中图分类号：V249.3　　单位代码：10335

学　　号：21115030

浙江大学

硕士学位论文

中文论文题目：GNSS/MEMS-DR 超紧耦合组合导航算法研究

英文论文题目：Algorithms for Ultra-tight GNSS / MEMS-DR Integration

申请人姓名：牟 文 杰

指导教师：宋开臣（教授）

专业名称：电子信息技术及仪器

所在学院：生物医学工程与仪器科学学院

论文提交日期　2014 年 5 月 15 日

GNSS/MEMS-DR 超紧耦合组合导航算法研究

论文作者签名：_____

指导教师签名：_____

论文评阅人 1：_____

评阅人 2：_____

评阅人 3：_____

评阅人 4：_____

评阅人 5：_____

答辩委员会主席：_____

委员 1：_____

委员 2：_____

委员 3：_____

委员 4：_____

委员 5：_____

答辩日期：2014 年 6 月 9 日

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 浙江大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解 浙江大学 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 浙江大学 可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

致　　谢

值此论文即将付梓之际，谨向所有关心我的老师、同学和亲友们致以最诚挚的感谢！

首先，感谢我的导师宋开臣教授。在攻读硕士学位期间的学习、科研和生活中，宋老师以他渊博的知识、丰富的工程实践经验和富有创造力的思维给予我深刻的启迪，使我思考问题的方式和实际动手能力得到了很大提高。宋老师严谨踏实的治学作风、认真求是的工作态度和以学生为本的高尚品德，给我留下极为深刻的印象，是我以后工作和生活中学习的榜样，在此论文成稿之际，向宋老师致以深刻的感谢和崇高的敬意，衷心的希望宋老师身体健康，事业顺利。

感谢叶凌云老师，叶老师乐于助人、平易近人的品格深深打动了我，他的关怀与帮助使我迅速成长，他分析问题的方法和待人处事的态度将会对我以后的职业生涯发挥重要作用。

感谢课题组里为我的学习和工作提供过帮助的康力、邓富城、李彩霞、黄添添、张新伟等博士生，感谢陈咨余、魏彬、罗云、孔令稳、陈曦、邓立新等硕士生，你们蓬勃的朝气深深鼓舞着我，你们上进心深深激励着我。感谢王大勇、刘建青、练斌、谢洁意、朱文戈、李嘉鸿等已毕业的师兄师姐，感谢你们给我的大力支持与帮助。感谢我在浙大两年多来认识的同学和朋友，在这里我找到了自己的兴趣和以后想过的生活，浙大的确是所好学校。

最后，衷心感谢我的父母和所有关心我的亲朋好友，多年来你们在我的求学道路上给予我最无私的关怀与帮助。在我即将告别学生时代之刻，向你们致以最真心的祝福和感谢，相信我会以自己的实际行动来承担起家庭的重任的，不负你们的厚望。

牟文杰

二零一四年春于浙大求是园

摘　要

针对卫星导航接收机在弱信号或高动态情况下容易失锁的问题，本文深入研究了卫星导航与微机电系统（Micro-Electro-Mechanical System, MEMS）航位推算系统的超紧耦合组合导航算法，并着重研究了 MEMS 航位推算系统辅助卫星导航的捕获和跟踪的算法，以及基于联邦无迹卡尔曼器的数据融合算法，提高组合导航系统的抗干扰能力和导航精度。同时针对 MEMS 器件噪声大的问题，提出了一种基于多传感器信息融合的 MEMS 航位推算算法，并以此得到的导航信息来辅助卫星导航，实现卫星导航与 MEMS 航位推算系统的超紧耦合算法。试验结果表明本文提出的算法相比传统的组合导航算法具有更高的抗干扰能力和导航精度。

本文主要研究了以下几个部分内容：

1. 针对 MEMS 惯性器件噪声大的特点，研究抑制 MEMS 航位推算系统累积误差的方法，提出了一种基于多传感器信息融合的航位推算算法，以低成本低精度的传感器实现了高精度导航。

2. 深入研究了卫星导航接收机的内部机理，研究 MEMS 惯性器件辅助卫星导航接收机捕获和跟踪的方法，实现了矢量跟踪环算法，提高对弱信号的跟踪能力。

3. 研究卫星导航与 MEMS 航位推算系统的数据融合算法，着重研究了基于联邦无迹卡尔曼滤波器的组合导航滤波算法，提高组合导航的精度和抗干扰能力。

4. 设计不同场景的试验，分析比较松耦合、紧耦合和超紧耦合等组合导航算法的性能，并验证超紧耦合方式相对于其他组合导航方式的优越性。

关键词：超紧耦合；卫星导航；航位推算；MEMS

Abstract

Global Navigation Satellite System (GNSS) receiver is vulnerable to lock under weak signals and high dynamics. To solve the problem, the ultra-tightly integration algorithm for GNSS and Dead Reckoning (DR) system has been researched in this paper. The research is focused on how to aid GNSS receiver with Micro-Electro-Mechanical System (MEMS) DR, and data fusion algorithm based on Federated Uncented Kalman Filter, to improve the robustness of the integrated navigation system. Meanwhile, to overcome the shortcomings that the noises of MEMS inertial measurement units are usually large, a Dead Reckoning system based on multi-sensor data fusion is designed. Then the designed Dead Reckoning system is integrated with GNSS using ultra-tightly integration algorithm. The experiments showed the advantages in accuracy and robustness over traditional integrated navigation system.

The contents of the paper includes the following parts:

1. To overcome the shortcomings that the noises of MEMS inertial measurement units are usually large, the method to restrain the accumulated errors of MEMS-DR is studied, and a Dead Reckoning system based on multi-sensor data fusion is designed, to achieve high accuracy navigation with low-cost and low-accuracy sensors.
2. Research on the principle of GNSS receiver, and the method to aid the GNSS acquisition and tracking with MEMS-DR, and the vector tracking loop, to improve the ability to track weak signals.
3. Study on the data fusion algorithm for GNSS/MEMS-DR integration. The navigation data fusion algorithm based on Federated Uncented Kalman Filter is studied, to improve accuracy and robustness.
4. Design different experiments to compare the performance of loosely coupled, tightly coupled and ultra-tightly coupled integrated algorithms, to

verify the advantages of ultra-tightly coupled algorithm over other algorithms.

Keywords: Ultra-tightly coupled; GNSS; DR; MEMS

目 录

插图目录

表格目录

术语缩写表

AHRS	姿态航向参考系统（Attitude and Heading Reference System）
BDS	北斗卫星导航系统（Beidou System）
C/N_0	载噪比（Carrier to Noise density ratio）
DLL	延时锁定环（Delay Locked Loop）
DR	航位推算（Dead Reckoning）
DVL	多普勒计程仪（Doppler Velocity Log）
EKF	扩展卡尔曼滤波器（Extended Kalman Filter）
FLL	锁频环（Frequency Locked Loop）
FUKF	联邦无迹卡尔曼器（Federated Uncented Kalman Filter）
GEO	地球静止轨道
GLONASS	前苏联的格洛纳斯系统
GNSS	全球导航卫星系统（Global Navigation Satellite System）
GPS	全球定位系统（Global Positioning System）
IGSO	倾斜地球同步轨道
INS	惯性导航系统（Inertial Navigation System）
KF	卡尔曼滤波器（Kalman Filter）
MEMS	微机电系统（Micro-Electro-Mechanical System）
MEO	中圆地球轨道
NCO	数字振荡器（Numerically Controlled Oscillator）
PLL	锁相环（Phase Locked Loop）
SDR	软件定义的无线电（Software Defined Radio）
SLL	标量跟踪环（Scalar Tracking Loop）
UKF	无迹卡尔曼滤波（Unscented Kalman Filter）
VLL	矢量跟踪环（Vector Tracking Loop）

第1章 绪论

1.1 论文研究背景

将运载体从起点引导到目的地的技术或方法称为导航。导航系统测量并解算出运载体的瞬时运动状态和位置，提供给驾驶员或自动驾驶仪实现对运载体的正确操纵或控制[1]。

随着科学技术的发展，可资利用的导航信息源越来越多，导航系统的种类也越来越多。以航海导航为例，目前可供装备的机载导航系统有惯性导航系统（Inertial Navigation System，INS）、全球卫星导航系统(Global Navigation Satellite System，GNSS)、多普勒计程仪（Doppler Velocity Log，DVL）等，这些导航系统各有特色，优缺点共存。比如 INS 的优点是：不需要任何外来信息也不向外辐射任何信息，可在任何介质和任何环境下实现导航，且能输出飞机的位置、速度和姿态等多种导航参数，系统频带宽，能跟踪运载体的任何机动运动，导航输出数据平稳，短期稳定性好[2-6]。但 INS 具有固有的缺点：导航精度随时间而发散，即长期稳定性差。GNSS 精度高，且不随时间发散，这种高精度和长期稳定性是 INS 望尘莫及的。但 GNSS 也有其致命弱点：频带窄，当运载体作高机动运动时，接收机码环和载波环极易失锁而丢失信号，从而完全丧失导航能力；完全依赖于 GNSS 卫星发射的导航信息，受制于他人，且易受人为干扰和电子欺骗[1]。DVL 的优点是：能进行完全自主的导航，不需要外部设备的支持；反应速度快，使用方便；由于发射波束很窄，且以很陡的角度发射到海底，所以隐蔽性和抗干扰性好；测得的平均速度精度很高。但是，多普勒速度计程仪也存在着不足，它需要外部的航向和垂直基准信息，定位误差随时间积累[7]。其余导航系统有各自的优缺点。

各种导航系统单独使用时很难满足实际导航要求，于是人们提出了组合导航的概念：即将多种不同的导航手段组合在一起，各种手段性能互补，取长补短，以获得比单独使用任一导航系统时更高的导航性能。目前使用两种或两种以上系统的组合导航是发展趋势和研究的热点。

从以上对 GNSS 和 INS 各自性能分析可以看出，两者可以实现很好的优势互补，将 GNSS 和 INS 组合成一个导航系统，可以提高系统的整体导航性能，得到比单一系统更加优越的性能：

1) 将具有全球、全天候不间断连续导航能力；

2) 能够输出高更新率的导航数据；

3) 能实现低成本的 GNSS / INS 设备高精度导航参数输出；

4) 能增强系统的动态性、抗干扰性、稳定性、可用性和完整性。

从结构上或信息交换及组合程度看，INS 与 GNSS 的组合可分为：松耦合(Loosely Coupled)、紧耦合(Tightly Coupled)和超紧耦合(Ultra-tightly Coupled)等三种耦合方式[8]。

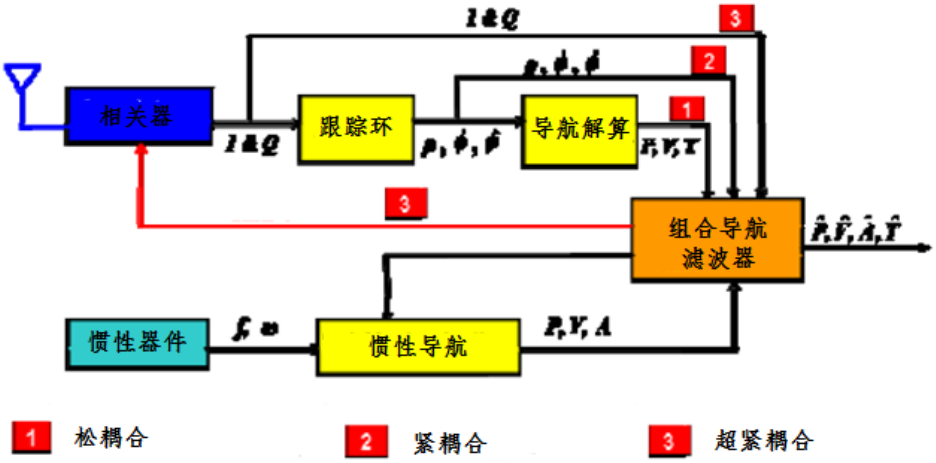

图 1-1 松耦合、紧耦合和超紧耦合结构示意图[9]

松耦合和紧耦合是一种软件形式的组合，分别将 GNSS 输出的位置、速度信息和伪距、伪距率信息同 INS 解算结果进行数据融合，利用卡尔曼滤波器产生各导航参数的误差修正量，通过输出或反馈校正 INS，达到改善 INS 精度的目的。深组合则是将 GNSS / INS 进行软硬件一体化设计，将 GNSS 接收机基带、导航解算与 INS 的导航解算进行整体化设计，利用 GNSS 接收机的输出信息去辅助 INS 的同时，将短时高精度的 INS 速度、加速度信息提供给接收机的基带环路，为 GNSS 接收机的信号捕获、跟踪提供辅助，提高接收机的灵敏度、消除环路的动态特性影响、减小多路径效应。再利用 Kalman 滤波器对 INS 和 GNSS 接收机的原始观测误差进行最优估计，使得系统输出最优的导航参数。

正是由于这些优点，超紧耦合使低成本的惯性器件（如 MEMS 惯性传感器）与 GNSS 的高性能组合导航成为可能。

以下是松耦合、紧耦合和超紧耦合的比较。

	松耦合	紧耦合	超紧耦合	说明
跟踪灵敏度	差	好	非常好	可提升20~30dB，一般室内卫星信号与室外相比衰减约20~30dB
捕获灵敏度	差	好	非常好	利用加速度信息，延长捕获积分时间，可提高捕获灵敏度
重新捕获能力	差	一般	非常好	利用多普勒频移，减少二维搜索的空间，从而减小捕获时间
数据更新率	低	低	高	捕获时间减少，从而可提高数据更新率，可高达100Hz
定位精度	低	一般	高	利用多普勒频移，可减小跟踪环路带宽，提高噪声抑制能力
动态响应	差	差	非常好	利用多普勒频移，可防止高动态情况下跟踪环路失锁
多径干扰抑制	差	一般	非常好	窄相关技术可消除多路径误差，从而获得更高精度的观测量
实现复杂度	低	中等	高	超紧耦合需要对接收机内部结构重新编排，融合算法也更加复杂

图 1-2 松耦合、紧耦合和超紧耦图合的比较图[9]

可见，在弱信号或高动态的复杂环境下的高精度导航，超紧耦合技术具有很大的优越性，超紧耦合是以后组合导航的发展趋势。

1.1.1 发展中的北斗卫星导航系统

全球卫星导航系统(Global Navigation Satellite System，GNSS)是一种能够提供全球、全天候、全天时的高精度地理位置信息以及导航、授时信息的卫星系统。作为国家重要的信息基础和战略设施，GNSS 是国家综合国力的一个重要标志，是保障国家安全的重要基石，对于满足国民经济、国防建设的需要，具有重大的经济和社会意义。[10]

目前，只有美国的全球定位系统(Global Position System, GPS) 及前苏联的格洛纳斯系统是完全覆盖全球的定位系统。中国的北斗卫星导航系统 (Bei Dou System, BDS) 则于 2012 年 12 月开始服务于亚太区 (共由 16 颗卫星组成)，预计于约 2020 年覆盖全球。欧洲联盟的伽利略定位系统()则为在初期部署阶段的全球导航卫星系统，预定最早到 2020 年才能够充分的运作。一些国家，包括法国、日本和印度，都在发展区域导航系统。[11]

北斗卫星导航系统简称北斗系统，其空间星座由 5 颗地球静止轨道（Geostationary Earth Orbit, GEO）卫星、27 颗中圆地球轨道（Medium Earth Orbit, MEO）卫星和 3 颗倾斜地球同步轨道（Inclined Geosynchronous Satellite Orbit, IGSO）卫星组成。GEO 卫星轨道高度 35786 千米，分别定点于东经 58.75 度、80 度、110.5 度、140 度和 160 度；MEO 卫星轨道高度 21528 千米，轨道倾角 55 度；IGSO 卫星轨道高度 35786 千米，轨道倾角 55 度。截止到 2012 年底，在轨工作卫星有 5 颗 GEO 卫星、4 颗 MEO 卫星和 5 颗 IGSO 卫星。[12]

和其他卫星定位系统一样，北斗二代同时提供普通精度的开放服务和高精度的授权服务。每颗北斗二代卫星都发射 4 个频率的载波信号用于导航，包括 B1（1561.098MHz）、B1-2（1589.742MHz）、B2（1207.14MHz）和 B3（1268.52MHz）。每个载波信号均有正交调制的普通测距码（I 支路）和精密测距码（Q 支路）。和 GPS 相同，北斗二代采用码分多址技术，卫星之前通过不同地址码进行区分。该系统将能够提供 10 米的位置精度、0.2m/s 的速度精度和 20ns 的授时精度。同时，还能够提供和北斗一代相同的短消息通讯服务。

以 GPS 为代表的卫星导航应用产业已成为一个全球性的高新技术产业，成为继通信和互联网之后的第三个 IT 新增长点。然而，当前中国卫星导航应用产业绝大多数基于 GPS。发展拥有自主知识产权的安全可靠的“北斗二号”全球卫星导航定位系统，不仅是发展中国国民经济新增长点的有效手段，也是加强国家安全的必要措施。[10]

1.2 国内外研究现状

1.2.1 超紧耦合算法研究现状

国外对于 GPS 和 INS 组合的研究起步较早，特别是作为 GPS 的拥有国的美国在对 GPS 和 INS 的组合导航研究方面一直处于世界前列，其它发达国家也都对组合导航系统进行了全面细致的研究。

自 20 世纪 80 年代起的过去二十多年中，INS 与 GPS 的耦合方式绝大多数是松耦合或紧耦合两种方式。

早在 1980 年 Copps EM 等学者的关于 GPS 信号最优化处理的文献中，就最先认识到 INS/GPS 超紧耦合结构的优势[13]。直到 20 世纪末，超紧耦合的组合导航方式才受到极大的关注，其中 Draper 实验室 Donald Gustafson 等人在 2000 年明确了超紧耦合方式[14]，提出了扩展范围码跟踪环的 INS/GPS 超紧耦合方法，以提高接收机的抗干扰能力，并通过仿真验证了这一方法。2003 年，美国斯坦福大学的 Gautier J D 则进一步完善了这种概念[15]，对 INS/GPS 的超紧方法进行了研究，并分析了超紧耦合下卡尔曼滤波器的基本结构及状态观测方程。2003 年，加拿大的 Abbott 和 Lillo 则为 INS/GPS 的联邦卡尔曼滤波器实现方法申请了专利[16]，并受美国关键军事技术出口限制。

国内对超紧耦合的研究起步较晚，基本还处于仿真阶段。2009 年，哈尔滨工业大学的 Gannan, Yuan 等人提出用基于无味卡尔曼滤波器（UKF）的超紧耦合方案[18]，仿真表明性能得到提升。2010 年，南京理工大学的胡锐自行设计了高精度的 GPS 接收机，增加了 INS 对 GPS 的辅助，实现了 INS/GPS 的超紧耦合[19]，但实现过程是基于紧耦合卡尔曼滤波器的超紧耦合，并没有将卡尔曼滤波器引入到跟踪环路中，与国外的超紧耦合实现方案还存在一定差距。2010 年，国立台湾海洋大学的 Dah-Jing 等人利用模糊自适应无味卡尔曼滤波器对超紧耦合进行分析，取得较好的滤波精度[17]。

1.2.2 组合导航系统信息融合方法研究现状

信息融合方法是组合导航系统的核心问题，直接影响了组合导航系统的性能，对组合导航信息融合方法的研究已经成为一个重要的课题[20]。

在现有的 INS/GPS 组合导航系统中主要应用卡尔曼滤波的方法进行多传感器的信息融合。卡尔曼滤波方法是由 Kalman[21]于 1960 年提出的一种线性最小方差估计方法，在时域中滤波，采用状态空间方法描述系统，算法采用递推形式，数据存储量小，不仅可以处理平稳随机过程，也可以处理多维和时变过程。正是由于卡尔曼滤波方法的以上优点，在多传感器信息融合中得到了广泛的应用。[20]

标准卡尔曼滤波方法仅能处理线性系统，扩展卡尔曼滤波器(Extended

Kalman Filter，EKF)方法将其扩展应用于非线性系统，EKF 方法通过线性化将非线性系统转变为线性系统后进行卡尔曼滤波。其后，一些广义卡尔曼滤波方法被提出并应用于非线性系统，广义卡尔曼滤波方法通过增加非线性系统线性化后泰勒级数展开项的数量调节滤波精度，由于减小了线性化的误差，因此滤波精度得到提高，但因此增加了计算负担[22]。

集中式卡尔曼滤波方法随着系统状态维数的增加计算量急剧增大，影响了系统对实时性的要求。其次，集中式卡尔曼滤波方法的可靠性较差，一个传感器的故障将导致整体导航性能的迅速下降。1978 年 Hassan 提出了一种分散卡尔曼滤波方法[23]，增强了滤波的可靠性，但当各传感器的状态噪声和初始条件相关联时，各分散滤波器无法独立的给出相应的滤波估计，同时各分散滤波器之间需要交换大量的信息，极大的增加了滤波计算量。近年来分散卡尔曼滤波方法得到了广泛的关注，提出了很多的改进方法，并被成功应用于组合导航系统[24]。

1988 年 Carlson 提出了一种联邦卡尔曼滤波方法[25]，该方法将系统分解为多个子系统，各子系统分别由各自的滤波器进行滤波估计，各分滤波器的估计结果通过一个主滤波器进行信息的融合。滤波过程中的信息是经由主滤波器向各子滤波器分配的，因此增强了对信息的利用率，消除了各子状态间的相关性，使各子滤波器的估计为最优估计，设计灵活，容错性好，只需进行简单、有效的融合，就能得到全局最优的估计，因此得到了广泛的关注和应用。

标准卡尔曼滤波方法要求我们知道系统状态和观测噪声的统计特性，统计特性的不准确或不可知，将导致滤波精度下降甚至是发散的[26]。Sage 于 1969 年提出的自适应卡尔曼滤波[27]算法很好地解决了这个问题，自适应滤波通过观测量估计修正预测值的同时对噪声特性和模型的不确定性进行估计并修正。现有的自适应滤波方法很多，包括渐消记忆卡尔曼滤波方法[28]、新息自适应卡尔曼滤波方法[29]、虚拟噪声补偿滤波方法[30]、动态偏差去耦合估计方法[31]等。

导航系统多为非线性系统，EKF 方法通过将系统误差函数线性化后再进行线性滤波，由于线性化误差大，影响了滤波的精度。非线性滤波方法通过对误差模型矢量的统计特性进行近似来达到将模型线性化的目的，由 Julier[32]等学

者于 1995 年提出的无迹卡尔曼滤波（Unscented Kalman Filter，UKF）就是其中最具代表性的非线性卡尔曼滤波方法。UKF 方法通过设计少量的 sigma 点，经过 UT 变换和非线性函数的传播，估计得到状态矢量的统计特性，然后进行滤波计算[32, 33]。UKF 方法假设系统噪声和观测噪声均为高斯白噪声，而这一条件有时不能满足，这也限制了该方法的应用[34]。UKF 不需要系统进行线性化近似，不需要计算雅可比矩阵，减小了线性化所带来的误差。UKF 方法已经得到了广泛的应用[35, 36]，2003 年 Crassidis[37]等对飞行器初始对准和姿态的 UKF 估计研究表明具有更大的容差性；2004 年 Merwe[35]将 UKF 方法应用于 INS/GPS 松散组合飞行器上，导航精度相对 EKF 方法提高了三分之一。

非线性滤波是以贝叶斯理论为基础的最优滤波过程。1977 年，Ho 等[38]对贝叶斯滤波问题进行了深入的研究，结论表明卡尔曼滤波为贝叶斯滤波的一个特例。基于同样的理论基础，粒子滤波也是解决非线性滤波的另一个有效方法，粒子滤波可以解决非高斯的非线性滤波问题，近几年来，Carvalho[39, 40]和 Babak[41]对组合导航系统滤波方法的研究表明粒子滤波可以很好的解决非线性滤波问题，提高了导航的精度。粒子滤波已经成为近年来非线性滤波研究的一个重要方向，由于退化问题和计算复杂使得粒子滤波方法的实际应用还有很多问题需要解决。

1.3 论文研究内容

本课题深入研究了 GNSS/MEMS-DR 超紧耦合算法，主要包括以下几个部分：

1. 针对 MEMS 惯性器件噪声大的特点，研究抑制 MEMS-DR 系统误差的方法，以低精度低成本的传感器实现高精度导航：
   a) 研究基于多传感器信息融合的 DR 算法，实现多种传感器优势互补，抑制 MEMS-DR 单独导航时；
   b) 研究 MEMS 惯性传感器的标定方法，消除传感器确定性误差。
2. 研究 MEMS-DR 辅助 GNSS 接收机捕获和跟踪的方法：
   a) 深入研究 GNSS 接收机的内部机理，特别是基带信号处理算法；

b) 研究基于卡尔曼滤波器的矢量跟踪环路算法，实现利用 MEMS-DR 的信息辅助 GNSS 接收机对卫星信号的捕获和跟踪，增强对弱信号的跟踪能力，从而提高 GNSS 接收机的抗干扰能力。

3. 研究 GNSS/MEMS-DR 超紧耦合组合滤波算法：
   a) 研究超紧耦合下卡尔曼滤波器的基本结构，利用载波环跟踪误差、伪距率量测误差与惯导误差之间的关系模型，建立卡尔曼滤波器的状态方程和观测方程；
   b) 研究各种卡尔曼滤波器的滤波性能，并对超紧耦合滤波算法进行仿真，寻找一种最佳滤波算法。
4. 设计不同场景的实验，分析比较松耦合、紧耦合和超紧耦合等组合导航算法的性能，并验证超紧耦合方式相对其他组合导航方式在抗干扰能力、导航精度和重捕时间等方面的优越性。

1.4 论文研究的意义

1) 超紧耦合使低成本的惯性器件（如 MEMS 惯性传感器）与 GNSS 的高性能组合导航成为可能，实现低成本、高精度的组合导航系统。

高精度、低成本是导航系统发展的方向。对于 INS 来说，在短时间内可以得到高精度的导航数据输出，由于基于航位推算的工作原理，误差将随时间的平方不断增加[5]。而能获得长时间高精度导航数据的惯性器件价格则非常的昂贵，一个具有 0.01°/h 的导航系统其价格达上百万[4]，因此对于惯性导航来说目前低成本和高精度是一对矛盾体[19]。

对于 GNSS 接收机来说，定位精度虽然没有短时间内的 INS 精度高，但误差稳定，不会随时间积累。目前一个 100 美金的接收机 OEM 的单频定位误差在 15m 左右[19]，接收机的成本相对于惯性来说就非常廉价了。

因此，利用低成本 GNSS 接收机与低精度 INS 集成，通过组合充分使用 INS 的短时间高精度和 GNSS 接收机误差不随时间累积的特点，采用 GNSS 实时修正 INS 的输出累积误差，提高 INS 精度，同时将 INS 测量的载体状念信息引入 GNSS 接收机基带坏路和导航算法中，实现 GNSS／INS 超紧耦合，提高基带信

号捕获、跟踪性能和导航定位精度，同时这也将提高接收机辅助 INS 时的精度，最终使得低成本组合系统获得高精度输出变成可能。[19]

2) 用北斗导航系统取代 GPS，有利于加强国家安全。

以 GPS 为代表的卫星导航应用产业已成为一个全球性的高新技术产业，成为继通信和互联网之后的第三个 IT 新增长点。然而，当前中国卫星导航应用产业绝大多数基于 GPS。发展拥有自主知识产权的北斗卫星导航系统，用北斗代替 GPS，不仅是发展中国国民经济新增长点的有效手段，也是加强国家安全的必要措施[10]。

第2章 基于多传感器信息融合的航位推算系统设计

航位推算（Dead Reckoning，DR）系统采集运动传感器信息进行自主连续的导航，短时间精度高，但误差随时间不断累积。DR 系统通常与 GNSS 进行组合导航，作为 GNSS 失效期间的一种补充导航方式。

已经有很多文献对车载 DR 系统进行研究，提出了很多实现方案。文献[42]中利用单轴陀螺和里程计组成 2D 车载 DR 系统，但当车辆行驶在立交桥和山区道路等高度变化较大的场合时，导航精度将会明显下降。文献[43]中利用 3 轴加速度计和 3 轴磁力计组成惯性导航系统，但位置误差随时间以 3 次方的速度增长，且 MEMS 惯性器件噪声大，60s 的导航误差将达到 150m。文献[44, 45]提出利用里程计、单轴陀螺和双轴加速度计组成 3D 车载 DR 系统，能够有效提高导航精度。但航向角由陀螺积分得到，最终的位置误差随时间以平方关系增长，通常情况下只能保证短时间的定位精度。

为降低车载 DR 系统导航误差随时间增长的速度，本文提出一种基于多传感器信息融合的车载航位推算系统，以卡尔曼滤波器作为数据融合算法，综合利用里程计、MEMS 陀螺、MEMS 加速度计和 MEMS 磁力计等多种传感器的信息，抑制车载 DR 系统导航误差的累积，以低成本、低精度的传感器实现高精度导航。

2.1 MEMS 航位推算系统总体设计

车载 DR 系统的组成如图 2-1 所示，ARM 处理器完成对 MEMS 陀螺、MEMS 加速度计、MEMS 磁力计和里程计的数据采集，然后进行航位推算，最后将导航结果上传至上位机。

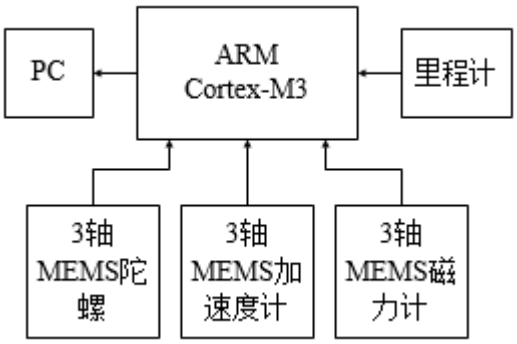

图 2-1 车载 DR 系统的硬件组成

DR 系统算法主要包括 AHRS 算法和位置更新算法，如图 2-2 所示。AHRS 利用卡尔曼滤波器对陀螺、磁力计和加速度计的数据融合，估算得到姿态矩阵 $\boldsymbol{C}_{b,k}^{n}$。利用里程计可以计算得到车辆的前向速度 v_{od}，对速度微分可得前向加速度 a_{od}。将前向速度投影到地理坐标系，得到速度矢量 $\boldsymbol{v}_{k}^{n}$，然后执行位置更新算法，得到位置矩阵 $\boldsymbol{C}_{n,k}^{e}$，最后转换成经度、纬度和高度等位置信息。

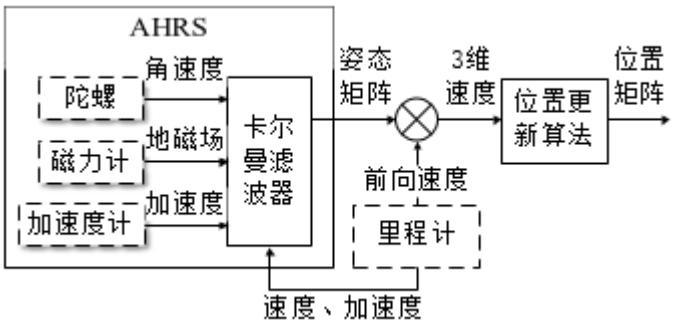

图 2-2 DR 系统算法框图

2.2 航向姿态参考系统算法推导

这一部分详细推导了适用于 MEMS-DR 的航向姿态参考系统（Attidtude and Heading Reference System, AHRS）算法，这是 MEMS-DR 系统的关键。

姿态信息解算过程采用方向余弦矩阵法，即用车体坐标系变换到地理坐标系的方向余弦矩阵 $\boldsymbol{C}_{b,k}^{n}$ 来表示姿态信息。方向余弦矩阵的更新过程可表示为[46-48]：

$$\boldsymbol{C}_{b,k}^{n} \approx \boldsymbol{C}_{b,k-1}^{n}\left(\boldsymbol{I}+(\boldsymbol{\alpha}\times)\right) \tag{2.1}$$

式中：$\boldsymbol{\alpha}$ 是车体坐标系从时刻 t_k 变换到时刻 t_{k+1} 的旋转矢量，即车体坐标系绕 $\boldsymbol{\alpha}$ 旋转，其转角等于 $\boldsymbol{\alpha}$ 的幅值。$(\boldsymbol{\alpha}\times)$ 为 $\boldsymbol{\alpha}$ 的斜反对称矩阵，即：

$$(\boldsymbol{\alpha}\times)=\begin{bmatrix} 0 & -\alpha_z & \alpha_y \\ \alpha_z & 0 & -\alpha_x \\ -\alpha_y & \alpha_x & 0 \end{bmatrix} \tag{2.2}$$

旋转矢量 $\boldsymbol{\alpha}$ 的计算方法有两种：（1）由 MEMS 陀螺积分得到，这种方法短期精度高，但误差随时间累积；（2）由重力和地磁场确定，这种方法误差不随时间累积，但由于 MEMS 加速度计测得的重力受运动加速度的影响，MEMS 磁力计测得的地磁场受环境电磁干扰影响，因此计算得到的水平姿态和航向角噪声较大。

采用卡尔曼滤波器对 MEMS 陀螺、MEMS 加速度计和 MEMS 磁力计的数据进行融合，估计出最佳的旋转矢量 $\boldsymbol{\alpha}$，最后根据式 2.1 计算得到当前时刻的姿态矩阵，从而实现短期精度高，误差又不随时间累积的 AHRS。

2.2.1 MEMS 航位推算中的卡尔曼滤波器算法

卡尔曼滤波是一种高效率的递归滤波器，它能够从一系列的不完全及包含噪声的测量中，估计动态系统的状态。

卡尔曼滤波器由状态方程和观测方程来描述[21, 49, 50]：

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{A}\boldsymbol{x}_{k-1}+\boldsymbol{B}\boldsymbol{u}_{k-1}+\boldsymbol{w}_{k-1} \\ \boldsymbol{z}_k &= \boldsymbol{H}\boldsymbol{x}_k+\boldsymbol{v}_k \end{aligned} \tag{2.3}$$

上两式子中，$\boldsymbol{x}_k$ 是 k 时刻的系统状态，$\boldsymbol{u}_{k-1}$ 是 k-1 时刻对系统的控制量。$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是系统参数。$\boldsymbol{z}_k$ 是 k 时刻的测量值，$\boldsymbol{H}$ 是测量系统的参数。$\boldsymbol{w}_{k-1}$ 和 $\boldsymbol{v}_k$ 分别表示过程和测量的噪声，他们被假设成高斯白噪声，协方差分别是 $\boldsymbol{Q}$，$\boldsymbol{R}$。

卡尔曼滤波器更新过程主要包括以下 5 个公式[49]：

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1}+\boldsymbol{B}\boldsymbol{u}_{k-1} \tag{2.4}$$

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T+\boldsymbol{Q} \tag{2.5}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^T+\boldsymbol{R}\right)^{-1} \tag{2.6}$$

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\left(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^-\right) \tag{2.7}$$

$$\boldsymbol{P}_k = \left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}\right)\boldsymbol{P}_k^- \tag{2.8}$$

根据式(2.4)-(2.8)得到状态量的最佳估计 $\hat{\boldsymbol{x}}_k$ 以及协方差 $\boldsymbol{P}_k$，然后重复以上公式进行下一周期的卡尔曼滤波。

2.2.2 离散状态方程的建立

卡尔曼滤波器状态量选取为：$\boldsymbol{x}_k = \begin{bmatrix}\alpha_x & \alpha_y & \alpha_z\end{bmatrix}^T$，其中 α_x，α_y，α_z 为旋转矢量 $\boldsymbol{\alpha}$ 在车体坐标系中的三个分量。控制量选取为陀螺的输出值：$\boldsymbol{u}_k = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T$，其中 ω_x，ω_y，ω_z 分别为三轴陀螺的输出值。

于是，离散状态方程可表示为以下形式：

$$\begin{aligned}\boldsymbol{x}_k &= \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1} \\ &= \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\boldsymbol{x}_{k-1} + \begin{bmatrix}\Delta t & 0 & 0\\ 0 & \Delta t & 0\\ 0 & 0 & \Delta t\end{bmatrix}\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1}\end{aligned} \tag{2.9}$$

其中 $\boldsymbol{w}_k$ 为系统噪声，满足 $\boldsymbol{w}_k \sim N(0, \boldsymbol{Q}_k)$，即 $\boldsymbol{w}_k$ 是均值为 0，协方差为 $\boldsymbol{Q}_k$ 的高斯型噪声。$\boldsymbol{Q}_k$ 的取值与陀螺噪声大小有关，实际仿真过程中，$\boldsymbol{Q}_k$ 选取为 $diag\{(\mathrm{T}^2, \mathrm{T}^2, \mathrm{T}^2)*10^{-4}\}$，其中 T=0.01s 为仿真更新时间间隔。

2.2.3 量测方程的建立

由加速度计和磁力计测得的重力和地磁场可确定姿态矩阵 $\tilde{\boldsymbol{C}}_{b,k}^n$，具体如下式得到：

$$\tilde{\boldsymbol{C}}_{b,k}^n = \begin{bmatrix}\left(\boldsymbol{m}^0 \times \boldsymbol{f}^0\right)^T & \left(\boldsymbol{f}^0 \times \left(\boldsymbol{m}^0 \times \boldsymbol{f}^0\right)\right)^T & \left(\boldsymbol{f}^0\right)^T\end{bmatrix}^T \tag{2.10}$$

其中：$\boldsymbol{f}^0$ 是重力加速度进行归一化后的数值，加速度计的输出值减去运动加速度，剩下的即为重力加速度，即 $\boldsymbol{f} = \begin{bmatrix}f_x + v_{od}\omega_z & f_y - a_{od} & f_z\end{bmatrix}^T$。$\boldsymbol{m}^0$ 是磁力计输出值进行归一化后的数值。

观测量 $\boldsymbol{z}_k$ 选取为由 $\boldsymbol{C}_{b,k-1}^{n}$ 变换到 $\tilde{\boldsymbol{C}}_{b,k}^{n}$ 的旋转矢量，可由下式计算得到：

$$(\boldsymbol{z}_k \times) = \left(\boldsymbol{C}_{b,k-1}^{n}\right)^T \tilde{\boldsymbol{C}}_{b,k}^{n} - \boldsymbol{I} \tag{2.11}$$

于是，观测方程可表示为：

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} \sigma_p \\ \sigma_r \\ \sigma_\psi \end{bmatrix} \tag{2.12}$$

式中：σ_p、σ_r、σ_ψ 分别为俯仰角、横滚角和航向角的噪声标准差，如果噪声标准差数值变大，状态量的估计值对观测量的信任度就下降。

实际应用中，由于磁力计易受环境的磁干扰，需要算法实时估计磁干扰的大小，自适应地调整测量噪声协方差矩阵，调整滤波增益，改善滤波性能，从而减小磁干扰造成的航向角误差。当磁干扰较大时，磁力计的信任程度就下降，主要靠 MEMS 陀螺来估算航向角。定义规则如下：

$$\begin{aligned} \Delta m &= \left| \|\boldsymbol{m}\| - m_0 \right| \\ \sigma_\psi &= \begin{cases} 0.1, & \Delta m \le 0.1 \\ \Delta m, & 0.1 < \Delta m \le 1 \\ \infty, & \Delta m > 1 \end{cases} \end{aligned} \tag{2.13}$$

对于存在较长时间的磁干扰情况，可采用精度更高的陀螺仪，进一步减小磁干扰对航向角估算的影响。

仿真过程中，σ_ψ 由式(2.13)确定，σ_p 和 σ_r 分别取为 0.1 和 0.1，于是观测量协方差矩阵取值为 $\boldsymbol{R}_k = diag\{\sigma_p^2, \sigma_r^2, \sigma_\psi^2\} = diag\{10^{-2}, 10^{-2}, \sigma_\psi^2\}$。

2.3 MEMS 航位推算系统的位置更新算法

水平姿态和航向确定后，将里程计输出的速度投影到地理坐标系：

$$\boldsymbol{v}_k^n = \boldsymbol{C}_{b,k}^{n} \cdot \begin{bmatrix} 0 & v_{od} & 0 \end{bmatrix}^T \tag{2.14}$$

式中，$\boldsymbol{v}_k^n$ 为地理坐标系下的速度矢量。

积分速度矢量就得到位置增量，利用辛普森法则的位置更新方程如下[4]：

$$\Delta \boldsymbol{r}_k = \left(\frac{\boldsymbol{v}_{k-2}^n + 4\boldsymbol{v}_{k-1}^n + \boldsymbol{v}_k^n}{6} \right) \cdot \Delta t \tag{2.15}$$

位置信息常以经度、纬度和高度来表示，在更新过程中，经度和纬度可以用地理坐标系到地球坐标系的方向余弦矩阵 $\boldsymbol{C}_{n,k}^e$ 来表示。位置方向余弦矩阵及高度的更新过程由下列离散公式表示[4, 48, 51]：

$$\boldsymbol{C}_{n,k}^e \approx \boldsymbol{C}_{n,k-1}^e \left(\boldsymbol{I} + (\boldsymbol{\zeta} \times) \right) \tag{2.16}$$

$$h_k = h_{k-1} + \Delta \boldsymbol{r}_{z,k} \tag{2.17}$$

式中，$\boldsymbol{\zeta}$ 是具有幅值和方向的旋转矢量，即地理坐标系绕 $\boldsymbol{\zeta}$ 旋转，其转角等于 $\boldsymbol{\zeta}$ 的幅值。$\boldsymbol{\zeta}$ 可根据以下公式计算得到[48]：

$$\boldsymbol{\zeta} \approx \begin{bmatrix} 0 & -\dfrac{1}{R_y + h_k} & 0 \\ \dfrac{1}{R_x + h_k} & 0 & 0 \\ \dfrac{\tan\varphi}{R_y + h_k} & 0 & 0 \end{bmatrix} \Delta \boldsymbol{r}_k \tag{2.18}$$

式中：R_x 和 R_y 分别为东向和北向的地球曲率半径，详细计算方法见文献[44, 48, 51-53]。

将式（2.18）代入式（2.16）和（2.17）最终得到位置信息。

2.4 MEMS 传感器标定方法设计

MEMS 惯性器件在组合导航系统的应用当中也日益受到重视，MEMS 惯性器件体积小，功耗低，便于低成本系统的实现。由于 MEMS 传感器输出噪声大，必须经过标定才能够发挥 MEMS 传感器的性能。

2.4.1 磁偏角在线补偿方法

在不同的地区，地磁北和地理北之间的夹角也不同，如果不对磁偏角进行补偿，会造成 DR 系统的导航轨迹与真实轨迹成一定夹角，在这种情况下可利用 DR 系统的导航轨迹和 GNSS 的导航轨迹在线估算出磁偏角，进而对航向角

做磁偏角补偿。航向角偏角由下式估算得到：

$$\Delta\psi = \frac{1}{N}\sum_{k=1}^{N}\left(\psi_{GPS,k} - \psi_{AHRS,k}\right) \tag{2.19}$$

式中：$\Delta\psi$ 为航向角偏角，$\psi_{GPS,k}$ 是由 GNSS 速度估算得到的航向角，$\psi_{AHRS,k}$ 是由 AHRS 估计得到的航向角，N 为样本大小，一般取几分钟的数据。

实际应用过程中，磁力计的零偏值随温度而变化，需要在不同温度下对磁力计进行标定，建立磁力计零偏随温度的变化模型，进而对零偏作温度补偿。

2.4.2 传感器确定性误差标定方法

单个陀螺的测量模型：

$$\tilde{\omega} = k\omega + b + \varepsilon(t) \tag{2.20}$$

式中，$\tilde{\omega}$：实际测量值（rad/s），即陀螺输出值；

ω：真实值，即真实转速在陀螺敏感轴上的分量，（rad/s）；

k：陀螺标定因数；

b：陀螺零偏，（rad/s）；

$\varepsilon(t)$：随机误差。

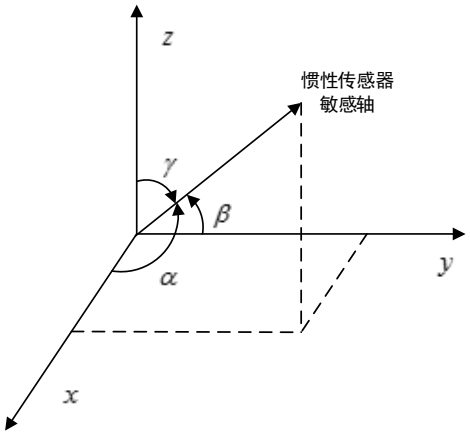

图 2-3 标定原理示意图

三轴陀螺测量模型[54-57]：

$$\tilde{\omega}_{ib}^{b} = ST^{-1}\bar{\omega}_{ib}^{b} + \vec{b} + \varepsilon(t) \tag{2.21}$$

式中，$\tilde{\omega}_{ib}^{b}$：三个陀螺实际测量值（rad/s）；

$\bar{\omega}_{ib}^{b}$：标定坐标系下的真实转速（rad/s），即标定后的目标值；

T：安装误差矩阵；

S：$\begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{bmatrix}$，$k_1$，$k_2$，$k_3$ 分别为 x，y，z 三个方向标定因数。

同理，三轴加速度计的测量模型[54-57]：

$$\tilde{f}_{ib}^{b} = ST^{-1}\bar{f}_{ib}^{b} + \vec{b} + \varepsilon(t) \tag{2.22}$$

式中，$\tilde{f}_{ib}^{b}$：三个加速度计的实际测量值（m/s^2）；

$\bar{f}_{ib}^{b}$：标定坐标系下的真实比力（m/s^2），即标定后的目标值；

约定如下：六面体的 6 个面分别编号为 a，b，c，d，e，f，其中 a，b 面垂直于 x 轴，c，d 面垂直于 y 轴，e，f 面垂直于 z 轴。3 个加速度计分别编号为 1，2，3；3 个陀螺分别编号为 1，2，3。

因此，当 j（j=a,b,c,d,e,f）面朝下时，标定方程可表示为：

$$\begin{bmatrix} k_1 A_{1,j} + b_1 \\ k_2 A_{2,j} + b_2 \\ k_3 A_{3,j} + b_3 \end{bmatrix} = \begin{bmatrix} \cos\alpha_1 & \cos\beta_1 & \cos\gamma_1 \\ \cos\alpha_2 & \cos\beta_2 & \cos\gamma_2 \\ \cos\alpha_3 & \cos\beta_3 & \cos\gamma_3 \end{bmatrix} \cdot \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = T \cdot \bar{g}_j \tag{2.23}$$

