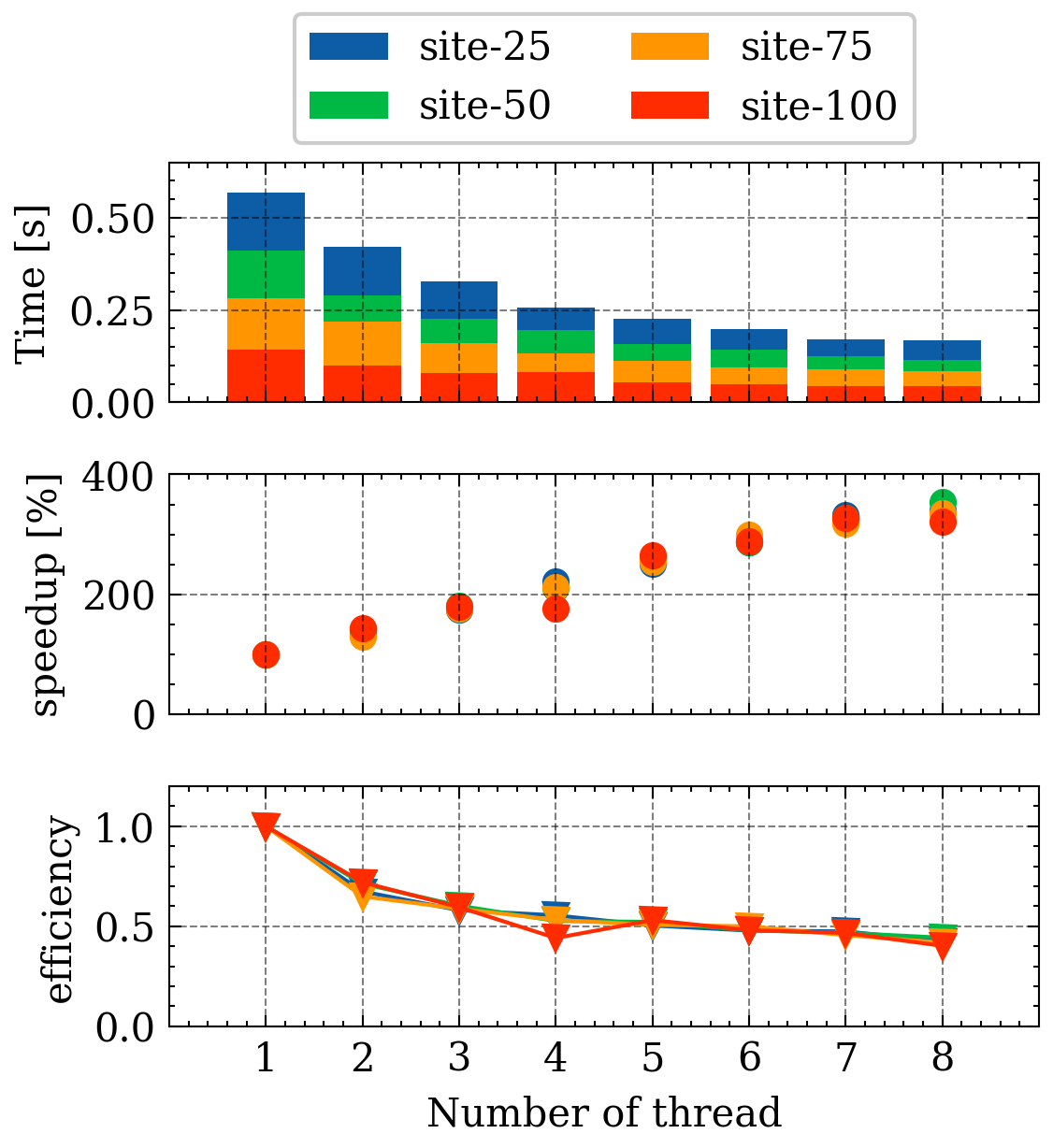


其中，为所用的所处理器核心数，为原始串行算法的的时间开销，为运行在处理器核心数为的平台上的时间开销。可以看到加速比其值必然大于等于1，而在接近1的情况下，说明并行化对程序性能几乎没有改善。另一方面，当有的时候，我们称此时程序并行的性能具有理想加速比（linear speedup），即代表程序执行速度随处理器个数呈现线性增加的关系。但在实际过程中一般很难达到如此高的加速比，因为通常存在线程创建以及线程锁同步的时间开销。也因此由加速比派生出了一个衡量性并行化性能效率的指标，其定义如下式所示：



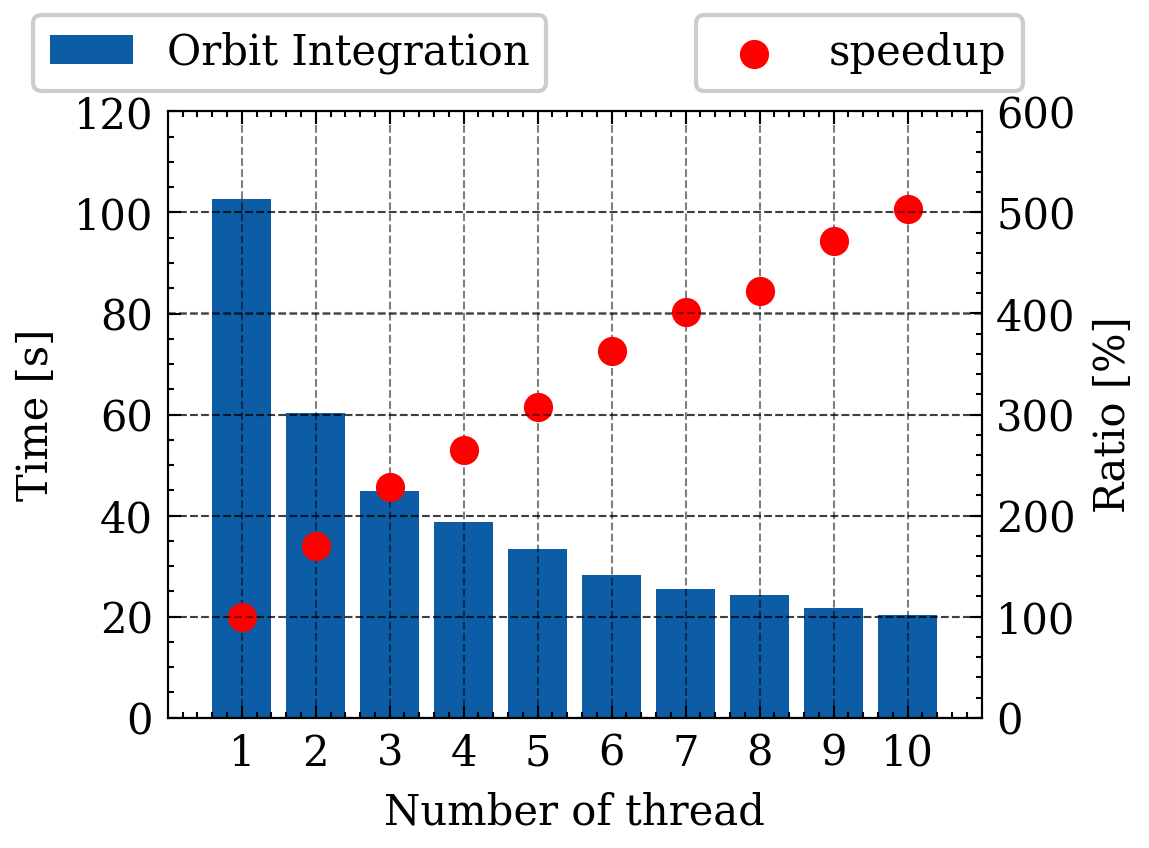
可以看到效率指标其值范围位于0~1之间，越接近于1，说明并行程序在同步以及线程通信等方面的额外开销越小。本质上效率指标和加速比是可以等价转换的。