式中：

k_i：第 i 号加速度计的标度因数；

b_i：第 i 号加速度计的零偏；

$A_{i,j}$：当 j 面朝下时，第 i 号惯性传感器的输出值；

$\bar{g}_j$：当 j 面朝下时，重力在六面体坐标系下的投影；

将 j=a 和 j=b 代入式 2.21，并相减得：

$$\begin{bmatrix} k_1(A_{1,\mathrm{a}} - A_{1,\mathrm{b}}) \\ k_2(A_{2,\mathrm{a}} - A_{2,\mathrm{b}}) \\ k_3(A_{3,\mathrm{a}} - A_{3,\mathrm{b}}) \end{bmatrix} = \begin{bmatrix} \cos\alpha_1 & \cos\beta_1 & \cos\gamma_1 \\ \cos\alpha_2 & \cos\beta_2 & \cos\gamma_2 \\ \cos\alpha_3 & \cos\beta_3 & \cos\gamma_3 \end{bmatrix} \cdot \begin{bmatrix} 2g \\ 0 \\ 0 \end{bmatrix} = 2g \cdot \begin{bmatrix} \cos\alpha_1 \\ \cos\alpha_2 \\ \cos\alpha_3 \end{bmatrix} \tag{2.24}$$

同理，对于 c 和 d、e 和 f 面朝下时可得：

$$\begin{bmatrix} k_1(A_{1,\mathrm{c}}-A_{1,\mathrm{d}}) \\ k_2(A_{2,\mathrm{c}}-A_{2,\mathrm{d}}) \\ k_3(A_{3,\mathrm{c}}-A_{3,\mathrm{d}}) \end{bmatrix} = 2g\cdot\begin{bmatrix} \cos\beta_1 \\ \cos\beta_2 \\ \cos\beta_3 \end{bmatrix} \tag{2.25}$$

$$\begin{bmatrix} k_1(A_{1,\mathrm{e}}-A_{1,\mathrm{f}}) \\ k_2(A_{2,\mathrm{e}}-A_{2,\mathrm{f}}) \\ k_3(A_{3,\mathrm{e}}-A_{3,\mathrm{f}}) \end{bmatrix} = 2g\cdot\begin{bmatrix} \cos\gamma_1 \\ \cos\gamma_2 \\ \cos\gamma_3 \end{bmatrix} \tag{2.26}$$

令 $\begin{bmatrix} A_{1,x} & A_{1,y} & A_{1,z} \\ A_{2,x} & A_{2,y} & A_{2,z} \\ A_{3,x} & A_{3,y} & A_{3,z} \end{bmatrix} = \begin{bmatrix} A_{1,\mathrm{a}} & A_{1,\mathrm{c}} & A_{1,\mathrm{e}} \\ A_{2,\mathrm{a}} & A_{2,\mathrm{c}} & A_{2,\mathrm{e}} \\ A_{3,\mathrm{a}} & A_{3,\mathrm{c}} & A_{3,\mathrm{e}} \end{bmatrix} - \begin{bmatrix} A_{1,\mathrm{b}} & A_{1,\mathrm{d}} & A_{1,\mathrm{f}} \\ A_{2,\mathrm{b}} & A_{2,\mathrm{d}} & A_{2,\mathrm{f}} \\ A_{3,\mathrm{b}} & A_{3,\mathrm{d}} & A_{3,\mathrm{f}} \end{bmatrix}$

于是：

$$\left.\begin{aligned} \cos\alpha_i &= \frac{k_i A_{i,x}}{2g} \\ \cos\beta_i &= \frac{k_i A_{i,\mathrm{y}}}{2g} \\ \cos\gamma_i &= \frac{k_i A_{i,\mathrm{z}}}{2g} \\ \cos^2\alpha_i+\cos^2\beta_i+\cos^2\gamma_i &= 1 \end{aligned}\right\} \Rightarrow \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} = 2g\cdot\begin{bmatrix} \dfrac{1}{\sqrt{A_{i,x}^2+A_{i,\mathrm{y}}^2+A_{i,\mathrm{z}}^2}} \\ \dfrac{1}{\sqrt{A_{i,x}^2+A_{i,\mathrm{y}}^2+A_{i,\mathrm{z}}^2}} \\ \dfrac{1}{\sqrt{A_{i,x}^2+A_{i,\mathrm{y}}^2+A_{i,\mathrm{z}}^2}} \end{bmatrix} \tag{2.27}$$

$$T = \begin{bmatrix} \cos\alpha_1 & \cos\beta_1 & \cos\gamma_1 \\ \cos\alpha_2 & \cos\beta_2 & \cos\gamma_2 \\ \cos\alpha_3 & \cos\beta_3 & \cos\gamma_3 \end{bmatrix} = \frac{1}{2g}K\begin{bmatrix} A_{1,x} & A_{1,y} & A_{1,z} \\ A_{2,x} & A_{2,y} & A_{2,z} \\ A_{3,x} & A_{3,y} & A_{3,z} \end{bmatrix} \tag{2.28}$$

将所有面的结果相加，可得：

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = -\frac{1}{6}\begin{bmatrix} k_1\sum_j A_{1,j} \\ k_2\sum_j A_{2,j} \\ k_3\sum_j A_{3,j} \end{bmatrix} \tag{2.29}$$

根据式 2.25-2.27 就确定了传感器的标度因数、安装误差矩阵和零偏。于是确定性误差补偿公式为：

$$\begin{bmatrix} \hat{\omega}_x \\ \hat{\omega}_y \\ \hat{\omega}_z \end{bmatrix} = T^{-1}\begin{bmatrix} k_1\tilde{\omega}_1+b_1 \\ k_2\tilde{\omega}_2+b_2 \\ k_3\tilde{\omega}_3+b_3 \end{bmatrix} \tag{2.30}$$

2.5 试验结果与分析

为验证本章提出的车载 DR 系统的导航性能，进行了实际道路导航实验。

基于多传感器数据融合的 DR 系统实物图如图 2-4 所示，DR 系统装在山地自行车上。里程计由型号为 ROCKYOU 的自行车码表改装而成，包括霍耳传感器、磁头和处理电路。为提高里程计的速度更新率，在前轮的每根辐条上都安装一个磁头。当磁头靠近霍耳传感器时，霍耳传感器便产生一个脉冲信号，由 MEMS 开发板处理得到前向速度和前向加速度信息。MEMS 开发板采用 ST 公司型号为 STM32F3DISCOVERY 的开发板，实物如图 2-5 所示。MEMS 开发板上安装有 3 轴 MEMS 加速度计、3 轴 MEMS 陀螺和 3 轴 MEMS 磁力计等传感器。为减少笔记本电脑对磁力计的电磁干扰，MEMS 开发板安装在远离笔记本电脑的地方。笔记本电脑通过串口采集各传感器信息并进行导航解算。

图 2-4 基于多传感器融合的 DR 系统实物图

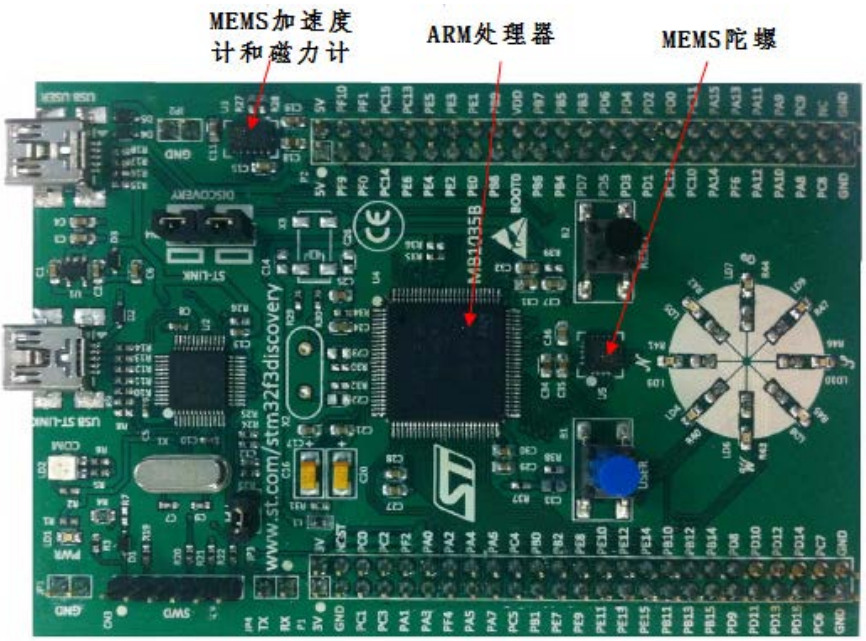

图 2-5 MEMS 开发板实物图

实验中采用的传感器均为低成本、低精度的传感器，具体型号和误差特性如表 2-1 所示。

表 2-1 传感器误差特性

传感器	型号	误差特性
陀螺	L3GD20	标定后零偏：0.05°/s
加速度计	LSM303DLHC	标定后零偏：0.1m/s^2
磁力计	LSM303DLHC	噪声标准差：0.01G
里程计	ROCKYOU	相对误差小于 0.5%
GPS	GARMIN	定位精度：5m

实验路线为绕杭州黄龙体育中心周围一圈，历时 760s，总里程约为 4km。实验中，由笔记本电脑采集各传感器的数据，并在 MATLAB 中进行仿真后处理。由 GPS 提供 DR 系统初始位置，并以 GPS 轨迹作为参考，利用本文提出的基于多传感器信息融合的 DR 算法进行导航，结果如图 2-6 所示。

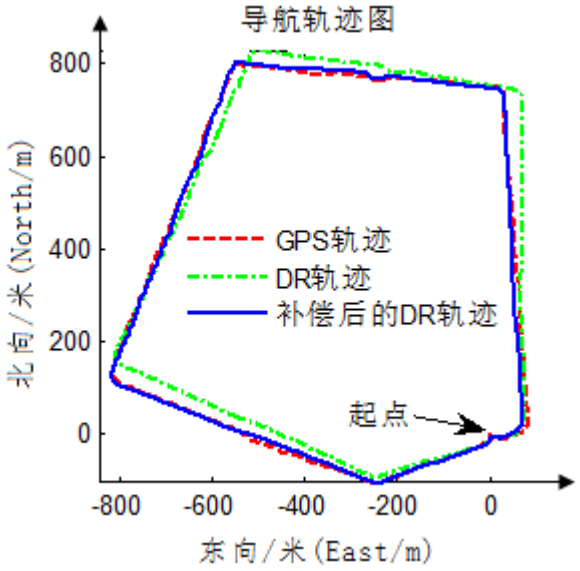

图 2-6 导航轨迹图

图 2-6 中的点划线为 DR 系统的导航轨迹，可以明显看出与 GPS 轨迹（虚线）有一固定夹角，这是由于存在航向角偏差引起的。利用在线航向角补偿算法可估算出航向角偏差约为 3.2°，经过航向角补偿后的导航轨迹如图 2-6 中的实线所示，可以看出基本与 GPS 轨迹重合。补偿后的 DR 轨迹相对于 GPS 轨迹的位置误差如图 2-7 所示，整个导航过程位置误差随时间缓慢漂移，最大误差约为 15m，小于总航程的 1%。

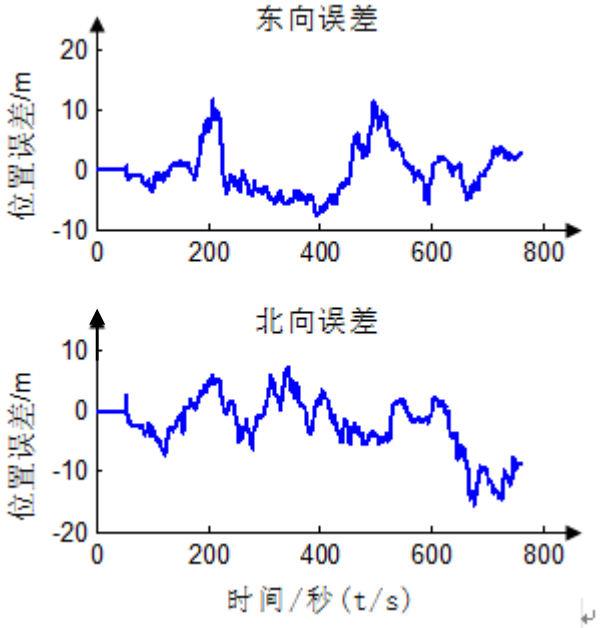

图 2-7 补偿后的 DR 系统的位置误差

进一步分析 AHRS 的航向角误差以及里程计投影到地理坐标系后的速度误

差，分别如图 2-8 和图 2-9 所示。从图 2-8 中可以看出 AHRS 输出的航向角与 GPS 估算出的航向角基本一致，误差不随时间累积（注：600s~720s 之间的航向角处于 180°到-180°之间的临界边缘，故出现上下跳变的现象，属于正常现象）。从图 2-9 中可以看出速度误差不随时间累积，误差积分后趋向于 0，因此积分后表现为位置误差的缓慢漂移，如图 2-7 所示。

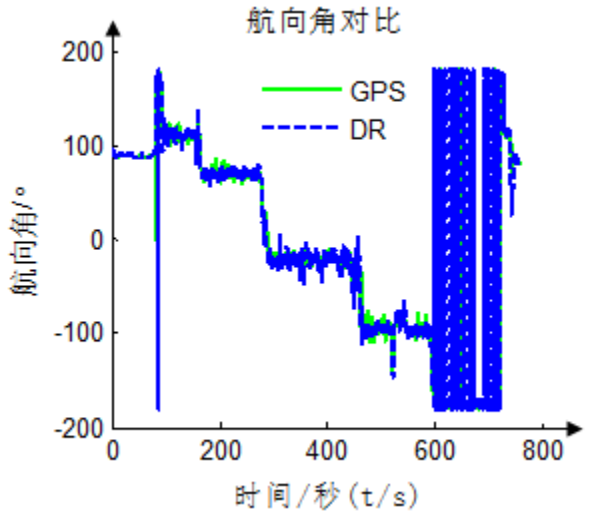

图 2-8 DR 系统与 GPS 测得的航向角比较

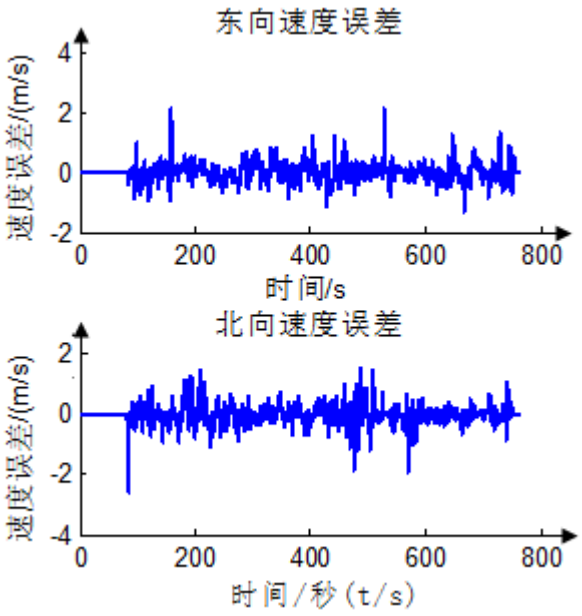

图 2-9 补偿后的 DR 系统的速度误差

综上所述，速度误差、姿态和航向误差不随时间积累，如果对航向角进行补偿，位置误差随时间的积累速度就会变得很慢，能够维持较长时间（至少 12min）的高精度导航。

2.6 本章小结

本章针对 MEMS-DR 系统单独导航时导航误差迅速累积的问题，设计并实现了一种基于多传感器信息融合的 MEMS-DR 系统，综合利用里程计、陀螺、加速度计、磁力计等多种传感器的信息，有效抑制了 DR 系统的累积误差。实验结果表明，位置误差小于总航程的 1%，与传统 DR 系统相比，导航误差随时间积累速度慢，能保持更长时间的高精度导航。本文提出的 3D 车载 DR 系统具有导航精度高、成本低、实现容易等特点。

第3章 卫星导航与 MEMS 航位推算系统的紧耦合算法研究

GNSS 与航位推算系统（Dead Reckoning，DR）具有很强的优势互补性[42, 58-63]，GNSS/DR 组合导航是应用最广泛的车载导航方式之一。从信息交换以及组合程度看，GNSS/DR 组合导航可分为松耦合、紧耦合、超紧耦合等方式[8, 44]。目前以松耦合组合方式为主，即 GNSS 接收机与 DR 系统各自独立工作，由融合算法融合两者数据给出最优结果。但当车辆穿行在城市高楼区、林荫道、隧道等处时，常出现卫星信号遮挡问题[64-67]，可见卫星数少于 4 颗时，GNSS 接收机无法正常定位，在这种情况下组合导航系统的可靠性无法得到保证。紧耦合方式是一种相对复杂的组合方式，卫星接收机提供给融合算法并用于 DR 误差修正的是伪距和伪距率等接收机用于定位解算的原始信息[44]。与松耦合相比，紧耦合的优势在于当可见卫星数少于 4 颗时也能进行组合导航，从而有效减少 GNSS 失效的时间，提高导航系统的抗干扰能力和导航精度。

本章针对车辆穿行在城市高楼区、林荫道、隧道等 GNSS 卫星信号遮挡严重的场合，设计并实现了一种基于联邦无迹卡尔曼滤波器（Federated Uncented Kalman Filter, FUKF）的 GNSS/DR 紧耦合组合导航算法。

在 GNSS/DR 紧耦合数据融合算法中，如果信息分配不合理，导航定位精度易受接收机弱信号通道的影响。最简单的方法是直接将弱信号通道的信息丢弃，但这样的做法不能充分利用各通道的信息，特别是当可见卫星数比较少的时候。为减少数据融合算法中弱信号通道对定位精度的影响但又充分利用各通道的信息，提出了一种自适应信息权重分配算法，即根据每个通道的载噪比（Carrier to Noise density ratio，C/N_0）来自适应地调整各子滤波器的信息分配权重，实现最佳的导航性能。

本章首先介绍了 GNSS/MEMS-DR 紧耦合组合导航原理，然后详细推导了基于联邦滤波器的 GNSS/MEMS-DR 紧耦合组合导航的数学模型和自适应信息权重分配算法。最后通过在山区道路和隧道导航试验，分析基于 FUKF 的 GNSS/MEMS-DR 紧耦合组合导航算法的导航性能。

3.1 紧耦合系统总体设计

GNSS/MEMS-DR 紧耦合组合导航主要包括 GNSS 接收机和 MEMS-DR 系统两部分，原理如图 3-1 所示。DR 系统采集里程计、加速度计、陀螺仪和磁力计等多种传感器的信息，根据当前的位置、速度、姿态等信息，推算下一时刻的位置和速度等导航信息。GNSS 接收机输出所有可见卫星的伪距和伪距率等用于导航定位解算的原始数据，与 DR 系统输出的导航信息进行数据融合，输出最终的组合导航信息，并反馈给 DR 进行误差修正。

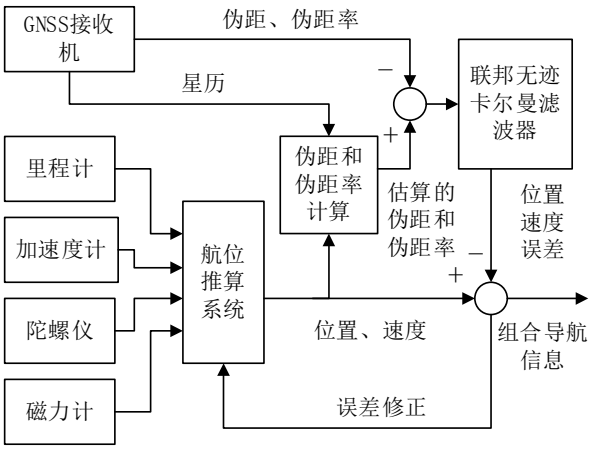

图 3-1 GNSS/MEMS-DR 紧耦合系统组成和原理框图

GNSS 采用星源北斗公司的卫星导航开发板，输出伪距和伪距率等信息。惯性传感器数据由 ST 公司的 STM32F3DISCOVERY 开发板提供，板上安装有型号为 L3GD20 的 3 轴 MEMS 陀螺仪，以及型号为 LSM303DLHC 的 3 轴 MEMS 加速度计和 3 轴 MEMS 磁力计。里程计脉冲信号采集自改进后的山地车码表，码表采用霍耳效应原理，试验过程中，在车轮辐条上安装了 32 个磁头以提高里程计脉冲更新率。对于 DR 系统单独导航的性能已经经过多次测试，导航定位精度约为总里程的 1%。

传感器信息融合算法是组合导航系统的核心问题，直接影响组合导航系统的性能。文中提出的紧耦合组合导航算法建立在联邦卡尔曼滤波器的基础之上。联邦卡尔曼滤波器是 Carlson[68-72]于 1988 年提出的一种基于信息分配原则的分散式滤波方法，由一系列子滤波器和一个主滤波器组成，每个子滤波器估计得到局部最优估计，通过主滤波器对各个子系统的信息进行融合。联邦滤波器的

优势在于降低了滤波器阶数，从而减小计算量，同时，联邦滤波器能够容易地检测出每个子滤波器出现的错误，提高滤波的可靠性[1]。

导航系统多为非线性系统，EKF 方法通过将系统误差函数线性化后再进行线性滤波，由于线性化误差大，影响了滤波的精度。非线性滤波方法通过对误差模型矢量的统计特性进行近似来达到将模型线性化的目的，无迹卡尔曼滤波（Unscented Kalman Filter，UKF）就是其中最具代表性的非线性卡尔曼滤波方法。UKF 方法通过设计少量的 sigma 点，经过 UT 变换和非线性函数的传播，估计得到状态矢量的统计特性，然后进行滤波计算[32, 73-75]。

UKF 由以下方程确定[32-34, 73-82]：

$$\begin{cases} \boldsymbol{x}_k = f\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}\right) + \boldsymbol{w}_{k-1} \\ \boldsymbol{z}_k = h\left(\boldsymbol{x}_k\right) + \boldsymbol{\eta}_k \end{cases} \tag{3.1}$$

式中，$\boldsymbol{x}_k$ 为系统状态变量，$\boldsymbol{z}_k$ 为观测量，$\boldsymbol{w}_k \sim N\left(0, \boldsymbol{Q}_k\right)$ 为高斯型的系统噪声，$\boldsymbol{\eta}_k \sim N\left(0, \boldsymbol{R}_k\right)$ 为高斯型的量测噪声。

3.2 紧耦合组合导航数学模型推导

每个子滤波器具有相同的结构，均采用 UKF 作为状态估计算法，且具有相同的状态方程和量测方程，不同的是观测量不同，即每个子滤波器接收对应接收机跟踪通道的伪距和伪距率等信息作为观测量。

下面详细推导了基于联邦无迹卡尔曼滤波器（FUKF）的状态方程和量测方程。

3.2.1 离散状态方程的建立

每个子滤波器具有相同的状态方程，包括 DR 系统的误差模型和 GNSS 接收机时钟误差模型。

以东北天地理坐标系作为导航坐标系，DR 系统的误差主要包括导航坐标系下的位置误差$\left[\delta r_e, \delta r_n, \delta r_u\right]^T$、导航坐标系下的速度误差$\left[\delta v_e, \delta v_n, \delta v_u\right]^T$。GNSS 接收机时钟误差[44, 83-86]主要有时钟零偏δb（单位为 m）和时钟漂移δd（单位

为 m/s）。

因此状态变量选取为：$\boldsymbol{x}=\left[\delta r_e,\delta r_n,\delta r_u,\delta v_e,\delta v_n,\delta v_u,\delta b,\delta d\right]^T$。于是 GNSS/DR 紧耦合系统的离散状态方程可表示为：

$$\boldsymbol{x}_k = f\left(\boldsymbol{x}_{k-1},\boldsymbol{u}_{k-1}\right)+\boldsymbol{w}_{k-1} \tag{3.2}$$

其中 F 为状态转移矩阵，是 9x9 维的矩阵；$\boldsymbol{w}_k$ 为 9x1 的系统噪声向量，具体表达式如下：

$$\boldsymbol{F}=\begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$\boldsymbol{w}_k=\left[\tfrac{1}{2}\omega_v\Delta t^2,\tfrac{1}{2}\omega_v\Delta t^2,\tfrac{1}{2}\omega_v\Delta t^2,\omega_v\Delta t,\omega_v\Delta t,\omega_v\Delta t,\tfrac{1}{2}\omega_t\Delta t^2,\omega_t\Delta t\right]^T \tag{3.4}$$

式中，ω_v 和 ω_t 分别为 DR 系统的速度噪声和 GNSS 接收机的时钟漂移噪声，仿真过程中分别取为 2 和 2。Δt 为 k 时刻到 k+1 时刻的时间间隔，仿真过程取值为 0.1，即组合导航算法更新率为 10Hz。于是，系统噪声协方差矩阵可表示为 $\boldsymbol{Q}_k=E\{\boldsymbol{w}_k\boldsymbol{w}_k^T\}$。

3.2.2 量测方程的建立

每个子滤波器接收对应通道的伪距和伪距率信息。根据载体位置和卫星位置计算得到的距离和距离变化率分别称为 DR 伪距和 DR 伪距率。则 DR 伪距和伪距率与 GNSS 伪距和伪距率之差作为对应子滤波器的观测量，即

$$\boldsymbol{z}_k^j=\begin{bmatrix}\rho_{DR}^j-\rho_{GNSS}^j\\ \dot{\rho}_{DR}^j-\dot{\rho}_{GNSS}^j\end{bmatrix} \tag{3.5}$$

其中：ρ_{DR}^j 和 $\dot{\rho}_{DR}^j$ 分别为 j 号卫星的 DR 伪距和伪距率，ρ_{GNSS}^j 和 $\dot{\rho}_{GNSS}^j$ 分别为 j 号卫星的 GNSS 伪距和伪距率，其中，ρ_{GPS}^j 为经过卫星时钟误差、对流层

延时误差和电离层延时误差等误差修正后的伪距值。

量测方程推导如下：

设载体在地球坐标系下的真实位置为 $\boldsymbol{r}=[x,y,z]^T$，DR 解算出的载体位置为 $\hat{\boldsymbol{r}}=[\hat{x},\hat{y},\hat{z}]^T$，由卫星星历解算出的 j 号卫星的位置为 $\boldsymbol{r}_s^j=\left[x^j,y^j,z^j\right]^T$，于是：

$$\rho_{DR}^j-\rho_{GNSS}^j=(\hat{\boldsymbol{r}}-\boldsymbol{r})\cdot\boldsymbol{l}^j-\delta b+\varepsilon_\rho^j=\boldsymbol{C}_n^e\delta\boldsymbol{r}\cdot\boldsymbol{l}^j-\delta b+\varepsilon_\rho^j \tag{3.6}$$

其中：$\boldsymbol{C}_n^e$ 为导航坐标系与地球坐标系之间的变换矩阵，$\delta\boldsymbol{r}$ 为导航坐标系下的位置误差，即 $\delta\boldsymbol{r}=[\delta r_e,\delta r_n,\delta r_u]^T$。$\varepsilon_\rho^j$ 为经误差修正后的残留的伪距噪声，包括大气层延时和接收机通道噪声等。$\boldsymbol{l}^j$ 表示载体与 j 号卫星的视线（Line of Sight）单位向量[44]，定义为：

$$\boldsymbol{l}^j=\frac{\hat{\boldsymbol{r}}-\boldsymbol{r}_s^j}{\left\|\hat{\boldsymbol{r}}-\boldsymbol{r}_s^j\right\|} \tag{3.7}$$

同理，设载体在地球坐标系下的真实速度为 $\boldsymbol{v}=\left[v_x,v_y,v_z\right]^T$，DR 解算出的载体速度为 $\hat{\boldsymbol{v}}=\left[\hat{v}_x,\hat{v}_y,\hat{v}_z\right]^T$，由卫星星历解算出的 j 号卫星的速度为 $\boldsymbol{v}_s^j=\left[v_x^j,v_y^j,v_z^j\right]^T$，于是：

$$\dot{\rho}_{DR}^j-\dot{\rho}_{GNSS}^j=(\hat{\boldsymbol{v}}-\boldsymbol{v})\cdot\boldsymbol{l}^j-\delta d+\dot{\varepsilon}_{\dot{\rho}}^j=\boldsymbol{C}_n^e\delta\boldsymbol{v}\cdot\boldsymbol{l}^j-\delta d+\dot{\varepsilon}_{\dot{\rho}}^j \tag{3.8}$$

其中 $\delta\boldsymbol{v}$ 为导航坐标系下的速度误差，即 $\delta\boldsymbol{v}=[\delta v_e,\delta v_n,\delta v_u]$。$\dot{\varepsilon}_{\dot{\rho}}^j$ 为经误差修正后的残留的伪距率噪声。

综合式（3.5）和式（3.7）可得 GNSS/DR 紧耦合的量测方程为：

$$\boldsymbol{z}_k^j=\begin{bmatrix}\rho_{DR}^j-\rho_{GNSS}^j\\ \dot{\rho}_{DR}^j-\dot{\rho}_{GNSS}^j\end{bmatrix}=\boldsymbol{H}^j\boldsymbol{x}_k+\begin{bmatrix}\varepsilon_\rho^j\\ \dot{\varepsilon}_{\dot{\rho}}^j\end{bmatrix} \tag{3.9}$$

其中：$\boldsymbol{H}^j$ 为子滤波器 j 的观测矩阵，维度为 2×9，表达式如下：

$$\boldsymbol{H}^j=\begin{bmatrix}(\boldsymbol{C}_n^e\boldsymbol{l}^j)^T & \boldsymbol{0}_{3\times1} & 0 & -1 & 0\\ \boldsymbol{0}_{3\times1} & (\boldsymbol{C}_n^e\boldsymbol{l}^j)^T & 0 & 0 & -1\end{bmatrix} \tag{3.10}$$

仿真过程中，观测量协方差 $\boldsymbol{R}_k^j=diag\{(\varepsilon_\rho^j)^2,(\dot{\varepsilon}_{\dot{\rho}}^j)^2\}=diag\{100,0.25\}$。

3.3 联邦滤波器权重分配方法研究

为减少弱信号通道对全局滤波精度的影响，研究了联邦卡尔曼滤波信息分配因子调整算法，即根据 GNSS 接收机每个通道的载噪比（C/N_0）的大小估计对应子滤波器观测量噪声大小，从而调节每个子滤波器信息在主滤波器中的权重，优化 GNSS 和 DR 信息组合滤波性能。

3.3.1 主滤波器信息分配

主滤波器负责对各个子系统的信息进行融合，子滤波器估计与主滤波器全局估计之间的关系按以下融合算法确定[69, 87]：

$$\begin{cases} \boldsymbol{P}_g^{-1} = \sum_{j=1}^{N} \boldsymbol{P}_j^{-1} \\ \hat{\boldsymbol{x}}_g = \boldsymbol{P}_g \cdot \sum_{j=1}^{N} \boldsymbol{P}_j^{-1} \hat{\boldsymbol{x}}_j \end{cases} \tag{3.11}$$

式中：$\hat{\boldsymbol{x}}_g$ 和 $\boldsymbol{P}_g$ 为主滤波器的全局估计和协方差阵，$\hat{\boldsymbol{x}}_j$ 和 $\boldsymbol{P}_j (j = 1,2,\cdots,N)$ 是子滤波器 j 的状态误差和协方差阵。其中，主滤波器的信息通过信息分配因子 γ_i 分配到各子滤波器，即[69, 87]：

$$\begin{cases} \boldsymbol{P}_j = \gamma_j^{-1} \boldsymbol{P}_g \\ \hat{\boldsymbol{x}}_j = \hat{\boldsymbol{x}}_g \\ \boldsymbol{Q}_j = \gamma_j^{-1} \boldsymbol{Q}_g \end{cases} \tag{3.12}$$

式中，$\boldsymbol{Q}_g$ 和 $\boldsymbol{Q}_j$ 分别为主滤波器和子滤波器 j 的噪声方差阵。

信息分配满足信息守恒原理[69, 87]，即 $\sum_{j=1}^{N} \gamma_j = 1$。通过调整 γ_j，即可使滤波性能具有更好的适应性。

3.3.2 权重分配算法

文献[84, 85, 88]中指出，GNSS 接收机中码跟踪环和载波跟踪环的环路噪声与 C/N_0 存在以下关系：

$$\sigma_\tau^2 = \frac{B_\tau}{2C/N_0}\left(1+\frac{2}{T\cdot C/N_0}\right) \tag{3.13}$$

$$\sigma_\theta^2 = \frac{B_\theta}{C/N_0}\left(1+\frac{1}{2T\cdot C/N_0}\right) \tag{3.14}$$

其中：σ_τ 和 σ_θ 分别为码跟踪环和载波跟踪环的环路噪声标准差，B_τ 和 B_θ 分别为码跟踪环和载波跟踪环的带宽，T 为环路积分时间。

伪距和伪距率标准与通道噪声之间存在如下关系：

$$\sigma_\rho = \lambda_{code}\sigma_\tau \tag{3.15}$$

$$\sigma_{\dot\rho} = \frac{\lambda_{carr}}{2\pi T}\sigma_\theta \tag{3.16}$$

其中：λ_{code} 是 GNSS 测距码每个码片的长度，λ_{carr} 为载波波长。

于是，根据信息分配原则，设计的自适应信息分配算法如下：

$$\begin{cases} \gamma_j = \dfrac{1}{2}\dfrac{\left(\sigma_\rho^2\right)_j^{-1}}{\sum_{i=1}^{N}\left(\sigma_\rho^2\right)_i^{-1}} + \dfrac{1}{2}\dfrac{\left(\sigma_{\dot\rho}^2\right)_j^{-1}}{\sum_{i=1}^{N}\left(\sigma_{\dot\rho}^2\right)_i^{-1}} \approx \dfrac{\alpha_j^{-1}}{\sum_{i=1}^{N}\alpha_i^{-1}} \\ \alpha = \dfrac{1}{C/N_0}\left(1+\dfrac{1}{T\cdot C/N_0}\right) \end{cases} \tag{3.17}$$

从上式中可以看出，信号越弱的通道，观测量方差就越大，分配的信息权重就越小，从而可有效减小对全局滤波精度的影响。

3.4 试验结果和分析

为验证基于 FUKF 的 GNSS/DR 紧耦合组合导航算法的性能，进行了在山区道路和隧道的导航试验。试验过程中采集 GNSS、惯性传感器和里程计信息，然后在 MATLAB 中进行仿真后处理，并将试验结果在 Google 地图中显示。

3.4.1 实验设备和条件

实验装置实物图如图 3-2 所示。实验装置包括 GNSS 开发板、MEMS-DR 系统、GNSS 天线、笔记本电脑等组成。

图 3-2 GNSS/MEMS-DR 紧耦合实验装置实物图

GNSS 开发板采用星源北斗公司的高级 GNSS 开发板 HG-RE03-D，提供 GNSS 卫星伪距和伪距率等信息。HG-RE03-D 是卫星导航的高级开发工具，包含射频板、处理器板、FPGA 板等，能够满足 GNSS 的各种开发需求，实物图和特性分别如图 3-3 和表 3-1 所示。MEMS-DR 系统输出位置、速度和姿态等信息。为增强信号的接收，GNSS 安装在头顶的帽子上，如图 3-2 所示。

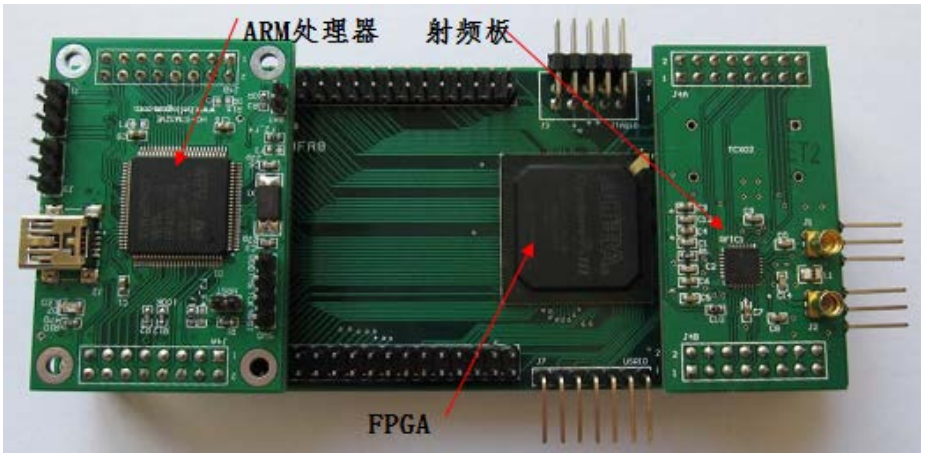

图 3-3 GNSS 高级开发板 HG-RE03-D 实物图

表 3-1 GNSS 高级开发板 HG-RE03-D 特性

组成部分	特性
射频芯片	MAX2769ETI+
TCXO 频率	16.368MHz
处理器	STM32F407VET6
FPGA	EP3C55F484C8N
采样位数	2 位
采样速率	16.368MHz
信号中心频率	4.092MHz
支持系统	GPS L1、BD2 B1

实验路线沿着梅灵北路和灵溪南路，总里程约为 8km，历时约 30min，途经长度分别为 1.2km 和 1.7km 的梅灵隧道和灵溪隧道。整条路线为山区道路，并且一路上 GNSS 卫星信号受树木遮挡严重，平均 C/N_0 为 30dB-Hz，与空旷的情况下[9, 85]（45dB-Hz）相比，平均衰减约 15dB。可见卫星数随时间变化的曲线如图 3-4 所示，可以看出，一路上可见卫星数较少，有很多路段卫星数少于 4 颗，其中 180s~460s 和 1250s~1510s 时行驶在隧道，卫星数为 0，460s~800s 时为比较紧耦合与松耦合的导航能力，人为地屏蔽了 3 颗信号最弱的卫星，使卫星数一直小于 4。

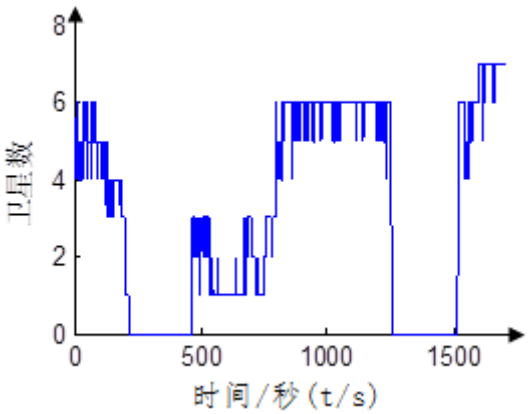

图 3-4 卫星数随时间的变化曲线

3.4.2 实验数据处理步骤

数据采集完成后，在 MATLAB 中进行仿真后处理。本章提出的 GNSS/MEMS-DR 紧耦合组合导航算法按如下步骤执行：

1) 采集里程计、加速度计、陀螺仪和磁力计等传感器的信息，经过 DR 算法计算得到车辆位置和速度等信息；
2) 采集 GNSS 的伪距和伪距率等原始观测量，并根据每个通道的 C/N_0 自适应地调整相应子滤波器的权重分配因子；
3) 每个子滤波器利用对应接收机通道的信息和 DR 的信息估计得到局部最优估计；
4) 主滤波器融合所有子滤波器的信息得到最终的导航结果；
5) 如果数据融合中可见卫星数大于 0，则反馈修正 DR 误差。

具体流程如图 3-5 所示。

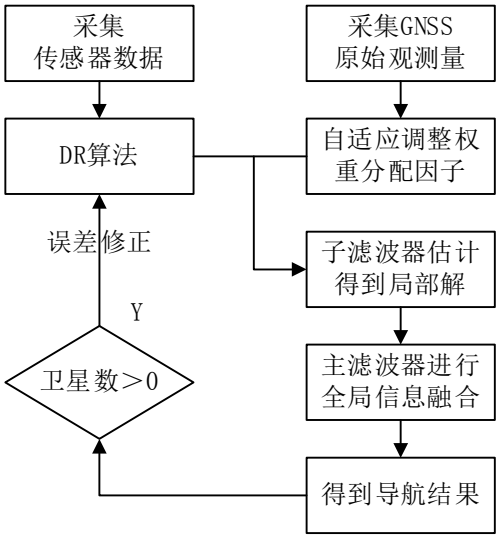

图 3-5 紧耦合算法流程图

3.4.3 结果与分析

分别采用目前常用的松耦合算法和本文提出的紧耦合算法后，导航结果如图 3-6 所示。

图 3-6 松耦合和紧耦合导航轨迹图

从图中可以看出在 460s~800s 期间（即 B 点到 C 点之间），松耦合算法由于卫星数小于 4，GNSS 不能正常定位，导航轨迹不断偏离真实轨迹，在 C 点位置误差约为 35m。紧耦合算法的导航轨迹与地图上的道路基本重合，B 点到 C 点之间误差在 10m 以内。在梅灵隧道出口处（B 点）和灵溪隧道出口处（E 点）分别有 15m 和 20m 的误差，这是由于在隧道内，只能依靠 DR 导航，误差约为隧道长度的 1%；其他路段导航误差均在 10m 以内，能够满足现实中的导航要求。

表 3-2 路线中特定点的导航误差

位置	时刻(s)	松耦合误差(m)	紧耦合误差(m)
A	180	7	7
B	450	17	20
C	840	34	10
D	1250	6	7
E	1510	21	20
F	1650	5	5

从路线中选取了 6 个点，以型号为 GARMIN GPS15xL 的 GPS 作为参考基

准，进行导航误差分析，如图 3-7 和表 3-2 所示，可以看出在信号遮挡严重的场合，紧耦合算法比松耦合算法具有更高的导航精度。

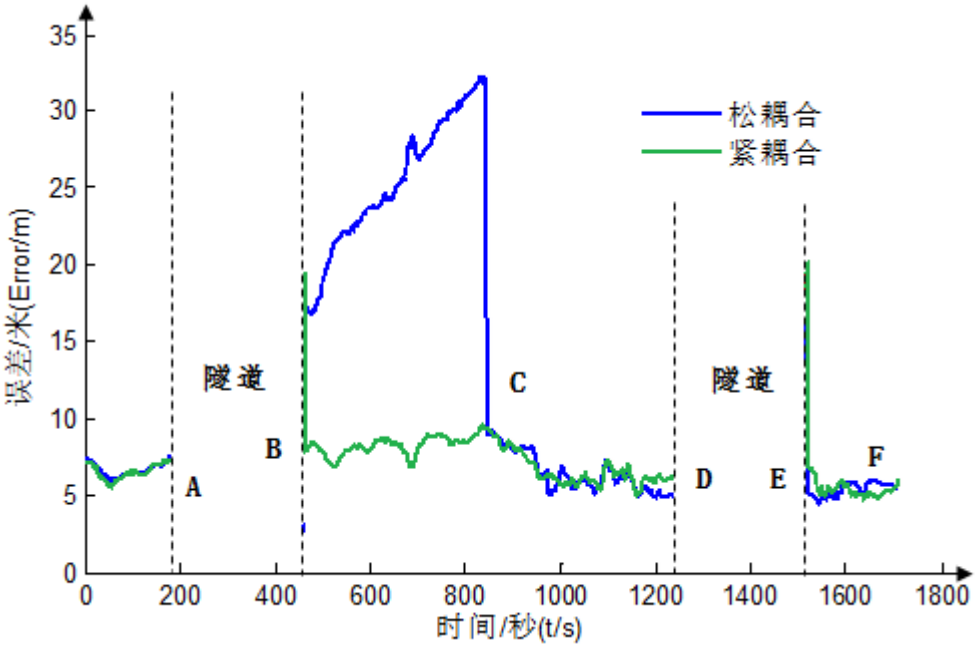

图 3-7 紧耦合与松耦合导航误差曲线

综合以上试验结果验证了本文提出的基于 FUKF 的 GNSS/DR 紧耦合组合导航算法能够在山区道路和隧道等卫星信号遮挡严重的场合保持较高精度的导航，与松耦合相比，具有更高的导航精度和抗干扰能力。

3.5 本章小结

本章针对 GNSS 卫星信号遮挡严重的场合，设计了一种基于 FUKF 的 GNSS/MEMS-DR 紧耦合组合导航算法。通过在山区道路和隧道的导航试验，并与目前常用的松耦合算法对比。实验结果验证了本章提出的紧耦合组合导航算法在卫星信号遮挡严重的场合导航的优越性，即使卫星数量少于 4 颗时，GNSS 仍能参与组合导航，从而有效提高组合导航系统的定位精度和可靠性。

本章所涉及的紧耦合算法是为了后续的 GNSS/MEMS-INS 超紧耦合作铺垫。

第4章 卫星导航与 MEMS 航位推算系统的超紧耦合算法研究

前一章研究了 GNSS/MEMS-DR 的紧耦合算法，相比松耦合算法，紧耦合算法有效提高了组合导航系统的导航精度和抗干扰能力。但在弱信号或高动态情况下，GNSS 接收机跟踪环路容易失锁[8, 19, 89-95]，导致组合导航性能下降。这一章将进一步研究 GNSS/MEMS-DR 的超紧耦合算法，利用 MEMS-DR 的信息辅助 GNSS 接收对卫星的跟踪，提高组合导航系统在弱信号场合的抗干扰能力。

本章深入研究了 GNSS/MEMS-DR 超紧耦合算法，主要包括 MEMS-DR 辅助 GNSS 的方法和超紧耦合的数据融合算法，设计并实现了一种基于矢量跟踪环和 FUKF 的超紧耦合算法。最后，通过实验验证了超紧耦合算法的有效性和优越性。

4.1 超紧耦合算法结构设计

相比紧耦合算法，超紧耦合算法中增加了惯导对接收机的辅助，以增强组合导航系统在强干扰和高动态情况下的适应能力。超紧耦合使低成本的惯性器件（如 MEMS 惯性器件）与 GNSS 的高性能组合导航成为可能[8]，超紧耦合是组合导航的发展趋势。

本文设计的 GNSS/MEMS-DR 超紧耦合算法主要由 MEMS 航位推算算法、矢量跟踪环算法、组合导航滤波器和 NCO 反馈方法等组成，总体方案结构如图 4-1 所示。