这里首先为了衡量基于测站间并行的观测方程构建算法的运行性能，首先开展了对GPS/Galileo/BDS共79颗卫星的三天的仿实时滤波轨道确定的实验。为分析不同测站数与不同线程数下，并行算法对观测方程构建过程的效率提升，设计了如下的实验方案：首先选取了数量为25、50、75和100四组测站列表，并分别进行了线程数为1-8依次递增的仿实时定轨实验，其中单独统计了观测方程构建部分的计算耗时。这里需注意的是，当线程数设置为1时，此时OpenMP并行化的代码域将与原有的串行化代码等价，即可直接作为串行程序运行性能的参考。图 4.9 给出了相应的实验统计结果，其中位于上方的子图表示了不同测站数量和不同线程数下观测方程构建部分的单历元耗时，中间的子图为对应的加速比，下方的子图为对应的效率指标，横轴均为运算过程中所采用的线程数。首先可以看到，对于不同测站数下，相较于串行程序模型（线程数1），并行算法均能有效提升了观测方程构建的处理效率，且均随着线程数增加，计算耗时逐渐越少。从加速比的统计结果中也可以看出，对于不同测站数其加速比结果基本一致，即测站数量并不会对并行算法的性能提升效果产生影响，这是符合实际情况的。根据前述加速比（效率指标）的定义可知，其大小反映了程序实际并行部分在原始串行部分的所占比例，而这并不受所处理数据影响，因此是一个程序本身并行性能的一个体现。除此之外，可以看到加速比随着线程数的增加，其增长趋势逐渐减缓。其主要原因是随着线程数增加，线程创建的时间开销和同步区域导致的线程间通信时间都逐渐增加，从而导致程序并行部分的占比不断减小。除此之外，效率指标也随着线程数的增加呈现不断减少的趋势，也印证了前述在线程通信及同步上的开销逐渐增大的现象。



* + - * 1. 不同线程数下观测方程构建的耗时占比（这个图Y轴名称Ratio要改一下，X轴也和下面一致算到10）

类似地，为了衡量基于卫星间并行的轨道积分算法的运行性能，这里同样开展了与上述相同的三系统79颗卫星的仿实时滤波轨道确定实验。这里为了确定不同线程数下，轨道积分并行算法提升的效果。这里分别测试了线程数为1-10依次递增的情况下，轨道积分部分的计算耗时，图 4.10 给出了统计结果。图中柱状图给出了不同线程数下轨道积分的计算耗时，散点图则给出了对应的加速比。显然，并行算法大幅度改善了轨道积分的计算耗时，且随着线程数的增加计算耗时越少。同时从加速比的变换中也可以看出，相对观测方程构建的并行算法，轨道积分的并行算法的加速比要更高，且增长趋势基本维持着线性增长的趋势。相对于观测方程构建中涉及了更多共有内存（如星历数据、潮汐数据、测站坐标数据等）的同步区域，卫星间的轨道积分则相对更为独立，因此加速比也相对更高。但加速比同样存在着随线程数增加增长趋势放缓的现象，其原因和之前对观测方程构建结果分析是相同的。



* + - * 1. 不同线程数下卫星轨道积分计算耗时统计图

综上所述，上述实验均验证了基于OpenMP构建的并行算法对观测方程构建以及轨道积分的过程有较好地改善效果，其中轨道积分的性能提升效果要好于观测方程构建的过程，在六线程的情况下，轨道积分平均性能提升程度为360%，观测房间构建性能提升程度为300%。同时随着线程数增多，尽管两者的性能均会随之进一步提升，但由于同步等操作等事件开销增加，对多处理器的利用效率实际上都降低了。因此在具体应用过程中，需要根据实际运行环境选择合适的并行线程数。

## SRIF参数估计的效率优化

本节首先对实时滤波轨道处理中SRIF参数估计中所使用的QR分解的基本原理和实现做了简要介绍，并重点梳理和分析了基于矩阵块操作的QR分解方法。接着在GREAT软件平台上基于高性能C++矩阵库Eigen重新实现SRIF参数估计模块，并通过与原有的基于C-Lapack实现进行对比实验，分析了SRIF参数估计模块的性能提升效果，验证了算法的有效性。

* + 1. QR分解的实现与优化

根据3.2节中推导SRIF的量测更新和时间更新的原理，可以看出SRIF的实现的核心关键部分依赖于对构造的单位权标准化后的系数矩阵（原始信息阵/观测方程，原始信息阵/状态转移方程）进行QR分解。QR分解的功能是将原有矩阵分解成一个单位正交阵和上三角阵，如下式所示：



其中，为待分解的矩阵，在精密轨道确定问题中代表了观测数以及待估参数，为了保证参数的可求解性，矩阵一般为满秩（full-rank），即有。分别为分解后的正交矩阵和上三角矩阵。这里需要指出，QR分解的结果并不唯一，取决于具体的实现方式，目前常用的实现方式有以下三种：首先是基于Gram-Schmidt分解的方式，其核心思路是根据施密特正交化（Schmidt orthogonalization）方法实现的，其不适用于稀疏矩阵的分解，且对病态矩阵的分解结果有较大的计算误差，其算法时间复杂度为（这里针对的分解矩阵为N阶方阵的情况，下同）；接着使基于Givens旋转的方式，其核心思路是基于坐标轴间旋转的方式进行实现的，由于它的每次迭代操作仅涉及两个向量，因此可以避免对零元素的无效操作，从而使得该方法十分适用于稀疏矩阵得分解中，其算法时间复杂度为；以及基于Householder变换得方式，其核心思路为对向量进行平面反射变换，使得除某一维度外其余分量均为0，其较适合稠密矩阵得计算，算法时间复杂度为。对比上述三种方式，可以看到基于Householder变换的QR分解的时间复杂度最低，同时考虑到SRIF中待分解的系数矩阵通常为稠密矩阵（因为通常需要包含上一历元的上三角信息阵），因此SRIF的实现通常采用了这种分解方式。这里对其实现作简要介绍，不失一般性，令为待分解矩阵A中第一列向量，则可以构造如下的变换矩阵Q：



这里我们称变换矩阵Q为Householder变换矩阵。基于上述的变换，我们可以依次对A矩阵中余下的向量构造类似的变换矩阵，最终得到上三角矩阵，具体如下式所式：



这里可以看到，在计算分解矩阵R的过程中主要涉及了大量的向量-向量，矩阵-向量的操作，这些操作在4.2.1节中介绍的多层缓存的存储器结构中并不具备良好内存时空局部性，因此可以将对分解矩阵中的向量Householder变换操作进行聚合，以形成矩阵-矩阵这类对缓存友好的操作类型，这也是基于块分解的Householder变换（blocked-Householder）的基本思路。具体实现方法则是需要对原有的Householder矩阵改造为WY表达式，对于分块维度为k的矩阵，Householder矩阵可以从如下的递推式得到：