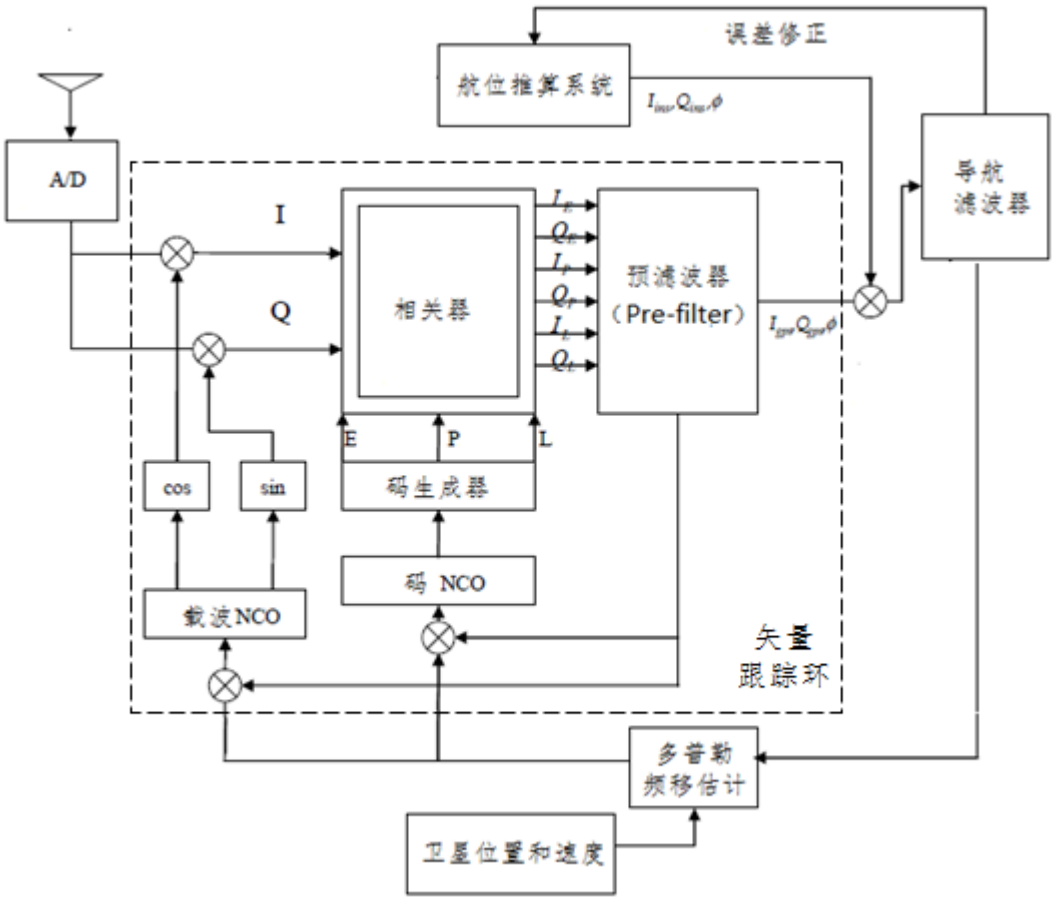

图 4-1 GNSS/MEMS-DR 超紧耦合算法结构框图

GNSS 接收机输出同相-正交信号（I-Q 信号），IMU 提供加速度和角速度等信息进行航位推算，并通过组合滤波器进行数据融合，估计出最佳导航信息，同时修正 IMU 误差，以及估算出多普勒频移等信息辅助 GNSS 接收机的捕获和跟踪。

超紧耦合算法的难点在于如何利用 MEMS-DR 的信息辅助 GNSS 接收机的载波环和码环跟踪。本文设计的超紧耦合算法中引入基于卡尔曼滤波器的矢量跟踪法，代替传统的标量跟踪法，将原始的 I 和 Q 路信号作为卡尔曼滤波器的观测量，估计出载波多普勒频移和测距码相位差分别反馈给载波环和码环的数控振荡器（NCO）。利用速度信息，估算载波多普勒频移量，调整数控振荡器（NCO）输出频率，提高动态跟踪性能，防止机体高动态运动时信号失锁；同时减小环路滤波器带宽，抑制环路噪声，从而获得更高精度的观测量。

矢量跟踪环算法、组合导航滤波器和 NCO 反馈方法是实现超紧耦合算法不可缺少的三个部分。下面分别研究了这个部分。

4.2 矢量跟踪环算法研究

传统的标量跟踪环的跟踪通道之间相互独立，即一个通道跟踪一颗卫星信号[96-98]。每个通道的跟踪环路由相关器、鉴相器、环路滤波器和数字振荡器（Numerically Controlled Oscillator，NCO）等组成，如图 4-2 所示。每个通道分别计算得到伪距和多普勒频移（伪距率）等原始观测量，输送给导航解算部分得到位置和速度信息。

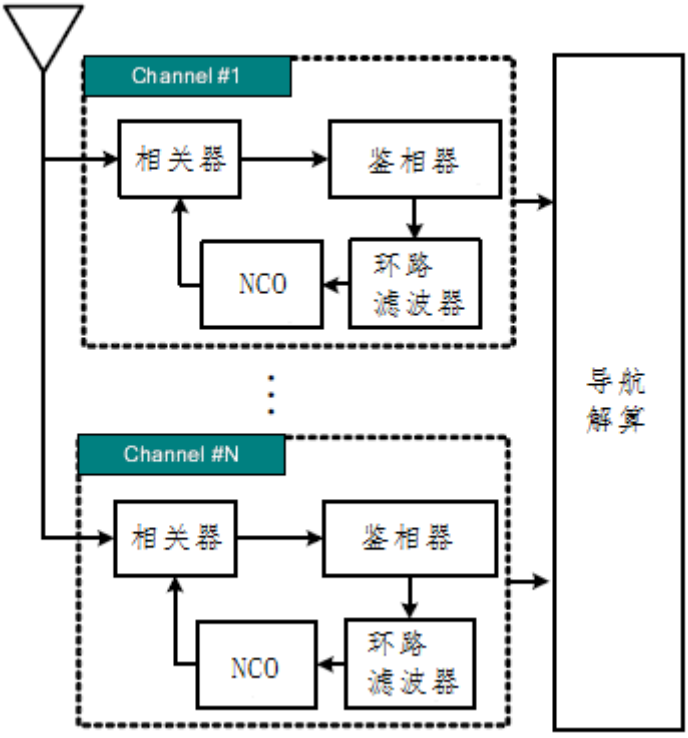

图 4-2 传统标量跟踪环

近年来，许多学者研究了矢量跟踪环以提高 GNSS 接收机的跟踪性能[96-101]。与标量跟踪环相比，矢量跟踪环能有效降低跟踪通道的噪声，提高精度；同时，矢量跟踪环具有更强的抗干扰能力，特别是在弱信号情况下，当标量跟踪环失锁时，矢量跟踪环仍然可以保持对信号的跟踪。

本文设计的矢量跟踪环基于文献[97]，并在此基础上设计了一种根据每个通道的载噪比自适应调整滤波器协方差阵的方法。在矢量跟踪环中，每个通道中的环路滤波器被一个卡尔曼滤波器取代，通常称作预滤波器（Pre-filter），如图 4-3 所示。鉴相器输出值作为预滤波器的观测值，预滤波器估计得到相应通道的伪距和多普勒频移，输送给导航滤波器进行全局导航解算，并估算出每个

通道的伪距和多普勒频移信息，反馈给相应通道的预滤波器。

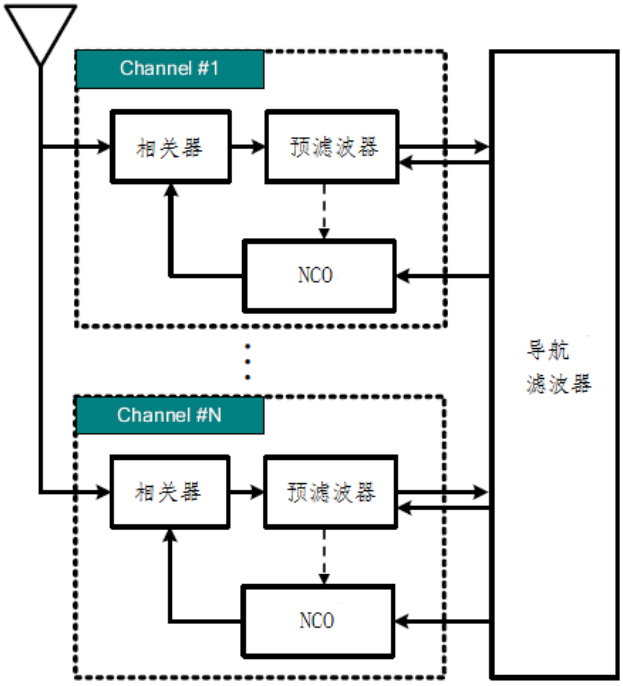

图 4-3 矢量跟踪环结构图

下面详细推导了矢量跟踪环算法。

4.2.1 预滤波器的建立

在矢量跟踪环中，引入了预滤波器来估计信号的伪距和伪距率（或多普勒频移）等原始观测信息。在接收机低动态情况下，即静止或以接近均速运动时，伪距和伪距率的的离散更新方程可表示为：

$$\begin{bmatrix}\rho_{k+1}\\ \dot{\rho}_{k+1}\end{bmatrix}=\begin{bmatrix}1 & \Delta t\\ 0 & 1\end{bmatrix}\begin{bmatrix}\rho_k\\ \dot{\rho}_k\end{bmatrix}+\begin{bmatrix}\Delta t^2/2\\ \Delta t\end{bmatrix}w_k^{\rho} \tag{4.1}$$

码相位和载波多普勒频移与伪距和伪距率存在以下关系：

$$\tau_k=\frac{1}{cT_c}\rho_k=\frac{1}{\lambda_\tau}\rho_k \tag{4.2}$$

$$f_D=f_k=-\frac{1}{\lambda_\varphi}\dot{\rho}_k \tag{4.3}$$

式中，c 为光速；T_c 为码片时间长度；λ_τ 为码片波长，对于 GPS C/A 码，取值为 293m，对于北斗 B1 则取值为 146.5m；λ_φ 为载波波长，对于 GPS C/A

码，取值为 0.19m，对于北斗 B1 频段载波则取值为 0.192m。

码相位误差动态模型可表示为：

$$\begin{aligned}&\hat{\tau}_{k+1}=\hat{\tau}_k+\hat{\dot{\tau}}_k\Delta t\\&\Delta\tau_k=\tau_k-\hat{\tau}_k\\&\Delta\tau_{k+1}=\Delta\tau_k+\frac{\dot{\rho}_k\Delta t}{\lambda_\tau}-\hat{\dot{\tau}}_k\Delta t+\frac{w_k\Delta t^2}{2\lambda_\tau}\end{aligned}\tag{4.4}$$

式中，τ_k 和 $\hat{\tau}_k$ 分别为真实码相位和码相位估计值。

同理，载波相位误差动态模型可表示为：

$$\begin{aligned}&\hat{\varphi}_{k+1}=\hat{\varphi}_k+2\pi\hat{f}_k\Delta t\\&\Delta\varphi_k=\varphi_k-\hat{\varphi}_k\\&\Delta\varphi_{k+1}=\Delta\varphi_k-\frac{2\pi}{\lambda_\varphi}\Delta t\dot{\rho}_k-2\pi\hat{f}_k\Delta t-\frac{\pi}{\lambda_\varphi}w_k\Delta t^2\end{aligned}\tag{4.5}$$

式中，φ_k 和 $\hat{\varphi}_k$ 分别为真实载波相位和载波相位估计值。

综合上述模型，就能得到 GNSS 信号跟踪的离散状态空间模型：

$$\boldsymbol{x}_{k+1}=\begin{bmatrix}\delta\varphi_k\\\delta\tau_k\\\dot{\rho}_k\end{bmatrix}=\begin{bmatrix}1&0&-\frac{2\pi\Delta t}{\lambda_\varphi}\\0&1&\frac{\Delta t}{\lambda_\tau}\\0&0&1\end{bmatrix}\boldsymbol{x}_k+\begin{bmatrix}-2\pi\Delta t&0\\0&-\Delta t\\0&0\end{bmatrix}\boldsymbol{u}_k+\begin{bmatrix}-\frac{\pi\Delta t^2}{\lambda_\varphi}\\\frac{\Delta t^2}{2\lambda_\tau}\\\Delta t\end{bmatrix}\boldsymbol{w}_k\tag{4.6}$$

式中，$\boldsymbol{x}_k=\begin{bmatrix}\delta\varphi_k&\delta\tau_k&\dot{\rho}_k\end{bmatrix}^T$ 是系统状态矢量，$\boldsymbol{u}_k=\begin{bmatrix}\hat{f}_k^{NCO}&\hat{\dot{\tau}}_k^{NCO}\end{bmatrix}^T$ 为系统反馈量，即输入 NCO 的控制量，$\boldsymbol{w}_k$ 为系统过程噪声。仿真过程中，系统噪声协方差矩阵 $\boldsymbol{Q}_k=diag\{6.7\,\mathrm{e}{-5},2.9\,\mathrm{e}{-11},2.5\,\mathrm{e}{-3}\}$。

量测方程利用 PLL、DLL 和 FLL 的鉴相器的输出作为观测量，具体如下：

$$\boldsymbol{z}_k=\begin{bmatrix}\Delta\varphi_k^{PLL}\\\Delta\tau_k^{DLL}\\\Delta f_k^{FLL}\end{bmatrix}=\boldsymbol{H}\boldsymbol{x}_k+\boldsymbol{D}\boldsymbol{u}_k+\boldsymbol{\eta}_k=\begin{bmatrix}1&0&\frac{\pi\Delta t}{\lambda_\varphi}\\0&1&-\frac{\Delta t}{2\lambda_\tau}\\0&0&-\frac{1}{\lambda_\varphi}\end{bmatrix}\boldsymbol{x}_k+\begin{bmatrix}\pi\Delta t&0\\0&\frac{\Delta t}{2}\\-1&0\end{bmatrix}\boldsymbol{u}_k+\begin{bmatrix}\eta_{PLL}\\\eta_{DLL}\\\eta_{FLL}\end{bmatrix}\tag{4.7}$$

式中，$\Delta\varphi_k^{PLL}$、$\Delta\tau_k^{DLL}$和Δf_k^{FLL}分别为 PLL、DLL 和 FLL 的鉴相器的输出值。η_{PLL}、η_{DLL}、η_{FLL}分别为 PLL、DLL 和 FLL 的输出噪声，标准差分别为σ_θ、σ_τ和σ_f，具体计算方法见下一小节（4.2.2）。

4.2.2 协方差阵的自适应调整方法

文献[84, 85, 88]指出可利用每个通道的 C/N_0 可估算出通道跟踪环路噪声，PLL、DLL 和 FLL 输出噪声标准差可表示为：

$$\sigma_\theta = \sqrt{\frac{1}{T\cdot C/N_0}\left(1+\frac{1}{2T\cdot C/N_0}\right)} \ \text{(rad)} \tag{4.8}$$

$$\sigma_\tau = \sqrt{\frac{D}{2T\cdot C/N_0}\left(1+\frac{1}{T\cdot C/N_0\left(1-0.5D\right)}\right)} \ \text{(chips)} \tag{4.9}$$

$$\sigma_f = \sqrt{\frac{1}{\pi^2T^3\cdot C/N_0}\left(1+\frac{1}{T\cdot C/N_0}\right)} \ \text{(Hz)} \tag{4.10}$$

式中，σ_θ、σ_τ和σ_f分别为 PLL、DLL 和 FLL 输出噪声标准差，T 为相关器的积分时间，实际的算法中，T 设为 20ms。

于是根据式（5.8）~式（5.10）设计了自适应预滤波器，即预滤波器观测量协方差阵 $\boldsymbol{R}$ 可表示为：

$$\boldsymbol{R}=\begin{bmatrix}\sigma_\theta^2 & 0 & 0\\ 0 & \sigma_\tau^2 & 0\\ 0 & 0 & \sigma_f^2\end{bmatrix} \tag{4.11}$$

根据 GNSS 接收机每个通道的载噪比（C/N_0）的大小估计对应子滤波器观测量噪声大小，从而自适应调节每个预滤波器的协方差阵，优化组合滤波性能。

4.3 组合导航滤波器研究

本章设计的 MEMS-DR 超紧耦合的组合导航滤波基于联邦无迹卡尔曼滤波器（FUKF）。即每个通道的信息输送至联邦滤波器的对应子滤波器，子滤波器由 UKF 实现。下面详细推导了基于联邦无迹卡尔曼滤波器（FUKF）的非线

性状态方程和量测方程。

4.3.1 非线性离散状态方程的建立

每个子滤波器具有相同的状态方程，包括 DR 系统的误差模型和 GNSS 接收机时钟误差模型。

以东北天地理坐标系作为导航坐标系，DR 系统误差主要包括导航坐标系下的位置误差 $\delta \boldsymbol{r}^n = [\delta r_e, \delta r_n, \delta r_u]^T$、导航坐标系下的速度误差 $\delta \boldsymbol{v}^n = [\delta v_e, \delta v_n, \delta v_u]^T$。具体模型如下：

$$\begin{cases} \delta r_{e,k+1} = \delta r_{e,k} + \delta v_{e,k}\Delta t + \frac{1}{2}\omega_v \Delta t^2 \\ \delta r_{n,k+1} = \delta r_{n,k} + \delta v_{n,k}\Delta t + \frac{1}{2}\omega_v \Delta t^2 \\ \delta r_{u,k+1} = \delta r_{u,k} + \delta v_{u,k}\Delta t + \frac{1}{2}\omega_v \Delta t^2 \end{cases} \tag{4.12}$$

$$\begin{cases} \delta v_{e,k+1} = \delta v_{e,k} + \omega_v \Delta t \\ \delta v_{n,k+1} = \delta v_{n,k} + \omega_v \Delta t \\ \delta v_{u,k+1} = \delta v_{u,k} + \omega_v \Delta t \end{cases} \tag{4.13}$$

式中，ω_v DR 系统的速度噪声，Δt 为 k 时刻到 k+1 时刻的时间间隔。

GNSS 接收机时钟误差主要有时钟零偏 δb（单位为 m）和时钟漂移 δd（单位为 m/s），具体模型如下：

$$\begin{cases} \delta b_{k+1} = \delta b_k + \delta d_k \Delta t + \frac{1}{2}\omega_t \Delta t^2 \\ \delta d_{k+1} = \delta d_k + \omega_t \Delta t \end{cases} \tag{4.14}$$

式中，ω_t 为 GNSS 接收机的时钟漂移噪声。

因此状态变量选取为：$\boldsymbol{x} = [\delta r_e, \delta r_n, \delta r_u, \delta v_e, \delta v_n, \delta v_u, \delta b, \delta d]^T$。于是 GNSS/DR 组合导航滤波器的非线性离散状态方程可表示为：

$$\boldsymbol{x}_{k+1}=\begin{bmatrix}\delta r_{e,k+1}\\ \delta r_{n,k+1}\\ \delta r_{u,k+1}\\ \delta v_{e,k+1}\\ \delta v_{n,k+1}\\ \delta v_{u,k+1}\\ \delta b_{k+1}\\ \delta d_{k+1}\end{bmatrix}=f\left(\boldsymbol{x}_k,\boldsymbol{u}_k\right)+\boldsymbol{w}_k=\begin{bmatrix}\delta r_{e,k}+\delta v_{e,k}\Delta t\\ \delta r_{n,k}+\delta v_{n,k}\Delta t\\ \delta r_{u,k}+\delta v_{u,k}\Delta t\\ \delta v_{e,k}\\ \delta v_{n,k}\\ \delta v_{u,k}\\ \delta b_k+\delta d_k\Delta t\\ \delta d_k\end{bmatrix}+\begin{bmatrix}\frac{1}{2}\omega_v\Delta t^2\\ \frac{1}{2}\omega_v\Delta t^2\\ \frac{1}{2}\omega_v\Delta t^2\\ \omega_v\Delta t\\ \omega_v\Delta t\\ \omega_v\Delta t\\ \frac{1}{2}\omega_t\Delta t^2\\ \omega_t\Delta t\end{bmatrix} \tag{4.15}$$

式中，ω_v，ω_θ 和 ω_t 分别为 DR 系统的速度噪声、航向角噪声和 GNSS 接收机的时钟漂移噪声，仿真过程中分别取为 2，0.17 和 2。Δt 为 k 时刻到 k+1 时刻的时间间隔，仿真过程取值为 0.1，即组合导航算法更新率为 10Hz。于是，系统噪声协方差矩阵可表示为$\boldsymbol{Q}_k=E\{\boldsymbol{w}_k\boldsymbol{w}_k^T\}$。

4.3.2 非线性量测方程的建立

每个子滤波器接收对应通道的伪距和伪距率信息,对应子滤波器的观测量，即

$$\boldsymbol{z}_k^j=\begin{bmatrix}\rho^j\\ \dot{\rho}^j\end{bmatrix} \tag{4.16}$$

式中，ρ^j 和 $\dot{\rho}^j$ 分别为 j 号卫星的伪距和伪距率，其中，ρ^j 为经过卫星时钟误差、对流层延时误差和电离层延时误差等误差修正后的伪距值。

根据文献[44, 85]中指出，j 号卫星与 GNSS 接收机之间的伪距率可表示为：

$$\tilde{\rho}^j=r^j+c\delta t_r-c\delta t_s+cI^j+cT^j+\tilde{\varepsilon}_\rho^j \tag{4.17}$$

式中：

$\tilde{\rho}^j$ 为未经误差修正的伪距值，单位为 m；

r^j 为 j 号卫星与 GNSS 接收机天线之间的真实距离，单位为 m；

c 为光速，单位为 m/s；

δt_r 为接收机时钟零偏，单位为 s；

δt_s 为卫星时钟零偏，单位为 s；

I^j 为电离层传播延时误差，单位为 s；

T^j 为对流层传播延时误差，单位为 s；

$\tilde{\varepsilon}_\rho^j$ 为残留误差，包括建模误差、环路噪声和多径干扰误差等，单位为 m。

卫星时钟零偏、电离层延时误差、对流层延时误差可以根据相应模型进行修正，因此修正后的伪距可表示为：

$$\rho^j = r^j + c\delta t_r + \tilde{\varepsilon}_\rho^j = \left\| \boldsymbol{r}^e - \boldsymbol{r}_s^j \right\| + \delta b + \tilde{\varepsilon}_\rho^j \tag{4.18}$$

式中，$\tilde{\varepsilon}_\rho^j$ 表示所有残留误差。$\boldsymbol{r}_s^j$ 为 j 号卫星在地球坐标下的位置，$\boldsymbol{r}^e$ 为 GNSS 接收机在地球坐标系下的真实位置，可由以下表达式计算得到：

$$\boldsymbol{r}^e = \hat{\boldsymbol{r}}^e + \boldsymbol{C}_n^e \delta \boldsymbol{r}^n \tag{4.19}$$

式中，$\hat{\boldsymbol{r}}^e$ 为由 DR 输出的地球坐标系下的位置。

同理，修正后的伪距率可表示为：

$$\dot{\rho}^j = \left(\boldsymbol{v}^e - \boldsymbol{v}_s^j \right) \cdot \boldsymbol{l}^j + \delta d + \dot{\varepsilon}_\rho^j \tag{4.20}$$

式中，$\boldsymbol{v}^e$ 为 GNSS 接收机在地球坐标系下真实速度，可由以下表达式计算得到：

$$\boldsymbol{v}^e = \hat{\boldsymbol{v}}^e + \boldsymbol{C}_n^e \delta \boldsymbol{v}^n \tag{4.21}$$

式中，$\hat{\boldsymbol{v}}^e$ 为由 DR 输出的地理坐标系下的速度。

综上所述，组合导航滤波器的非线性量测方程为：

$$\boldsymbol{z}_k^j = \begin{bmatrix} \rho^j \\ \dot{\rho}^j \end{bmatrix} = \boldsymbol{h}\left(\boldsymbol{x}_k \right) + \boldsymbol{\eta}_k = \begin{bmatrix} \left\| \boldsymbol{r}^e - \boldsymbol{r}_s^j \right\| + \delta b \\ \left(\boldsymbol{v}^e - \boldsymbol{v}_s^j \right) \cdot \boldsymbol{l}^j + \delta d \end{bmatrix} + \begin{bmatrix} \varepsilon_\rho^j \\ \dot{\varepsilon}_\rho^j \end{bmatrix} \tag{4.22}$$

每个子滤波器观测量协方差阵 $\boldsymbol{R}_k$ 由预滤波器传递过来。

4.4 跟踪环路反馈算法研究

导航滤波器估计得到全局最优解后，估算每个通道的伪距、多普勒频移信

息，反馈给相应通道的预滤波器。每个通道的反馈结构如图 4-4 所示。

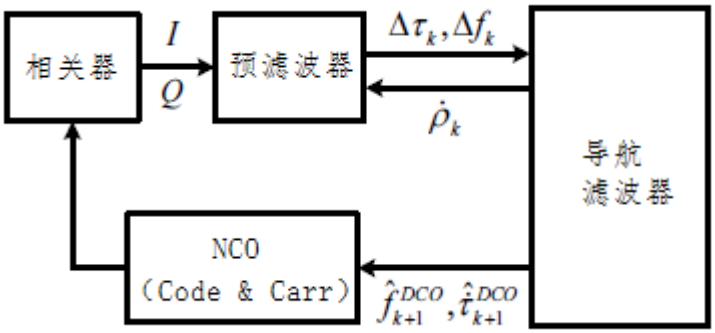

图 4-4 通道反馈结构

根据前面的推导过程可以得到载波 NCO 和码 NCO 的控制输入量为：

$$\hat{f}_k^{NCO} = -\frac{\hat{\dot{\rho}}_k}{\lambda_\varphi} + \frac{\Delta\hat{\varphi}_k}{2\pi\Delta t} \tag{4.23}$$

$$\hat{\tau}_k^{NCO} = \frac{\hat{\dot{\rho}}_k}{\lambda_\tau} + \frac{\Delta\hat{\tau}_k}{\Delta t} \tag{4.24}$$

式中，$\hat{f}_k^{NCO}$ 和 $\hat{\tau}_k^{NCO}$ 分别为载波 NCO 和码 NCO 的控制输入量。

4.5 试验结果与分析

为验证超紧耦合算法的有效性和优越性，分别从定位精度、抗干扰能力和重新捕获时间等方面进行了 3 组对比试验。

4.5.1 定位精度试验

为了比较本文设计的矢量跟踪环（Vector Tracking Loop，VLL）和传统的标量跟踪环（Scalar Tracking Loop, SLL）算法的定位精度，首先进行了静止定位实验。

实验设备主要包括 GNSS 中频信号采集器、GNSS 天线和笔记本电脑。GNSS 中频信号采集器采用了星源北斗公司的型号为HG-SOFTGPS02 的中频信号采集器，实物图和特性分别见图 4-5 和表 4-1，用于采集 GNSS 卫星信号经 A/D 采样后数字中频信号，并通过 USB 2.0 口传输至笔记本电脑。

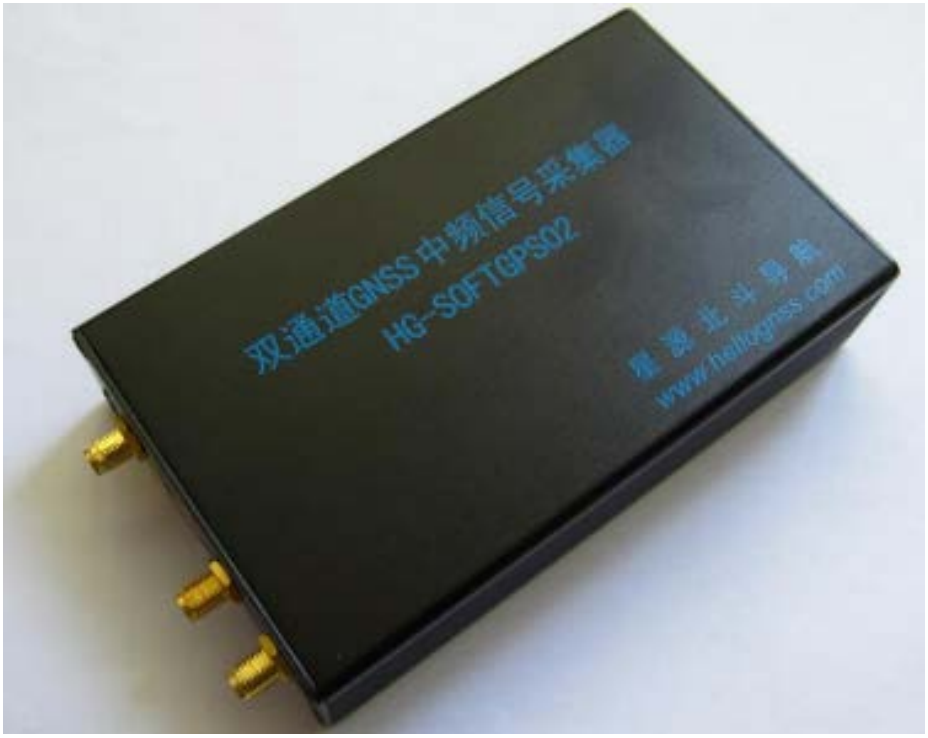

图 4-5 中频信号采集器实物图

表 4-1 中频信号采集器特性

组成部分	特性
射频芯片	MAX2769
TCXO 频率	16.368MHz
采样位数	2 位
采样速率	16.368MHz
信号中心频率	4.092MHz
支持系统	GPS L1、BD2 B1

实验地点选在浙江大学玉泉校区周亦卿大楼前的草坪上，场地开阔，可见卫星多，卫星信号基本不受遮挡，卫星信号载噪比基本维持在 45dB-Hz 左右。试验场景如图 4-6 所示。

图 4-6 GNSS 静止导航试验场景

实验过程中，利用笔记本电脑采集 GNSS 中频信号采集器的信息，并在 MATLAB 中进行仿真后处理，分别利用矢量跟踪环（Vector Tracking Loop，VLL）和标量跟踪环（Scalar Tracking Loop, SLL）进行导航定位，分析导航定位误差。导航误差如图 4-7 所示。

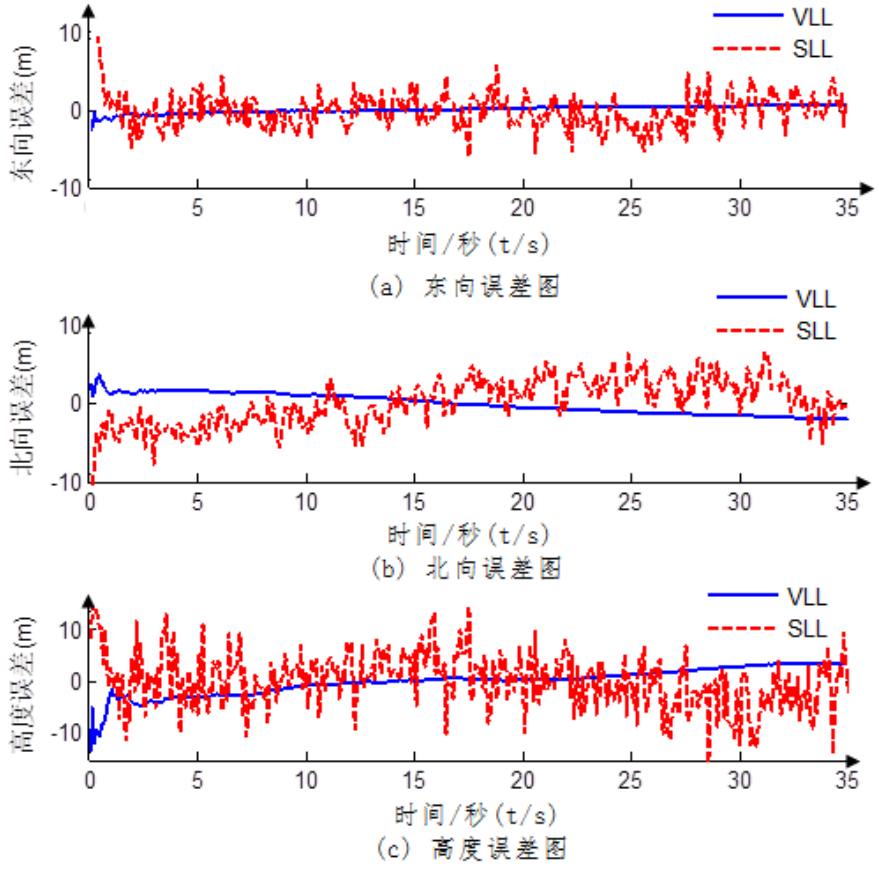

图 4-7 VLL 与 SLL 在静止情况下的定位误差

从图中可以看出，矢量跟踪环的定位精度约为 5m，而标量跟踪环的定位精度约为 10m。（注：这里的定位精度不是绝对定位精度，而是相对定位精度，即在一定时间内的相对于平均位置的误差）。由此可以看出采用矢量跟踪环后，定位精度得到大幅度提升，原因在于环路中引入了卡尔曼滤波器，环路噪声得到有效抑制。

4.5.2 抗干扰能力试验

为验证矢量跟踪环对弱信号的跟踪性能，进行了信号遮挡实验，并与标量跟踪环进行比较。

实验设备主要包括 GNSS 中频信号采集器、GNSS 天线、笔记本电脑和厚度约为 0.5mm 的金属盖。实验场景如图 4-8 所示。

图 4-8 矢量跟踪环实验场景

在 GNSS 接收卫星信号过程中，人为地用金属盖将天线盖住，遮挡时间为 60s，分别进行标量跟踪环和矢量跟踪环对 GNSS 卫星信号进行跟踪，比较并分析跟踪结果。仿真过程采用 6 个跟踪通道，各个通道的跟踪结果如图 4-9 所示。

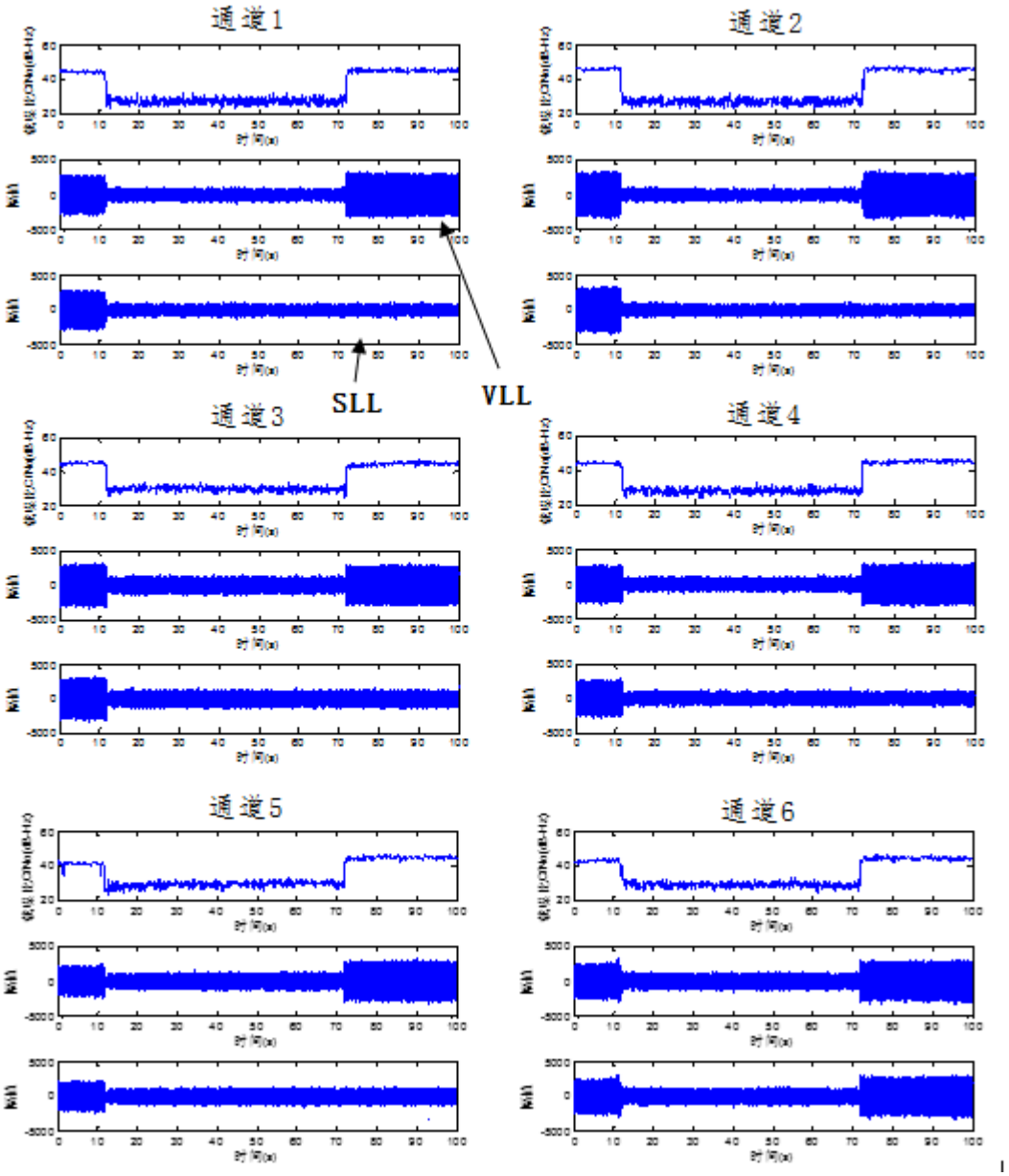

图 4-9 矢量跟踪环对弱信号的跟踪结果

从跟踪结果中可以看出，当没有金属盖遮挡时，所有跟踪通道的 C/N_0 均大于 40dB-Hz。在 13s~73s 之间，信号受到严重遮挡，通道 C/N_0 约为 25dB-Hz，此时通道 1、3、5 和 6 没有失锁；在标量跟踪环路中，只有通道 6 没有失锁。当移去金属盖遮挡后，矢量跟踪环的所有通道又重新锁定信号，而标量跟踪环只有通道 6 锁定信号。

因此，可以看出：与传统标量跟踪环算法相比，矢量跟踪环算法对弱信号具有更强的跟踪能力。同时在矢量跟踪环中，失锁的通道仍然保持跟踪状态，由其他通道一旦卫星信号重新出现，失锁的通道会迅速锁定卫星信号。

4.5.3 穿越隧道实验

为了验证 GNSS/MEMS-DR 超紧耦合组合导航算法的性能(包括导航精度、重捕时间等），进行了穿越隧道的导航试验。

4.5.3.1 实验设备与条件

实验装置实物图如图 4-10 所示。实验装置包括 GNSS 中频信号采集器、MEMS-DR 系统、GNSS 天线、笔记本电脑等组成，整个导航装置安装于山地自行车上。

其中，MEMS-DR 系统采用的是第 2 章中设计的基于多传感器信息融合的 DR 系统，输出位置、速度和姿态信息，前面已经验证导航精度优于总航程的 1%；GNSS 中频信号采集器采用了星源北斗公司的型号为 HG-SOFTGPS02 的中频信号采集器，用于采集 GNSS 卫星信号经 A/D 采样后数字中频信号，并通过 USB 2.0 口传输至笔记本电脑；为增强信号的接收，GNSS 安装在头顶的帽子上，具体如图 4-10 所示。

图 4-10 GNSS/MEMS-DR 超紧耦合实验装置实物图

实验中所用到的组件的型号与特性具体如表 4-2 所示：

表 4-2 超紧耦合试验装置组成部分特性

组成部分	型号	特性
中频信号采集器	星源北斗 HG-SOFTGPS02	详见表 4-1
陀螺	L3GD20	标定后零偏：0.05°/s
加速度计	LSM303DLHC	标定后零偏：0.1m/s^2
磁力计	LSM303DLHC	噪声标准差：0.01G
里程计	ROCKYOU	相对速度误差小于 0.5%
笔记本电脑	联想 Yoga	采用 intel i3 处理器

实验过程中，骑山地自行车进行导航，路线沿着梅灵北路，总里程约为 2km，历时约 8min，途经长度为 1.2km 的梅灵隧道。笔记本电脑采集各传感器的信息，并在 MATLAB 中进行仿真后处理，分别采用本章提出的超紧耦合算法和第 3 章提出的紧耦合算法进行导航解算，分析并比较导航结果，导航结果在 Google 地图中显示。

4.5.3.2 实验数据处理方法

数据采集完成后，在 MATLAB 中进行仿真后处理。本章提出的 GNSS/MEMS-DR 超紧耦合算法按如下步骤执行：

1) 采集里程计、加速度计、陀螺仪和磁力计等传感器的信息，经过 DR 算法计算得到车辆位置和速度等信息。DR 算法详细解算过程见第 2 章。
2) 采集 GNSS 数字中频信号，与本地产生的载波和测距码进行相关运算，得到 PLL、DLL 和 FLL 等鉴相器输出，作为预滤波器的观测量。
3) 计算每个通道信号的载噪比，估算通道噪声，由此自适应调整预滤波器观测量协方差阵，提高滤波性能。
4) 利用预滤波器估算得到对应通道的多普勒频移和码相位等信息，进而得到伪距和伪距率信息，作为导航滤波器的观测量。
5) 导航滤波器融合 DR 系统和 GNSS 的信息，估算得到最终的导航结果；
6) 利用导航滤波器得到的导航信息，估算出每个通道信号的多普勒频移和码相位信息，对通道的载波 NCO 和码 NCO 进行反馈修正。
7) 如果数据融合中可见卫星数大于 0，则反馈修正 DR 误差。

具体流程如图 4-11 所示。

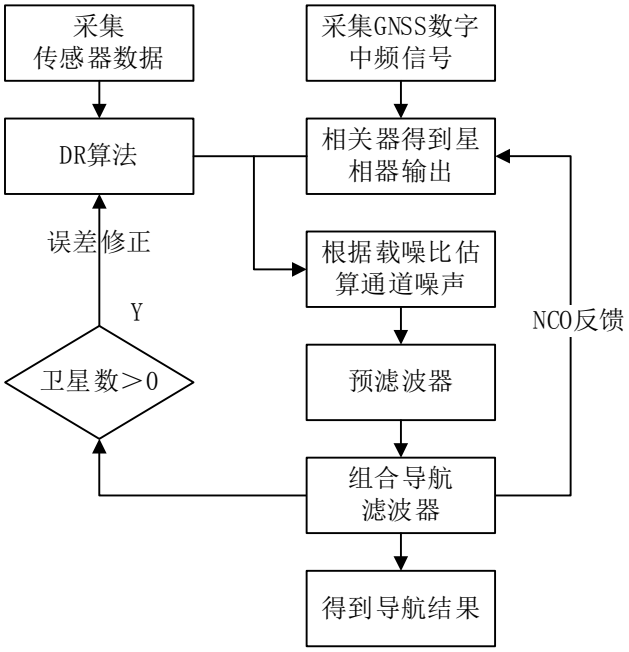

图 4-11 超紧耦合算法流程图

4.5.3.3 结果与分析

分别采用本章提出的超紧耦合算法和第 3 章提出的紧耦合算法进行导航解算，结果如图 4-12 和图 4-13 所示，其中图 4-13 为图 4-12 在隧道出口处的局部放大图。

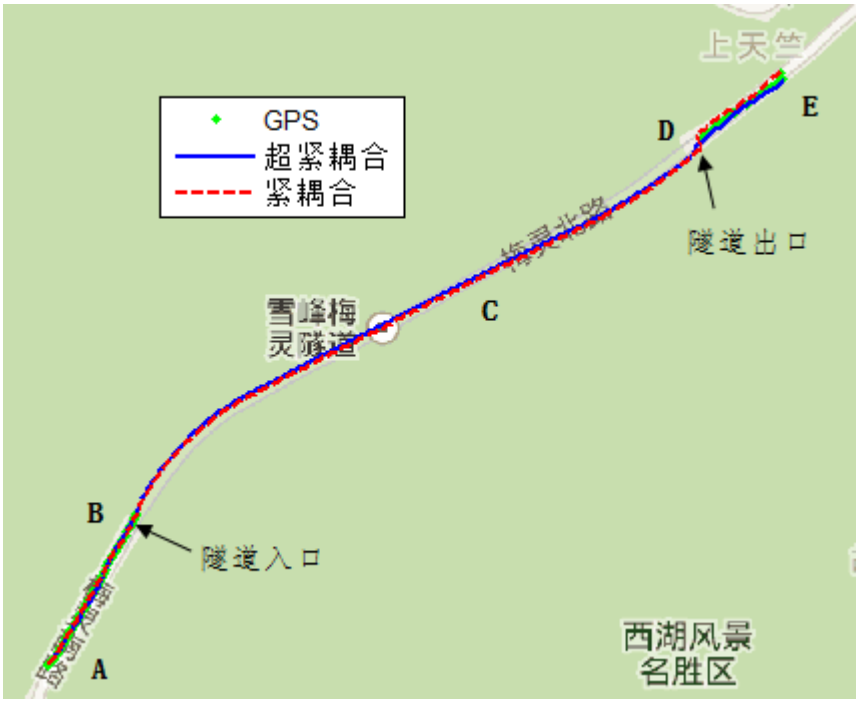

图 4-12 超紧耦合和紧耦合导航轨迹

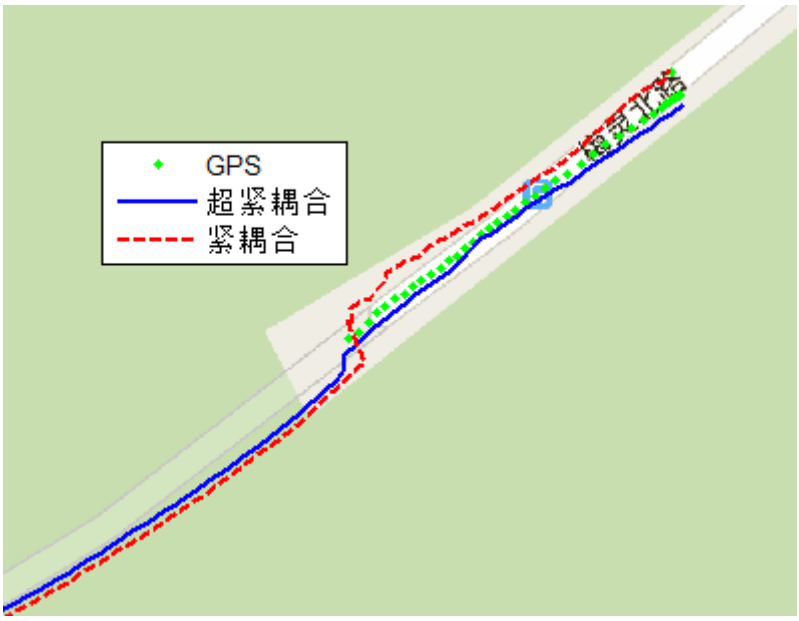

图 4-13 导航轨迹在隧道出口处的局部放大图

从图 4-12 和图 4-13 中可以看出，超紧耦合的导航轨迹基本与地图上的道路基本重合，最大误差小于 15m；而紧耦合最大误差在隧道出口处，约为 20m。

进一步从轨迹中的不同路段中选取了如图 4-12 中的 A、B、C、D、E 等 5 个点，以型号为 GARMIN GPS15xL 的 GPS 作为参考基准，分析导航误差，具体如图 4-14 和表 4-3 所示，可以看出，超紧耦合比紧耦合具有更高的导航精度。

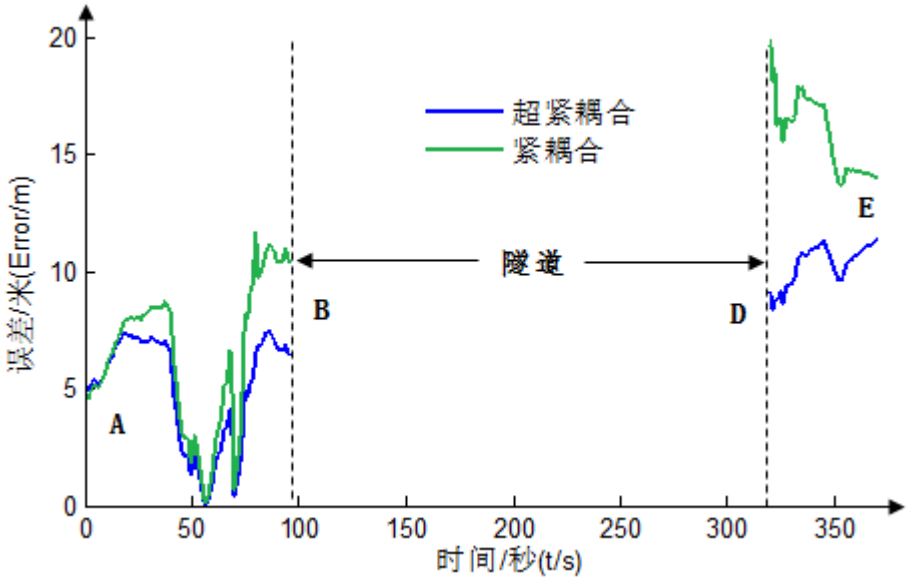

图 4-14 超紧耦合与紧耦合导航误差曲线

表 4-3 路线中特定点的导航误差

位置	时刻(s)	紧耦合误差(m)	超紧耦合误差(m)
A	0	5	5
B	95	12	7
C	210	16*	11*
D	320	20	9
E	370	15	11

*注：由于在隧道中没有 GPS 信号，因此导航误差以 GOOGLE 地图为基准。

在隧道中，GNSS 信号完全受到遮挡，只能进行航位推算。在隧道出口处，GNSS 信号重新有效，由于在超紧耦合算法中，航位推算系统能够辅助 GNSS 接收机对信号的捕获，利用航位推算系统提供的位置和速度信息作为先验知识，推算出每个通道的载波频率和码相位，从而快速捕获卫星信号。从图 4-13 可以看出，超紧耦合算法比紧耦合算法提前重新捕获卫星信号，提前时间大约为 1s。

从穿隧道实验中可以看出，与紧耦合算法相比，超紧耦合具有更高的导航精度，并且当信号失锁后，能够更快重新捕获卫星信号。

4.6 本章小结

本章研究了 GNSS/MEMS-DR 超紧耦合算法，设计并实现了一种基于矢量跟踪环和 FUKF 的超紧耦合算法。通过实验验证了超紧耦合算法相对其他组合导航的优越性。超紧耦合算法的的优势主要体现在以下三个方面：

1) 在跟踪环路中引入卡尔曼滤波器后，对伪距和多普勒频移等原始信息的估计准确，从而提高导航精度。
2) 对于弱信号通道，能够利用全局的位置和速度信息推断出信号的载波频率和测距码相位，从而锁定微弱信号，增强对弱信号的跟踪能力。
3) 同时，即使有通道失锁时，允许该通道不退出跟踪状态，由其他通道的信息来推断信号的载波频率和测距码相位，保持跟踪节奏。当信号再次有效时，该通道能够迅速地锁定信号，不需要进行重新捕获。

第5章 总结与展望

5.1 超紧耦合算法研究总结

本文针对 GNSS 接收机在弱信号或高动态情况下容易失锁的问题，本文研究了 GNSS 与 MEMS-DR 的超紧耦合组合导航算法，并着重研究了 MEMS-DR 辅助 GNSS 捕获和跟踪的算法，以及基于联邦无迹卡尔曼滤波器的数据融合算法，提高组合导航系统的抗干扰能力和导航精度。同时针对 MEMS 器件噪声大的问题，研究了抑制 MEMS-DR 系统累积误差的方法，提高 MEMS-DR 在单独导航时的导航精度。

主要取得了以下研究成果：

1. 针对 MEMS 惯性器件噪声大的特点，提出了一种基于多传感器信息融合的 MEMS-DR 算法。实验结果表明提出的 MEMS-DR 系统导航精度优于总航程的 1%，有效抑制了 MEMS-DR 系统在单独导航时的累积误差，以低成本低精度的传感器实现了高精度导航。

2. 深入研究了 GNSS 接收机的内部机理，实现了矢量跟踪环算法，并在此基础上有效实现了利用 MEMS-DR 的信息来辅助 GNSS 接收机对卫星信号的跟踪，提高了对弱信号的跟踪能力。

4. 最后实现了一种基于矢量跟踪环和联邦无迹卡尔曼滤波器的 GNSS/MEMS-DR 超紧耦合算法。通过不同场景的实验验证了超紧耦合算法相对其他组合导航的优越性。超紧耦合算法的的优势主要体现在以下三个方面：

1) 在跟踪环路中引入卡尔曼滤波器后，对伪距和多普勒频移等原始信息的估计更加准确，从而提高导航精度。
2) 对于弱信号通道，能够利用全局的位置和速度信息推断出信号的载波频率和测距码相位，从而锁定微弱信号，增强对弱信号的跟踪能力。
3) 同时，即使有通道失锁时，允许该通道不退出跟踪状态，由其他通道的信息来推断信号的载波频率和测距码相位，保持跟踪节奏。当信号再次有效时，该通道能够迅速地锁定信号，不需要进行重新捕获。

5.2 超紧耦合算法研究展望

GNSS/MEMS-DR 的超紧耦合能有效地提高组合导航系统在强干扰和高动态情况下的导航精度、提高整体的鲁棒性，从而实现在海洋中的长时间高精度导航。

超紧耦合的思想不仅仅局限于组合导航中，也可以应用于多传感器数据融合的场合（如无人驾驶汽车等），从而提高信息融合的效果以及鲁棒性。

本文对超紧耦合算法的研究时间有限，还存在许多不足的地方，还有很多内容需进一步研究：

1) 目前实现的超紧耦合处于半仿真阶段，即采集实际的 MEMS 传感器数据和 GNSS 数字中频信号，然后在 MATLAB 中进行仿真后处理，得到导航结果。下一步要做的就是在 FPGA/DSP/ARM 中实现超紧耦合，实现实时导航。
2) 本文主要研究了超紧耦合在弱信号情况下的导航性能，由于条件受限，未对超紧耦合的高动态性能进行研究。可进一步研究超紧耦合在高动态情况下的应用。
3) 研究差分 GNSS 以及载波相位跟踪原理，实现差分 GNSS 与 INS 的超紧耦合，进一步提高导航精度。

参考文献

[1] 秦永元, 张洪钺, 汪叔华. 卡尔曼滤波与组合导航原理[M]. 西北工业大学出版社, 1998.

[2] 维基百科. 惯性导航系统[EB/OL]. http://zh.wikipedia.org/wiki/惯性导航系统.

[3] 陈哲. 捷联惯导系统原理[M]. 宇航出版社, 1986.

[4] Titterton D. Strapdown inertial navigation technology[M]. IET, 2004.

[5] Gade K. Introduction to inertial navigation and Kalman filtering[R]. 2008.

[6] King A D. Inertial navigation-forty years of evolution[J]. GEC review, 1998, 13(3): 140-149.

[7] 刘慧, 王丰乐. INS/DVL 组合导航系统在水下载体中运用研究[J]. 青岛大学学报(工程技术版); JOURNAL OF QINGDAO UNIVERSITY(ENGINEERING & TECHNOLOGY EDITION), 2002, 17(1): 4.

[8] 黄汛, 高启孝, 李安, 等. INS/GPS 超紧耦合技术研究现状及展望[J]. 飞航导弹, 2009(4): 42-47.

[9] Gao G, Lachapelle G. A novel architecture for ultra-tight HSGPS-INS inte-gration[J]. Journal of Global Positioning Systems, 2008,7(1):46-61.

[10]任江涛. 北斗接收机基带信号的捕获算法研究[D]. 合肥工业大学信号与信息处理, 2010.

[11]维基百科. 卫星导航系统[EB/OL]. http://zh.wikipedia.org/wiki/卫星导航系统.

[12]中国卫星导航系统管理办公室. 北斗卫星导航系统空间信号接口控制文件 2.0 版[S]. 2013.

[13]Copps E M, Geier G J, Fidler W C, et al. OPTIMAL PROCESSING OF GPS SIGNALS[J]. NAVIGATION, 1980(Vol. 27 No. 3): 171-182.

[14]Gustafson D, Dowdle J, Flueckiger K. A deeply integrated adaptive GPS-based navigator with extended range code tracking: Position Location and Navigation Symposium, IEEE 2000, 2000[C].

[15]Gautier J D. GPS/INS generalized evaluation tool (GIGET) for the design and testing of integrated navigation systems[D]. Stanford University, 2003.

[16]Abbott A S. Global positioning systems and inertial measuring unit ultratight coupling method:

[17]Dah-Jing J, Fong-Chi C. Fuzzy Adaptive Unscented Kalman Filter for Ultra-Tight GPS/INS Integration: Computational Intelligence and Design (ISCID), 2010 International Symposium on, 2010[C].

[18]Gannan Y, Tao Z. Unscented Kalman filtering for ultra-tightly coupled GPS/INS integration: Mechatronics and Automation, 2009. ICMA 2009. International Conference on, 2009[C].

[19]胡锐. 惯性辅助 GPS 深组合导航系统研究与实现[D]. 中国, 2010.

[20]张涛. GPS/SINS 超紧密组合导航系统的关键技术研究[D]. 中国, 2010.

[21]Kalman R E. A new approach to linear filtering and prediction problems[J]. Journal of basic Engineering, 1960, 82(1): 35-45.

[22]Vathsal S. SPACE ATTITUDE DETERMINATION USING A SECOND-ORDER NONLINEAR FILTER.[J]. Journal of Guidance, Control, and Dynamics, 1987, 10(6): 559-566.

[23]Hassan M F, Salut G, Singh M G, et al. DECENTRALIZED COMPUTATIONAL ALGORITHM FOR THE GLOBAL KALMAN FILTER.[J]. IEEE Transactions on Automatic Control, 1978, AC-23(2): 262-268.

[24]Zhu Y, You Z, Zhao J, et al. The optimality for the distributed Kalman filtering fusion with feedback[J]. Automatica, 2001, 37(9): 1489-1493.

[25]Carlson N A. Federated square root filter for decentralized parallel processors [J]. Aerospace and Electronic Systems, IEEE Transactions on, 1990, 26(3): 517-525.

[26]岳晓奎, 袁建平, Xiaokui Yue, 等. 一种基于极大似然准则的自适应卡尔曼滤波算法[J]. 西北工业大学学报; JOURNAL OF NORTHWESTERN POLYTECHNICAL UNIVERSITY, 2005, 23(4): 6.

[27]SAGE A P, HUSA G W. ADAPTIVE FILTERING WITH UNKNOWN PRIOR STATISTICS[J]. 1969: 760-769.

[28]耿延睿, 崔中兴, 张洪钺, 等. 衰减因子自适应滤波及在组合导航中的应用[J]. 北京航空航天大学学报; JOURNAL OF BEIJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS, 2004, 30(5): 4.

[29]Hide C, Moore T, Smith M. Adaptive Kalman filtering algorithms for integrating GPS and low cost INS: PLANS - 2004 Position Location and Navigation Symposium, April 26, 2004 - April 29, 2004, Monterey, CA, United states, 2004[C]. Institute of Electrical and Electronics Engineers Inc..

[30]Yoshimura T, Soeda T. A technique for compensating the filter performance by a fictitious noise[J]. Transactions of the ASME. Journal of Dynamic Systems, Measurement and Control, 1978, 100(2): 154-156.

[31]尚捷, 顾启泰, Jie SHANG, 等. MIMS/GPS 组合导航系统中卡尔曼滤波器设计与实验研究[J]. 中国惯性技术学报; JOURNAL OF CHINESE INERTIAL TECHNOLOGY, 2005, 13(2): 5.

[32]Julier S J, Uhlmann J K, Durrant-Whyte H F. A New approach for filtering nonlinear systems: Proceedings of the 1995 American Control Conference. Part 1 (of 6), June 21, 1995 - June 23, 1995, Seattle, WA, USA, 1995[C].

[33]Julier S, Uhlmann J, Durrant-Whyte H F. New method for the nonlinear transformation of means and covariances in filters and estimators[J]. IEEE Transactions on Automatic Control, 2000,45(3):477-482.

[34]Julier S J, Uhlmann J K. Corrections to "Unscented filtering and nonlinear estimation": Deep Space Kas Band Link Management and Mars Reconnaissance Orbiter, 2004[C]. Institute of Electrical and Electronics Engineers Inc..

[35]Van Merwe R D, Wan E A, Julier S I. Sigma-point kalman filters for nonlinear estimation and sensor-fusion - Applications to integrated navigation: Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference, August 16, 2004 - August 19, 2004, Providence, RI, United states, 2004[C]. American Institute of Aeronautics and Astronautics Inc..

[36]Crassidis J L. Sigma-point Kalman filtering for integrated GPS and inertial navigation: AIAA Guidance, Navigation, and Control Conference 2005, August 15, 2005 - August 18, 2005, San Francisco, CA, United states, 2005[C]. American Institute of Aeronautics and Astronautics Inc..

[37]Crassidis J L, Markley F L. Unscented filtering for spacecraft attitude estimation[J]. Journal of Guidance, Control, and Dynamics, 2003, 26(4): 536-542.

[38]Upadhyaya B R, Sorenson H W. BAYESIAN DISCRIMINANT APPROACH TO INPUT SIGNAL SELECTION IN PARAMETER ESTIMATION PROBLEMS.[J]. Information Sciences, 1977, 12(1): 61-91.

[39]Carvalho H, Del Moral P, Monin A, et al. Optimal nonlinear filtering in GPS/INS integration[J]. IEEE Transactions on Aerospace and Electronic Systems, 1997, 33(3): 835-850.

[40]Nordlund P J, Gustafsson F. Sequential Monte Carlo filtering techniques applied to integrated navigation systems: 2001 American Control Conference, June 25, 2001 - June 27, 2001, Arlington, VA, United states, 2001[C]. Institute of Electrical and Electronics Engineers Inc..

[41]Azimi-Sadjadi B. Approximate nonlinear filtering with applications to navigation[D]. University of Maryland College ParkDepartment of Electrical and Computer Engineering, 2001.

[42]常青，郑平方．车载 GPS/DR 组合导航系统数据融合算法研究[J]．通信学报, 2000, 21(2): 42-48.

[43]Woodman O J. An introduction to inertial navigation[J]. University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696, 2007.

[44]Noureldin A, Karamat T B, Georgy J. Fundamentals of Inertial Navigation, Satellite-Based Positioning and Their Integration[M]. Springer, 2013.

[45]Abdel-Hamid W. Accuracy enhancement of integrated MEMS-IMU/GPS systems for land vehicular navigation applications[M]. University of Calgary, Department of Geomatics Engineering, 2005.

[46]Savage P G. Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms[J]. Journal of guidance, control, and dynamics, 1998, 21(1): 19-28.

[47]Savage P G. Erratum to Savage, P.G., Strapdown Inertial Navigation Integration Algorithm Design Part 1: Attitude Algorithms[J]. Journal of Guidance, Control, and Dynamics, 1999,22(2):384.

[48]Savage P G. Strapdown Analytics[M]. Strapdown Associates, 2000.

[49]Welch G, Bishop G. An introduction to the Kalman filter[J]. 1995.

[50]Grewal M S, Andrews A P. Kalman filtering: theory and practice using MATLAB[M]. Wiley-IEEE press, 2011.

[51]Savage P G. Strapdown inertial navigation integration algorithm design part 2: Velocity and position algorithms[J]. Journal of Guidance, Control, and Dynamics, 1998,21(2):208-221.

[52]Savage P G. Erratum To Savage, P.G., "Strapdown Inertial Navigation Algorithm Design Part 2: Velocity and Position Algorithms", JGCD, Vol. 21, No. 2, 1998, pp. 208-221[J]. Journal of Guidance, Control, and Dynamics, 2004, 27(2): 318.

[53]Savage P G. Strapdown Analytics Errata[M]. Strapdown Associates, 2006.

[54]Stockwell W. Bias stability measurement: Allan variance[J]. Crossbow Technology, Inc. http://www. xbow, 2004.

[55]李颖, 陈兴林, 宋申民. 光纤陀螺漂移误差动态 Allan 方差分析 3 3[J]. 光电子激光, 2008, 19(2).

[56]邹学锋, 卢新艳. 基于 Allan 方差的 MEMS 陀螺仪性能评价方法 [J][J]. 微纳电子技术, 2010,47(8):490-493.

[57]赵思浩, 陆明泉, 冯振明. MEMS 惯性器件误差系数的 Allan 方差分析方法[J]. 中国科学: 物理学, 力学, 天文学, 2010(5):672-675.

[58]彭飞, 柳重堪, 张其善. 基于模糊逻辑的 GPS/DR 组合导航系统地图匹配算法[J]. 遥测遥控, 2001,22(1):32-36.

[59]王志刚. 车载导航 GPS/DR/MM 组合定位技术的研究 [D][Z]. 武汉大学, 2005.

[60]于德新, 杨兆升, 刘雪杰. 基于卡尔曼滤波的 GPS/DR 导航信息融合方法[J]. 交通运输工程学报, 2006,6(2):65-69.

[61]金亮良, 何文涛, 徐建华, 等. 一种低成本 GPS/DR 组合导航系统设计[J]. 计算机工程, 2012,38(18):228-230.

[62]李沁雪, 彭志平, 张锋. GPS/DR 组合导航系统数据融合优化设计[J]. 计算机工程, 2012,38(12):268-271.

[63]刘桂辛. 基于多传感器信息融合的 GPS/DR 车载组合导航系统的研究[Z]. 燕山大学, 2013.

[64]Abdel-Hamid W, Osman A, Noureldin A, et al. Improving the performance of MEMS-based inertial sensors by removing short-term errors utilizing wavelet multi-resolution analysis, 2001[C].2001.

[65]Neu J M. A tightly-coupled INS/GPS integration using a MEMS IMU[J]. 2004.

[66]Abdel-Hamid W. Accuracy enhancement of integrated MEMS-IMU/GPS systems for land vehicular navigation applications[M]. University of Calgary, Department of Geomatics Engineering, 2005.

[67]Noureldin A, Karamat T B, Eberts M D, et al. Performance enhancement of MEMS-based INS/GPS integration for low-cost navigation applications[J]. Vehicular Technology, IEEE Transactions on, 2009, 58(3): 1077-1096.

[68]Carlson N A. Federated filter for fault-tolerant integrated navigation systems: Position Location and Navigation Symposium, 1988. Record. Navigation into the 21st Century. IEEE PLANS'88., IEEE, 1988[C]. IEEE.

[69]Carlson N A. Federated square root filter for decentralized parallel processors[J]. Aerospace and Electronic Systems, IEEE Transactions on, 1990, 26(3): 517-525.

[70]Carlson N A, Berarducci M P. Federated Kalman filter simulation results[J]. Navigation, 1994, 41(3): 297-321.

[71]Carlson N A. Federated filter for computer-efficient, near-optimal GPS integration: Position Location and Navigation Symposium, 1996., IEEE 1996, 1996[C]. IEEE.

[72]Carlson N A. Federated filter for distributed navigation and tracking applications: Proceedings of the 58th Annual Meeting of The Institute of Navigation and CIGTF 21st Guidance Test Symposium, 2001[C].

[73]Van Der Merwe R, Wan E A. The square-root unscented Kalman filter for state and parameter-estimation: Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP'01). 2001 IEEE International Conference on, 2001[C]. IEEE.

[74]Wan E A, Van Der Merwe R. The unscented Kalman filter[J]. Kalman filtering and neural networks, 2001: 221-280.

[75]潘泉, 杨峰, 叶亮, 等. 一类非线性滤波器--UKF 综述[J]. 控制与决策, 2005, 20(5): 481-489, 494.

[76]Julier S J, Uhlmann J K. A general method for approximating nonlinear transformations of probability distributions[J]. Robotics Research Group, Department of Engineering Science, University of Oxford, Oxford, OC1 3PJ United Kingdom, Tech. Rep, 1996.

[77]Julier S J, Uhlmann J K. New extension of the Kalman filter to nonlinear systems: AeroSense'97, 1997[C]. International Society for Optics and Photonics.

[78]Wan E A, Van Der Merwe R. The unscented Kalman filter for nonlinear estimation: Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000, 2000[C]. IEEE.

[79]Julier S J. The scaled unscented transformation: American Control Conference, 2002. Proceedings of the 2002, 2002[C].

[80]Julier S J, Uhlmann J K. Unscented filtering and nonlinear estimation[J]. Proceedings of the IEEE, 2004, 92(3): 401-422.

[81]Daum F. Nonlinear filters: beyond the Kalman filter[J]. Aerospace and Electronic Systems Magazine, IEEE, 2005, 20(8): 57-69.

[82]Terejanu G A. Unscented Kalman filter tutorial: Workshop on Large-Scale Quantification of Uncertainty, Sandia National Laboratories, 2009[C].

[83]Tsui. Fundamentals of Global Positioning System Receivers: A Software Approach[M]. Wiley, 2005.

[84]Kaplan E D, Hegarty C J. Understanding GPS: principles and applications[M]. Artech House Publishers, 2006.

[85]Misra P, Enge P. Global positioning system: signals, measurements, and performance[M]. Ganga-Jamuna Press, 2006.

[86]Borre K. A software-defined GPS and Galileo receiver: a single-frequency approach[M]. Birkhauser, 2007.

[87]沈雪松, 刘建业, 孙永荣, 等. 基于多传感器信息融合城市车辆导航定位[J]. 传感器与微系统, 2006, 25(1): 85-88.

[88]Spiker J J. The Global Positioning System: Theory and Application[M]. 1. Aiaa, 1996.

[89]Gustafson D, Dowdle J, Flueckiger K. A deeply integrated adaptive GPS-based navigator with extended range code tracking: Position Location and Navigation Symposium, IEEE 2000, 2000[C]. IEEE.

[90]Psiaki M L, Jung H. Extended Kalman filter methods for tracking weak GPS signals, 2001[C]. 2001.

[91]Abbott A S, Lillo W E. Global positioning systems and inertial measuring unit ultratight coupling method:

[92]Ohlmeyer E J. Analysis of an ultra-tightly coupled GPS/INS system in jamming: Proceedings of IEEE/ION PLANS 2006, 2006[C].

[93]Pany T, Eissfeller B. Use of a vector delay lock loop receiver for GNSS signal power analysis in bad signal conditions: Proceedings of the IEEE-ION Position, Location and Navigation Symposium (PLANS) 2006, 2006[C].

[94]Petovello M G, Lachapelle G. Comparison of vector-based software receiver implementations with application to ultra-tight GPS/INS integration: ION GNSS, 2006[C].

[95]Won J, D?tterb?ck D, Eissfeller B. Performance comparison of different forms of Kalman filter approaches for a vector-based GNSS signal tracking loop[J]. Navigation, 2010,57(3):185-199.

[96]Zhao S, Akos D. An Open Source GPS/GNSS Vector Tracking Loop-Implementation, Filter Tuning, and Results: Proceedings of the 2011 International Technical Meeting of The Institute of Navigation, 2011[C].

[97]Kim K, Jee G, Im S. Adaptive vector-tracking loop for low-quality GPS signals[J]. International Journal of Control, Automation and Systems, 2011, 9(4): 709-715.

[98]赵思浩, 陆明泉, 冯振明. 基于自适应卡尔曼滤波的 GNSS 矢量锁定环路[J]. 哈尔滨工业大学学报, 2012, 44(007): 139-143.

[99]Petovello M G, Lachapelle G. Comparison of vector-based software receiver implementations with application to ultra-tight GPS/INS integration: ION GNSS, 2006[C].

[100] So H, Lee T, Jeon S, et al. Implementation of a vector-based tracking loop receiver in a pseudolite navigation system[J]. Sensors, 2010, 10(7): 6324-6346.

[101] Kanwal N, Hurskainen H, Nurmi J. Vector tracking loop design for degraded signal environment: Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), 2010, 2010[C]. IEEE.

攻读硕士学位期间所取得的科研成果

[1] 牟文杰，叶凌云. 基于多传感器融合的车载航位推算系统 [J]. 传感器与微系统. （已录用）

附录 程序 Matlab 源代码

附录部分是论文中涉及的算法的 Matlab 源代码，主要包括 MEMS 航位推算算法、GNSS 基带信号处理和导航解算、组合导航算法（松耦合、紧耦合、超紧耦合）等。

A. MEMS 航位推算程序结构

MEMS 航位推算的 Matlab 源代码结构如图 A-1 所示：

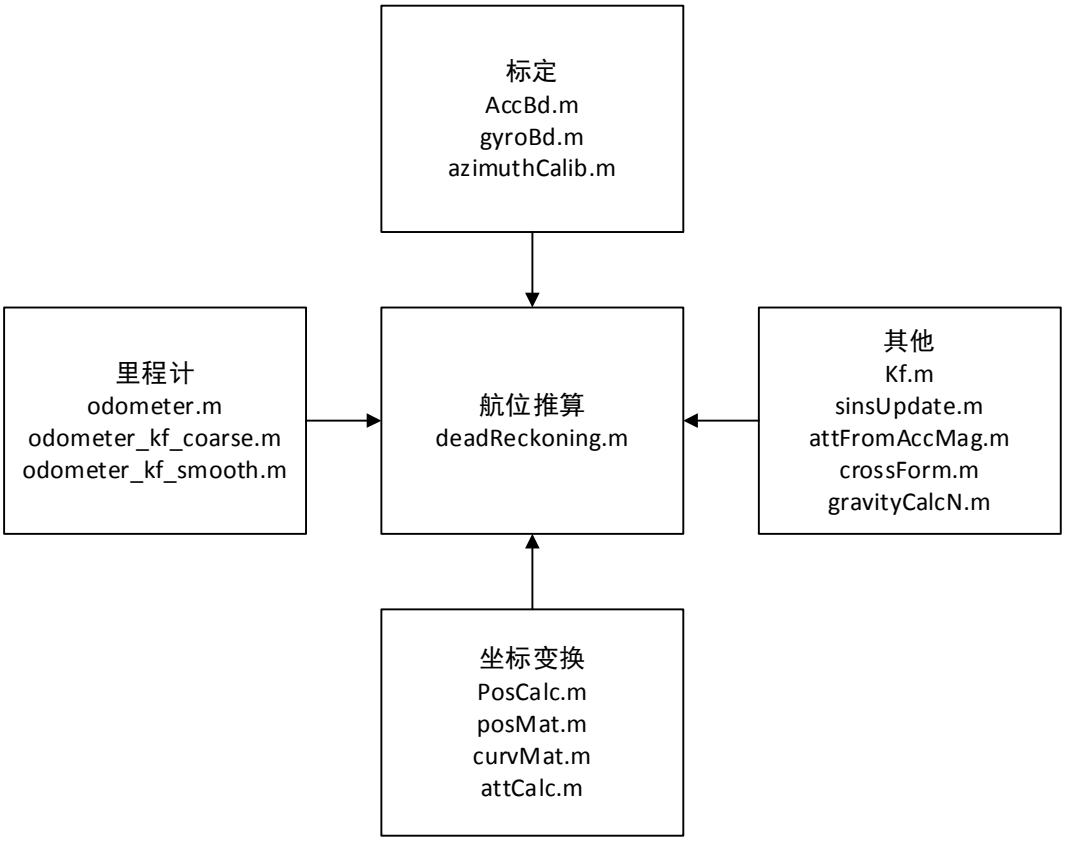

图 A-1 MEMS 航位推算源代码结构

函数具体说明如表 A-1 所示：

表 A-1 MEMS 航位推算程序的函数列表

函数或文件名	功能	对应章节
deadReckoning	航位推算主程序	第 2 章
accBd	加表标定程序	2.4.2

表 A-1 MEMS 航位推算程序的函数列表(续)

函数或文件名	功能	对应章节
gyroBd	陀螺标定程序	2.4.2
azimuthCalib	磁偏角标定程序	2.4.1
odometer	根据里程计的脉冲信号计算得到速度和加速度	2.1
odometer_kf_coarse	根据里程计的脉冲信号计算得到速度和加速度，加速度的信任度较高	2.1
odometer_kf_smooth	根据里程计的脉冲信号计算得到速度和加速度，速度的信任度较高	2.1
kf	标准卡尔曼滤波器	2.2
sinsUpdate	惯性导航更新算法	
attFromAccMag	根据加速度计和磁力计输出值计算载体的姿态矩阵	2.2.3
crossForm	将向量叉乘转换成矩阵形式	第 2 章
gravityCalcN	计算当地重力加速度大小	2.2 & 2.3
posCalc	根据位置矩阵计算经度、纬度和高度	2.3
posMat	根据经度、纬度和高度计算位置矩阵	2.3
attCalc	根据姿态角（航向、俯仰、横滚）计算姿态矩阵	2.2
curvMat	计算当时的地球曲率矩阵	2.3

B. 超紧耦合算法程序结构

超紧耦合算法源代码建立在 SoftGNSS v3.0 的基础之上。

超紧耦合算法的 Matlab 源代码结构如图 B-2 所示：

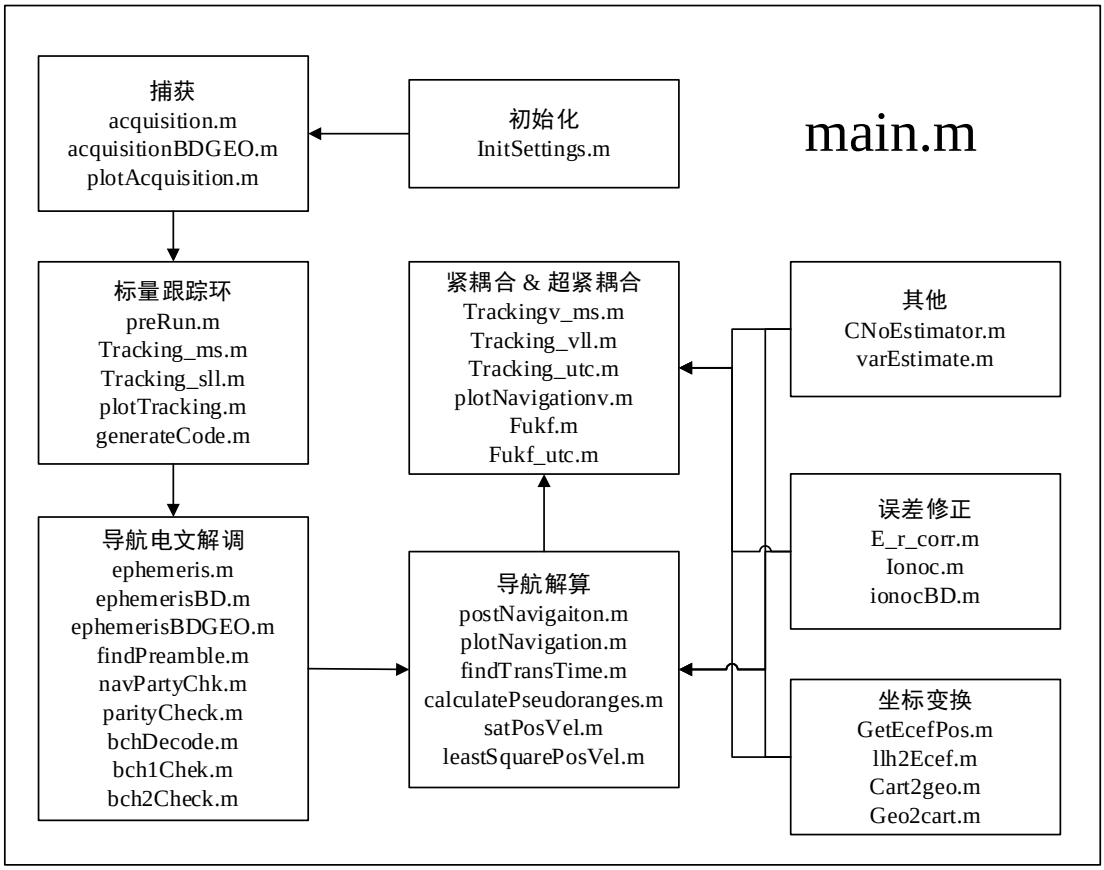

图 B-2 超紧耦合算法源代码结构

函数具体说明如表 B-2 所示：

表 B-2 超紧耦合程序的函数列表

函数或文件名	功能	对应章节
main	GNSS 接收机主程序	第 4 章
initSettings	初始化卫星导航的各种参数	第 3&4 章
acquisition	GPS 并行捕获算法程序	第 3&4 章
acquisitionBDGEO	北斗并行捕获算法程序	第 3&4 章
plotAcquisition	用于显示捕获结果	第 3&4 章
preRun	捕获完成后，初始化每个跟踪通道	第 3&4 章
tracking_ms	标量跟踪环算法，可设置跟踪的时间长度	4.2
tracking_sll	标量跟踪环算法	4.2
plotTracking	用于显示跟踪结果	第 3&4 章
generateCode	生成相应卫星的测距码序列	第 3&4 章

表 B-2 超紧耦合程序的函数列表(续 1)

函数或文件名	功能	对应章节
ephemerisBD	北斗 MEO 或 IGEO 卫星导航电文解调	第 3&4 章
ephemerisBDGEO	北斗 GEO 卫星导航电文解调	第 3&4 章
ephemeris	GPS 卫星导航电文解调	第 3&4 章
findPreamble	在跟踪结果中寻找前导码位置	第 3&4 章
navPartyChk	GPS 导航电文奇偶校验	第 3&4 章
parityCheck	奇偶校验程序	第 3&4 章
bchCode	北斗 BCH(15,11,1)译码	第 3&4 章
bch1Check	北斗每一子帧的第一个 word 的校验程序	第 3&4 章
bch2Check	北斗每一子帧的第 2~10 个 word 的校验程序	第 3&4 章
postNavigation	卫星接收机导航解算程序	4.2
plotNavigation	用于显示导航定位结果	4.2
findTransTime	计算信号发射时刻	第 3&4 章
calculatePseudoranges	计算伪距	3.2&4.3
satPosVel	计算每颗卫星的位置和速度	3.2&4.3
leastSquarePosVel	利用最小二乘法计算接收机位置和速度	3.2&4.3
trackingv_ms	矢量跟踪环预滤波器算法	4.2
tracking_vll	矢量跟踪环算法	4.2
tracking_utc	超紧耦合算法	4.2&4.3
plotNavigationv	用于显示矢量跟踪环或超紧耦合导航结果	4.2&4.3
fukf	紧耦合中基于 FUKF 的导航滤波器	3.2
fukf_utc	超紧耦合中基于 FUKF 的导航滤波器	4.3
CNoEstimator	估计相应通道的 C/N_0	4.2.2
varEstimate	根据通道的 C/N_0 计算 PLL、DLL、FLL 鉴相器输出噪声方差	4.2.2
e_r_corr	地球自转引起的导航误差修正	第 3&4 章
ionoc	计算 GPS 电离层的延时误差	第 3&4 章

表 B-2 超紧耦合程序的函数列表(续 2)

函数或文件名	功能	对应章节
ionocBD	计算北斗电离层的延时误差	第 3&4 章
GetEcefPos	根据位置矩阵计算在 ECEF 坐标系中的位置	4.3
llh2Ecef	将位置矩阵转换成在 ECEF 中的坐标	4.3
cart2Geo	直角坐标系到地理坐标系的变换	第 3&4 章
geo2Cart	地理坐标系到直角坐标系的变换	第 3&4 章

C. Matlab 源代码

1) accBd

```
function accBdRes = accBd(dir)
% 加表数据命名格式：ace.mat表示a面朝下，c朝东，e朝北时采集的数据

%% load data
% dir = '\';
dir1 = ['a','b','c','d','e','f'];

avgMeas = zeros(3,6);
for i = 1:6
    din = load([dir, dir1(i), '.mat']);
    avgMeas(:,i) = mean(din.imuData(4:6,:),2);
end

%% Calculate result
g = 9.7934;
% accBdRes = accBdCalc(avgMeas);
dMeas = avgMeas(:,[1,3,5]) - avgMeas(:,[2,4,6]);
% 标定因数，nx1
accBdRes.k = 2*g./sqrt(sum(dMeas.^2, 2));
% 零偏
accBdRes.b = -accBdRes.k.*mean(avgMeas, 2);
% 安装误差矩阵
T = diag(accBdRes.k)*dMeas/(2*g);
accBdRes.A = inv(T);     % 当传感器个数大于3时，应当相应修改

%% save results
```

```
save([dir,'accBdResults.mat'], 'accBdRes');
```

2) acquisition

```
function acqResults = acquisition(longSignal, settings)
%Function performs cold start acquisition on the collected "data". It
%searches for GPS signals of all satellites, which are listed in field
%"acqSatelliteList" in the settings structure. Function saves code phase
%and frequency of the detected signals in the "acqResults" structure.
%
%acqResults = acquisition(longSignal, settings)
%
%   Inputs:
%       longSignal    - 11 ms of raw signal from the front-end
%       settings      - Receiver settings. Provides information about
%                       sampling and intermediate frequencies and other
%                       parameters including the list of the satellites to
%                       be acquired.
%   Outputs:
%       acqResults    - Function saves code phases and frequencies of the
%                       detected signals in the "acqResults" structure. The
%                       field "carrFreq" is set to 0 if the signal is not
%                       detected for the given PRN number.
%----------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis and Dennis M. Akos
% Written by Darius Plausinaitis and Dennis M. Akos
% Based on Peter Rinder and Nicolaj Bertelsen
% Modified by Mwenjie @2014.03
%----------------------------------------------------------------------
%% Initialization ================================================
ms = 1;                    % coherent integration time [ms]
freqBin = 500/ms;          % frequency step

% Find number of samples per spreading code
samplesPerCode = round(settings.samplingFreq / ...
                       (settings.codeFreqBasis / settings.codeLength));

% Create two 1msec vectors of data to correlate with and one with zero
DC
signal1 = longSignal(1 : samplesPerCode*ms);
signal2 = longSignal(samplesPerCode*ms+1 : 2*samplesPerCode*ms);

signal0DC = longSignal - mean(longSignal);   %%Problems here....
```

```
% Find sampling period
ts = 1 / settings.samplingFreq;

% Find phase points of the local carrier wave
phasePoints = (0 : (samplesPerCode*ms-1)) * 2 * pi * ts;

% Number of the frequency bins for the given acquisition band (500Hz steps)
numberOfFrqBins = round(settings.acqSearchBand*1000 / freqBin) + 1;

% Generate all C/A codes and sample them according to the sampling freq.
caCodesTable = makeCaTable(settings);

%--- Initialize arrays to speed up the code ----------------------
% Search results of all frequency bins and code shifts (for one satellite)
results     = zeros(numberOfFrqBins, samplesPerCode*ms);

% Carrier frequencies of the frequency bins
frqBins     = zeros(1, numberOfFrqBins);

%--- Initialize acqResults ---------------------------------------
% Carrier frequencies of detected signals
acqResults.carrFreq     = zeros(1, 32);
% C/A code phases of detected signals
acqResults.codePhase    = zeros(1, 32);
% Correlation peak ratios of the detected signals
acqResults.peakMetric   = zeros(1, 32);

fprintf('(');

% Perform search for all listed PRN numbers ...
for PRN = settings.acqSatelliteList

%% Correlate signals ======================================
    %--- Perform FFT of C/A code ------------------------------
    caCodeFreqDom = conj(fft(repmat(caCodesTable(PRN, :),1,ms)));

    %--- Make the correlation for whole frequency band (for all freq. bins)
    for frqBinIndex = 1:numberOfFrqBins

        %--- Generate carrier wave frequency grid (0.5kHz step) -----
        frqBins(frqBinIndex) = settings.IF - ...
```

```
                        (settings.acqSearchBand/2) * 1000 + ...
                        freqBin * (frqBinIndex - 1);

        %--- Generate local sine and cosine -------------------------
        sigCarr = exp(1i*frqBins(frqBinIndex) * phasePoints);

        %--- "Remove carrier" from the signal -----------------------
        I1      = real(sigCarr .* signal1);
        Q1      = imag(sigCarr .* signal1);
        I2      = real(sigCarr .* signal2);
        Q2      = imag(sigCarr .* signal2);

        %--- Convert the baseband signal to frequency domain --------------
        IQfreqDom1 = fft(I1 + 1i*Q1);
        IQfreqDom2 = fft(I2 + 1i*Q2);

        %--- Multiplication in the frequency domain (correlation in time
        %domain)
        convCodeIQ1 = IQfreqDom1 .* caCodeFreqDom;
        convCodeIQ2 = IQfreqDom2 .* caCodeFreqDom;

        %--- Perform inverse DFT and store correlation results ------------
        acqRes1 = abs(ifft(convCodeIQ1)) .^ 2;
        acqRes2 = abs(ifft(convCodeIQ2)) .^ 2;

        %--- Check which msec had the greater power and save that, will
        %"blend" 1st and 2nd msec but will correct data bit issues
        if (max(acqRes1) > max(acqRes2))
            results(frqBinIndex, :) = acqRes1;
        else
            results(frqBinIndex, :) = acqRes2;
        end

    end % frqBinIndex = 1:numberOfFrqBins

%% Look for correlation peaks in the results ====================
    % Find the highest peak and compare it to the second highest peak
    % The second peak is chosen not closer than 1 chip to the highest peak

    %--- Find the correlation peak and the carrier frequency -----------
    [peakSize, frequencyBinIndex] = max(max(results, [], 2));
```

```
    %--- Find code phase of the same correlation peak -------------
    [peakSize, codePhase] = max(max(results));
    codePhase = rem(codePhase, samplesPerCode);

    %--- Find 1 chip wide C/A code phase exclude range around the peak
    samplesPerCodeChip   = round(settings.samplingFreq /
settings.codeFreqBasis);
    excludeRangeIndex1 = codePhase - samplesPerCodeChip;
    excludeRangeIndex2 = codePhase + samplesPerCodeChip;

    %--- Correct C/A code phase exclude range if the range includes array
    %boundaries
    if excludeRangeIndex1 < 2
        codePhaseRange = excludeRangeIndex2 : ...
                       (samplesPerCode + excludeRangeIndex1);

    elseif excludeRangeIndex2 >= samplesPerCode
        codePhaseRange = (excludeRangeIndex2 - samplesPerCode) : ...
                       excludeRangeIndex1;
    else
        codePhaseRange = [1:excludeRangeIndex1, ...
                        excludeRangeIndex2 : samplesPerCode];
    end

    %--- Find the second highest correlation peak in the same freq. bin
    secondPeakSize = max(results(frequencyBinIndex, codePhaseRange));

    %--- Store result --------------------------------------------
    acqResults.peakMetric(PRN) = peakSize/secondPeakSize;

    % If the result is above threshold, then there is a signal ...
    if (peakSize/secondPeakSize) > settings.acqThreshold

%% Fine resolution frequency search =============================

        %--- Indicate PRN number of the detected signal --------------
        fprintf('%02d ', PRN);

        %--- Generate 10msec long C/A codes sequence for given PRN ----
        caCode = generateCAcode(PRN);

        codeValueIndex = floor((ts * (1:10*samplesPerCode)) / ...
```

```
                        (1/settings.codeFreqBasis));

        longCaCode = caCode((rem(codeValueIndex, 1023) + 1));

        %--- Remove C/A code modulation from the original signal -------
        % (Using detected C/A code phase)
        xCarrier = signal0DC(codePhase:(codePhase +
10*samplesPerCode-1)) ...
            .* longCaCode;

        %--- Compute the magnitude of the FFT, find maximum and the
        %associated carrier frequency

        %--- Find the next highest power of two and increase by 8x --------
        fftNumPts = 8*(2^(nextpow2(length(xCarrier))));

        %--- Compute the magnitude of the FFT, find maximum and the
        %associated carrier frequency
        fftxc = abs(fft(xCarrier, fftNumPts));

        uniqFftPts = ceil((fftNumPts + 1) / 2);
        [fftMax, fftMaxIndex] = max(fftxc);
        fftFreqBins = (0 : uniqFftPts-1) *
settings.samplingFreq/fftNumPts;
        if (fftMaxIndex > uniqFftPts) %and should validate using complex
data
            if (rem(fftNumPts,2)==0)  %even number of points, so DC and Fs/2
computed
                fftFreqBinsRev=-fftFreqBins((uniqFftPts-1):-1:2);
                [fftMax, fftMaxIndex] =
max(fftxc((uniqFftPts+1):length(fftxc)));
                acqResults.carrFreq(PRN) = -fftFreqBinsRev(fftMaxIndex);
            else  %odd points so only DC is not included
                fftFreqBinsRev=-fftFreqBins((uniqFftPts):-1:2);
                [fftMax, fftMaxIndex] = max(fftxc((uniqFftPts+1):
length(fftxc)));
                acqResults.carrFreq(PRN)  = fftFreqBinsRev(fftMaxIndex);
            end
        else
            acqResults.carrFreq(PRN) =
(-1)^(settings.fileType-1)*fftFreqBins(fftMaxIndex);
        end
```

```
        acqResults.codePhase(PRN) = codePhase;

    else
        %--- No signal with this PRN --------------------------------
        fprintf('. ');
    end   % if (peakSize/secondPeakSize) > settings.acqThreshold

end    % for PRN = satelliteList

%=== Acquisition is over ==========================================
fprintf(')\n');
```