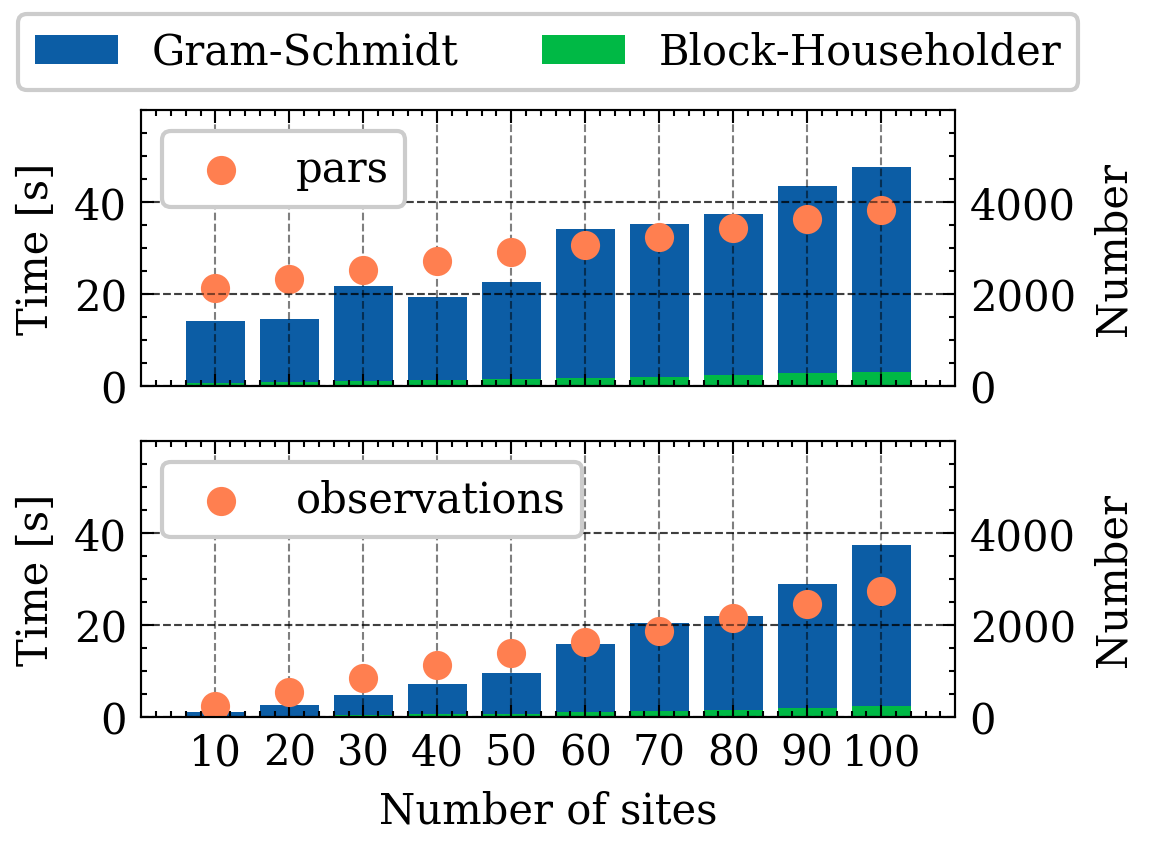


式中，为对k维向量的Householder变换矩阵。通过该变换矩阵，即可将分解矩阵中k列向量进行反射变换，迭代该过程即可完成QR分解。

这里值得一提的是，Lapack (Linear Algebra PACKage)已经实现了许多线性代数的数值计算函数，并提供了使用接口，其中就包括了基于Householder实现的QR分解过程，其初始采用了Fortran编程语言进行开发，后续也提供了对C语言的版本（C-Lapack）。本文的开发平台GREAT软件即采用C-Lapack作为数值计算基础库，SRIF参数估计模块的初始版本也基于此开发完成。但对于C-Lapack中的Householder变换实现依旧采用了式的实现方式，无法很好利用现代处理器和存储器的相关性能，因此这里采用了Eigen库（基于C++开发的现代线性代数计算库）重新实现了SRIF的参数估计模块，其中Eigen库中所提供了HouseholderQR分解接口即是基于块操作实现的（参考文献）。

* + 1. 实验结果和分析

为进一步对比使用高性能矩阵库开发的SRIF参数估计模块的性能提升效果，这里分别使用优化前后的SRIF参数估计算法进行了多系统仿实时滤波轨道确定的对比实验。由于SRIF中待分解矩阵维度通常由观测数与待估参数个数所共同决定，且对于时间更新和量测更新中的待分解矩阵所涉及的维度并不统一，这里具体设计的实验方案如下：依次采用了测站数10-100（间隔为10）进行三系统（GPS/Galileo/BDS）的多天仿实时定轨，分别单独统计其中SRIF时间更新和量测更新部分的计算耗时以及待分解矩阵的维度，最后得到单历元的平均结果。统计结果如下图 4.11 所示。图中上方的子图表示了基于C-Lapack实现的SRIF参数估计模块的计算耗时，下方的子图表示了基于Eigen实现的计算耗时。从上下子图纵轴的数值范围首先可以明显看出，相较于原始方法，新方法无论是在时间更新还是量测更新上的计算耗时都下降了一个数量级，在处理性能上有着显著改善。在测站数为100的情况下，SRIF时间更新的单历元计算耗时从原有的47.7s改善为3.0s，提升了94.1%；SRIF的量测更新的单历元计算耗时则从原有的37.5s改善至了2.4s，提升了93.6%。如此大的改善效果正是通过采用矩阵块操作所带来的优势。另一方面，随着测站数的增加，观测方程数以及参数个数基本保持对应的线性增长，此时两者方法的计算耗时都基本一致地展现了指数级别的增长趋势，主要原因在于新方法提升程序性能的原理是通过增加了数据缓存的时空局部性，从而减少了处理器从存储器中获取数据的时间，本质上并没有改变算法时间复杂度，因此其仍将保持着随矩阵维度增加而显现高次方的增长趋势。总而言之，新方法对SRIF参数估计模块的大幅度改善满足了目前实时轨道处理的计算效率的要求，也为将来以更高频的采样率进行数据处理成为可能。同时在未来可用测站数逐渐增多的情况下，量测更新的计算耗时由于观测数的增加仍将快速增长，成为SRIF参数估计模块中主要的计算耗时瓶颈。



* + - * 1. 不同测站数下，不同QR分解实现的计算耗时对比图（这里还是说改成对比块和非块之间矩阵的之间的对比方法差距其实也有这么大）

## 本章小结

本章主要分析了

# Multi-GNSS实时精密轨道实验与结果分析

## 引言

为了进一步验证

## 实时精密定轨实验方案

测站分布图

导航卫星的基本定轨策略

## 实时精密轨道结果分析

* + 1. 轨道比较结果分析
    2. 轨道收敛时间分析
    3. 轨道效率提升分析

## 基于实时精密轨道的PPP定位分析

* + 1. 实验方案
    2. 实验结果和分析

## 本章小结

# 结论

## 工作内容总结

## 未来展望