3) acquisitionBDGEO

```
function acqResults = acquisitionBDGEO(longSignal, settings)
%Function performs cold start acquisition on the collected "data". It
%searches for Beidou signals of all satellites, which are listed in field
%"acqSatelliteList" in the settings structure. Function saves code phase
%and frequency of the detected signals in the "acqResults" structure.
%
%acqResults = acquisitionBDGEO(longSignal, settings)
%
%   Inputs:
%       longSignal    - 11 ms of raw signal from the front-end
%       settings      - Receiver settings. Provides information about
%                       sampling and intermediate frequencies and other
%                       parameters including the list of the satellites to
%                       be acquired.
%   Outputs:
%       acqResults    - Function saves code phases and frequencies of the
%                       detected signals in the "acqResults" structure. The
%                       field "carrFreq" is set to 0 if the signal is not
%                       detected for the given PRN number.
%----------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis and Dennis M. Akos
% Written by Mwenjie @2014.03
% Based on SoftGNSS v3.0
%----------------------------------------------------------------------
%% Initialization ===================================================
corrNum = 10;          % non-coherent integration time [ms]
freqBin = 250;         % frequency step
```

```
% Find number of samples per spreading code
samplesPerCode = round(settings.samplingFreq / ...
    (settings.codeFreqBasis / settings.codeLength));

% Find sampling period
ts = 1 / settings.samplingFreq;
% C/A chip period in sec
tc = 1/settings.codeFreqBasis;

% Find phase points of the local carrier wave
phasePoints = (0 : (samplesPerCode-1)) * 2 * pi * ts;

% Number of the frequency bins for the given acquisition band (500Hz steps)
numberOfFrqBins = round(settings.acqSearchBand / freqBin) + 1;

%--- Initialize arrays to speed up the code ------------------------
% Carrier frequencies of the frequency bins
frqBins    = zeros(1, numberOfFrqBins);
signal0DC = longSignal - mean(longSignal);   %%Problems here....

%--- Initialize acqResults -----------------------------------
% Carrier frequencies of detected signals
acqResults.carrFreq    = zeros(1, settings.max_sv);
% C/A code phases of detected signals
acqResults.codePhase    = zeros(1, settings.max_sv);
% Correlation peak ratios of the detected signals
acqResults.peakMetric   = zeros(1, settings.max_sv);
fprintf('(');
% Perform search for all listed PRN numbers ...
for PRN = settings.acqSatelliteList
%% Correlate signals ======================================
    % Generate all C/A codes and sample them according to the sampling
freq.
    caCode = generateCode(PRN, settings.system);
    codeValueIndex = ceil((ts * (1:samplesPerCode)) / tc);
    codeValueIndex(end) = settings.codeLength;
    longCaCode = caCode( codeValueIndex );
    %--- Perform DFT of C/A code ------------------------------------
    caCodeFreqDom = conj(fft(longCaCode));
    % Search results of all frequency bins and code shifts (for one
satellite)
    results1    = zeros(numberOfFrqBins, samplesPerCode);
```

```
    results2    = zeros(numberOfFrqBins, samplesPerCode);

    for corrCnt = 0:corrNum-1
        % Create two 1msec vectors of data to correlate with and one with
zero DC
        signal1 = longSignal(corrCnt*samplesPerCode+1 :
(corrCnt+1)*samplesPerCode);
        signal2 = longSignal((corrCnt+1)*samplesPerCode+1 :
(corrCnt+2)*samplesPerCode);

        %--- Make the correlation for whole frequency band (for all freq.
bins)
        for frqBinIndex = 1:numberOfFrqBins

            %--- Generate carrier wave frequency grid (0.5kHz step) -------
            frqBins(frqBinIndex) = settings.IF -
(settings.acqSearchBand/2) + freqBin * (frqBinIndex - 1);
            %--- Generate local sine and cosine -------------------------
            sigCarr = exp(1i*frqBins(frqBinIndex) * phasePoints);
            %--- "Remove carrier" from the signal -----------------------
            I1      = real(sigCarr .* signal1);
            Q1      = imag(sigCarr .* signal1);
            I2      = real(sigCarr .* signal2);
            Q2      = imag(sigCarr .* signal2);

            %--- Convert the baseband signal to frequency domain
            IQfreqDom1 = fft(I1 + 1i*Q1);
            IQfreqDom2 = fft(I2 + 1i*Q2);
            %--- Multiplication in the frequency domain (correlation in
time domain)
            convCodeIQ1 = IQfreqDom1 .* caCodeFreqDom;
            convCodeIQ2 = IQfreqDom2 .* caCodeFreqDom;
            %--- Perform inverse DFT and store correlation results --------
            acqRes1 = abs(ifft(convCodeIQ1)) .^ 2;
            acqRes2 = abs(ifft(convCodeIQ2)) .^ 2;
            %--- Check which msec had the greater power and save that, will
            %"blend" 1st and 2nd msec but will correct data bit issues
            results1(frqBinIndex, :) = results1(frqBinIndex, :) + acqRes1;
            results2(frqBinIndex, :) = results2(frqBinIndex, :) + acqRes2;
        end % frqBinIndex = 1:numberOfFrqBins
    end % for corrCnt = 0:corrNum-1
```

```
    %--- Check which msec had the greater power and save that, will
    % "blend" 1st and 2nd msec but will correct data bit issues
    if max(max(results1)) > max(max(results2))
        results = results1 / corrNum;
    else
        results = results2 / corrNum;
    end

    %% Look for correlation peaks in the results ========================
    % Find the highest peak and compare it to the second highest peak
    % The second peak is chosen not closer than 1 chip to the highest peak

    %--- Find the correlation peak and the carrier frequency --------------
    [~, frequencyBinIndex] = max(max(results, [], 2));
    %--- Find code phase of the same correlation peak ----------------------
    [peakSize, codePhase] = max(max(results));
    % codePhase = rem(codePhase, samplesPerCode);
    %--- Find 1 chip wide C/A code phase exclude range around the peak
    samplesPerCodeChip   = round(settings.samplingFreq /
settings.codeFreqBasis);
    excludeRangeIndex1 = codePhase - samplesPerCodeChip;
    excludeRangeIndex2 = codePhase + samplesPerCodeChip;
    %--- Correct C/A code phase exclude range if the range includes array
    % boundaries
    if excludeRangeIndex1 < 2
        codePhaseRange = excludeRangeIndex2 : ...
                        (samplesPerCode + excludeRangeIndex1);
    elseif excludeRangeIndex2 >= samplesPerCode
        codePhaseRange = (excludeRangeIndex2 - samplesPerCode) : ...
                        excludeRangeIndex1;
    else
        codePhaseRange = [1:excludeRangeIndex1, ...
                         excludeRangeIndex2 : samplesPerCode];
    end
    %--- Find the second highest correlation peak in the same freq. bin
    secondPeakSize = max(results(frequencyBinIndex, codePhaseRange));
    %--- Store result -----------------------------------------------------
    acqResults.peakMetric(PRN) = peakSize/secondPeakSize;

    %% Plot result
    % figure(PRN)
    % x_axis = codeValueIndex;
```

```
    % y_axis = frqBins/1e6;
    % s=surf(x_axis,y_axis,results);
    % set(s,'EdgeColor','none','Facecolor','interp');
    % axis([min(x_axis) max(x_axis) min(y_axis) max(y_axis) ...
    % min(min(results)) max(max(results))]);
    % caxis([0 max(max(results))]);
    % xlabel('Code Phase [chips]');
    % ylabel('Frequency [MHz]');
    % zlabel('Magnitude');
    % text=sprintf('SVN %i',PRN);
    % title(text);

    %% Fine resolution frequency search ==============================
    % If the result is above threshold, then there is a signal ...
    if (peakSize/secondPeakSize) > settings.acqThreshold
        %--- Indicate PRN number of the detected signal -------------------
        fprintf('%02d ', PRN);
        %--- Remove C/A code modulation from the original signal ----------
        % (Using detected C/A code phase)
        xCarrier = signal0DC(codePhase:(codePhase + samplesPerCode-1)) .*
longCaCode;
        %--- Find the next highest power of two and increase by 8x --------
        fftNumPts = 8*(2^(nextpow2(length(xCarrier))));
        %--- Compute the magnitude of the FFT, find maximum and the
        %associated carrier frequency
        fftxc = abs(fft(xCarrier, fftNumPts));
        uniqFftPts = ceil((fftNumPts + 1) / 2);
        [~, fftMaxIndex] = max(fftxc);
        fftFreqBins = (0 : uniqFftPts-1) *
settings.samplingFreq/fftNumPts;
        if (fftMaxIndex > uniqFftPts) %and should validate using complex
data
            if (rem(fftNumPts,2)==0)  %even number of points, so DC and Fs/2
computed
                fftFreqBinsRev=-fftFreqBins((uniqFftPts-1):-1:2);
                [~, fftMaxIndex] =
max(fftxc((uniqFftPts+1):length(fftxc)));
                acqResults.carrFreq(PRN)  =
-fftFreqBinsRev(fftMaxIndex);
            else  %odd points so only DC is not included
                fftFreqBinsRev=-fftFreqBins((uniqFftPts):-1:2);
                [~, fftMaxIndex] =
```

```
max(fftxc((uniqFftPts+1):length(fftxc)));
            acqResults.carrFreq(PRN)  = fftFreqBinsRev(fftMaxIndex);
        end
    else
        acqResults.carrFreq(PRN)  =
(-1)^(settings.fileType-1)*fftFreqBins(fftMaxIndex);
    end

    acqResults.codePhase(PRN) = codePhase;
else
    %--- No signal with this PRN --------------------------------
    fprintf('. ');
end    % if (peakSize/secondPeakSize) > settings.acqThreshold
end    % for PRN = satelliteList
```

4) attCalc.m

```
function [ pitch, roll, azimuth ] = attCalc( C_b2n )
% 根据姿态矩阵计算航向角、俯仰角和横滚角
% 输入:
%   C_b2n   --姿态矩阵，3x3，由机体坐标系变换到导航坐标系
% 输出:
%   pitch   --俯仰角，rad
%   roll    --横滚角，rad
%   azimuth     --航向角，rad

%% 计算航向角、俯仰角和横滚角
% 俯仰角
pitch = asin(C_b2n(3,2));
% 横滚角
roll = atan2(-C_b2n(3,1),C_b2n(3,3));
% 航向角
azimuth = atan2(-C_b2n(1,2),C_b2n(2,2));
% if azimuth < 0
%    azimuth = azimuth + 2*pi;
% end
```

5) attFromAccMag

```
function C_b2n = attFromAccMag( acc, mag, dAzimuth )
% 根据加速度值、磁场值和磁偏角计算姿态矩阵
% 加速度值正交化
acc = acc/norm(acc);
```

```
% 计算东向单位矢量
v1 = cross(mag,acc);
v1 = v1/norm(v1);
% 计算北向单位矢量
v2 = cross(acc,v1);
v2 = v2/norm(v2);
% 姿态矩阵
C_b2n = [v1'; v2'; acc'];
% 磁偏角修正
R = [ cos(dAzimuth), -sin(dAzimuth), 0;
    sin(dAzimuth), cos(dAzimuth), 0;
    0, 0, 1 ];
C_b2n = R*C_b2n;
```

6) azimuthCalib

```
% 磁偏角标定程序
Vx = 0;
Vy = 0;
for jj = 1:760
    % 获得GPS水平速度
    vg = vel_gps(:,jj);
    vg(3) = 0;
    if norm(vg) == 0
        continue;
    end
    vg = vg/norm(vg);
    % 获得航位推算水平速度
    vd = mean(vel(:,100*(jj-1)+90:100*jj+10),2);
    vd(3) = 0;
    % cos
    y = dot(vd,vg);
    % sin
    x = norm(vd - y*vg);
    vz = cross(vd,vg);
    if vz(3) < 0
        x = -x;
    end
    Vy = Vy+y;
    Vx = Vx+x;
end
 % 计算磁偏角
theta = atan2(Vx,Vy)*180/pi;
```

7) bch1Check

```
function codeout = bch1Check(codein)
 % 每个子帧第1个字的前15比特信息不进行纠错编码，后11比特信息采用BCH(15,11,1)方式进行纠错
decode_2 = bchDecode(codein(16:end));
codeout = [codein(1:15),decode_2];
```

8) bch2Check

```
function codeout = bch2Check(codein)
% 第2~10个字均采用BCH(15,11,1)加交织方式进行纠错编码
codeout1 = bchDecode(codein(1:2:end));
codeout2 = bchDecode(codein(2:2:end));
codeout = [codeout1(1:11) codeout2(1:11) codeout1(12:end)
codeout1(12:end)];
```

9) bchDecode

```
function code = bchDecode(code)
% BCH(15,11,1) check
% bchROM = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;  % 0
%     0 0 0 0 0 0 0 0 0 0 0 0 0 0 1;         % 15
%     0 0 0 0 0 0 0 0 0 0 0 0 0 1 0;         % 14
%     0 0 0 0 0 0 0 0 0 0 1 0 0 0 0;         % 11
%     0 0 0 0 0 0 0 0 0 0 0 0 1 0 0;         % 13
%     0 0 0 0 0 0 1 0 0 0 0 0 0 0 0;         % 7
%     0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;         % 10
%     0 0 0 0 1 0 0 0 0 0 0 0 0 0 0;         % 5
%     0 0 0 0 0 0 0 0 0 0 0 1 0 0 0;         % 12
%     1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;         % 1
%     0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;         % 6
%     0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;         % 8
%     0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;         % 9
%     0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;         % 2
%     0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;         % 4
%     0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ];       % 3
% 纠错信号表
bchTable = [ 0 15 14 11 13 7 10 5 12 1 6 8 9 2 4 3 ];

code_logic = zeros(1,15);
for jj = 1:15
    if code(jj) == '1'
```

```
        code_logic(jj) = 1;
    else
        code_logic(jj) = 0;
    end
end
% 初始时移位寄存器清零
d0 = 0;
d1 = 0;
d2 = 0;
d3 = 0;
% BCH码组逐位输入到除法电路和15级纠错缓存器中
for jj = 1:length(code_logic)
    temp = d3;
    d3 = d2;
    d2 = d1;
    d1 = xor(d0,temp);
    d0 = xor(temp,code_logic(jj));
end
% 当BCH 码的15位全部输入后，纠错信号ROM表利用除法电路的4级移位寄存器的状态D3、D2、
D1、D0查表，得到纠错位
index = bchTable(d3*8+d2*4+d1*2+d0+1);
% 纠错
if index > 0
    if code(index) == '1'
        code(index) = '0';
    else
        code(index) = '1';
    end
end
```

10) calculatePseudoranges

```
function pseudoranges = calculatePseudoranges(...
                    transmitTime,rxTime,channelList,settings)
%calculatePseudoranges finds relative pseudoranges for all satellites
%listed in CHANNELLIST at the specified millisecond of the processed
%signal. The pseudoranges contain unknown receiver clock offset. It can
be
%found by the least squares position search procedure.
%
% function pseudoranges = calculatePseudoranges(...
%                     transmitTime,rxTime,channelList,settings)
%
```

```
%   Inputs:
%       transmitTime    - transmitting time all satellites on the list
%       rxTime          - receiver time
%       channelList     - list of channels to be processed
%       settings        - receiver settings
%
%   Outputs:
%       pseudoranges    - relative pseudoranges to the satellites.
%--------------------------------------------------------------------
% Copyright (C) D.M.Akos
% Written by Darius Plausinaitis
% Modified by Xiaofan Li at University of Colorado at Boulder
%--------------------------------------------------------------------
 %--- Set initial travel time to infinity -------------------------
% Later in the code a shortest pseudorange will be selected. Therefore
% pseudoranges from non-tracking channels must be the longest - e.g.
% infinite.
travelTime = inf(1, settings.numberOfChannels);

%--- For all channels in the list ...
for channelNr = channelList
    %--- Compute the travel times ------------------------------
    travelTime(channelNr) = rxTime - transmitTime(channelNr);
end

%--- Convert travel time to a distance --------------------------
% The speed of light must be converted from meters per second to meters
% per millisecond.
pseudoranges = travelTime * settings.c;
```

11) cart2geo

```
function [phi, lambda, h] = cart2geo(X, Y, Z, i)
%CART2GEO Conversion of Cartesian coordinates (X,Y,Z) to geographical
%coordinates (phi, lambda, h) on a selected reference ellipsoid.
%
%[phi, lambda, h] = cart2geo(X, Y, Z, i);
%
%   Choices i of Reference Ellipsoid for Geographical Coordinates
%           1. International Ellipsoid 1924
%           2. International Ellipsoid 1967
%           3. World Geodetic System 1972
%           4. Geodetic Reference System 1980
```

```
%           5. World Geodetic System 1984
%           6. CGCS2000

%Kai Borre 10-13-98
%Copyright (c) by Kai Borre
%Revision: 1.0   Date: 1998/10/23

% Constants
a = [6378388 6378160 6378135 6378137 6378137 6378137.0];
f = [1/297 1/298.247 1/298.26 1/298.257222101 1/298.257223563
1/298.257222101];
% longitude
lambda = atan2(Y,X);
% Initialize latitude
ex2 = (2-f(i))*f(i)/((1-f(i))^2);
c = a(i)*sqrt(1+ex2);
phi = atan(Z/((sqrt(X^2+Y^2)*(1-(2-f(i)))*f(i))));
% iterate to approach accurate latitude within given error
h = 0.1; oldh = 0;
iterations = 0;
while abs(h-oldh) > 1.e-12
    oldh = h;
    N = c/sqrt(1+ex2*cos(phi)^2);
    phi = atan(Z/((sqrt(X^2+Y^2)*(1-(2-f(i))*f(i)*N/(N+h)))));
    h = sqrt(X^2+Y^2)/cos(phi)-N;
    iterations = iterations + 1;
    if iterations > 100
        fprintf('Failed to approximate h with desired precision.
h-oldh: %e.\n', h-oldh);
       break;
    end
end
% change to radian form
phi = phi*180/pi;
lambda = lambda*180/pi;
```

12) CNoEstimator

```
function CNo = CNoEstimator(I_P, Q_P, dT)
% Estimate Carrier-to-Noise ratio of each channel
M = length(I_P);
% Pre-detection integration time
T = M*dT;
```

```
ip = zeros(1,M);
qp = zeros(1,M);
ip(1) = abs(I_P(1));
qp(1) = abs(Q_P(1));
% low pass filter
for jj = 2:M
    ip(jj) = ip(jj-1) + (abs(I_P(jj)) - ip(jj-1)) * 0.0247;
    qp(jj) = qp(jj-1) + (abs(Q_P(jj)) - qp(jj-1)) * 0.0247;
end
% Carrier power
C = 2*sum( ip.^2 + qp.^2 );
% Noise power
No = 4*T*var(abs(I_P+1j*Q_P));
% Carrier-to-Noise ratio
CNo = 10*log10(C/No);
```

13) crossForm

```
function x_cross = crossForm(x)
% 生成叉乘的矩阵形式，即(ax)
x_cross = [ 0,-x(3),x(2);
    x(3),0,-x(1);
    -x(2),x(1),0 ];
```

14) curvMat

```
function F_curv = curvMat( C_n2e, hgt, NORTH_SYSTEM )
% 根据位置矩阵计算当前位置的曲率矩阵
% 输入:
%   C_n2e -- 位置矩阵，3x3，将导航坐标系转换到地球坐标系
%   hgt   -- 高度，m
% 输出:
%   F_curv -- 曲率矩阵，3x3，用于更新w_en_n

% 具体模型见Savage的Strapdown Analytics中的第5章

%% 地球相关常数定义
Ellip1 = -0.006694379990141;    % Ellip1 = (1-Ellip)^2 - 1;
Ra = 6378137.0;                 % 地球长轴,m

%% 计算当前位置的曲率矩阵
% Rs', modified magnitude of Rs
Rs1_len = Ra/sqrt(1+C_n2e(3,3)^2*Ellip1);
```

```
% equivalent radii of curvature corresponding to latitude
r_lat = (Ellip1+1)*Rs1_len^3/Ra^2 + hgt;

% 计算曲率矩阵F_curv
F_curv = zeros(3,3);

if NORTH_SYSTEM == 1    % 指北方位系统
    r_lon = Rs1_len + hgt;  % equivalent radii of curvature corresponding
to latitude
    F_curv(1,2) = -1/r_lat;
    F_curv(2,1) = 1/r_lon;
    F_curv(3,1) = C_n2e(3,3)/C_n2e(3,2)/r_lon;
else                    % 游动自由方位系统
    feh = 1/(1+h/Rs1_len)*Ellip1/(1+C_n2e(3,3)^2*Ellip1);
    F_curv(1,1) = C_n2e(3,1)*C_n2e(3,2)*feh/r_lat;
    F_curv(1,2) = -(1+C_n2e(3,1)^2*feh)/r_lat;
    F_curv(2,1) = (1+C_n2e(3,2)^2*feh)/r_lat;
    F_curv(2,2) = -C_n2e(3,1)*C_n2e(3,2)*feh/r_lat;
End
```

15) deadReckoning.m

```
% 基于多传感器信息融合的航位推算系统主程序
len = length(time);
dAzimuth = 3*pi/180;
 %% Initialize kalman filter
dT = 1/100;           % 更新时间时间间隔
F = eye(3);
G = eye(3)*dT;
H = eye(3);
 % Q = diag( ([dt, dt, dt]*1e-2).^2 );
Q = diag( ([dT, dT, dT]*1e-1).^2 );
R = diag( [0.1, 0.1, 0.2].^2 );
x = zeros(3,1);
P = diag( [pi/2, pi/2, pi/2].^2 );
 %% Initialize attitude
C_b2n = attFromAccMag(mean(acc(:,1:500),2), mean(mag(:,1:500),2),
dAzimuth);
% 输出变量初始化
pos = zeros(3,len);             % 位置
vel = zeros(3,len);             % 速度
att = zeros(3,len);             % 姿态
C_n2e = posMat(mean(pos_gps(1,1:10)),mean(pos_gps(2,1:10)),0);
```

```
hgt = mean(pos_gps(3,1:10));
vel_1 = zeros(3,1);
vel_2 = zeros(3,1);

%% 开始航位推算
for jj = 1:len
    %% AHRS
    u = gyro(:,jj)*pi/180;
    grav_b = acc(:,jj);
    grav_b(1) = grav_b(1)+speed(jj)*u(3);
    grav_b(2) = grav_b(2) - acceleration(jj);
%     grav_b(3) = sqrt(9.7934^2-grav_b(1).^2-grav_b(2).^2);
    C_b2n_est = attFromAccMag(grav_b,mag(:,jj),dAzimuth);
    dAtt_cross = C_b2n'*C_b2n_est - eye(3);
    z = [ dAtt_cross(3,2); dAtt_cross(1,3); dAtt_cross(2,1) ];
    % 自适应调整观测量协方差矩阵
    dAcc = abs(norm(acc(:,jj)) - 9.7934);
    if dAcc < 0.1
        dAcc = 0.1;
    end
    dMag = abs(norm(mag(:,jj)) - 0.45);
    if dMag < 0.2
        dMag = 0.2;
    elseif dMag > 1.0
        dMag = 100.0;
    end
    R = diag( [dAcc, dAcc, 0.2+10*(dMag-0.2)].^2 );
    % 卡尔曼滤波器
    [x,P] = kf(x,P,F,Q,G,u,z,H,R);

    % 误差修正
    dAtt = x(1:3);
    x(1:3) = zeros(3,1);
    dAtt_cross = crossForm(dAtt);
    C_b2n = C_b2n*(eye(3)+dAtt_cross); % +1/2*dAtt_cross*dAtt_cross
    % 正交化
    if mod(jj,10) == 0
        C_b2n = ( 1.5*eye(3) - 0.5*C_b2n*(C_b2n') )*C_b2n;
    end
    % 计算姿态角
    [att(1,jj), att(2,jj), att(3,jj)] = attCalc(C_b2n);
```

```
    %% 计算速度
    vel(:,jj) = C_b2n*[0;speed(jj);0];

    %% 计算位置
    dX = (vel_2 + 4*vel_1 + vel(:,jj))/6*dT;
    vel_2 = vel_1;
    vel_1 = vel(:,jj);
    F_curv = curvMat(C_n2e,hgt,1);
    zeta = F_curv*dX;
    zeta_cross = crossForm(zeta);
    C_n2e = C_n2e*(eye(3)+zeta_cross);
    hgt = hgt + dX(3);
    [pos(1,jj),pos(2,jj),~] = posCalc(C_n2e);
    pos(3,jj) = hgt;

end

% plotTrace;
% save('navResults.mat','len','time','pos','vel','att');
```

16) ephemeris

```
function [eph, TOW] = ephemeris(bits,D29Star,D30Star)
%Function decodes ephemerides and TOW from the given bit stream. The stream
%(array) in the parameter BITS must contain 1500 bits. The first element
%in the array must be the first bit of a subframe. The subframe ID of
%the first subframe in the array is not important.
%%
%[eph, TOW] = ephemeris(bits, D30Star)
%
%   Inputs:
%       bits       - bits of the navigation messages (5 subframes).
%                 Type is character array and it must contain only
%                 characters '0' or '1'.
%       D29Star     - The 29th bit of the previous nav-word. Refer to the
%                 GPS interface control document ICD (IS-GPS-200D) for
%                 more details on the parity checking. Parameter type is
%                 char. It must contain only characters '0' or '1'.
%       D30Star     - The 30th (last) bit of the previous nav-word. Refer
to
%                 the GPS interface control document ICD (IS-GPS-200D) for
%                 more details on the parity checking. Parameter type is
%                 char. It must contain only characters '0' or '1'.
```

```
%   Outputs:
%       TOW         - Time Of Week (TOW) of the first sub-frame in the bit
%                     stream (in seconds)
%       eph         - SV ephemeris
%--------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis and Kristin Larson
% Written by Darius Plausinaitis and Kristin Larson
% Modified by Xiaofan Li at University of Colorado at Boulder
%--------------------------------------------------------------------

%% Check if there is enough data ===================================
if length(bits) < 1500
    error('The parameter BITS must contain 1500 bits!');
end

%% Check if the parameters are strings =============================
if ~ischar(bits)
    error('The parameter BITS must be a character array!');
end

if ~ischar(D29Star)
    error('The parameter D29Star must be a char!');
end

if ~ischar(D30Star)
    error('The parameter D30Star must be a char!');
end

% Pi used in the GPS coordinate system
gpsPi = 3.1415926535898;
%% Decode all 5 sub-frames =========================================
for i = 1:5

    %--- "Cut" one sub-frame's bits ------------------------------
    subframe = bits(300*(i-1)+1 : 300*i);

    %--- Correct polarity of the data bits in all 10 words ----------
    for j = 1:10
        [subframe(30*(j-1)+1 : 30*j)] = ...
            parityCheck(subframe(30*(j-1)+1 : 30*j), D29Star, D30Star);
        D29Star = subframe(30*j-1);
        D30Star = subframe(30*j);
```

```
    end

    %--- Decode the sub-frame id -------------------------------
    % For more details on sub-frame contents please refer to GPS IS.
    subframeID = bin2dec(subframe(50:52));
    % If a wrong subframe ID is still found after the parity check, then
    % the IF Data used in the post-processing is not reliable
    if subframeID >5 || subframeID <1
        error('Wrong subframe ID , IF Data is not reliable!');
    end

    %--- Decode sub-frame based on the sub-frames id -------------
    % The task is to select the necessary bits and convert them to decimal
    % numbers. For more details on sub-frame contents please refer to GPS
    % ICD (IS-GPS-200D).
    switch subframeID
        case 1  %--- It is subframe 1 -----------------------------
            % It contains WN, SV clock corrections, health and accuracy
            eph.weekNumber = bin2dec(subframe(61:70)) + 1024;
            eph.accuracy = bin2dec(subframe(73:76));
            eph.health = bin2dec(subframe(77:82));
            eph.T_GD = twosComp2dec(subframe(197:204)) * 2^(-31);
            eph.IODC = bin2dec([subframe(83:84) subframe(197:204)]);
            eph.t_oc = bin2dec(subframe(219:234)) * 2^4;
            eph.a_f2 = twosComp2dec(subframe(241:248)) * 2^(-55);
            eph.a_f1 = twosComp2dec(subframe(249:264)) * 2^(-43);
            eph.a_f0 = twosComp2dec(subframe(271:292)) * 2^(-31);

        case 2  %--- It is subframe 2 -----------------------------
            % It contains first part of ephemeris parameters
            eph.IODE_sf2 = bin2dec(subframe(61:68));
            eph.C_rs = twosComp2dec(subframe(69: 84)) * 2^(-5);
            eph.deltan = twosComp2dec(subframe(91:106)) * 2^(-43) * gpsPi;
            eph.M_0 = twosComp2dec([subframe(107:114) subframe(121:144)])
* 2^(-31) * gpsPi;
            eph.C_uc = twosComp2dec(subframe(151:166)) * 2^(-29);
            eph.e = bin2dec([subframe(167:174) subframe(181:204)]) *
2^(-33);
            eph.C_us = twosComp2dec(subframe(211:226)) * 2^(-29);
            eph.sqrtA = bin2dec([subframe(227:234) subframe(241:264)]) *
2^(-19);
            eph.t_oe = bin2dec(subframe(271:286)) * 2^4;
```

```
       case 3  %--- It is subframe 3 ----------------------------
          % It contains second part of ephemeris parameters
          eph.C_ic = twosComp2dec(subframe(61:76)) * 2^(-29);
          eph.omega_0 = twosComp2dec([subframe(77:84)
subframe(91:114)]) * 2^(-31) * gpsPi;
          eph.C_is = twosComp2dec(subframe(121:136)) * 2^(-29);
          eph.i_0 = twosComp2dec([subframe(137:144) subframe(151:174)])
* 2^(-31) * gpsPi;
          eph.C_rc = twosComp2dec(subframe(181:196)) * 2^(-5);
          eph.omega = twosComp2dec([subframe(197:204)
subframe(211:234)]) * 2^(-31) * gpsPi;
          eph.omegaDot = twosComp2dec(subframe(241:264)) * 2^(-43) *
gpsPi;
          eph.IODE_sf3 = bin2dec(subframe(271:278));
          eph.iDot = twosComp2dec(subframe(279:292)) * 2^(-43) * gpsPi;

       case 4  %--- It is subframe 4 ----------------------------
          % Almanac, ionospheric model, UTC parameters.
          % SV health (PRN: 25-32).
          % Not decoded at the moment.

       case 5  %--- It is subframe 5 ----------------------------
          % SV almanac and health (PRN: 1-24).
          % Almanac reference week number and time.
          % Not decoded at the moment.

   end % switch subframeID ...

end % for all 5 sub-frames ...

%% Compute the time of week (TOW) of the first sub-frames in the array
% Also correct the TOW. The transmitted TOW is actual TOW of the next
% subframe and we need the TOW of the first subframe in this data block
% (the variable subframe at this point contains bits of the last subframe).
TOW = bin2dec(subframe(31:47)) * 6 - 30;
```

17) ephemerisBD

```
function [eph, SOW] = ephemerisBD(bits)
%Function decodes ephemerides and TOW from the given bit stream. The stream
%(array) in the parameter BITS must contain 1500 bits. The first element
```

```
in
%the array must be the first bit of a subframe. The subframe ID of the
%first subframe in the array is not important.
%
%[eph, TOW] = ephemerisBD(bits)
%
%   Inputs:
%       bits       - bits of the navigation messages (5 subframes).
%                  Type is character array and it must contain only
%                  characters '0' or '1'.
%   Outputs:
%       TOW        - Time Of Week (TOW) of the first sub-frame in the bit
%                  stream (in seconds)
%       eph        - SV ephemeris
%--------------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis and Kristin Larson
% Written by Mwenjie @2014.03
% Based on SoftGNSS v3.0
%--------------------------------------------------------------------------
%% Check if there is enough data ===========================================
if length(bits) < 1500
    error('The parameter BITS must contain 1500 bits!');
end

%% Check if the parameters are strings =====================================
if ~ischar(bits)
    error('The parameter BITS must be a character array!');
end

% Pi used in the GPS coordinate system
gnssPi = 3.1415926535898;

% Initialize eph
eph.weekNumber  = [];    % + 1024;
eph.accuracy    = [];    % URAI
eph.health      = [];      %
eph.alpha       = [];
eph.beta        = [];
eph.T_GD        = [];
eph.IODC        = [];
eph.t_oc        = [];
eph.a_f0        = [];
```

```
eph.a_f1       = [];
eph.a_f2       = [];
eph.IODE       = [];
eph.t_oe       = [];
eph.sqrtA      = [];
eph.e          = [];
eph.omega      = [];
eph.deltan     = [];
eph.M_0        = [];
eph.omega_0    = [];
eph.omegaDot   = [];
eph.i_0        = [];
eph.iDot       = [];
eph.C_uc       = [];
eph.C_us       = [];
eph.C_rc       = [];
eph.C_rs       = [];
eph.C_ic       = [];
eph.C_is       = [];

%% Decode all 5 sub-frames ========================================
subframeBak = zeros(5,300);
for i = 1:5

    %--- "Cut" one sub-frame's bits ---------------------------------
    subframe = bits(300*(i-1)+1 : 300*i);

    %--- BCH check of the data bits in all 10 words ----------------
    [subframe(1 : 30)] = bch1Check(subframe(1 : 30));
    for j = 2:10
        [subframe(30*(j-1)+1 : 30*j)] = ...
            bch2Check(subframe(30*(j-1)+1 : 30*j));
    end

    %--- Decode the sub-frame id -----------------------------------
    % For more details on sub-frame contents please refer to GPS IS.
    subframeID = bin2dec(subframe(16:18));
    % If a wrong subframe ID is still found after the parity check, then
    % the IF Data used in the post-processing is not reliable
    if subframeID >5 || subframeID <1
        error('Wrong subframe ID , IF Data is not reliable!');
    end
```

```
    % backup
    subframeBak(subframeID,:) = subframe;

    %--- Decode sub-frame based on the sub-frames id ----------------
    % The task is to select the necessary bits and convert them to decimal
    % numbers. For more details on sub-frame contents please refer to GPS
    % ICD (IS-GPS-200D).
    switch subframeID
        case 1  %--- It is subframe 1 -------------------------------
            % It contains WN, SV clock corrections, health and accuracy
            eph.weekNumber = bin2dec(subframe(61:73));    % + 1024;
            eph.accuracy   = bin2dec(subframe(49:52));    % URAI
            eph.health     = bin2dec(subframe(43));
            eph.T_GD       = twosComp2dec(subframe(99:108)) * 1e-10;
            eph.IODC       = bin2dec(subframe(44:48));
            eph.t_oc       = bin2dec([subframe(74:82) subframe(91:98)]) *
2^3;
            eph.a_f2       = twosComp2dec(subframe(215:225)) * 2^(-66);
            eph.a_f1       = twosComp2dec([subframe(226:232)
subframe(241:257)]) * 2^(-50);
            eph.a_f0       = twosComp2dec([subframe(258:262)
subframe(271:287)]) * 2^(-30);

            alpha0         = twosComp2dec(subframe(127:134)) * 2^(-30);
            alpha1         = twosComp2dec(subframe(135:142)) * 2^(-27);
            alpha2         = twosComp2dec(subframe(151:158)) * 2^(-24);
            alpha3         = twosComp2dec(subframe(159:166)) * 2^(-24);
            beta0          = twosComp2dec([subframe(167:172)
subframe(181:182)]) * 2^(11);
            beta1          = twosComp2dec(subframe(183:190)) * 2^(14);
            beta2          = twosComp2dec(subframe(191:198)) * 2^(16);
            beta3          = twosComp2dec([subframe(199:202)
subframe(211:214)]) * 2^(16);
            eph.alpha   = [ alpha0 alpha1 alpha2 alpha3 ];
            eph.beta    = [ beta0 beta1 beta2 beta3 ];
            eph.IODE       = bin2dec(subframe(288:292));

        case 2  %--- It is subframe 2 -------------------------------
            % It contains first part of ephemeris parameters
            eph.deltan = twosComp2dec([subframe(43:52) subframe(61:66)])
* 2^(-43) * gnssPi;
            eph.C_uc = twosComp2dec([subframe(67:82) subframe(91:92)]) *
```

```
2^(-31);
          eph.M_0 = twosComp2dec([subframe(93:112) subframe(121:132)])
* 2^(-31) * gnssPi;
          eph.e = bin2dec([subframe(133:142) subframe(151:172)]) *
2^(-33);
          eph.C_us = twosComp2dec(subframe(181:198)) * 2^(-31);
          eph.C_rc = twosComp2dec([subframe(199: 202) subframe(211:
224)]) * 2^(-6);
          eph.C_rs = twosComp2dec([subframe(225: 232) subframe(241:
250)]) * 2^(-6);
          eph.sqrtA = bin2dec([subframe(251:262) subframe(271:290)]) *
2^(-19);
          eph.t_oe = bin2dec([subframe(291:292) subframeBak(3,43:52)
subframeBak(3,61:65)]) * 2^3;

       case 3  %--- It is subframe 3 ---------------------------------
          % It contains second part of ephemeris parameters
          eph.t_oe = bin2dec([subframeBak(2,291:292) subframe(43:52)
subframe(61:65)]) * 2^3;
          eph.i_0 = twosComp2dec([subframe(66:82) subframe(91:105)]) *
2^(-31) * gnssPi;
          eph.C_ic = twosComp2dec([subframe(106:112)
subframe(121:131)]) * 2^(-31);
          eph.omegaDot = twosComp2dec([subframe(132:142)
subframe(151:163)]) * 2^(-43) * gnssPi;
          eph.C_is = twosComp2dec([subframe(164:172)
subframe(181:189)]) * 2^(-31);
          eph.iDot = twosComp2dec([subframe(190:202) subframe(211)]) *
2^(-43) * gnssPi;
          eph.omega_0 = twosComp2dec([subframe(212:232)
subframe(241:251)]) * 2^(-31) * gnssPi;
          eph.omega = twosComp2dec([subframe(252:262)
subframe(271:291)]) * 2^(-31) * gnssPi;

       case 4  %--- It is subframe 4 ----------------------------
          % Almanac, ionospheric model, UTC parameters.
          % SV health (PRN: 25-32).
          % Not decoded at the moment.

       case 5  %--- It is subframe 5 ---------------------------------
          % SV almanac and health (PRN: 1-24).
          % Almanac reference week number and time.
```

```
            % Not decoded at the moment.

    end % switch subframeID ...

end % for all 5 sub-frames ...

%% Compute the time of week (TOW) of the first sub-frames in the array
% Also correct the TOW. The transmitted TOW is actual TOW of the next
% subframe and we need the TOW of the first subframe in this data block
% (the variable subframe at this point contains bits of the last subframe).
SOW = bin2dec([subframe(19:26) subframe(31:42)]) - 24;
```

18) ephemerisBDGEO

```
function [eph, SOW] = ephemerisBDGEO(bits)
%Function decodes ephemerides and TOW from the given bit stream. The stream
%(array) in the parameter BITS must contain 1500 bits. The first element
in
%the array must be the first bit of a subframe. The subframe ID of the
%first subframe in the array is not important.
%
%
%[eph, TOW] = ephemerisBDGEO(bits)
%
%   Inputs:
%       bits       - bits of the navigation messages (5 subframes).
%                  Type is character array and it must contain only
%                  characters '0' or '1'.
%   Outputs:
%       TOW        - Time Of Week (TOW) of the first sub-frame in the bit
%                  stream (in seconds)
%       eph        - SV ephemeris
%--------------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis and Kristin Larson
% Written by Mwenjie @2014.03
% Based on SoftGNSS v3.0
%--------------------------------------------------------------------------

%% Check if there is enough data ===========================================
if length(bits) < 1500*10
    error('The parameter BITS must contain 1500 bits!');
end
```

```
%% Check if the parameters are strings ===============================
if ~ischar(bits)
    error('The parameter BITS must be a character array!');
end

% Pi used in the GPS coordinate system
gnssPi = 3.1415926535898;

% Initialize eph
eph.weekNumber  = [];    % + 1024;
eph.accuracy    = [];    % URAI
eph.health      = [];      %
eph.alpha       = [];
eph.beta        = [];
eph.T_GD        = [];
eph.IODC        = [];
eph.t_oc        = [];
eph.a_f0        = [];
eph.a_f1        = [];
eph.a_f2        = [];
eph.IODE        = [];
eph.t_oe        = [];
eph.sqrtA       = [];
eph.e          = [];
eph.omega       = [];
eph.deltan      = [];
eph.M_0        = [];
eph.omega_0     = [];
eph.omegaDot    = [];
eph.i_0        = [];
eph.iDot        = [];
eph.C_uc        = [];
eph.C_us        = [];
eph.C_rc        = [];
eph.C_rs        = [];
eph.C_ic        = [];
eph.C_is        = [];

%% Decode all 5 sub-frames ========================================
subframe1Bak = zeros(10,300);
for pageCnt = 1:10
    for i = 1:5
```

```
        %--- "Cut" one sub-frame's bits --------------------------------
        subframe = bits( 300*((pageCnt-1)*5+i-1)+1 :
300*((pageCnt-1)*5+i) );

        %--- BCH check of the data bits in all 10 words ----------------
        [subframe(1 : 30)] = bch1Check(subframe(1 : 30));
        for j = 2:10
            [subframe(30*(j-1)+1 : 30*j)] = ...
                bch2Check(subframe(30*(j-1)+1 : 30*j));
        end

        %--- Decode the sub-frame id -----------------------------------
        % For more details on sub-frame contents please refer to GPS IS.
        subframeID = bin2dec(subframe(16:18));
        % If a wrong subframe ID is still found after the parity check,
then
        % the IF Data used in the post-processing is not reliable
        if subframeID >5 || subframeID <1
            error('Wrong subframe ID , IF Data is not reliable!');
        end

        %--- Decode sub-frame based on the sub-frames id ----------------
        % The task is to select the necessary bits and convert them to decimal
        % numbers. For more details on sub-frame contents please refer to
GPS
        % ICD (IS-GPS-200D).
        switch subframeID
            case 1  %--- It is subframe 1 -----------------------------
                Pnum = bin2dec(subframe(43:46));
                % backup
                subframe1Bak(Pnum,:) = subframe;

                switch Pnum
                    case 1    % Page 1
                        eph.health = bin2dec(subframe(47));       %
                        eph.IODC = bin2dec(subframe(48:52));
                        eph.accuracy = bin2dec(subframe(61:64));  % URAI
                        eph.weekNumber = bin2dec(subframe(65:77));
                        eph.t_oc = bin2dec([subframe(78:82)
subframe(91:102)]) * 2^3;
                    eph.T_GD1 = twosComp2dec(subframe(103:112)) * 1e-10;
```

```
                case 2    % Page 2
                    alpha0 = twosComp2dec([subframe(47:52)
subframe(61:62)]) * 2^(-30);
                    alpha1 = twosComp2dec(subframe(63:70)) * 2^(-27);
                    alpha2 = twosComp2dec(subframe(71:78)) * 2^(-24);
                    alpha3 = twosComp2dec([subframe(79:82)
subframe(91:94)]) * 2^(-24);
                    beta0 = twosComp2dec(subframe(95:102)) * 2^(11);
                    beta1 = twosComp2dec(subframe(103:110)) * 2^(14);
                    beta2 = twosComp2dec([subframe(111:112)
subframe(121:126)]) * 2^(16);
                    beta3 = twosComp2dec(subframe(127:134)) * 2^(16);
                    eph.alpha = [ alpha0 alpha1 alpha2 alpha3 ];
                    eph.beta = [ beta0 beta1 beta2 beta3 ];

                case 3    % Page 3
                    eph.a_f0 = twosComp2dec([subframe(101:112)
subframe(121:132)]) * 2^(-30);
                    eph.a_f1 = twosComp2dec([subframe(133:136) ...
subframe1Bak(4,47:52) subframe1Bak(4,61:72)]) * 2^(-50);

                case 4    % Page 4
                    eph.a_f1 = twosComp2dec([subframe1Bak(3,133:136)
                       subframe(47:52) subframe(61:72)]) * 2^(-50);
                    eph.a_f2 = twosComp2dec([subframe(73:82)
subframe(91)]) * 2^(-66);
                    eph.IODE = bin2dec(subframe(92:96));
                    eph.deltan =
twosComp2dec(subframe(97:112)) * 2^(-43) * gnssPi;
                    eph.C_uc = twosComp2dec([subframe(121:134) ...
                       subframe1Bak(5,47:50)]) * 2^(-31);

                case 5    % Page 5
                    eph.C_uc = twosComp2dec([subframe1Bak(4,121:134)
                       subframe(47:50)]) * 2^(-31);
                    eph.M_0 =
twosComp2dec([subframe(51:52) subframe(61:82) subframe(91:92)]) *
2^(-31) * gnssPi;
                    eph.C_us = twosComp2dec([subframe(99:112)
subframe(121:124)]) * 2^(-31);
                    eph.e = bin2dec([subframe(125:134)
```

```
subframe1Bak(6,47:52) subframe1Bak(6,61:76)]) * 2^(-33);

                case 6   % Page 6
                    eph.e =  bin2dec([subframe1Bak(5,125:134)
subframe(47:52) subframe(61:76)]) * 2^(-33);
                    eph.sqrtA = bin2dec([subframe(77:82)
subframe(91:112) subframe(121:124)]) * 2^(-19);
                    eph.C_ic = twosComp2dec([subframe(125:134)
subframe1Bak(7,47:52) subframe1Bak(7,61:62)]) * 2^(-31);

                case 7   % Page 7
                    eph.C_ic = twosComp2dec([subframe1Bak(6,125:134)
subframe(47:52) subframe(61:62)]) * 2^(-31);
                    eph.C_is = twosComp2dec(subframe(63:80)) * 2^(-31);
                    eph.t_oe = bin2dec([subframe(81:82)
subframe(91:105)]) * 2^3;
                    eph.i_0 = twosComp2dec([subframe(106:112)
subframe(121:134) subframe1Bak(8,47:52) subframe1Bak(8,61:65) ]) *
2^(-31) * gnssPi;

                case 8   % Page 8
                    eph.i_0 = twosComp2dec([subframe1Bak(7,106:112)
subframe1Bak(7,121:134) subframe(47:52) subframe(61:65) ])  * 2^(-31) *
gnssPi;
                    eph.C_rc = twosComp2dec([subframe(66: 82)
subframe(91)]) * 2^(-6);
                    eph.C_rs = twosComp2dec(subframe(92: 109)) * 2^(-6);
                    eph.omegaDot = twosComp2dec([subframe(110:112)
subframe(121:136) subframe1Bak(9,47:51)]) * 2^(-43) * gnssPi;

                case 9   % Page 9
                    eph.omegaDot =
twosComp2dec([subframe1Bak(8,110:112) subframe1Bak(8,121:136)
subframe(47:51)]) * 2^(-43) * gnssPi;
                    eph.omega_0 =
twosComp2dec([subframe(52) subframe(61:82) subframe(91:99)])
* 2^(-31) * gnssPi;
                    eph.omega = twosComp2dec([subframe(100:112)
subframe(121:134) subframe1Bak(10,47:51) ]) * 2^(-31) * gnssPi;

                case 10   % Page 10
                    eph.omega = twosComp2dec([subframe1Bak(9,100:112)
```

```
subframe1Bak(9,121:134) subframe(47:51) ]) * 2^(-31) * gnssPi;
                    eph.iDot = twosComp2dec([subframe(52)
subframe(61:73)]) * 2^(-43) * gnssPi;
             end

          case 2  %--- It is subframe 2 ---------------------------
          case 3  %--- It is subframe 3 -------------------------
          case 4  %--- It is subframe 4 ----------------------
             % Almanac, ionospheric model, UTC parameters.
             % SV health (PRN: 25-32).
             % Not decoded at the moment.
          case 5  %--- It is subframe 5 --------------------
             % SV almanac and health (PRN: 1-24).
             % Almanac reference week number and time.
             % Not decoded at the moment.
       end % switch subframeID ...
    end % for all 5 sub-frames ...
end

%% Compute the time of week (TOW) of the first sub-frames in the array
====
% Also correct the TOW. The transmitted TOW is actual TOW of the next
% subframe and we need the TOW of the first subframe in this data block
% (the variable subframe at this point contains bits of the last subframe).
SOW = bin2dec([subframe(19:26) subframe(31:42)]) - 29.4 +
(subframeID-1)*0.6;
```

19) e_r_corr

```
function X_sat_rot = e_r_corr(traveltime, X_sat)
%E_R_CORR  Returns rotated satellite ECEF coordinates due to Earth
%rotation during signal travel time
%
%X_sat_rot = e_r_corr(traveltime, X_sat);
%
%   Inputs:
%       travelTime  - signal travel time
%       X_sat       - satellite's ECEF coordinates
%
%   Outputs:
%       X_sat_rot   - rotated satellite's coordinates (ECEF)

%Written by Kai Borre
```

```
%Copyright (c) by Kai Borre
%
% CVS record:
% $Id: e_r_corr.m,v 1.1.1.1.2.6 2006/08/22 13:45:59 dpl Exp $
%=========================================================================

Omegae_dot = 7.292115147e-5;         %  rad/sec

%--- Find rotation angle -------------------------------------------
omegatau   = Omegae_dot * traveltime;

%--- Make a rotation matrix ----------------------------------
R3 = [ cos(omegatau)    sin(omegatau)   0;
     -sin(omegatau)    cos(omegatau)   0;
      0             0             1];

%--- Do the rotation -----------------------------------------
X_sat_rot = R3 * X_sat;
%%%%%%%% end e_r_corr.m %%%%%%%%%%%%%%%%%%%%%%
```

20) findPreambles

```
function [firstSubFrame, isReverse, activeChnList] =
findPreambles(trackResults, settings)
% findPreambles finds the first preamble occurrence in the bit stream of
% each channel. The preamble is verified by check of the spacing between
% preambles (6sec) and parity checking of the first two words in a
% subframe. At the same time function returns list of channels, that are
in
% tracking state and with valid preambles in the nav data stream.
%
%[firstSubFrame, activeChnList] = findPreambles(trackResults, settings)
%
%   Inputs:
%       trackResults    - output from the tracking function
%       settings        - Receiver settings.
%
%   Outputs:
%       firstSubframe   - the array contains positions of the first
%                       preamble in each channel. The position is ms count
%                       since start of tracking. Corresponding value will
%                       be set to 0 if no valid preambles were detected in
%                       the channel.
```

```
%      activeChnList   - list of channels containing valid preambles
%--------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis, Peter Rinder and Nicolaj Bertelsen
% Written by Darius Plausinaitis, Peter Rinder and Nicolaj Bertelsen
% Modified by Mwenjie @2014.03
%--------------------------------------------------------------------

% Preamble search can be delayed to a later point in the tracking results
% to avoid noise due to tracking loop transients
searchStartOffset = 0;

%--- Initialize the firstSubFrame array
-----------------------------------
firstSubFrame = zeros(1, settings.numberOfChannels);
isReverse = ones(1, settings.numberOfChannels);

%--- Generate the preamble pattern
---------------------------------------
switch settings.system
    case 1  % GPS
        preamble_bits = [1 -1 -1 -1 1 -1 1 1];
    case 2  % Beidou
        preamble_bits = [1 1 1 -1 -1 -1 1 -1 -1 1 -1];
end

%--- Make a list of channels excluding not tracking channels --------------
activeChnList = find([trackResults.status] ~= '-');

%=== For all tracking channels ...
for channelNr = activeChnList
    % Generate Secondary Code
    switch settings.system
        case 1  % GPS
            msPerBit = 20;
            secondaryCode = ones(1,msPerBit);
            xcorrTheshold = 153;
        case 2  % Beidou
            if trackResults(channelNr).PRN <= 5
                msPerBit = 2;
                secondaryCode = ones(1,msPerBit);
                xcorrTheshold = 20;
            else
```

```
                msPerBit = 20;
                % Neumann-Hoffman (NH) code
                secondaryCode = [-1 -1 -1 -1 -1 1 -1 -1 1 1 -1 1 -1 1 -1
-1 1 1 1 -1];
                xcorrTheshold = 213;
            end
        end

        % "Upsample" the preamble - make 20 values per one bit. The preamble
must be
        % found with precision of a sample.
        preamble_ms = kron(preamble_bits, secondaryCode);

    %% Correlate tracking output with preamble ==========================
        % Read output from tracking. It contains the navigation bits. The start
        % of record is skiped here to avoid tracking loop transients.
        bits = trackResults(channelNr).I_P(1 + searchStartOffset : end);

        % Now threshold the output and convert it to -1 and +1
        bits(bits > 0)  =  1;
        bits(bits <= 0) = -1;

        % Correlate tracking output with the preamble
        tlmXcorrResult = xcorr(bits, preamble_ms);

    %% Find all starting points off all preamble like patterns ===============
        clear index
        clear index2

        xcorrLength = (length(tlmXcorrResult) +  1) /2;
        xcorrResult = tlmXcorrResult(xcorrLength : xcorrLength * 2 - 1);

        %--- Find at what index/ms the preambles start --------------------
        index = find( abs(xcorrResult) > xcorrTheshold )' + searchStartOffset;

    %% Analyze detected preamble like patterns ==========================
        for i = 1:size(index) % For each occurrence

            %--- Find distances in time between this occurrence and the rest
of
            %preambles like patterns. If the distance is 6000 milliseconds (one
            %subframe), the do further verifications by validating the parities
```

```
        %of two GPS words

        index2 = index - index(i);

        if (~isempty(find(index2 == 300*msPerBit, 1)))

            %=== Re-read bit vales for preamble verification ==============
            % Preamble occurrence is verified by checking the parity of
            % the first two words in the subframe. Now it is assumed that
            % bit boundaries a known. Therefore the bit values over 20ms
are
            % combined to increase receiver performance for noisy signals.
            % in Total 62 bits mast be read :
            % 2 bits from previous subframe are needed for parity checking;
            % 60 bits for the first two 30bit words (TLM and HOW words).
            % The index is pointing at the start of TLM word.
            switch settings.system
                case 1  % GPS
                    bits = trackResults(channelNr).I_P(index(i)-40 : ...
                                                index(i) + 20 * 60 -1)';

                    %--- Combine the 20 values of each bit ----------------
                    bits = reshape(bits, msPerBit, (size(bits, 1) /
msPerBit));
                    bits = sum(bits);

                    % Now threshold and make it -1 and +1
                    bits(bits > 0)  = 1;
                    bits(bits <= 0) = -1;

                    %--- Check the parity of the TLM and HOW words
----------------
                    if (navPartyChk(bits(1:32)) ~= 0) && ...
                       (navPartyChk(bits(31:62)) ~= 0)
                        % Parity was OK. Record the preamble start position.
Skip
                        % the rest of preamble pattern checking for this
channel
                        % and process next channel.
                        firstSubFrame(channelNr) = index(i);
                        break;
                    end % if parity is OK ...
```

```
            case 2  % Beidou
                firstSubFrame(channelNr) = mod(index(i),6000);
                break;
        end

    end % if (~isempty(find(index2 == 6000)))
end % for i = 1:size(index)

% Exclude channel from the active channel list if no valid preamble
was
% detected
if firstSubFrame(channelNr) ~= 0
    if xcorrResult(firstSubFrame(channelNr)) < 0
        isReverse(channelNr) = -1;
    end
else
    % Exclude channel from further processing. It does not contain any
    % valid preamble and therefore nothing more can be done for it.
    activeChnList = setdiff(activeChnList, channelNr);

    disp(['Could not find valid preambles in channel ', ...
        num2str(channelNr),'!']);
end

end % for channelNr = activeChnList
```

21) findTransTime

```
function transmitTime =
findTransTime(sampleNum,readyChnList,svTimeTable,trackResults,setting
s,offset)
% findTransTime finds the transmitting time of each satellite at a
specified
% sample number using the interpolation
% function transmitTime=...
%   findTransTime(sampleNum,readyChnList,svTimeTable,trackResults)
%  Inputs:
%     sampleNum    - absolute sample number from the tracking loop
%     readyChnList  - a list of satellites ready for nav solution
%     svTimeTable   - the transmitting time table
%     trackResults  - output from the tracking function
%
%  Outputs:
```

```
%      transmitTime  - transmitting time all ready satellites
%--------------------------------------------------------------------------
% Copyright (C) D.M.Akos
% Written by Xiaofan Li at University of Colorado at Boulder
%--------------------------------------------------------------------------
 % Initialize the transmitting time
transmitTime=zeros(1,settings.numberOfChannels);
% transmitTime=zeros(1,length(readyChnList));

% Calculate the transmitting time of each satellite using interpolations
for channelNr = readyChnList
    % Find the index of the sampleNum in the tracking results
    index_a=find(trackResults(channelNr).absoluteSample<=sampleNum, 1,
'last' );
    index_b=find(trackResults(channelNr).absoluteSample>=sampleNum,
1 );
    x1=trackResults(channelNr).absoluteSample(index_a:index_b);
    y1=[index_a,index_b];
    index_c=offset + interp1(x1,y1,sampleNum);
    x2=offset + [index_a,index_b];
    y2=svTimeTable(channelNr).time(x2);
    % Find the transmitting time based on the index calculated
    transmitTime(channelNr)=interp1(x2,y2,index_c);
end
```

22) fukf

```
function [x,P,N] = fukf(x,P,Q,fkfData,dt)
 % 紧耦合数据融合
L = length(x);
N = sum(fkfData.flag);

% if N == 0
%    return;
% end

%% Calculate sigma points
alpha = 1e-3;
beta = 2;
kappa = 0;

lamda = alpha*alpha*(L+kappa) - L;
c1 = 0.5/(L+lamda);
```

```
c2 = (lamda-0.5)/(L+lamda);
c3 = c2+1-alpha*alpha+beta;

% Generate sigma point
x_1 = x;
if N == 0
    X_1 = repmat(x_1,1,2*L+1) + sqrt(L+lamda)*[zeros(L,1),
chol(P,'lower'),...
       -chol(P,'lower')];
else
    X_1 = repmat(x_1,1,2*L+1) + sqrt(L+lamda)*[zeros(L,1),
chol(N*P,'lower'),...
       -chol(N*P,'lower')];
end

%% Time update -- predict

% Integ_f, to get X_
X_ = zeros(L,2*L+1);
for ii = 1:2*L+1
    pos = X_1(1:3,ii);
    vel = X_1(4:6,ii);
    b = X_1(7,ii);
    d = X_1(8,ii);

    pos = pos + vel*dt;
    b = b + d*dt;

    X_(1:3,ii) = pos;
    X_(4:6,ii) = vel;
    X_(7,ii) = b;
    X_(8,ii) = d;
end

x_ = c1*sum(X_,2) + c2*X_(:,1);
dX_ = X_ - repmat(x_,1,2*L+1);
dx_ = X_(:,1) - x_;

if N == 0
    x = x_;
    P = c1*(dX_*dX_') + c3*(dx_*dx_') + Q;
    return;
```

```
else
    P_ = c1*(dX_*dX_') + c3*(dx_*dx_') + N*Q;
end


%% Initialize master filter
P_mm = zeros(size(P_));
x_m = zeros(size(x));

%% Local filters
MAX_SV = length(fkfData.flag);

Z_ = zeros(2,2*L+1);
for ii = 1:MAX_SV
    if fkfData.flag(ii) == 0
        continue;
    end

    % Tight_h, to get Z_
    for jj = 1:2*L+1
        r_e = X_(1:3,jj);
        v_e = X_(4:6,jj);

        rho = r_e-fkfData.r_sat(:,ii);
        los = rho/norm(rho);
        Z_(1,jj) = norm(rho) + X_(7,jj);                    % pseudorange
        Z_(2,jj) = dot(v_e-fkfData.v_sat(:,ii),los) + X_(8,jj); %
deltarange
    end

    z_ = c1*sum(Z_,2) + c2*Z_(:,1);

    %% Measurement update -- correct
    dZ_ = Z_ - repmat(z_,1,2*L+1);
    dz_ = Z_(:,1) - z_;
    Pzz = c1*(dZ_*dZ_') + c3*(dz_*dz_') + fkfData.R(:,:,ii); %R;
    Pxz = c1*(dX_*dZ_') + c3*(dx_*dz_');

    K = Pxz / Pzz;
    x1 = x_ + K*( fkfData.z(:,ii) - z_ );
    P1 = P_ - K*Pzz*K';
```

```
    P_mm = P_mm + inv(P1);
    x_m = x_m + P1\x1;

end

%% Master filter
P = inv(P_mm);
x = P_mm\x_m;
```

23) fukf_utc

```
function [x,P,N] = fkf_utc(x,P,Q,fkfData,sinsBak,dt)

L = length(x);
N = sum(fkfData.flag);

% if N == 0
%     return;
% end

%% Calculate sigma points
alpha = 1e-3;
beta = 2;
kappa = 0;

lamda = alpha*alpha*(L+kappa) - L;
c1 = 0.5/(L+lamda);
c2 = (lamda-0.5)/(L+lamda);
c3 = c2+1-alpha*alpha+beta;

% Generate sigma point
x_1 = x;
if N == 0
    X_1 = repmat(x_1,1,2*L+1) + sqrt(L+lamda)*[zeros(L,1),
chol(P,'lower'),...
        -chol(P,'lower')];
else
    X_1 = repmat(x_1,1,2*L+1) + sqrt(L+lamda)*[zeros(L,1),
chol(N*P,'lower'),...
        -chol(N*P,'lower')];
end

%% Time update -- predict
```

```
% Integ_f, to get X_
X_ = zeros(L,2*L+1);
for ii = 1:2*L+1
    dPos = X_1(1:3,ii);
    dVel = X_1(4:6,ii);
    b = X_1(7,ii);
    d = X_1(8,ii);

    dPos = dPos + dVel*dt;
    b = b + d*dt;

    X_(1:3,ii) = dPos;
    X_(4:6,ii) = dVel;
    X_(7,ii) = b;
    X_(8,ii) = d;
end

x_ = c1*sum(X_,2) + c2*X_(:,1);
dX_ = X_ - repmat(x_,1,2*L+1);
dx_ = X_(:,1) - x_;

if N == 0
    x = x_;
    P = c1*(dX_*dX_') + c3*(dx_*dx_') + Q;
    return;
else
    P_ = c1*(dX_*dX_') + c3*(dx_*dx_') + N*Q;
end

%% Initialize master filter
P_mm = zeros(size(P_));
x_m = zeros(size(x));

%% Local filters
MAX_SV = length(fkfData.flag);

Z_ = zeros(2,2*L+1);

F_curv = curvMat(sinsBak.C_n2e,sinsBak.h_n,1);
Rx = 1/(F_curv(2,1)/sinsBak.C_n2e(3,2));
Ry = -1/F_curv(1,2);
```

```
for ii = 1:MAX_SV
    if fkfData.flag(ii) == 0
        continue;
    end

    % Tight_h, to get Z_
    for jj = 1:2*L+1
        dPos = [X_(2,jj)/Ry; X_(1,jj)/Rx; X_(3,jj)];
        dVel = X_(4:6,jj);

        r_e = getEcefPos( sinsBak.C_n2e, sinsBak.h_n, dPos );
        v_e = sinsBak.C_n2e * ( sinsBak.v_n - dVel );

        rho = r_e-fkfData.r_sat(:,ii);
        los = rho/norm(rho);
        Z_(1,jj) = norm(rho) + X_(7,jj);                        %
pseudorange
        Z_(2,jj) = dot(v_e-fkfData.v_sat(:,ii),los) + X_(8,jj);    %
deltarange
    end

    z_ = c1*sum(Z_,2) + c2*Z_(:,1);

    %% Measurement update -- correct
    dZ_ = Z_ - repmat(z_,1,2*L+1);
    dz_ = Z_(:,1) - z_;
    Pzz = c1*(dZ_*dZ_') + c3*(dz_*dz_') + fkfData.R(:,:,ii); %R;
    Pxz = c1*(dX_*dZ_') + c3*(dx_*dz_');

    K = Pxz / Pzz;
    x1 = x_ + K*( fkfData.z(:,ii) - z_ );
    P1 = P_ - K*Pzz*K';

    P_mm = P_mm + inv(P1);
    x_m = x_m + P1\x1;

end

%% Master filter
P = inv(P_mm);
x = P_mm\x_m;
```

24) generateCode

```
function code = generateCode(PRN, system)
% generateCode.m generates GNSS satellite PRN codes.
%
% code = generateCode(PRN, system)
%
%   Inputs:
%       PRN         - PRN number of the sequence.
%       system      - 1: GPS; 2: Beidou
%
%   Outputs:
%       code        - a vector containing the desired code sequence
%                   (chips).
%--------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis
% Written by Mwenjie @2014.03
% Based on SoftGNSS v3.0
%--------------------------------------------------------------------

switch system
    case 1  % GPS
        index = [2,6; 3,7; 4,8; 5,9; 1,9; 2,10; 1,8; 2,9; 3,10; 2,3; 3,4;
            5,6; 6,7; 7,8; 8,9; 9,10; 1,4; 2,5; 3,6; 4,7; 5,8; 6,9; 1,3;
            4,6; 5,7; 6,8; 7,9; 8,10; 1,6; 2,7; 3,8; 4,9; 5,10; 4,10; 1,7;
            2,8; 4,10];
        bitLen = 10;
        codeLen = 1023;
    case 2  % Beidou
        index = [1,3; 1,4; 1,5; 1,6; 1,8; 1,9; 1,10; 1,11; 2,7; 3,4; 3,5;
            3,6; 3,8; 3,9; 3,10; 3,11; 4,5; 4,6; 4,8; 4,9; 4,10; 4,11; 5,6;
            5,8; 5,9; 5,10; 5,11; 6,8; 6,9; 6,10; 6,11; 8,9; 8,10; 8,11;
            9,10; 9,11; 10,11];
        bitLen = 11;
        codeLen = 2046;
end

%--- Generate G1 code -----------------------------------------

%--- Initialize g1 output to speed up the function ---
g1 = zeros(1, codeLen);
%--- Load shift register ---
switch system
```

```
    case 1  % GPS
        reg = -1*ones(1, bitLen);
    case 2  % Beidou
        reg = [1,-1,1,-1,1,-1,1,-1,1,-1,1];
end

%--- Generate all G1 signal chips based on the G1 feedback polynomial -----
for ii = 1:codeLen
    g1(ii) = reg(bitLen);
    switch system
        case 1  % GPS
            saveBit = reg(3)*reg(10);
        case 2  % Beidou
            saveBit = reg(1)*reg(7)*reg(8)*reg(9)*reg(10)*reg(11);
    end
    reg(2:bitLen) = reg(1:bitLen-1);
    reg(1) = saveBit;
end

%--- Generate G2 code ------------------------------------------------

%--- Initialize g2 output to speed up the function ---
g2 = zeros(1, codeLen);
%--- Load shift register ---
switch system
    case 1  % GPS
        reg = -1*ones(1, bitLen);
    case 2  % Beidou
        reg = [1,-1,1,-1,1,-1,1,-1,1,-1,1];
end

%--- Generate all G2 signal chips based on the G2 feedback polynomial -----
for ii = 1:codeLen
    g2(ii) = reg(index(PRN,1))*reg(index(PRN,2));
    switch system
        case 1  % GPS
            saveBit = reg(2)*reg(3)*reg(6)*reg(8)*reg(9)*reg(10);
        case 2  % Beidou
            saveBit =
reg(1)*reg(2)*reg(3)*reg(4)*reg(5)*reg(8)*reg(9)*reg(11);
    end
    reg(2:bitLen) = reg(1:bitLen-1);
```

```
    reg(1) = saveBit;
end

%--- Form single sample C/A code by multiplying G1 and G2 -----------------
code = -(g1 .* g2);
```

25) geo2cart

```
function [X, Y, Z] = geo2cart(phi, lambda, h, i)
%GEO2CART Conversion of geographical coordinates (phi, lambda, h) to
%Cartesian coordinates (X, Y, Z).
%
%[X, Y, Z] = geo2cart(phi, lambda, h, i);
%
%Format for phi and lambda: [degrees minutes seconds].
%h, X, Y, and Z are in meters.
%
%Choices i of Reference Ellipsoid
%   1. International Ellipsoid 1924
%   2. International Ellipsoid 1967
%   3. World Geodetic System 1972
%   4. Geodetic Reference System 1980
%   5. World Geodetic System 1984
%
%   Inputs:
%      phi      - geocentric latitude (format [degrees minutes seconds])
%      lambda    - geocentric longitude (format [degrees minutes
seconds])
%      h        - height
%      i        - reference ellipsoid type
%
%   Outputs:
%      X, Y, Z   - Cartesian coordinates (meters)

%Kai Borre 10-13-98
%Copyright (c) by Kai Borre

b   = phi(1) + phi(2)/60 + phi(3)/3600;
b   = b*pi / 180;
l   = lambda(1) + lambda(2)/60 + lambda(3)/3600;
l   = l*pi / 180;

a   = [6378388 6378160 6378135 6378137 6378137];
```

```
f   = [1/297 1/298.247 1/298.26 1/298.257222101 1/298.257223563];

ex2 = (2-f(i))*f(i) / ((1-f(i))^2);
c   = a(i) * sqrt(1+ex2);
N   = c / sqrt(1 + ex2*cos(b)^2);

X   = (N+h) * cos(b) * cos(l);
Y   = (N+h) * cos(b) * sin(l);
Z   = ((1-f(i))^2*N + h) * sin(b);
%%%%%%%%%%%%%% end geo2cart.m  %%%%%%%%%%%%%%%%%%%%%%%%%%
```

26) getEcefPos

```
function r = getEcefPos( C_n2e, h, dPos )
% 根据位置矩阵和位置误差计算真实的地球坐标系下的位置矢量
% 地球常数
Ra_EARTH = 6378137.0;
e2_EARTH = 6.69437999014e-3;
% Normal Radius
c33 = C_n2e(3,3);
c23 = C_n2e(2,3);
c13 = C_n2e(1,3);
Rn = Ra_EARTH/sqrt(1-e2_EARTH*c33*c33);
% 计算地球坐标系下的位置矢量
r_e = [(Rn+h)*c13;(Rn+h)*c23;(Rn*(1-e2_EARTH)+h)*c33];
A = [ (Rn+h)*C_n2e(1,2), -(Rn+h)*C_n2e(2,3), C_n2e(1,3);
    (Rn+h)*C_n2e(2,2), (Rn+h)*C_n2e(1,3), C_n2e(2,3);
    (Rn*(1-e2_EARTH)+h)*C_n2e(3,2), 0, C_n2e(3,3) ];
dr_e = A*dPos;
% 计算真实的地球坐标系下的位置矢量
r = r_e - dr_e;
```

27) gravityCalcN

```
function g_n = gravityCalcN( varargin )
% 输入:
%   lon -- 经度，rad
%   lat -- 纬度，rad
%   hgt -- 高度，m
%   alpha --游动角,在指北方位系统中为0，rad
% 输出:
%   g_n -- 导航坐标系下的重力加速度，m/s^2
```

```
% 具体地球模型见Savage的Strapdown Analytics中的第5章

%% 函数重载
if nargin == 1
    r_n = varargin{1};
    sin_lat = sin(r_n(1));
    hgt = r_n(3);
elseif nargin == 2
% 输入:
%   C_n2e -- 位置矩阵，3x3
%   hgt -- 高度，m
% 输出:
%   g -- 导航坐标系下的重力加速度，m/s^2
    C_n2e = varargin{1};
    sin_lat = C_n2e(3,3);
    hgt = varargin{2};
elseif nargin == 3
% 输入:
%   lon -- 经度，rad
%   lat -- 纬度，rad
%   hgt -- 高度，m
% 输出:
%   g -- 导航坐标系下的重力加速度，m/s^2
%     lon = varargin{1};
    lat = varargin{1};
    sin_lat = sin(lat);
    hgt = varargin{3};
else
    error('Incorrect number of input arguments. ');
end

%% 地球相关常数定义
a1 = 9.7803267714;         % m/s^2;
a2 = 0.0052790414;
a3 = 0.0000232718;
a4 = -0.0000030876910891;    % 1/s^2
a5 = 0.0000000043977311;    % 1/s^2
a6 = 0.0000000000007211;    % 1/(m*s^2)

%% 计算重力加速度
g_n = a1 * (1+a2*sin_lat^2+a3*sin_lat^4) + (a4+a5*sin_lat^2) * hgt +
a6*hgt^2;
```

28) gyroBd

```
function gyroBdRes = gyroBd( dir, w )
% 陀螺数据命名格式：a1.mat, a2.mat, a3.mat, a4.mat
% 分别表示a面朝下时，正转，反转，摆放面旋转180度后正转和反转

%% load data
% dir = '\';
dir1 = ['a','b','c','d','e','f'];
% dir2 = ['+','-'];
% 加载标定数据
avgMeas = zeros(3,6);
for i = 1:6
    data = load([dir, dir1(i), '+.mat']);
    avgMeas(:,i) = avgMeas(:,i) + mean(data.imuData(1:3,:),2);

    data = load([dir, dir1(i), '-.mat']);
    if mod(i,2) == 1
        avgMeas(:,i+1) = avgMeas(:,i+1) + mean(data.imuData(1:3,:),2);
    else
        avgMeas(:,i-1) = avgMeas(:,i-1) + mean(data.imuData(1:3,:),2);
    end
end
avgMeas = avgMeas/2;

%% Calculate result
% gyroBdRes = gyroBdCalc(avgMeas);
dMeas = avgMeas(:,[1,3,5]) - avgMeas(:,[2,4,6]);
% 标定因数，nx1
gyroBdRes.k = 2*w./sqrt(sum(dMeas.^2, 2));
% 零偏
gyroBdRes.b = -gyroBdRes.k.*mean(avgMeas,2);
% 安装误差矩阵
T = diag(gyroBdRes.k)*dMeas/(2*w);
gyroBdRes.A = inv(T);     % 如果传感器个数多于3个，则应当修改

%% save results
save([dir,'gyroBdResults.mat'], 'gyroBdRes');
```

29) initSettings

```
function settings = initSettings()
```

```
%Functions initializes and saves settings. Settings can be edited inside
of
%the function, updated from the command line or updated using a dedicated
%GUI - "setSettings".
%
%All settings are described inside function code.
%
%settings = initSettings()
%
%   Inputs: none
%
%   Outputs:
%       settings     - Receiver settings (a structure).
%--------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis
% Written by Darius Plausinaitis
% Modified by Mwenjie @2014.03
%--------------------------------------------------------------------

%% select satellite navigation system
% 1 - GPS
% 2 - Beidou
% 3 - Galileo
% 4 - Glonass
settings.system = 2;

%% Processing settings ====================================================
% Number of milliseconds to be processed used 36000 + any transients (see
% below - in Nav parameters) to ensure nav subframes are provided
settings.msToPreSll        = 7e3;         %[ms]
settings.msToSll           = 36e3;        %[ms]
settings.msToVll           = 20e3;        %[ms]
% settings.msToProcess       = settings.msToSll +
settings.msToVll;    %[ms]
settings.msToProcess       = 100e3;       %[ms]
settings.msOfPDI           = 10;          %[ms]

% Number of channels to be used for signal processing
settings.numberOfChannels   = 6;

% Move the starting point of processing. Can be used to start the signal
% processing at any point in the data record (e.g. for long records). fseek
```

```
% function is used to move the file read point, therefore advance is byte
% based only. For Real sample files it skips the number of bytes as indicated
% here. For I/Q files it skips twice the number of bytes as indicated here
% to consider both I and Q samples
settings.skipNumberOfSamples    = 95e6;
settings.skipNumberOfBytes      = 95e6;

%% Raw signal file name and other parameter ========================
% This is a "default" name of the data file (signal record) to be used
in
% the post-processing mode

settings.fileName = '..\usbdata.bin';
%
% Data type used to store one sample
settings.dataType         = 'schar';

% File Types
%1 - 8 bit real samples S0,S1,S2,...
%2 - 8 bit I/Q samples I0,Q0,I1,Q1,I2,Q2,...
settings.fileType         = 1;

% Intermediate, sampling and code frequencies
settings.IF               = 4.092e6;     %[Hz]
settings.samplingFreq      = 16.368e6;    %[Hz]

switch settings.system
    case 1  % GPS
        settings.codeFreqBasis     = 1.023e6;     %[Hz]
        settings.carrFreqBasis     = 1575.42e6;   %[Hz]
        % Define number of chips in a code period
        settings.codeLength        = 1023;
        settings.max_sv            = 32;
    case 2  % Beidou
        settings.codeFreqBasis     = 2.046e6;     %[Hz]
        settings.carrFreqBasis     = 1561.098e6;  %[Hz]
        % Define number of chips in a code period
        settings.codeLength        = 2046;
        settings.max_sv            = 37;
end

%% Acquisition settings ============================================
```

```
% Skips acquisition in the script postProcessing.m if set to 1
settings.skipAcquisition    = 0;
% List of satellites to look for. Some satellites can be excluded to speed
% up acquisition
% settings.acqSatelliteList   = 1:32;        %[PRN numbers]
settings.acqSatelliteList   = 1:37;        %[PRN numbers]
% Band around IF to search for satellite signal. Depends on max Doppler
settings.acqSearchBand      = 16e3;         %[Hz]
% Threshold for the signal presence decision rule
settings.acqThreshold       = 2.5;
% % No. of code periods for coherent integration (multiple of 2)
% settings.acquisition.cohCodePeriods=2;  %10;%
% % No. of non-coherent summations
% settings.acquisition.nonCohSums=4;

%% Tracking loops settings ========================================
settings.enableFastTracking    = 0;

% Code tracking loop Correlator Spacing -- (E-L)/2
settings.dllCorrelatorSpacing    = 0.5;    %[chips]

% Code tracking loop parameters
settings.dllDampingRatio        = 0.7;
settings.dllNoiseBandwidth       = 2;      %[Hz]

% Carrier tracking loop parameters
settings.pllDampingRatio        = 0.7;
settings.pllNoiseBandwidth       = 25;     %[Hz]
settings.fllDampingRatio        = 0.7;
settings.fllNoiseBandwidth       = 10;     %[Hz]


%% Navigation solution settings ===================================

switch settings.system
    case 1  % GPS
        settings.startOffset       = 68.802;      % [ms] Initial sign.
travel time
    case 2  % Beidou
        settings.startOffset       = 120;      % [ms] Initial sign.
travel time
%        settings.startOffset       = 71.81;      % [ms] Initial sign.
```

```
travel time
end


% Rate for calculating pseudorange and position
settings.navSolRate       = 10;          %[Hz]
settings.navSolPeriod     = 1000/settings.navSolRate; %[ms]

% Elevation mask to exclude signals from satellites at low elevation
settings.elevationMask    = 5;         %[degrees 0 - 90]
% Enable/dissable use of tropospheric correction
settings.useTropCorr      = 1;          % 0 - Off
                                  % 1 - On

% True position of the antenna in UTM system (if known). Otherwise enter
% all NaN's and mean position will be used as a reference .
settings.truePosition.E   = nan;
settings.truePosition.N   = nan;
settings.truePosition.U   = nan;

%% Plot settings ============================================
% Enable/disable plotting of the tracking results for each channel
settings.plotTracking     = 0;          % 0 - Off
                                  % 1 - On

%% Constants ================================================
settings.c               = 299792458;   % The speed of light, [m/s]

%% CNo Settings==============================================
% Accumulation interval in Tracking (in Sec)
settings.CNo.accTime = 0.001;
% Show C/No during Tracking;1-on;0-off;
settings.CNo.enableVSM = 1;
% Accumulation interval for computing VSM C/No (in ms)
settings.CNo.VSMinterval = 200;

% C/No initial value
settings.initCNo = 40;
% C/No default value
settings.defaultCNo = 30;

%% Phase lock detector Settings ===================================
```

```
settings.phaseLockDetector.k1 = 0.0247;
settings.phaseLockDetector.k2 = 1.5;
settings.phaseLockDetector.Lp = 100;
settings.phaseLockDetector.Lo = 250;
```

30) ionoc

```
function   ionoCorr = ionoc(lat,lon,el,az,tgps,alpha,beta)
%  Scope:   This MATLAB macro computes L1 iono correction for a specified
user
%         by using the Klobuchar model; WGS-84 constants are used.
%  Usage:   ionocorr = ionoc(lat,lon,el,az,tgps,alpha,beta)
%  Description of parameters:
%         lat      - input, user's geodetic latitude, in radians
%         lon      - input, user's geodetic longitude, in radians
%         el       - input, user's elevation angle, in radians
%         az       - input, user's azimuth angle, in radians
%         tgps     - input, GPS system time, in seconds
%         alpha    - input, coefficients of cubic fit to the amplitude
of
%                    vertical delay (array with 4 elements)
%         beta     - input, coefficients of cubic fit to the period of
model
%                    (array with 4 elements)
%         ionocorr - output, L1 iono correction, in meters
%  Remark:  L2 iono correction can be determined as follows
%             L2 Iono correction = (L1 iono correction) * ((77/60)**2)
%  References:
%         [1] Parkinson, B. W., Spilker, J. J., Editors, Global
Positioning
%             System: Theory and Applications. AIAA, 1996 (Chapter 12,
%             Ionospheric effects on GPS by J. A. Klobuchar).
%         [2] Farrell, J., Barth, M., The Global Positioning System and
%             Inertial Navigation. McGraw Hill, 1998, Appendix E.
%  External Matlab macros used:  wgs84con
%  Last update:  07/22/00
%  Copyright (C) 1999-00 by LL Consulting. All Rights Reserved.

%  Calculate angles in semicircles
elsm = el / pi;
azsm = az / pi;
latsm = lat / pi;
lonsm = lon / pi;
```

```
%  Compute the earth-centered angle
psi = 0.0137 / (elsm + 0.11) - 0.022;

%  Compute the subionospheric latitude
iono_lat = latsm + psi * cos(az);
if (iono_lat > 0.416)
   iono_lat = 0.416;
elseif (iono_lat < -0.416)
   iono_lat = -0.416;
end

%  Compute the subionospheric longitude
iono_lon = lonsm + (psi * sin(az)) / (cos(iono_lat * pi));

%  Find the local time at the subionospheric point
localt = 43200. * iono_lon + tgps;
kk = 0;
while ( (localt >= 86400.) && (kk < 10) )
  localt = localt - 86400.;
  kk = kk + 1;
end
while ( (localt < 0.) && (kk < 10) )
  localt = localt + 86400.;
  kk = kk + 1;
end
if (kk == 10)
   error('IONOC.m - error in local time computation');
end

%  Calculate the geomagnetic latitude of the earth projection of the
%  ionospheric intersection point
latm = iono_lat + 0.064 * cos((iono_lon - 1.617) * pi);

%  Calculate the period by using the beta terms from the GPS message
per = ((beta(4)*latm + beta(3))*latm + beta(2))*latm + beta(1);
if (per < 72000.)
   per = 72000.;
end

%  Calculate the argument of the cosine term
x = (pi + pi) * (localt - 50400.) / per;
```

```
%  Calculate the slant factor
slantf = 0.53 - elsm;
slantf = 1. + 16. * slantf * slantf * slantf;

%  Calculate the amplitude by using the alpha terms from the GPS message
amp = ((alpha(4)*latm + alpha(3))*latm + alpha(2))*latm + alpha(1);
if (amp < 0.)
   amp = 0.;
end

%  Calculate the L1 ionospheric correction (in meters)
if (abs(x) < 1.57)
   xx = x*x;
   ionoCorr = slantf * (5.e-9 + amp * ( 1. - 0.5*xx + (xx*xx/24) ) ) *
299792458;
else
   ionoCorr = slantf * 5.e-9 * 299792458;
end
```

31) ionocBD

```
function ionoErr = ionocBD(lat, lon, el, az, tgps, alpha, beta)
%TROPO  Calculation of ionospheric correction.
%      The range correction ddr in m is to be subtracted from
%      pseudo-ranges and carrier phases
%
%ddr = iono(sinel, hsta, p, tkel, hum, hp, htkel, hhum);
%
%   Inputs:
%
%
%   Outputs:
%      ddr     - range correction (meters)

R    = 6378.137;     % semi-major axis of earth ellipsoid
h    = 375;        % height of ionospheric layer

psi = pi/2 - el - asin(R/(R+h)*cos(el));
latm = asin(sin(lat)*cos(psi)+cos(lat)*sin(psi)*cos(az))/pi; % [pi]

A2 = ((alpha(4)*latm + alpha(3))*latm + alpha(2))*latm + alpha(1);
```

```
if A2 < 0
    A2 = 0;
end

A4 = ((beta(4)*latm + beta(3))*latm + beta(2))*latm + beta(1);
if A4 < 72000
    A4 = 72000;
elseif A4 > 172800
    A4 = 172800;
end

if abs(tgps-50400) >= A4/4
    Iz = 5e-9;
else
    Iz = 5e-9 + A2*cos(2*pi(t-50400)/A4);
end

ionoErr = 299792458/sqrt(1-(R/(R+h)*cos(el))^2) * Iz;
```

32) kf.m

```
function [x,P] = kf(x,P,A,Q,B,u,z,C,R)
% Created by Mwenjie @20140216
% Perform Kalman Filter
%
% Syntax:
%   [x,P] = kf(x,P,A,Q,B,u,z,H,R)
%
% In:
%   x - Nx1 mean state estimate of previous step
%   P - NxN state covariance of previous step
%   A - Transition matrix of discrete model
%   Q - Process noise of discrete model
%   B - Input effect matrix
%   u - Constant input
%   z - Dx1 measurement vector.
%   H - Measurement matrix.
%   R - Measurement noise covariance.
%
% Out:
%   x - Predicted state mean
%   P - Predicted state covariance
%
```

```
% Description:
%  Perform Kalman Filter, including predict step and update step.
%  The model is
%
%    x[k] = A*x[k-1] + B*u[k-1] + q,   q ~ N(0,Q)
%    z[k] = C*x[k] + D*u[k-1]  + r,    r ~ N(0,R)

% Kalman 预测
x_ = A * x + B * u;           % Predicted state mean
P_ = A * P * A' + Q;          % Predicted state covariance

% Kalman 更新
Pxz = P_*C';                    % Covariance between x and z
Pzz = R + C*P_*C';               % Covariance of z
K = Pxz / Pzz;                  % Kalman gain
x = x_ + K * (z - C*x_);     % Estimated state mean
P = P_ - K*Pzz*K';               % Estimated state covariance
```

33) leastSquarePosVel

```
function [pos, vel, el, az, dop] = leastSquarePosVel(satpos, satvel, ...
    pseudorange, deltarange, settings)
%Function calculates the Least Square Solution.
%
%[pos, el, az, dop] = leastSquarePos(satpos, obs, settings);
%
%  Inputs:
%      satpos     - Satellites positions (in ECEF system: [X; Y; Z;] -
%                 one column per satellite)
%      obs        - Observations - the pseudorange measurements to each
%                 satellite:
%                 (e.g. [20000000 21000000 .... .... .... .... ....])
%      settings    - receiver settings
%
%  Outputs:
%      pos        - receiver position and receiver clock error
%                 (in ECEF system: [X, Y, Z, dt])
%      el         - Satellites elevation angles (degrees)
%      az         - Satellites azimuth angles (degrees)
%      dop        - Dilutions Of Precision ([GDOP PDOP HDOP VDOP TDOP])
%--------------------------------------------------------------------------
% Copyright (c) by Kai Borre
% Updated by Darius Plausinaitis, Peter Rinder and Nicolaj Bertelsen
```

```
% Updated by Mwenjie @2014.03
%-----------------------------------------------------------------

%=== Initialization ================================================
nmbOfIterations = 7;

% dtr    = pi/180;
pos    = zeros(4, 1);
vel    = zeros(4,1);      % calculate velocity and ddt
X      = satpos;
vX     = satvel;
nmbOfSatellites = size(satpos, 2);

A      = zeros(nmbOfSatellites, 4);
omc    = zeros(nmbOfSatellites, 1);
az     = zeros(1, nmbOfSatellites);
el     = zeros(1, nmbOfSatellites);

%=== Iteratively find receiver position ============================
for iter = 1:nmbOfIterations

   for i = 1:nmbOfSatellites
      if iter == 1
         %--- Initialize variables at the first iteration --------------
         Rot_X = X(:, i);
         trop = 2;
      else
         %--- Update equations -----------------------------------
         rho2 = (X(1, i) - pos(1))^2 + (X(2, i) - pos(2))^2 + ...
            (X(3, i) - pos(3))^2;
         traveltime = sqrt(rho2) / settings.c ;

         %--- Correct satellite position (do to earth rotation) --------
         Rot_X = e_r_corr(traveltime, X(:, i));

         %--- Find the elevation angel of the satellite ----------------
         [az(i), el(i), ~] = topocent(pos(1:3, :), Rot_X - pos(1:3, :));

%           if (settings.useTropCorr == 1)
             %--- Calculate tropospheric correction --------------
             trop = 2.47 / (sin(el(i) * pi/180) + 0.0121);
%              trop = tropo(sin(el(i) * dtr), ...
```

```
%                   0.0, 1013.0, 293.0, 50.0, 0.0, 0.0, 0.0);
%           else
%               % Do not calculate or apply the tropospheric corrections
%               trop = 0;
%           end
        end % if iter == 1 ... ... else

        %--- Apply the corrections -------------------------------
        omc(i) = (pseudorange(i) - norm(Rot_X - pos(1:3), 'fro') - pos(4)
- trop);

        %--- Construct the A matrix ------------------------------
        A(i, :) =  [ (-(Rot_X(1) - pos(1))) / pseudorange(i) ...
            (-(Rot_X(2) - pos(2))) / pseudorange(i) ...
            (-(Rot_X(3) - pos(3))) / pseudorange(i) ...
            1 ];
    end % for i = 1:nmbOfSatellites

    % These lines allow the code to exit gracefully in case of any errors
    if rank(A) ~= 4
        pos    = zeros(1, 4);
        return
    end

    %--- Find position update ----------------------------------
    x   = A \ omc;

    %--- Apply position update ---------------------------------
    pos = pos + x;

end % for iter = 1:nmbOfIterations


% calculate velocity from carrier frequency
HH = zeros(nmbOfSatellites,4);
d = zeros(nmbOfSatellites,1);
for ii = 1:nmbOfSatellites
    r = pos(1:3)-X(:,ii);
    los = r/norm(r);
    %if Doppler is normal
    d(ii,1) =
-settings.c*(deltarange(ii)-settings.IF)/settings.carrFreqBasis + ...
```

```
        dot(vX(:,ii),los);
    HH(ii,:) = [los',1];
end
vel = HH \ d;

%=== Calculate Dilution Of Precision ============================
if nargout  == 5
    %--- Initialize output ------------------------------------
    dop     = zeros(1, 5);

    %--- Calculate DOP ----------------------------------------------
    Q       = inv(A'*A);

    dop(1)  = sqrt(trace(Q));                   % GDOP
    dop(2)  = sqrt(Q(1,1) + Q(2,2) + Q(3,3));      % PDOP
    dop(3)  = sqrt(Q(1,1) + Q(2,2));             % HDOP
    dop(4)  = sqrt(Q(3,3));                     % VDOP
    dop(5)  = sqrt(Q(4,4));                     % TDOP
end
```

34) llh2ecef

```
function r_e = llh2ecef( C_n2e, h )
% 根据位置矩阵计算地球坐标系下的位置坐标
% 地球常数
Ra_EARTH = 6378137.0;
e2_EARTH = 6.69437999014e-3;
% 计算曲率半径
c33 = C_n2e(3,3);
c23 = C_n2e(2,3);
c13 = C_n2e(1,3);
Rn = Ra_EARTH/sqrt(1-e2_EARTH*c33*c33);    % Normal Radius
% 地球坐标系下的位置矢量
r_e = [(Rn+h)*c13;(Rn+h)*c23;(Rn*(1-e2_EARTH)+h)*c33];
```

35) main

```
%-------------------------------------------------------------------------
%                        SoftGNSS v3.0
%
% Copyright (C) Darius Plausinaitis
% Written by Darius Plausinaitis, Dennis M. Akos
% Some ideas by Dennis M. Akos
```

```
% Modified by Mwenjie @2014.03
%--------------------------------------------------------------------
%This program is free software; you can redistribute it and/or
%modify it under the terms of the GNU General Public License
%as published by the Free Software Foundation; either version 2
%of the License, or (at your option) any later version.
%
%This program is distributed in the hope that it will be useful,
%but WITHOUT ANY WARRANTY; without even the implied warranty of
%MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%GNU General Public License for more details.
%
%You should have received a copy of the GNU General Public License
%along with this program; if not, write to the Free Software
%Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
02110-1301,
%USA.
%--------------------------------------------------------------------

%% Initialization =================================================
init;

gnssStart = input('Enter "1" to initiate GNSS processing or "0" to exit :
');
% gnssStart = 1;

if (gnssStart ~= 1)
    disp(' ');
    %start things rolling...
    return
end

disp ('Starting processing...');

[fid, message] = fopen(settings.fileName, 'rb');

%Initialize the multiplier to adjust for the data type
if (settings.fileType==1)
    dataAdaptCoeff=1;
else
    dataAdaptCoeff=2;
end
```

```
if (fid <= 0)
    % Error while opening the data file.
    error('Unable to read file %s: %s.', settings.fileName, message);
end

%If success, then process the data
% Move the starting point of processing. Can be used to start the
% signal processing at any point in the data record (e.g. good for long
% records or for signal processing in blocks).
fseek(fid, dataAdaptCoeff*settings.skipNumberOfSamples, 'bof');

%% Acquisition ============================================================

% Do acquisition if it is not disabled in settings or if the variable
% acqResults does not exist.
if ((settings.skipAcquisition == 0 || ~exist('acqResults', 'var')))

    % Find number of samples per spreading code
    samplesPerCode = round(settings.samplingFreq / ...
        (settings.codeFreqBasis / settings.codeLength));

    % Read the required amount of data depending on the data file type
    % and the number of code period of coherent and non-coherent
    % integration and invoke the acquisition function

    data  = fread(fid, dataAdaptCoeff*40*samplesPerCode,
settings.dataType)';
    if (dataAdaptCoeff == 2)
        data1 = data(1:2:end);
        data2 = data(2:2:end);
        data = data1 + 1i .* data2;
    end

    %--- Do the acquisition -------------------------------------------
    disp ('   Acquiring satellites...');
    switch settings.system
        case 1  % GPS
            acqResults = acquisition(data, settings);
        case 2  % Beidou
            acqResults = acquisitionBDGEO(data, settings);
```

```
    end

    % Plot the acquisition results
    plotAcquisition(acqResults);

elseif(settings.skipAcquisition == 1)
    disp('skipAcquistion == 1');
    load acqresults

    % Plot the acquisition results
    plotAcquisition(acqResults);
end

%% Initialize channels and prepare for the run ==================

% Start further processing only if a GNSS signal was acquired (the
% field FREQUENCY will be set to 0 for all not acquired signals)
if (any(acqResults.peakMetric > settings.acqThreshold))
    channel = preRun(acqResults, settings);
    showChannelStatus(channel, settings);
else
    % No satellites to track, exit
    disp('No GNSS signals detected, signal processing finished.');
    trackResults = [];
    return;
end

%% Track the signal ============================================

sllStart = input('Enter "1" to start scalar tracking or "0" to exit : ');
% sllStart = 1;

if sllStart ~= 1
    return
end

startTime = now;
disp (['   Tracking started at ', datestr(startTime)]);

% Process all channels for given data block
[trackResults_1ms, channel]= tracking_ms(fid, channel, settings, 1000,
1);
```

```
[trackResults_2ms, channel]= tracking_ms(fid, channel, settings, 1000,
2);
[trackResults_5ms, channel]= tracking_ms(fid, channel, settings, 2000,
5);
[trackResults_10ms, channel]= tracking_ms(fid, channel, settings, 3000,
5);
% % PDI: 10ms
 [trackResults, channel] = tracking_sll(fid, channel, settings,
settings.msToSll);

disp(['   Scaler tracking is over (elapsed time ', ...
    datestr(now - startTime, 13), ')'])

% Auto save the acquisition & tracking results to a file to allow
% running the positioning solution afterwards.
disp('   Saving Acq & Tracking results to file "trackingResults.mat"')
save('trackingResults', ...
    'trackResults', 'settings', 'acqResults', 'channel');

%% Calculate navigation solutions ==================================
disp('   Calculating navigation solutions...');
[navSolutions, eph, svTimeTable, activeChnList] =
postNavigation(trackResults, settings);
save('navSolutions','navSolutions','eph','svTimeTable','activeChnList
')

disp('   Processing is complete for this data block');

%% Plot all results ================================================
disp ('   Ploting results...');
if settings.plotTracking
    plotTracking(1:settings.numberOfChannels, trackResults, settings);
end

plotNavigation(navSolutions, settings);

%% Vector tracking ================================================
disp(' ');
vllStart = input('Enter "1" to start vector tracking or "0" to exit : ');
% vllStart = 1;

if (vllStart == 1)
```

```
    startTime = now;
    disp (['   Tracking started at ', datestr(startTime)]);

    [trackResultsVll, navSolutionsVll, channel, vllSettings] =
tracking_vll( fid, ...
        channel, activeChnList, navSolutions, eph, svTimeTable,
settings );

    disp(['   Vector tracking is over (elapsed time ', ...
        datestr(now - startTime, 13), ')'])

    % Auto save the acquisition & tracking results to a file to allow
    % running the positioning solution afterwards.
    disp('   Saving vector tracking results to file
"trackingResultsv.mat"')
    save('trackingResultsv', 'trackResultsVll', 'navSolutionsVll',
'vllSettings');

    % plot vector tracking result
    plotNavigationv(navSolutionsVll, settings);
end

%%
% Close the data file
fclose(fid);

disp('Post processing of the signal is over.');
disp(' ');
```

36) navPartyChk

```
function status = navPartyChk(ndat)
% This function is called to compute and status the parity bits on GPS
word.
% Based on the flowchart in Figure 2-10 in the 2nd Edition of the GPS-SPS
% Signal Spec.
%
%status = navPartyChk(ndat)
%
%   Inputs:
%       ndat       - an array (1x32) of 32 bits represent a GPS navigation
%                  word which is 30 bits plus two previous bits used in
%                  the parity calculation (-2 -1 0 1 2 ... 28 29)
```

```
%
%   Outputs:
%       status      - the test value which equals EITHER +1 or -1 if parity
%                 PASSED or 0 if parity fails.  The +1 means bits #1-24
%                 of the current word have the correct polarity, while -1
%                 means the bits #1-24 of the current word must be
%                 inverted.
%-------------------------------------------------------------------------
% Written by Darius Plausinaitis, Kristin Larson
%-------------------------------------------------------------------------

% In order to accomplish the exclusive or operation using multiplication
% this program represents a '0' with a '-1' and a '1' with a '1' so that
% the exclusive or table holds true for common data operations
%
%   a   b   xor           a   b   product
%  --------------        -----------------
%   0   0    1           -1  -1     1
%   0   1    0           -1   1    -1
%   1   0    0            1  -1    -1
%   1   1    1            1   1     1

%--- Check if the data bits must be inverted
------------------------------
if (ndat(2) ~= 1)
    ndat(3:26)= -1 .* ndat(3:26);  % Also could just negate
end

%--- Calculate 6 parity bits -----------------------------------
% The elements of the ndat array correspond to the bits showed in the table
% 20-XIV (ICD-200C document) in the following way:
% The first element in the ndat is the D29* bit and the second - D30*.
% The elements 3 - 26 are bits d1-d24 in the table.
% The elements 27 - 32 in the ndat array are the received bits D25-D30.
% The array "parity" contains the computed D25-D30 (parity) bits.

parity(1) = ndat(1)  * ndat(3)  * ndat(4)  * ndat(5)  * ndat(7)  * ...
          ndat(8)  * ndat(12) * ndat(13) * ndat(14) * ndat(15) * ...
          ndat(16) * ndat(19) * ndat(20) * ndat(22) * ndat(25);

parity(2) = ndat(2)  * ndat(4)  * ndat(5)  * ndat(6)  * ndat(8)  * ...
          ndat(9)  * ndat(13) * ndat(14) * ndat(15) * ndat(16) * ...
```

```
          ndat(17) * ndat(20) * ndat(21) * ndat(23) * ndat(26);

parity(3) = ndat(1)  * ndat(3)  * ndat(5)  * ndat(6)  * ndat(7)  * ...
          ndat(9)  * ndat(10) * ndat(14) * ndat(15) * ndat(16) * ...
          ndat(17) * ndat(18) * ndat(21) * ndat(22) * ndat(24);

parity(4) = ndat(2)  * ndat(4)  * ndat(6)  * ndat(7)  * ndat(8)  * ...
          ndat(10) * ndat(11) * ndat(15) * ndat(16) * ndat(17) * ...
          ndat(18) * ndat(19) * ndat(22) * ndat(23) * ndat(25);

parity(5) = ndat(2)  * ndat(3)  * ndat(5)  * ndat(7)  * ndat(8)  * ...
          ndat(9)  * ndat(11) * ndat(12) * ndat(16) * ndat(17) * ...
          ndat(18) * ndat(19) * ndat(20) * ndat(23) * ndat(24) * ...
          ndat(26);

parity(6) = ndat(1)  * ndat(5)  * ndat(7)  * ndat(8)  * ndat(10) * ...
          ndat(11) * ndat(12) * ndat(13) * ndat(15) * ndat(17) * ...
          ndat(21) * ndat(24) * ndat(25) * ndat(26);

%--- Compare if the received parity is equal the calculated parity --------
if ((sum(parity == ndat(27:32))) == 6)

   % Parity is OK. Function output is -1 or 1 depending if the data bits
   % must be inverted or not. The "ndat(2)" is D30* bit - the last  bit
of
   % previous subframe.
   status = -1 * ndat(2);
else
   % Parity failure
   status = 0;
end
```

37) odometer

```
% 融合odometer_kf_coarse和odometer_kf_smooth的信息,得到准确的速度和加速度
odometer_kf_coarse;
odometer_kf_smooth;

%% Initialize kalman filter
dT = 0.01;
F = [ 1, dT; 0, 1 ];
Q = diag( ([0.5*dT^2, dT]*1).^2 );
```

```
H = eye(2);
R = diag( ([0.5, 0.5]).^2 );

x = zeros(2,1);
P = diag( ([0.1, 0.5]*1).^2 );

%% Calculate speed and acceleration
len = length(cnt);
speed  = zeros(1,len);
acceleration = zeros(1,len);
 % Kalman filter
for jj = 1:len
    z = [speed_smooth(jj); acceleration_smooth(jj)];
    [x,P] = kf(x,P,F,Q,z,H,R);
    if dt(jj) >= 1.0
        x = zeros(2,1);
    end

    speed(jj) = x(1);
    acceleration(jj) = x(2);
end
% 绘制结果
time = (1:len)/100;
figure('Name','Speed');
plot(time,speed);
figure('Name','Acceleration');
plot(time,acceleration);
% 保存结果
save('odometer.mat','time','len','dt','cnt','speed','acceleration');
```

38) odometer_kf_coarse

```
% 根据里程计的脉冲信号计算得到速度和加速度，加速度的信任度较高，因此加速度准确，速度相对不准确。
% 车胎直径
ds = 0.0628;
 %% Initialize kalman filter
dT = 0.01;
F = [ 1, dT; 0, 1 ];
Q = diag( ([0.5*dT^2, dT]*10).^2 );
H = eye(2);
R = diag( ([0.2, 50]).^2 );
```

```
x = zeros(2,1);
P = diag( ([0.1, 1]*1).^2 );

%% Calculate speed and acceleration
len = length(cnt);
speed_coarse  = zeros(1,len);
acceleration_coarse = zeros(1,len);

nOrder = 4;
nOrder1 = 8;
dt_ = ones(nOrder1,1);
dt_sum = 0;

speed_ = zeros(nOrder1,1);
t_ = zeros(nOrder1,1);

count = cnt(1);
captureNr = 0;
% Kalman filter
for jj = 2:len
    if dt(jj) < 1
        if cnt(jj) >= count+1
            captureNr = captureNr+1;

            dt_0 = dt(jj);
            if captureNr <= nOrder
                dt_sum = dt_sum + dt_0;
                speed_0 = captureNr*ds/dt_sum;
            else
                dt_sum = dt_sum + dt_0 - dt_(nOrder);
                speed_0 = nOrder*ds/dt_sum;
            end
            dt_ = [dt_0; dt_(1:nOrder1-1)];
            speed_ = [speed_0;speed_(1:nOrder1-1)];
            if captureNr == 1
                acc_0 = speed_0/2/dt(jj);
            elseif captureNr <= nOrder1
                acc_0 = (speed_(1) - speed_(captureNr)) /
sum(dt_(1:captureNr));
            else
                acc_0 = (speed_(2) - speed_(end)) / sum(dt_(2:end));
            end
```

```
            z = [speed_0; acc_0];
            [x,P] = kf(x,P,F,Q,z,H,R);
            speed_(1) = x(1);
        else
            [x,P] = kf_predict(x,P,F,Q);
        end
    else
        x = zeros(2,1);
        captureNr = 0;
        dt_sum = 0.0;
    end

    speed_coarse(jj) = x(1);
    acceleration_coarse(jj) = x(2);
    count = cnt(jj);
end
% 绘制结果
figure('Name','Speed');
plot((1:len)/100,speed_coarse);
figure('Name','Acceleration');
plot((1:len)/100,acceleration_coarse);
```

39) odometer_kf_smooth

```
% 根据里程计的脉冲信号计算得到速度和加速度，速度的信任度较高，因此速度准确，加速度相对不准确。
% 车胎直径
%% Initialize kalman filter
dT = 0.01;
F = [ 1, dT; 0, 1 ];
Q = diag( ([0.5*dT^2, dT]*5).^2 );
H = eye(2);
R = diag( ([1, 50]).^2 );

x = zeros(2,1);
P = diag( ([0.1, 1]*1).^2 );

%% Calculate speed and acceleration
len = length(cnt);
speed_smooth  = zeros(1,len);
acceleration_smooth = zeros(1,len);

nOrder = 4;
```

```
nOrder1 = 16;
dt_ = ones(nOrder1,1);
dt_sum = 0;

speed_ = zeros(nOrder1,1);
t_ = zeros(nOrder1,1);

count = cnt(1);
captureNr = 0;

for jj = 2:len
    if dt(jj) < 1
        if cnt(jj) >= count+1
            captureNr = captureNr+1;
%           dt_0 = ds/speed_coarse(jj);
            dt_0 = dt(jj);
            if captureNr <= nOrder
                dt_sum = dt_sum + dt(jj);
                speed_0 = captureNr*ds/dt_sum;
            else
                dt_sum = dt_sum + dt_0 - dt_(nOrder);
                speed_0 = nOrder*ds/dt_sum;
            end
            dt_ = [dt_0; dt_(1:nOrder1-1)];
            speed_ = [speed_0;speed_(1:nOrder1-1)];
            if captureNr == 1
                acc_0 = speed_0/2/dt(jj);
            elseif captureNr <= nOrder1
                acc_0 = (speed_(1) - speed_(captureNr)) /
sum(dt_(1:captureNr));
            else
                acc_0 = (speed_(2) - speed_(end)) / sum(dt_(2:end));
            end

            z = [speed_0; acc_0];
            [x,P] = kf(x,P,F,Q,z,H,R);

            speed_(1) = x(1);
%           dt_(1) = ds/x(1);
        else
            [x,P] = kf_predict(x,P,F,Q);
        end
```

```
    else
        x = zeros(2,1);
        captureNr = 0;
        dt_sum = 0.0;
    end

    speed_smooth(jj) = x(1);
    acceleration_smooth(jj) = x(2);

    count = cnt(jj);
end
% 绘制结果
figure('Name','Speed');
plot((1:len)/100,speed_smooth);
figure('Name','Acceleration');
plot((1:len)/100,acceleration_smooth);
```

40) parityCheck

```
function word = parityCheck(word, D29Star, D30Star)
%Checks the parity of the supplied 30bit word.If the parity check is not
%passed, the program will be quit with error messages shown
%The last two parity bits of the previous word is used for the calculation.
%A note on the procedure is supplied by the GPS standard positioning
%service signal specification.
%
%word = parityCheck(word, D29Star, D30Star)
%
%   Inputs:
%       word       - an array with 30 bit long word from the navigation
%                  message (a character array, must contain only '0' or
%                  '1').
%       D29Star     - the 29th bit of the previous word (char type).
%       D30Star     - the 30th bit of the previous word (char type).
%
%   Outputs:
%       word       - word with corrected polarity of the data bits
%                  (character array).

%--------------------------------------------------------------------------
% Copyright (C) D.M.Akos
% Written by Xiaofan Li
%--------------------------------------------------------------------------
```

```
% Convert the word,D29Star,D30Start from 0,1
% in char type to 1,-1 in double type
word=1-2*(word-48);
D29Star=1-2*(D29Star-48);
D30Star=1-2*(D30Star-48);

% Correct the polarity of the data word if necessary
word(1:24)=word(1:24)*D30Star;

% Establish matrices for the parity check of the last six bits
h1=[1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0];
h2=[0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 1];
h3=[1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0];
h4=[0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0];
h5=[1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1];
h6=[0 0 1 0 1 1 0 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1 1];
H=[h1;h2;h3;h4;h5;h6];
d=word(1:24);
DStarMat=[D29Star,D30Star,D29Star,D30Star,D30Star,D29Star];

% Calculation of the D
for i=1:6
    temp=H(i,:).*d;
    D(i)=prod(temp(temp~=0))*DStarMat(i);
end;

% Convert the word and D back into 0,1 in char type
word=dec2bin((1-word)/2);
D=dec2bin((1-D)/2);

% If the D does not match the last six bits in the word
% An error message will be reported
if D~=word(25:30)
    error('Parity Check Failed!');
end
```

41) plotAcquisition

```
function plotAcquisition(acqResults)
%Functions plots bar plot of acquisition results (acquisition metrics).
No
%bars are shown for the satellites not included in the acquisition list
```

```
(in
%structure SETTINGS).
%
%plotAcquisition(acqResults)
%
%   Inputs:
%       acqResults    - Acquisition results from function acquisition.
%--------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis
% Written by Darius Plausinaitis
%--------------------------------------------------------------------
%% Plot all results ===================================================
figure(101);

hAxes = newplot();

bar(hAxes, acqResults.peakMetric);

title (hAxes, 'Acquisition results');
xlabel(hAxes, 'PRN number (no bar - SV is not in the acquisition list)');
ylabel(hAxes, 'Acquisition Metric');

oldAxis = axis(hAxes);
axis  (hAxes, [0, 33, 0, oldAxis(4)]);
set   (hAxes, 'XMinorTick', 'on');
set   (hAxes, 'YGrid', 'on');

%% Mark acquired signals ==============================================
acquiredSignals = acqResults.peakMetric .* (acqResults.carrFreq ~= 0);

hold(hAxes, 'on');
bar (hAxes, acquiredSignals, 'FaceColor', [0 0.8 0]);
hold(hAxes, 'off');

legend(hAxes, 'Not acquired signals', 'Acquired signals');
```

42) plotNavigation

```
function plotNavigation(navSolutions, settings)
%Functions plots variations of coordinates over time and a 3D position
%plot. It plots receiver coordinates in UTM system or coordinate offsets
% if the true UTM receiver coordinates are provided.
%
```

```
%plotNavigation(navSolutions, settings)
%
%   Inputs:
%       navSolutions    - Results from navigation solution function. It
%                       contains measured pseudoranges and receiver
%                       coordinates.
%       settings        - Receiver settings. The true receiver coordinates
%                       are contained in this structure.
%--------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis
% Written by Darius Plausinaitis
%--------------------------------------------------------------------

%% Plot results in the necessary data exists ========================
if (~isempty(navSolutions))

    %% If reference position is not provided, then set reference position
    %% to the average postion
    if isnan(settings.truePosition.E) ||
isnan(settings.truePosition.N) ...
                                 || isnan(settings.truePosition.U)

        %=== Compute mean values ==================================
        % Remove NaN-s or the output of the function MEAN will be NaN.
        refCoord.E = mean(navSolutions.E(~isnan(navSolutions.E)));
        refCoord.N = mean(navSolutions.N(~isnan(navSolutions.N)));
        refCoord.U = mean(navSolutions.U(~isnan(navSolutions.U)));

        %Also convert geodetic coordinates to deg:min:sec vector format
        meanLongitude =
mean(navSolutions.longitude(~isnan(navSolutions.longitude)));
        meanLatitude  =
mean(navSolutions.latitude(~isnan(navSolutions.latitude)));

        formatSpect = '%10.6f';
        refPointLgText = ['Mean Position\newline  Lat: ',
num2str(meanLatitude,formatSpect), '{\circ}',
'\newline Lng: ',
num2str(meanLongitude,formatSpect), '{\circ}',
'\newline Hgt: ',
num2str(mean(navSolutions.height(~isnan(navSolutions.height))),
'%+6.1f')];
```

```
    else
        refPointLgText = 'Reference Position';
        refCoord.E = settings.truePosition.E;
        refCoord.N = settings.truePosition.N;
        refCoord.U = settings.truePosition.U;
    end

    figureNumber = 300;
    % The 300 is chosen for more convenient handling of the open
    % figure windows, when many figures are closed and reopened. Figures
    % drawn or opened by the user, will not be "overwritten" by this
    % function if the auto numbering is not used.

    %=== Select (or create) and clear the figure ==================
    figure(figureNumber);
    clf   (figureNumber);
    set   (figureNumber, 'Name', 'Navigation solutions');

    %--- Draw axes ------------------------------------------------
    handles(1, 1) = subplot(4, 2, 1 : 4);
    handles(3, 1) = subplot(4, 2, [5, 7]);
    handles(3, 2) = subplot(4, 2, [6, 8]);

%% Plot all figures ========================================================

    %--- Coordinate differences in UTM system ---------------------
    plot(handles(1, 1), [(navSolutions.E - refCoord.E)', ...
                         (navSolutions.N - refCoord.N)',...
                         (navSolutions.U - refCoord.U)']);

    title (handles(1, 1), 'Coordinates variations in UTM system');
    legend(handles(1, 1), 'E', 'N', 'U');
    xlabel(handles(1, 1), ['Measurement period: ', ...
                                num2str(settings.navSolPeriod), 'ms']);
    ylabel(handles(1, 1), 'Variations (m)');
    grid  (handles(1, 1));
    axis  (handles(1, 1), 'tight');

    %--- Position plot in UTM system ------------------------------
    plot3 (handles(3, 1), navSolutions.E - refCoord.E, ...
                          navSolutions.N - refCoord.N, ...
                          navSolutions.U - refCoord.U, '+');
```

```
    hold  (handles(3, 1), 'on');
    %Plot the reference point
    plot3 (handles(3, 1), 0, 0, 0, 'r+', 'LineWidth', 1.5, 'MarkerSize',
10);
    hold  (handles(3, 1), 'off');

    view  (handles(3, 1), 0, 90);
    axis  (handles(3, 1), 'equal');
    grid  (handles(3, 1), 'minor');

    legend(handles(3, 1), 'Measurements', refPointLgText);

    title (handles(3, 1), 'Positions in UTM system (3D plot)');
    xlabel(handles(3, 1), 'East (m)');
    ylabel(handles(3, 1), 'North (m)');
    zlabel(handles(3, 1), 'Upping (m)');

    %--- Satellite sky plot ---------------------------------
    skyPlot(handles(3, 2), ...
           navSolutions.channel.az, ...
           navSolutions.channel.el, ...
           navSolutions.channel.PRN(:, 1));

    title (handles(3, 2), ['Sky plot (mean PDOP: ', ...
                           num2str(mean(navSolutions.DOP(2,:))),
')']);

else
    disp('plotNavigation: No navigation data to plot.');
end % if (~isempty(navSolutions))
```

43) plotNavigationv

```
function plotNavigationv(navSolutions, settings)
%Functions plots variations of coordinates over time and a 3D position
%plot. It plots receiver coordinates in UTM system or coordinate offsets
if
%the true UTM receiver coordinates are provided.
%
%plotNavigation(navSolutions, settings)
%
%   Inputs:
%       navSolutions    - Results from navigation solution function. It
```

```
%                   contains measured pseudoranges and receiver
%                   coordinates.
%      settings       - Receiver settings. The true receiver coordinates
%                   are contained in this structure.
%--------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis
% Written by Mwenjie @2014.03
% Based on Darius Plausinaitis
%--------------------------------------------------------------------
%% Plot results in the necessary data exists ====================
if (~isempty(navSolutions))

    %% If reference position is not provided, then set reference position
    %% to the average postion
    if isnan(settings.truePosition.E) ||
isnan(settings.truePosition.N) ...
                                || isnan(settings.truePosition.U)

        %=== Compute mean values ==================================
        % Remove NaN-s or the output of the function MEAN will be NaN.
        refCoord.E = mean(navSolutions.E(~isnan(navSolutions.E)));
        refCoord.N = mean(navSolutions.N(~isnan(navSolutions.N)));
        refCoord.U = mean(navSolutions.U(~isnan(navSolutions.U)));

        %Also convert geodetic coordinates to deg:min:sec vector format
        meanLongitude =
mean(navSolutions.longitude(~isnan(navSolutions.longitude)));
        meanLatitude  =
mean(navSolutions.latitude(~isnan(navSolutions.latitude)));

        formatSpect = '%10.6f';
        refPointLgText = ['Mean Position\newline  Lat: ',
num2str(meanLatitude,formatSpect), '{\circ}', '\newline Lng: ',
num2str(meanLongitude,formatSpect), '{\circ}',
'\newline Hgt: ',
num2str(mean(navSolutions.height(~isnan(navSolutions.height))),
'%+6.1f')];
    else
        refPointLgText = 'Reference Position';
        refCoord.E = settings.truePosition.E;
        refCoord.N = settings.truePosition.N;
        refCoord.U = settings.truePosition.U;
```

```
    end

    figureNumber = 400;
    % The 400 is chosen for more convenient handling of the open
    % figure windows, when many figures are closed and reopened. Figures
    % drawn or opened by the user, will not be "overwritten" by this
    % function if the auto numbering is not used.

    %=== Select (or create) and clear the figure ================
    figure(figureNumber);
    clf   (figureNumber);
    set   (figureNumber, 'Name', 'Navigation solutions');

    %--- Draw axes ----------------------------------------------
    handles(1, 1) = subplot(4, 2, 1 : 4);
    handles(3, 1) = subplot(4, 2, [5, 7]);
    handles(3, 2) = subplot(4, 2, [6, 8]);

%% Plot all figures ========================================

    %--- Coordinate differences in UTM system ------------------
    plot(handles(1, 1), [(navSolutions.E - refCoord.E)', ...
                         (navSolutions.N - refCoord.N)', ...
                         (navSolutions.U - refCoord.U)']);

    title (handles(1, 1), 'Coordinates variations in UTM system');
    legend(handles(1, 1), 'E', 'N', 'U');
    xlabel(handles(1, 1), ['Measurement period: ', ...
                                num2str(settings.navSolPeriod), 'ms']);
    ylabel(handles(1, 1), 'Variations (m)');
    grid  (handles(1, 1));
    axis  (handles(1, 1), 'tight');

    %--- Position plot in UTM system ----------------------------
    plot3 (handles(3, 1), navSolutions.E - refCoord.E, ...
                          navSolutions.N - refCoord.N, ...
                          navSolutions.U - refCoord.U, '+');
    hold  (handles(3, 1), 'on');
    % Plot the reference point
    plot3 (handles(3, 1), 0, 0, 0, 'r+', 'LineWidth', 1.5, 'MarkerSize',
10);
    hold  (handles(3, 1), 'off');
```

```
    view  (handles(3, 1), 0, 90);
    axis  (handles(3, 1), 'equal');
    grid  (handles(3, 1), 'minor');
    legend(handles(3, 1), 'Measurements', refPointLgText);

    title (handles(3, 1), 'Positions in UTM system (3D plot)');
    xlabel(handles(3, 1), 'East (m)');
    ylabel(handles(3, 1), 'North (m)');
    zlabel(handles(3, 1), 'Upping (m)');

    %--- Plot velocity ------------------------------------------
    plot(handles(3, 2), [(navSolutions.Vx)', ...
                         (navSolutions.Vy)', ...
                         (navSolutions.Vz)']);

    title (handles(3, 2), 'Velocity in ECEF system');
    legend(handles(3, 2), 'Vx', 'Vy', 'Vz');
    xlabel(handles(3, 2), ['Measurement period: ', ...
                                num2str(settings.navSolPeriod), 'ms']);
    ylabel(handles(3, 2), 'Velocity (m/s)');
    grid  (handles(3, 2));
    axis  (handles(3, 2), 'tight');
else
    disp('plotNavigation: No navigation data to plot.');
end % if (~isempty(navSolutions))
```

44) plotTracking

```
function plotTracking(channelList, trackResults, settings)
%This function plots the tracking results for the given channel list.
%
%plotTracking(channelList, trackResults, settings)
%
%   Inputs:
%       channelList    - list of channels to be plotted.
%       trackResults   - tracking results from the tracking function.
%       settings       - receiver settings.
%--------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis
% Written by Darius Plausinaitis
% Modified by Mwenjie @2014.03
%--------------------------------------------------------------------
```

```
% ms of Predection Integration time
msOfPDI = settings.msOfPDI;
msToProcess = settings.msToSll;

% Protection - if the list contains incorrect channel numbers
channelList = intersect(channelList, 1:settings.numberOfChannels);

%=== For all listed channels ==================================
for channelNr = channelList

%% Select (or create) and clear the figure ======================
   % The number 200 is added just for more convenient handling of the
open
   % figure windows, when many figures are closed and reopened.
   % Figures drawn or opened by the user, will not be "overwritten" by
   % this function.

   figure(channelNr +200);
   clf(channelNr +200);
   set(channelNr +200, 'Name', ['Channel ', num2str(channelNr), ...
                           ' (PRN ', ...
                           num2str(trackResults(channelNr).PRN), ...
                           ') results']);

%% Draw axes ====================================================
      % Row 1
      handles(1, 1) = subplot(3, 3, 1);
      handles(1, 2) = subplot(3, 3, [2 3]);
      % Row 2
      handles(2, 1) = subplot(3, 3, 4);
      handles(2, 2) = subplot(3, 3, [5 6]);
      % Row 3
      handles(3, 1) = subplot(3, 3, 7);
      handles(3, 2) = subplot(3, 3, 8);
      handles(3, 3) = subplot(3, 3, 9);

%% Plot all figures =============================================

      timeAxisInSeconds1 = (1:msToProcess)/1000;
      timeAxisInSeconds = (msOfPDI:msOfPDI:msToProcess)/1000;
```

```
%----- Discrete-Time Scatter Plot ----------------------
plot(handles(1, 1), trackResults(channelNr).I_P,...
                trackResults(channelNr).Q_P, ...
                '.');

grid  (handles(1, 1));
axis  (handles(1, 1), 'equal');
title (handles(1, 1), 'Discrete-Time Scatter Plot');
xlabel(handles(1, 1), 'I prompt');
ylabel(handles(1, 1), 'Q prompt');

%----- Nav bits ----------------------------------------------
plot  (handles(1, 2), timeAxisInSeconds1, ...
                 trackResults(channelNr).I_P);

grid  (handles(1, 2));
title (handles(1, 2), 'Bits of the navigation message');
xlabel(handles(1, 2), 'Time (s)');
axis  (handles(1, 2), 'tight');

%----- PLL discriminator unfiltered -------------------------
plot  (handles(2, 1), timeAxisInSeconds, ...
                 trackResults(channelNr).pllDiscr, 'r');

grid  (handles(2, 1));
axis  (handles(2, 1), 'tight');
xlabel(handles(2, 1), 'Time (s)');
ylabel(handles(2, 1), 'Amplitude');
title (handles(2, 1), 'Raw PLL discriminator');

%----- Correlation --------------------------------------
plot(handles(2, 2), timeAxisInSeconds1, ...
                [sqrt(trackResults(channelNr).I_E.^2 + ...
                     trackResults(channelNr).Q_E.^2)', ...
                 sqrt(trackResults(channelNr).I_P.^2 + ...
                     trackResults(channelNr).Q_P.^2)', ...
                 sqrt(trackResults(channelNr).I_L.^2 + ...
                     trackResults(channelNr).Q_L.^2)'], ...
                '-*');

grid  (handles(2, 2));
title (handles(2, 2), 'Correlation results');
```

```
        xlabel(handles(2, 2), 'Time (s)');
        axis  (handles(2, 2), 'tight');

        hLegend = legend(handles(2, 2), '$\sqrt{I_{E}^2 + Q_{E}^2}$', ...
                                        '$\sqrt{I_{P}^2 + Q_{P}^2}$', ...
                                        '$\sqrt{I_{L}^2 + Q_{L}^2}$');

        %set interpreter from tex to latex. This will draw \sqrt correctly
        set(hLegend, 'Interpreter', 'Latex');

        %----- PLL discriminator filtered----------------------------
        plot  (handles(3, 1), timeAxisInSeconds, ...
                           trackResults(channelNr).pllDiscrFilt, 'b');

        grid  (handles(3, 1));
        axis  (handles(3, 1), 'tight');
        xlabel(handles(3, 1), 'Time (s)');
        ylabel(handles(3, 1), 'Amplitude');
        title (handles(3, 1), 'Filtered PLL discriminator');

        %----- DLL discriminator unfiltered-------------------------
        plot  (handles(3, 2), timeAxisInSeconds, ...
                           trackResults(channelNr).dllDiscr, 'r');

        grid  (handles(3, 2));
        axis  (handles(3, 2), 'tight');
        xlabel(handles(3, 2), 'Time (s)');
        ylabel(handles(3, 2), 'Amplitude');
        title (handles(3, 2), 'Raw DLL discriminator');

        %----- DLL discriminator filtered-------------------------
        plot  (handles(3, 3), timeAxisInSeconds, ...
                           trackResults(channelNr).dllDiscrFilt, 'b');

        grid  (handles(3, 3));
        axis  (handles(3, 3), 'tight');
        xlabel(handles(3, 3), 'Time (s)');
        ylabel(handles(3, 3), 'Amplitude');
        title (handles(3, 3), 'Filtered DLL discriminator');

end % for channelNr = channelList
```

45) posCalc

```
function [ lat, lon, wander ] = posCalc( C_n2e )
% 根据位置矩阵计算经度、纬度和游动角
% 输入:
%   C_n2e -- 位置矩阵，3x3，将导航坐标系转换到地球坐标系
% 输出:
%   lon -- 经度，rad
%   lat -- 纬度，rad
%   wander -- 游动角，rad

%% 计算经度、纬度和游动角
% 纬度
lat = asin( C_n2e(3,3) );
% 经度
lon = atan2( C_n2e(2,3),C_n2e(1,3) );
% 游动角
wander = atan2( C_n2e(3,1),C_n2e(3,2) );
%     if wander < 0
%         wander = wander + 2*pi;
%     end
```

46) posMat

```
function C_n2e = posMat(lat, lon, wander)
% 根据经度、纬度和游动角计算位置矩阵
% 输入:
%   lat -- 纬度，rad
%   lon -- 经度，rad
%   alpha -- 游动角，rad
% 输出:
%   C_n2e -- 位置矩阵，3x3，将导航坐标系转换到地球坐标系

%% 计算位置矩阵
C_n2e = [-sin(lon)*cos(wander)-sin(lat)*cos(lon)*sin(wander), ...
    sin(lon)*sin(wander)-sin(lat)*cos(lon)*cos(wander), ...
    cos(lat)*cos(lon);
    cos(lon)*cos(wander)-sin(lat)*sin(lon)*sin(wander), ...
    -cos(lon)*sin(wander)-sin(lat)*sin(lon)*cos(wander), ...
    cos(lat)*sin(lon);
    cos(lat)*sin(wander), cos(lat)*cos(wander), sin(lat) ];
```

47) postNavigation

```
function [navSolutions, eph, svTimeTable, activeChnList] =
postNavigation(trackResults, settings)
%Function calculates navigation solutions for the receiver (pseudoranges,
%positions). At the end it converts coordinates from the WGS84 system to
%the UTM, geocentric or any additional coordinate system.
%
%[navSolutions, eph] = postNavigation(trackResults, settings)
%
%   Inputs:
%       trackResults    - results from the tracking function (structure
%                       array).
%       settings        - receiver settings.
%   Outputs:
%       navSolutions    - contains measured pseudoranges, receiver
%                       clock error, receiver coordinates in several
%                       coordinate systems (at least ECEF and UTM).
%       eph             - received ephemerides of all SV (structure array).
%--------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis
% Written by Darius Plausinaitis with help from Kristin Larson
% Modified By Xiaofan Li at University of Colorado at Boulder
% Modified By Mwenjie @2014.03
%--------------------------------------------------------------------

%% Check is there enough data to obtain any navigation solution ===========
% It is necessary to have at least three subframes (number 1, 2 and 3)
to
% find satellite coordinates. Then receiver position can be found too.
% The function requires all 5 subframes, because the tracking starts at
% arbitrary point. Therefore the first received subframes can be any three
% from the 5.
% One subframe length is 6 seconds, therefore we need at least 30 sec long
% record (5 * 6 = 30 sec = 30000ms). We add extra seconds for the cases,
% when tracking has started in a middle of a subframe.
tic
if (settings.msToSll < 36000) || (sum([trackResults.status] ~= '-') < 4)
    % Show the error message and exit
    disp('Record is to short or too few satellites tracked. Exiting!');
    navSolutions = [];
    eph          = [];
    svTimeTable  = [];
    activeChnList = [];
```

```
    return
end

%% Find preamble start positions ==================================

[subFrameStart, isReverse, activeChnList] = findPreambles(trackResults,
settings);

%% Decode ephemerides =============================================

for channelNr = activeChnList

    switch settings.system
        case 1  % GPS
            msPerBit = 20;
            secondaryCode = ones(1,20);
            %--- Copy 5 sub-frames long record from tracking output -------
            navBitsSamples =
trackResults(channelNr).I_P(subFrameStart(channelNr) - 2*20 : ...
                subFrameStart(channelNr) + (1500 * 20) -1)';
        case 2  % Beidou
            if trackResults(channelNr).PRN <= 5    % GEO
                msPerBit = 2;
                secondaryCode = ones(1,2);
            else
                msPerBit = 20;
                secondaryCode = [-1 -1 -1 -1 -1 1 -1 -1 1 1 -1 1 -1 1 -1
-1 1 1 1 -1];
            end
            navBitsSamples =
trackResults(channelNr).I_P(subFrameStart(channelNr) : ...
                subFrameStart(channelNr) + (1500 * 20) -1)';
    end

    %=== Convert tracking output to navigation bits =================

    %--- Group every 20 vales of bits into columns ------------------------
    navBitsSamples = reshape(navBitsSamples, ...
        msPerBit, (size(navBitsSamples, 1) / msPerBit));

    %--- Sum all samples in the bits to get the best estimate -------------
%     navBits = sum(navBitsSamples);
```

```
    navBits = isReverse(channelNr) * secondaryCode * navBitsSamples;

    %--- Now threshold and make 1 and 0 -----------------------------
    % The expression (navBits > 0) returns an array with elements set to
1
    % if the condition is met and set to 0 if it is not met.
    navBits = (navBits > 0);

    %--- Convert from decimal to binary -----------------------------
    % The function ephemeris expects input in binary form. In Matlab it
is
    % a string array containing only "0" and "1" characters.
    navBitsBin = dec2bin(navBits);

    %=== Decode ephemerides and TOW of the first sub-frame ===========
    switch settings.system
        case 1  % GPS
            [eph(trackResults(channelNr).PRN), TOW(channelNr)] = ...
                ephemeris(navBitsBin(3:1502)', navBitsBin(1),
navBitsBin(2));
        case 2  % Beidou
            if trackResults(channelNr).PRN <= 5    % GEO
                [eph(trackResults(channelNr).PRN), TOW(channelNr)] = ...
                    ephemerisBDGEO(navBitsBin');
            else
                [eph(trackResults(channelNr).PRN), TOW(channelNr)] = ...
                    ephemerisBD(navBitsBin');
            end
    end

    %--- Exclude satellite if it does not have the necessary nav data -----
    % If the satellite accuracy or health is not in reliable values, then
    % this satellite is excluded as well
%     if (isempty(eph(trackResults(channelNr).PRN).IODC) || ...
%           isempty(eph(trackResults(channelNr).PRN).IODE_sf2) || ...
%           isempty(eph(trackResults(channelNr).PRN).IODE_sf3) || ...
%           eph(trackResults(channelNr).PRN).accuracy >=3 ||...
%           eph(trackResults(channelNr).PRN).health~=0)
%     if (isempty(eph(trackResults(channelNr).PRN).IODC) || ...
%           isempty(eph(trackResults(channelNr).PRN).IODE_sf2) || ...
%           isempty(eph(trackResults(channelNr).PRN).IODE_sf3) || ...
%           eph(trackResults(channelNr).PRN).accuracy >=3 )
```

```
%
%       %--- Exclude channel from the list (from further processing) ---
%       activeChnList = setdiff(activeChnList, channelNr);
%       s=sprintf('PRN %d is excluded from the active channel',...
%           trackResults(channelNr).PRN);
%       disp(s);
%    end
end

%% Check if the number of satellites is still above 3 ====================
if (isempty(activeChnList) || (size(activeChnList, 2) < 4))
    % Show error message and exit
    disp('Too few satellites with ephemeris data for postion calculations.
Exiting!');
    navSolutions = [];
    eph          = [];
    svTimeTable  = [];
    activeChnList = [];
    return
end

%% Initialization =========================================================

% Add by Mwenjie @20140121 {
% ms of Predection Integration time
% msOfPDI = 2;
% Length of trackResults
% trackLen = floor(settings.msToProcess / msOfPDI);
% } Added by Mwenjie

% Set the satellite elevations array to INF to include all satellites for
% the first calculation of receiver position. There is no reference point
% to find the elevation angle as there is no receiver position estimate
at
% this point.
satElev  = inf(1, settings.numberOfChannels);

% Save the active channel list. The list contains satellites that are
% tracked and have the required ephemeris data. In the next step the list
% will depend on each satellite's elevation angle, which will change over
% time.
readyChnList = activeChnList;
```

```
% Establish the transmitting time table
msToProcess = settings.msToProcess;
svTimeTable.time=zeros(1,msToProcess);
svTimeTable.PRN=[];
svTimeTable.subFrameStart = 0;
svTimeTable = repmat(svTimeTable, 1, max(activeChnList));

% Establish the time table based on the TOW and its position
for channelNr = activeChnList
    svTimeTable(channelNr).PRN = trackResults(channelNr).PRN;
    svTimeTable(channelNr).subFrameStart = subFrameStart(channelNr);
    for i=1:msToProcess
        svTimeTable(channelNr).time(i)=...
            TOW(channelNr)-subFrameStart(channelNr)*0.001+i*0.001;
    end;
end

% Find the last sample number in the tracking results
lastSample=inf(1,max(readyChnList));

for channelNr = readyChnList
    lastSample(channelNr) = ...
        trackResults(channelNr).absoluteSample(end);
end
% Find the step size for navigation solution
navStep=settings.samplingFreq/settings.navSolRate;

%#########################################################################
%#   Do the satellite and receiver position calculations                 #
%#########################################################################
sllCntOffset = floor( settings.msToPreSll/settings.navSolPeriod );

%% Initialization of current measurement ==========================
for currMeasNr = 1:floor( settings.msToSll/settings.navSolPeriod )
% for currMeasNr = 1:fix( (min(lastSample) - ...
%        settings.samplingFreq/settings.navSolRate-...
%        settings.skipNumberOfSamples) /navStep)

    % Exclude satellites, that are belove elevation mask
    activeChnList = intersect(find(satElev >=
settings.elevationMask), ...
```

```
        readyChnList);

    % Save list of satellites used for position calculation
    navSolutions.channel.PRN(activeChnList, currMeasNr) = ...
        [trackResults(activeChnList).PRN];

    % These two lines help the skyPlot function. The satellites excluded
    % do to elevation mask will not "jump" to possition (0,0) in the sky
    % plot.
    navSolutions.channel.el(:, currMeasNr) = ...
        NaN(settings.numberOfChannels, 1);
    navSolutions.channel.az(:, currMeasNr) = ...
        NaN(settings.numberOfChannels, 1);

%% Calculate the current sample number, corresponding satellites ======
%% transmitting time and raw receiver time ============================
    sampleNum =
(currMeasNr+sllCntOffset)*settings.samplingFreq/settings.navSolRate..
.
        + settings.skipNumberOfSamples;
    transmitTime =
findTransTime(sampleNum,activeChnList,svTimeTable,...
        trackResults,settings,0);
    if currMeasNr == 1
        rxTime = max(transmitTime)+settings.startOffset/1000;
    else
        rxTime = rxTime + 1/settings.navSolRate;
    end

%% Find pseudoranges ==================================================
    [navSolutions.channel.rawP(:, currMeasNr)] =
calculatePseudoranges(...
        transmitTime, rxTime, activeChnList, settings);
    % old version of calculateP
%    navSolutions.channel.rawP(:, currMeasNr) =
calculatePseudoranges(...
%           trackResults, ...
%           subFrameStart + 1000/settings.navSolRate *
(currMeasNr-1), ...
%           activeChnList, settings);

%% Find satellites positions and clocks corrections ================
```

```
%    [satPositions, satClkCorr] = satpos(transmitTime, ...
%       [trackResults(activeChnList).PRN],eph);
%    [satPos, satVel, satClkCorr] =
satPosVel(transmitTime(transmitTime>0), ...
%       [trackResults(activeChnList).PRN],eph, , settings.system);
    [satPos, satVel, satClkCorr] = satPosVel(transmitTime, ...
       [trackResults(activeChnList).PRN], eph, settings.system);

%% Find receiver position ==================================================

   %3D receiver position can be found only if signals from more than 3
   % satellites are available
   if size(activeChnList, 2) > 3

      %=== Calculate receiver position ==============================
      rawF = zeros(1,settings.numberOfChannels);
      for ii = activeChnList
         rawF(ii) =
trackResults(ii).carrFreq(currMeasNr*settings.navSolPeriod);
      end

      [pos, vel, ...
         navSolutions.channel.el(activeChnList, currMeasNr), ...
         navSolutions.channel.az(activeChnList, currMeasNr), ...
         navSolutions.DOP(:, currMeasNr)] = ...
         leastSquarePosVel( satPos(:,activeChnList),
satVel(:,activeChnList), ...
         navSolutions.channel.rawP(activeChnList, currMeasNr)' + ...
         satClkCorr(activeChnList) * settings.c, ...
         rawF(activeChnList), settings );

      %--- Save results -----------------------------------------
      navSolutions.X(currMeasNr)  = pos(1);
      navSolutions.Y(currMeasNr)  = pos(2);
      navSolutions.Z(currMeasNr)  = pos(3);
      navSolutions.dt(currMeasNr) = pos(4);

      navSolutions.Vx(currMeasNr)  = vel(1);
      navSolutions.Vy(currMeasNr)  = vel(2);
      navSolutions.Vz(currMeasNr)  = vel(3);
      navSolutions.ddt(currMeasNr) = vel(4);
```

```
        % Update the satellites elevations vector
        satElev = navSolutions.channel.el(:, currMeasNr);
        satElev = satElev';

        %=== Correct pseudorange measurements for clocks errors =======
        navSolutions.channel.correctedP(activeChnList, currMeasNr) = ...
            navSolutions.channel.rawP(activeChnList, currMeasNr) + ...
            satClkCorr(activeChnList)' * settings.c -
navSolutions.dt(currMeasNr);

        %% Coordinate conversion ================================================

        %=== Convert to geodetic coordinates =====================
        switch settings.system
            case 1     % GPS
                cordinateSystem = 5;
            case 2     % Beidou
                cordinateSystem = 6;
        end
        [navSolutions.latitude(currMeasNr), ...
            navSolutions.longitude(currMeasNr), ...
            navSolutions.height(currMeasNr)] = cart2geo(...
            navSolutions.X(currMeasNr), ...
            navSolutions.Y(currMeasNr), ...
            navSolutions.Z(currMeasNr), ...
            cordinateSystem );

        %=== Convert to UTM coordinate system ====================
        navSolutions.utmZone =
findUtmZone(navSolutions.latitude(currMeasNr), ...
            navSolutions.longitude(currMeasNr));

        [navSolutions.E(currMeasNr), ...
            navSolutions.N(currMeasNr), ...
            navSolutions.U(currMeasNr)] = cart2utm(pos(1), pos(2),
pos(3), ...
            navSolutions.utmZone);

        % Compute the corrected receiver time

navSolutions.rxTime(currMeasNr)=rxTime-navSolutions.dt(currMeasNr)/se
```

```
ttings.c;
       % Record the sample number and raw receiver time
       navSolutions.absoluteSample(currMeasNr) = sampleNum;
       navSolutions.rawRxTime(currMeasNr) = rxTime;

       rxTime = rxTime-navSolutions.dt(currMeasNr)/settings.c;

       %DMA add - get the precise time of the first sample and the avg
       %clock rate for the file (skip first 5 samples and should be
       %enough to get over transients)
       if (currMeasNr == fix((min(lastSample) - ...
              settings.samplingFreq/settings.navSolRate-...
              settings.skipNumberOfSamples) /navStep))
          dmaTime = polyfit(navSolutions.absoluteSample(5:end) - ...

settings.skipNumberOfSamples,navSolutions.rxTime(5:end),1);
          navSolutions.avgClock = 1/dmaTime(1);
          navSolutions.firstSampleTime = 1 * dmaTime(1) + dmaTime(2);
       end
   else % if size(activeChnList, 2) > 3
       %--- There are not enough satellites to find 3D position ----------
       disp(['   Measurement No. ', num2str(currMeasNr), ...
          ': Not enough information for position solution.']);

       %--- Set the missing solutions to NaN. These results will be
       %excluded automatically in all plots. For DOP it is easier to use
       %zeros. NaN values might need to be excluded from results in some
       %of further processing to obtain correct results.
       navSolutions.X(currMeasNr)         = NaN;
       navSolutions.Y(currMeasNr)         = NaN;
       navSolutions.Z(currMeasNr)         = NaN;
       navSolutions.dt(currMeasNr)        = NaN;
       navSolutions.DOP(:, currMeasNr)     = zeros(5, 1);
       navSolutions.latitude(currMeasNr)    = NaN;
       navSolutions.longitude(currMeasNr)   = NaN;
       navSolutions.height(currMeasNr)     = NaN;
       navSolutions.E(currMeasNr)         = NaN;
       navSolutions.N(currMeasNr)         = NaN;
       navSolutions.U(currMeasNr)         = NaN;
       navSolutions.rawRxTime            = NaN;
       navSolutions.absoluteSample        = NaN;
       navSolutions.rxTime               = NaN;
```

```
        navSolutions.channel.az(activeChnList, currMeasNr) = ...
            NaN(1, length(activeChnList));
        navSolutions.channel.el(activeChnList, currMeasNr) = ...
            NaN(1, length(activeChnList));

        % TODO: Know issue. Satellite positions are not updated if the
        % satellites are excluded do to elevation mask. Therefore rasing
        % satellites will be not included even if they will be above
        % elevation mask at some point. This would be a good place to
        % update positions of the excluded satellites.

    end % if size(activeChnList, 2) > 3

end
toc
```

48) preRun

```
function [channel] = preRun(acqResults, settings)
%Function initializes tracking channels from acquisition data. The
acquired
%signals are sorted according to the signal strength. This function can
be
%modified to use other satellite selection algorithms or to introduce
%acquired signal properties offsets for testing purposes.
%
%[channel] = preRun(acqResults, settings)
%
%   Inputs:
%       acqResults  - results from acquisition.
%       settings    - receiver settings
%
%   Outputs:
%       channel     - structure contains information for each channel (like
%                   properties of the tracked signal, channel status etc.).
%--------------------------------------------------------------------
% Copyright (C) Darius Plausinaitis
% Written by Darius Plausinaitis
% Based on Peter Rinder and Nicolaj Bertelsen
% Modified by Mwenjie @2014.03
%--------------------------------------------------------------------
%% Initialize all channels ========================================
```

```
channel              = [];   % Clear, create the structure

channel.PRN           = 0;    % PRN number of the tracked satellite
channel.acquiredFreq    = 0;    % Used as the center frequency of the NCO
channel.carrFreq       = 0;
channel.codePhase      = 0;    % Position of the C/A  start
channel.codeFreq       = settings.codeFreqBasis;

% channel.carrError      = 0.0;
channel.carrNco        = 0.0;
% channel.codeError      = 0.0;
channel.codeNco        = 0.0;
channel.remCarrPhase    = 0.0;
channel.remCodePhase    = 0.0;

channel.x = [ 0; 0; 0 ];
channel.P = diag( ([10, 1, 100]).^2 );

channel.status         = '-';  % Mode/status of the tracking channel
                           % "-" - "off" - no signal to track
                           % "T" - Tracking state

channel.startSample    = settings.skipNumberOfSamples;

channel.CNo           = settings.initCNo;

% phase lock detector
channel.phaseLockDetector.A = 2500;
channel.phaseLockDetector.B = 250;
channel.phaseLockDetector.optimisticLock = 0;
channel.phaseLockDetector.pessimisticLock = 0;
channel.phaseLockDetector.cnt1 = 0;
channel.phaseLockDetector.cnt2 = 0;

%--- Copy initial data to all channels
-------------------------------------
channel = repmat(channel, 1, settings.numberOfChannels);

%% Copy acquisition results ========================================

%--- Sort peaks to find strongest signals, keep the peak index information
[~, PRNindexes]         = sort(acqResults.peakMetric, 2, 'descend');
```

```
%--- Load information about each satellite ------------------------
% Maximum number of initialized channels is number of detected signals,
but
% not more as the number of channels specified in the settings.
for ii = 1:min([settings.numberOfChannels, sum(acqResults.carrFreq ~=
0)])
    channel(ii).PRN         = PRNindexes(ii);
    channel(ii).acquiredFreq = acqResults.carrFreq(PRNindexes(ii));
    channel(ii).carrFreq = channel(ii).acquiredFreq;
    channel(ii).codePhase    = acqResults.codePhase(PRNindexes(ii));

    channel(ii).startSample  = settings.skipNumberOfSamples +
channel(ii).codePhase - 1;

    channel(ii).x(3) = -(channel(ii).carrFreq - settings.IF) * ...
        settings.c / settings.carrFreqBasis;

    channel(ii).phaseLockDetector.optimisticLock = 1;
    channel(ii).phaseLockDetector.pessimisticLock = 1;

    % Set tracking into mode (there can be more modes if needed e.g. pull-in)
    channel(ii).status      = 'T';
end
```

49) satPosVel

```
function [satPos, satVel, satClkCorr] = satPosVel(transmitTime, prnList,
eph, satSystem)
%SATPOS Calculation of X,Y,Z satellites coordinates at TRANSMITTIME for
%given ephemeris EPH. Coordinates are calculated for each satellite in
the
%list PRNLIST.
%[satPositions, satClkCorr] = satpos(transmitTime, prnList, eph);
%
%   Inputs:
%       transmitTime  - transmission time for all satellites
%       prnList       - list of PRN-s to be processed
%       eph           - ephemeridies of satellites
%
%   Outputs:
%       satPositions  - positions of satellites (in ECEF system [X; Y; Z;])
%       satClkCorr    - correction of satellites clocks
```

```
%--------------------------------------------------------------------
% Based on Kai Borre 04-09-96
% Copyright (c) by Kai Borre
% Updated by Darius Plausinaitis, Peter Rinder and Nicolaj Bertelsen
% Modified by Xiaofan Li at University of Colorado at Boulder
% Modified by Mwenjie @2014.03
%--------------------------------------------------------------------
%% Initialize constants ================================================
numOfChannels = size(transmitTime, 2);

% GNSS constatns

gnssPi        = 3.1415926535898;  % Pi used in the GPS coordinate
% system

%--- Constants for satellite position calculation -----------------
switch satSystem
    case 1     % GPS
       Omegae_dot = 7.2921151467e-5;   % Earth rotation rate, [rad/s]
       GM = 3.986005e14;      % Earth's universal
                                    % gravitational parameter,
                                    % [m^3/s^2]
       F = 4.442807633393060e-10;  % Constant, [sec/(meter)^(1/2)]
    case 2     % Beidou
       Omegae_dot = 7.2921150e-5;   % Earth rotation rate, [rad/s]
       GM = 3.986004418e14;      % Earth's universal
                                    % gravitational parameter,
                                    % [m^3/s^2]
       F = 4.442807309043978e-10;  % Constant, [sec/(meter)^(1/2)]
end

%% Initialize results =================================================
satClkCorr = zeros(1, numOfChannels);
satPos = zeros(3, numOfChannels);
satVel = zeros(3, numOfChannels);

%% Process each satellite =============================================

numOfSatellites = size(prnList, 2);

for satNr = 1 : numOfSatellites
```

```
prn = prnList(satNr);

if length(eph) < prn || isempty(eph(prn).t_oc)
    continue;
end

%% Find initial satellite clock correction -------------------

%--- Find time difference ---------------------------------
dt = check_t(transmitTime(satNr) - eph(prn).t_oc);

%--- Calculate clock correction ---------------------------
satClkCorr(satNr) = (eph(prn).a_f2 * dt + eph(prn).a_f1) * dt + ...
    eph(prn).a_f0 - eph(prn).T_GD;

time = transmitTime(satNr) - satClkCorr(satNr);

%% Find satellite's position ----------------------------------

%Restore semi-major axis
a = eph(prn).sqrtA * eph(prn).sqrtA;

%Time correction
tk = check_t(time - eph(prn).t_oe);

%Initial mean motion
n0 = sqrt(GM / a^3);
%Mean motion
n = n0 + eph(prn).deltan;

%Mean anomaly
M   = eph(prn).M_0 + n * tk;
%Reduce mean anomaly to between 0 and 360 deg
M = rem(M + 2*gnssPi, 2*gnssPi);

%Initial guess of eccentric anomaly
E = M;

%--- Iteratively compute eccentric anomaly ----------------
for ii = 1:10
    E_old = E;
    E = M + eph(prn).e * sin(E);
```

```
        dE = rem(E - E_old, 2*gnssPi);

        if abs(dE) < 1.e-12
            % Necessary precision is reached, exit from the loop
            break;
        end
    end

    %Reduce eccentric anomaly to between 0 and 360 deg
    E = rem(E + 2*gnssPi, 2*gnssPi);
    dE = n/(1-eph(prn).e * cos(E));
    %Compute relativistic correction term
    dtr = F * eph(prn).e * eph(prn).sqrtA * sin(E);

    %Calculate the true anomaly
    nu = atan2(sqrt(1 - eph(prn).e^2) * sin(E), cos(E)-eph(prn).e);

    %Compute angle phi
    phi = nu + eph(prn).omega;
    dphi = sqrt(1-eph(prn).e^2)*dE/(1-eph(prn).e*cos(E));
    %Reduce phi to between 0 and 360 deg
    phi = rem(phi, 2*gnssPi);

    %Correct argument of latitude
    u = phi + ...
        eph(prn).C_uc * cos(2*phi) + ...
        eph(prn).C_us * sin(2*phi);
    du = (1+2*(eph(prn).C_us * cos(2*phi)-eph(prn).C_uc *
sin(2*phi)))*dphi;
    %Correct radius
    r = a * (1 - eph(prn).e*cos(E)) + ...
        eph(prn).C_rc * cos(2*phi) + ...
        eph(prn).C_rs * sin(2*phi);
    dr = a * eph(prn).e *sin(E) *dE + ...
        2 * (eph(prn).C_rs * cos(2*phi) - eph(prn).C_rc * sin(2*phi)) ...
        * dphi;
    %Correct inclination
    i = eph(prn).i_0 + eph(prn).iDot * tk + ...
        eph(prn).C_ic * cos(2*phi) + ...
        eph(prn).C_is * sin(2*phi);
    di = 2 * (eph(prn).C_is * cos(2*phi) - eph(prn).C_ic * sin(2*phi)) ...
        * dphi + eph(prn).iDot;
```

```
    %Compute the angle between the ascending node and the Greenwich
meridian
    if satSystem == 2 && prn <= 5 % Beidou GEO
        Omega = eph(prn).omega_0 + eph(prn).omegaDot*tk - ...
            Omegae_dot * eph(prn).t_oe;
        dOmega = eph(prn).omegaDot;
    else
        Omega = eph(prn).omega_0 + (eph(prn).omegaDot - Omegae_dot)*tk
- ...
            Omegae_dot * eph(prn).t_oe;
        dOmega = eph(prn).omegaDot - Omegae_dot;
    end
    %Reduce to between 0 and 360 deg
    Omega = rem(Omega + 2*gnssPi, 2*gnssPi);

    %--- Compute satellite coordinates ------------------------------
    x = cos(u)*r * cos(Omega) - sin(u)*r * cos(i)*sin(Omega);
    y = cos(u)*r * sin(Omega) + sin(u)*r * cos(i)*cos(Omega);
    z = sin(u)*r * sin(i);

    xdash = r * cos(u);
    ydash = r * sin(u);

    dxdash = dr * cos(u) - r * sin(u) * du;
    dydash = dr * sin(u) + r * cos(u) * du;

    Vx = dxdash * cos(Omega) -  dydash * cos(i) * sin(Omega) ...
        + ydash * sin(Omega) * sin(i) * di - (xdash * sin(Omega) + ...
        ydash * cos(i) * cos(Omega)) * dOmega;

    Vy = dxdash * sin(Omega) + dydash * cos(i) * cos(Omega) - ...
        ydash * sin(i) *cos(Omega) * di + (xdash * cos(Omega) - ...
        ydash * cos(i) * sin(Omega)) * dOmega;

    Vz = dydash * sin(i) + ydash * cos(i) * di;

    % Added by Mwenjie @20140310
    if satSystem == 2 && prn <= 5 % Beidou GEO
        Rx = [1, 0, 0;
            0, cos(-5*pi/180), sin(-5*pi/180);
            0, -sin(-5*pi/180), cos(-5*pi/180) ];
```

```
        Rz = [ cos(Omegae_dot*tk), sin(Omegae_dot*tk), 0;
            -sin(Omegae_dot*tk), cos(Omegae_dot*tk), 0;
            0, 0, 1 ];
        dRz = [ -sin(Omegae_dot*tk), cos(Omegae_dot*tk), 0;
            -cos(Omegae_dot*tk), -sin(Omegae_dot*tk), 0;
            0, 0, 0 ] * Omegae_dot;
        satPos(:, satNr) = Rz*Rx*[x; y; z];
        satVel(:, satNr) = Rz*Rx*[Vx; Vy; Vz] + dRz*Rx*[x; y; z];
    else
        satPos(:, satNr) = [x; y; z];
        satVel(:, satNr) = [Vx; Vy; Vz];
    end


    %% Include relativistic correction in clock correction ------------
    satClkCorr(satNr) = (eph(prn).a_f2 * dt + eph(prn).a_f1) * dt + ...
        eph(prn).a_f0 -  eph(prn).T_GD + dtr;
end % for satNr = 1 : numOfSatellites
```

50) sinsUpdate

```
function [pos, vel, att] = sinsUpdate(a_sf, w_ib, Len, dTl, lmRatio, pos0,
vel0, att0)
% 惯导更新算法
% 输入:
%   a_sf    : 加表输出值, 3xLen, m/s^2
%   w_ib    : 陀螺输出值, 3xLen, rad/s
%   Len     : 数据长度
%   dTl     : 高频更新时间间隔, s
%   lmRatio : l cycle to m cycle frequency ratio
%   pos0    : 初始位置(rad), 3x1, 包括lat, lon, hgt
%   vel0    : 初始速度, 3x1, 包括ve, vn, vu
%   att0    : 初始状态(rad), 3x1, 包括pitch, roll, azimuth
% 输出:
%   pos : 位置信息
%   vel : 速度信息
%   att : 姿态信息

%%配置选项
% HIGH_ACCURACY = 1;          %高精度 or 低精度
SURFACE_NAVIGATION = 1;     %平面导航 or 三维导航
NORTH_SYSTEM = 1;           %指北方位系统 or 游动自由方位系统
```

```
%% 常数定义
matI = [1,0,0;0,1,0;0,0,1];     %单位矩阵
We = 7.2921151467e-5;           %地球自转角速率,rad/s
w_ie_e = [0; 0; We];            %在地球坐标系中的地球自转矢量

%% 输出定义
vel = zeros(3,floor(Len/lmRatio));     % 记录速度输出
pos = zeros(3,floor(Len/lmRatio));     % 记录位置输出
att = zeros(3,floor(Len/lmRatio));     % 记录姿态输出

%% 中间变量初始化
dTm = dTl*lmRatio;       %低速更新时间间隔

% high frequency update
a_sf_b_1 = a_sf(:,1);

alpha = zeros(3,1);
beta = zeros(3,1);
nu = zeros(3,1);
dV_scul = zeros(3,1);
sAlpha = zeros(3,1);
sNu = zeros(3,1);
dR_scrl = zeros(3,1);

dAlpha = zeros(3,1);
dNu = zeros(3,1);

% low frequency update
C_b2n = attMat(att0(1), att0(2), att0(3));
v_n = vel0;
C_n2e = posMat(pos0(1), pos0(2), 0);
h_n = pos0(3);

%
v_n_1 = v_n;
v_n_2 = v_n;

g_n = gravityCalc(C_n2e, h_n);
g_n_1 = g_n;
g_n_2 = g_n_1;
w_ie_n_1 = C_n2e'*w_ie_e;
w_ie_n_2 = w_ie_n_1;
```

```
F_curv_1 = curvMat(C_n2e, h_n, NORTH_SYSTEM);
F_curv_2 = F_curv_1;
w_en_n_1 = F_curv_1*v_n;
w_en_n_2 = w_en_n_1;

%% 惯导更新

m_cnt = 1;     % m cycle计数值
m = 1;
for l_cnt = 1:Len      % l cycle
    a_sf_b = a_sf(:,l_cnt);
    w_ib_b = w_ib(:,l_cnt);
    %% High frequency update
    % 姿态更新 -- l cycle
    dAlpha_1 = dAlpha;
    dAlpha = w_ib_b*dTl;    % 角速率积分
    alpha_1 = alpha;
    alpha = alpha + dAlpha;
    dBeta = 0.5*cross( alpha_1+dAlpha_1/6, dAlpha );
    beta = beta + dBeta;
    % 速度更新 -- l cycle
    dNu_1 = dNu;
    dNu = (a_sf_b + a_sf_b_1)*(0.5*dTl);
    nu_1 = nu;
    nu = nu + dNu;
    ddV_scul = cross( alpha_1+dAlpha_1/6, dNu ) + cross( nu_1+dNu_1/6,
dAlpha );
    dV_scul_1 = dV_scul;
    dV_scul = dV_scul + ddV_scul*0.5;
    % 位置更新-- l cycle
    dSalpha = (alpha_1+(5*dAlpha+dAlpha_1)/12)*dTl;
    sAlpha_1 = sAlpha;
    sAlpha = sAlpha + dSalpha;
    dSnu = (nu_1+(5*dNu+dNu_1)/12)*dTl;
    sNu_1 = sNu;
    sNu = sNu + dSnu;
    ddAlpha = dAlpha - dAlpha_1;
    ddNu = dNu - dNu_1;
    ddR_scrlA = dV_scul_1*dTl ...
        + 0.5*cross(alpha_1-ddAlpha/12, dSnu-nu_1*dTl) ...
        + 0.5*cross(nu_1-ddNu/12, dSalpha-alpha_1*dTl);
    ddR_scrlB = 1/6*cross(sNu_1+dTl/24*ddNu, dAlpha) ...
```

```
        - 1/6*cross(sAlpha_1+dTl/24*ddAlpha, dNu) ...
        + dTl/6*cross(alpha_1-ddAlpha/6, nu_1-ddNu/6) ...
        - dTl/2160*cross(ddAlpha, ddNu);
    dR_scrl = dR_scrl + ddR_scrlA + ddR_scrlB;

    if m >= lmRatio    % m cycle
        %% low frequency update
        % 备份high update的数据,并清零
        % 姿态
        alpha_m = alpha;    % 备份alpha
        alpha = zeros(3,1);
        beta_m = beta;      % 备份beta
        beta = zeros(3,1);
        % 速度
        dV_scul_m = dV_scul;
        dV_scul = zeros(3,1);
        nu_m = nu;
        nu = zeros(3,1);
        % 位置
        sAlpha_m = sAlpha;
        sAlpha = zeros(3,1);
        sNu_m = sNu;
        sNu = zeros(3,1);
        dR_scrl_m = dR_scrl;
        dR_scrl = zeros(3,1);

        % 姿态更新 -- m cylce
        phi = alpha_m + beta_m;
        phi_len = norm(phi);
        phi_cross = [ 0, -phi(3), phi(2); phi(3), 0, -phi(1); -phi(2),
phi(1), 0 ];
        % 计算C_b2b，BI(m)->BI(m-1)
        coeff_1 = 1 - phi_len^2/6 + phi_len^4/120;% - phi_len^6/5040;    %
sin(x)/x
        coeff_2 = 0.5 - phi_len^2/24 + phi_len^4/720;% - phi_len^6/40320; %
(1-cos(x))/x^2
        C_b2b = matI + coeff_1*phi_cross + coeff_2*phi_cross*phi_cross;
        % 计算C_n2n, NI(m-1)->NI(m)
        w_ie_n_m = 1.5*w_ie_n_1 - 0.5*w_ie_n_2;     % w_ie_n(m-1/2)
        w_en_n_m = 1.5*w_en_n_1 - 0.5*w_en_n_2;     % w_en_n(m-1/2)
        zeta = (w_ie_n_m + w_en_n_m)*dTm;   % w_ie_n(m-1/2) + w_en_n(m-1/2)
        zeta_cross = [ 0, -zeta(3), zeta(2); zeta(3), 0, -zeta(1); -zeta(2),
```

```
zeta(1), 0 ];
        C_n2n = matI - zeta_cross + 0.5*zeta_cross*zeta_cross;
        % 计算C_b2n, B(m)->N(m)
        C_b2n_1 = C_b2n;
        C_b2n = C_n2n*C_b2n_1*C_b2b;
        % 正交化
        C_b2n = ( 1.5*matI - 0.5*C_b2n*(C_b2n') )*C_b2n;

        % 速度更新 -- m cylce
        % 计算dV_rot
        alpha_len = norm(alpha_m);
        coeff_1 = 0.5-alpha_len^2/24+alpha_len^4/720;      %
        coeff_2 = 1/6-alpha_len^2/120+alpha_len^4/5040;    %
        dV_rot = coeff_1*cross( alpha_m, nu_m ) ...
            + coeff_2*cross( alpha_m, cross(alpha_m,nu_m) );
        % 计算dV_sf_n
        dV_sf_b = nu_m + dV_rot + dV_scul_m;
        dV_sf_n = 0.5*(C_n2n + matI)*C_b2n_1*dV_sf_b;
        % 计算dV_gc_n
        g_n_m = 1.5*g_n_1 - 0.5*g_n_2;            % g_n(m-1/2)
        v_n_m = 1.5*v_n_1 - 0.5*v_n_2;            % v_n(m-1/2)
        dV_gc_n = ( g_n_m - cross(2*w_ie_n_m+w_en_n_m, v_n_m) ) * dTm;
        % 计算v_en_n
        v_n = v_n_1 + dV_gc_n + dV_sf_n;
        % 位置更新 -- m cycle
        % 计算dR_rot
        coeff_1 = 1/6 - alpha_len^2/120 + alpha_len^4/5040;
        coeff_2 = 1/24 - alpha_len^2/720 + alpha_len^4/40320;
        alpha_cross = [ 0, -alpha_m(3), alpha_m(2); alpha_m(3), 0,
-alpha_m(1); -alpha_m(2), alpha_m(1), 0 ];
        dR_rot = ( coeff_1*matI - coeff_2*alpha_cross )*( cross( sAlpha_m,
nu_m )...
            + cross( alpha_m, sNu_m ) );
        % 计算dR_sf_n
        dR_sf_b = sNu_m + dR_rot + dR_scrl_m;
        dR_sf_n = (C_n2n-matI)*C_b2n_1*dV_sf_b*(dTm/6) +
C_b2n_1*dR_sf_b;
        % 计算dR_n
        dR_n = (v_n_1 + 0.5*dV_gc_n)*dTm + dR_sf_n;
        % 计算C_n2e, NE(m)->E
        F_curv_m = 1.5*F_curv_1 - 0.5*F_curv_2;
        xi = F_curv_m*dR_n;
```

```
        xi_len = norm(xi);
        xi_cross = [ 0, -xi(3), xi(2); xi(3), 0, -xi(1); -xi(2), xi(1),
0 ];
        coeff_1 = 1 - xi_len^2/6 + xi_len^4/120;
        coeff_2 = 1/2 - xi_len^2/24 + xi_len^4/720;
        C_n2n_e = matI + coeff_1*xi_cross + coeff_2*xi_cross*xi_cross;  %
NE(m)->NE(m-1)
        C_n2e = C_n2e*C_n2n_e;
        % 正交化
        C_n2e = ( 1.5*matI - 0.5*C_n2e*(C_n2e') )*C_n2e;
        % 计算h_n
        if SURFACE_NAVIGATION == 0  % 三维导航
            h_n = h_n + dR_n(3);
        end
        % 计算g_n, w_ie_n, w_en_n, F_curv
        g_n = gravityCalc(C_n2e, h_n);
        w_ie_n = C_n2e'*w_ie_e;
        F_curv = curvMat(C_n2e, h_n, NORTH_SYSTEM);
        w_en_n = F_curv*v_n;
        % 延时
        % 速度
        v_n_2 = v_n_1;
        v_n_1 = v_n;
        % 与位置相关变量
        g_n_2 = g_n_1;
        g_n_1 = g_n;
        w_ie_n_2 = w_ie_n_1;
        w_ie_n_1 = w_ie_n;
        F_curv_2 = F_curv_1;
        F_curv_1 = F_curv;
        w_en_n_2 = w_en_n_1;
        w_en_n_1 = w_en_n;
        %% 记录输出值
        if NORTH_SYSTEM == 1       % 指北方位系统
            % 位置
            [ lat, lon, ~ ] = posCalc( C_n2e );
            pos(:,m_cnt) = [lat; lon; h_n];
            % 姿态
            [ pitch, roll, azimuth ] = attCalc( C_b2n );
            att(:,m_cnt) = [ pitch; roll; azimuth ];
            % 速度
            vel(:,m_cnt) = v_n;
```

```
        else                        % 游动自由方位系统
            % 位置
            [ lat, lon, wander ] = posCalc( C_n2e );
            pos(:,m_cnt) = [lat; lon; h_n];
            % 速度
            [ pitch, roll, azimuth ] = attCalc( C_b2n );
            azimuth = azimuth + wander;
            att(:,m_cnt) = [ pitch; roll; azimuth];
            % 姿态
            vel(:,m_cnt) = [cos(wander) -sin(wander) 0;
                sin(wander) cos(wander) 0;
                0 0 1] * vel_n;
        end
        % 更新m_cnt
        m_cnt = m_cnt+1;
        % m重新计数
        m = 1;
    else
        m = m+1;
    end
    % 延时
    a_sf_b_1 = a_sf_b;
end
```

51) trackingv_ms

```
function [trackResults, channel]= trackingv_ms(fid, channel, settings,
msToProcess)
% Performs code and carrier tracking for all channels.
%
%[trackResults, channel] = tracking(fid, channel, settings)
%
%   Inputs:
%       fid             - file identifier of the signal record for I
%       channel         - PRN, carrier frequencies and code phases of all
%                       satellites to be tracked (prepared by preRun.m from
%                       acquisition results).
%       settings        - receiver settings.
%   Outputs:
%       trackResults    - tracking results (structure array). Contains
%                       in-phase prompt outputs and absolute spreading
%                       code's starting positions, together with other
%                       observation data from the tracking loops. All are
```

```
%                   saved every millisecond.
%--------------------------------------------------------------------
% Copyright (C) Dennis M. Akos
% Written by Mwenjie @2014.03
% Based on SoftGNSS v3.0
%--------------------------------------------------------------------
%% Initialize result structure ====================================

% ms of Predection Integration time
msOfPDI = settings.msOfPDI;
% Predection Integration time
timeOfPDI = 0.001*msOfPDI;
% half of Predection Integration time
halfPDI = 0.001*floor(msOfPDI/2);
% Length of trackResults
trackLen = floor(msToProcess / msOfPDI);

% Channel status
trackResults.status        = zeros(1, msToProcess);     % No tracked
signal, or lost lock

% The absolute sample in the record of the C/A code start:
trackResults.absoluteSample = zeros(1, msToProcess);

% Freq of the C/A code:
trackResults.codeFreq      = inf(1, msToProcess);

% Frequency of the tracked carrier wave:
trackResults.carrFreq      = inf(1, msToProcess);

% Outputs from the correlators (In-phase):
trackResults.I_P           = zeros(1, msToProcess);
trackResults.I_E           = zeros(1, msToProcess);
trackResults.I_L           = zeros(1, msToProcess);

% Outputs from the correlators (Quadrature-phase):
trackResults.Q_E           = zeros(1, msToProcess);
trackResults.Q_P           = zeros(1, msToProcess);
trackResults.Q_L           = zeros(1, msToProcess);

% Loop discriminators
trackResults.dllDiscr      = inf(1, trackLen);
```

```
trackResults.dllDiscrFilt   = inf(1, trackLen);
trackResults.pllDiscr      = inf(1, trackLen);
trackResults.pllDiscrFilt   = inf(1, trackLen);
trackResults.fllDiscr      = inf(1, trackLen);

%--- Copy initial settings for all channels -------------------------
trackResults = repmat(trackResults, 1, settings.numberOfChannels);

%% Initialize tracking variables ====================================

codePeriods = msToProcess;    % For GPS one C/A code is one ms

%--- DLL variables --------------------------------------------------
% Define early-late offset (in chips)
earlyLateSpc = settings.dllCorrelatorSpacing;

% % early, late, and prompt parameters
% I_E_pdi = zeros(1,msOfPDI);
% Q_E_pdi = zeros(1,msOfPDI);
% I_P_pdi = zeros(1,msOfPDI);
% Q_P_pdi = zeros(1,msOfPDI);
% I_L_pdi = zeros(1,msOfPDI);
% Q_L_pdi = zeros(1,msOfPDI);

%--- Kalman filter variables
-----------------------------------------------------------
dTf = timeOfPDI;    % kalman filter updating interval
% beta = settings.codeFreqBasis / settings.carrFreqBasis;

lambda_code = settings.c / settings.codeFreqBasis;
lambda_carr = settings.c / settings.carrFreqBasis;

% Process transition matrix
F = [ 1, 0, -2*pi*dTf/lambda_carr;
    0, 1, dTf/lambda_code;
    0, 0, 1 ];
% Covariance of process noises
Q = diag( ([-pi*dTf^2/lambda_carr, dTf^2/2/lambda_code, dTf]*5).^2 );
% Input matrix
G = [ -2*pi*dTf, 0;
    0, -dTf;
    0, 0 ];
```

```
% Measurement matrix
H = [ 1, 0, pi*dTf/lambda_carr;
    0, 1, -0.5*dTf/lambda_code;
    0, 0, -1/lambda_carr ];
D = [ pi*dTf, 0;
    0, 0.5*dTf;
    -1, 0 ];
% Covariance of measurement noises
% R = diag( [0.25, 0.5, 100].^2 );
% R = diag( [0.2, 0.1, 25].^2 );

if (settings.fileType==1)
    dataAdaptCoeff=1;
else
    dataAdaptCoeff=2;
end

%% Start processing channels ====================================
for channelNr = 1:settings.numberOfChannels

    % Only process if PRN is non zero (acquisition was successful)
    if (channel(channelNr).PRN == 0)
        continue;
    end

    % Save additional information - each channel's tracked PRN
    trackResults(channelNr).PRN     = channel(channelNr).PRN;

    % Move the starting point of processing. Can be used to start the
    % signal processing at any point in the data record (e.g. for long
    % records). In addition skip through that data file to start at the
    % appropriate sample (corresponding to code phase). Assumes sample
    % type is schar (or 1 byte per sample)
    fseek(fid, dataAdaptCoeff*channel(channelNr).startSample, 'bof');

    % Get a vector with the C/A code sampled 1x/chip
    caCode0 = generateCode(channel(channelNr).PRN, settings.system);
    % Then make it possible to do early and late versions
    caCode = [caCode0(settings.codeLength) caCode0 caCode0(1)];

    %--- Perform various initializations ------------------------------
```

```
    % define initial code frequency basis of NCO
    codeFreq     = channel(channelNr).codeFreq;
    % define residual code phase (in chips)
    remCodePhase  = channel(channelNr).remCodePhase;
    % define carrier frequency which is used over whole tracking period
    carrFreq     = channel(channelNr).carrFreq;
    % define residual carrier phase
    remCarrPhase  = channel(channelNr).remCarrPhase;

    %code tracking loop parameters
    codeNco = settings.codeFreqBasis - codeFreq;    %
channel(channelNr).codeNco;
%    codeError = channel(channelNr).codeError;

    %carrier/Costas loop parameters
    carrNco = carrFreq - settings.IF;            %
channel(channelNr).carrNco;
%    carrError = channel(channelNr).carrError;

    %--- Kalman filter variables ---------------------------------
    x = channel(channelNr).x;      % [ 0; 0; -carrNco*lambda_carr ];
    P = channel(channelNr).P;      % diag( ([10, 1, 100]).^2 );
    % Covariance of measurement noises
    [varCarrPhase, varCodePhase, varCarrFreq] =
varEstimate( channel(channelNr).CNo, ...
        settings.dllCorrelatorSpacing, settings.msOfPDI*0.001 );
    R = diag( [varCarrPhase, varCodePhase, varCarrFreq]*5^2 );

    %--- Phase lock detector -------------------------------------
    phaseLockDetector = channel(channelNr).phaseLockDetector;

    % early, late, and prompt parameters
    I_Es = 0.0;
    Q_Es = 0.0;
    I_Ps = 0.0;
    Q_Ps = 0.0;
    I_Ls = 0.0;
    Q_Ls = 0.0;

    I_Ps1 = 0.0;
    Q_Ps1 = 0.0;
    I_Ps2 = 0.0;
```

```
    Q_Ps2 = 0.0;

    trackCnt = 1;
    pdiCnt = 1;

    %=== Process the number of specified code periods ================
    for loopCnt =  1:codePeriods


%% Read next block of data ------------------------------------
        % Find the size of a "block" or code period in whole samples

        % Update the phasestep based on code freq (variable) and
        % sampling frequency (fixed)
        codePhaseStep = codeFreq / settings.samplingFreq;

        blksize = ceil((settings.codeLength-remCodePhase) /
codePhaseStep);

        % Read in the appropriate number of samples to process this
        % interation
        [rawSignal, samplesRead] = fread(fid, ...
            dataAdaptCoeff*blksize, settings.dataType);

        rawSignal = rawSignal';    % transpose vector

        if (dataAdaptCoeff==2)
            rawSignal1=rawSignal(1:2:end);
            rawSignal2=rawSignal(2:2:end);
            rawSignal = rawSignal1 + 1i .* rawSignal2;  %transpose vector
        end

        % If did not read in enough samples, then could be out of
        % data - better exit
        if (samplesRead ~= dataAdaptCoeff*blksize)
            disp('Not able to read the specified number of samples  for
tracking, exiting!')
            fclose(fid);
            return
        end

%% Set up all the code phase tracking information -----------------
```

```
        % Define index into early code vector
        tcode      = (remCodePhase-earlyLateSpc) : codePhaseStep : ...
 ((blksize-1)*codePhaseStep+remCodePhase-earlyLateSpc);
        tcode2     = ceil(tcode) + 1;
        earlyCode   = caCode(tcode2);
        % Define index into late code vector
        tcode      = (remCodePhase+earlyLateSpc) : codePhaseStep : ...
 ((blksize-1)*codePhaseStep+remCodePhase+earlyLateSpc);
        tcode2     = ceil(tcode) + 1;
        lateCode    = caCode(tcode2);
        % Define index into prompt code vector
        tcode      = remCodePhase : codePhaseStep : ...
                    ((blksize-1)*codePhaseStep+remCodePhase);
        tcode2     = ceil(tcode) + 1;
        promptCode  = caCode(tcode2);
        remCodePhase = (tcode(blksize) + codePhaseStep) -
settings.codeLength;

%% Generate the carrier frequency to mix the signal to baseband -----------
        time    = (0:blksize) ./ settings.samplingFreq;
        % Get the argument to sin/cos functions
        trigarg = ((carrFreq * 2.0 * pi) .* time) + remCarrPhase;
        remCarrPhase = rem(trigarg(blksize+1), (2 * pi));
        % Finally compute the signal to mix the collected data to bandband
        carrsig = exp(1i .* trigarg(1:blksize));

%% Generate the six standard accumulated values ------------------
        % First mix to baseband
        qBasebandSignal = real(carrsig .* rawSignal);
        iBasebandSignal = imag(carrsig .* rawSignal);
        % Now get early, late, and prompt values for each
        I_E = sum(earlyCode  .* iBasebandSignal);
        Q_E = sum(earlyCode  .* qBasebandSignal);
        I_P = sum(promptCode .* iBasebandSignal);
        Q_P = sum(promptCode .* qBasebandSignal);
        I_L = sum(lateCode   .* iBasebandSignal);
        Q_L = sum(lateCode   .* qBasebandSignal);
        %
        signP = sign(I_P);
        I_Es = I_Es + I_E*signP;
        Q_Es = Q_Es + Q_E*signP;
        I_Ps = I_Ps + I_P*signP;
```

```
        Q_Ps = Q_Ps + Q_P*signP;
        I_Ls = I_Ls + I_L*signP;
        Q_Ls = Q_Ls + Q_L*signP;
        % I and Q output for FLL
        if( pdiCnt <= ceil(msOfPDI/2) )
            I_Ps1 = I_Ps1 + I_P*signP;
            Q_Ps1 = Q_Ps1 + Q_P*signP;
        end
        if( pdiCnt > msOfPDI - ceil(msOfPDI/2) )
            I_Ps2 = I_Ps2 + I_P*signP;
            Q_Ps2 = Q_Ps2 + Q_P*signP;
        end
        pdiCnt = pdiCnt + 1;

%% Phase lock detector
        phaseLockDetector.A = phaseLockDetector.A + ...
            (abs(I_P) - phaseLockDetector.A) *
settings.phaseLockDetector.k1;
        phaseLockDetector.B = phaseLockDetector.B + ...
            (abs(Q_P) - phaseLockDetector.B) *
settings.phaseLockDetector.k1;

        if (phaseLockDetector.A / settings.phaseLockDetector.k2) >
phaseLockDetector.B
            phaseLockDetector.optimisticLock = 1;
            phaseLockDetector.cnt2 = 0;
            if phaseLockDetector.cnt1 > settings.phaseLockDetector.Lp
                phaseLockDetector.pessimisticLock = 1;
            else
                phaseLockDetector.cnt1 = phaseLockDetector.cnt1 + 1;
            end
        else
            phaseLockDetector.pessimisticLock = 0;
            phaseLockDetector.cnt1 = 0;
            if phaseLockDetector.cnt2 > settings.phaseLockDetector.Lo
                phaseLockDetector.optimisticLock = 0;
            else
                phaseLockDetector.cnt2 = phaseLockDetector.cnt2 + 1;
            end
        end

        if (phaseLockDetector.optimisticLock == 1 && ...
```

```
                phaseLockDetector.pessimisticLock == 1)
            channel(channelNr).status = 'T';
        elseif (phaseLockDetector.optimisticLock == 0 && ...
                phaseLockDetector.pessimisticLock == 0)
            channel(channelNr).status = '-';
        end

%%       %=== Prefilter ========================================
        if (rem(loopCnt, msOfPDI) == 0)
            pdiCnt = 1;
            % PLL
            carrError = atan(Q_Ps / I_Ps);
            % DLL
            codeError = (1-earlyLateSpc) * ...
                ( sqrt(I_Es * I_Es + Q_Es * Q_Es) - ...
                sqrt(I_Ls * I_Ls + Q_Ls * Q_Ls) ) / ...
                ( sqrt(I_Es * I_Es + Q_Es * Q_Es) + ...
                sqrt(I_Ls * I_Ls + Q_Ls * Q_Ls) );
            % FLL
            Dot = I_Ps1 * I_Ps2 + Q_Ps1 * Q_Ps2;
            Cross = I_Ps1 * Q_Ps2 - I_Ps2 * Q_Ps1;
            freqError = atan2(Cross, Dot) / halfPDI / (2.0 * pi);
            %% Prefilter
            u = [carrNco; codeNco];
            z = [carrError; codeError; freqError];
            [x,P] = kf(x,P,F,Q,G,u,z,H,D,R);

            carrNco = -x(3) / lambda_carr + 1*x(1)/(2.0*pi)/dTf;  %
            codeNco = x(3) / lambda_code + 1*x(2)/dTf;   %
            carrFreq = settings.IF + carrNco;
            codeFreq = settings.codeFreqBasis - codeNco;

%% Record various measures to show in postprocessing ----------------
            % Record sample number (based on 8bit samples)

            trackResults(channelNr).dllDiscr(trackCnt)      =
codeError;
            trackResults(channelNr).dllDiscrFilt(trackCnt)   =
codeNco; % x(2);
            trackResults(channelNr).pllDiscr(trackCnt)      =
carrError;
            trackResults(channelNr).pllDiscrFilt(trackCnt)   =
```

```
carrNco; % x(1);

          trackResults(channelNr).fllDiscr(trackCnt)      =
freqError;

          % record channel information
          channel(channelNr).startSample  =
(ftell(fid))/dataAdaptCoeff;
%           channel(channelNr).codePhase    = 0;
          channel(channelNr).carrFreq    = carrFreq;
          channel(channelNr).codeFreq    = codeFreq;
          %    channel(channelNr).carrError      = carrError;
          %    channel(channelNr).carrNco        = carrNco;
          %    channel(channelNr).codeError      = codeError;
          %    channel(channelNr).codeNco        = codeNco;
          channel(channelNr).remCarrPhase    = remCarrPhase;
          channel(channelNr).remCodePhase    = remCodePhase;
          channel(channelNr).x = x;
          channel(channelNr).P = P;
          channel(channelNr).phaseLockDetector = phaseLockDetector;
          I_Es = 0.0;
          Q_Es = 0.0;
          I_Ps = 0.0;
          Q_Ps = 0.0;
          I_Ls = 0.0;
          Q_Ls = 0.0;
          I_Ps1 = 0.0;
          I_Ps2 = 0.0;
          Q_Ps1 = 0.0;
          Q_Ps2 = 0.0;
          trackCnt = trackCnt + 1;

      end % if (rem(loopCnt, msOfPDI) == 0)

      % Evaluate the tracking results status here to ensure the
      % plotTracking to plot the results tracked so far
      % (In case the tracking update window is closed)
      if channel(channelNr).status == 'T'
          trackResults(channelNr).status(loopCnt) = 1;
      end

      trackResults(channelNr).I_E(loopCnt) = I_E;
```

```
        trackResults(channelNr).I_P(loopCnt) = I_P;
        trackResults(channelNr).I_L(loopCnt) = I_L;
        trackResults(channelNr).Q_E(loopCnt) = Q_E;
        trackResults(channelNr).Q_P(loopCnt) = Q_P;
        trackResults(channelNr).Q_L(loopCnt) = Q_L;
        trackResults(channelNr).absoluteSample(loopCnt) = ...
            (ftell(fid))/dataAdaptCoeff - remCodePhase/codePhaseStep;
        trackResults(channelNr).carrFreq(loopCnt) = carrFreq;
%         trackResults(channelNr).remCarrPhase(loopCnt) = remCarrPhase;

        trackResults(channelNr).codeFreq(loopCnt) = codeFreq;
%         trackResults(channelNr).remCodePhase(loopCnt) = remCodePhase;

    end % for loopCnt

end % for channelNr

% % Close the waitbar
% close(hwb)
```

52) tracking_ms

```
function [trackResults, channel]= tracking_ms(fid, channel, settings,
msToProcess, msOfPDI)
% Performs code and carrier tracking for all channels.
%
%[trackResults, channel] = tracking(fid, channel, settings)
%
%   Inputs:
%       fid             - file identifier of the signal record for I
%       channel         - PRN, carrier frequencies and code phases of all
%                       satellites to be tracked (prepared by preRun.m from
%                       acquisition results).
%       settings        - receiver settings.
%   Outputs:
%       trackResults    - tracking results (structure array). Contains
%                       in-phase prompt outputs and absolute spreading
%                       code's starting positions, together with other
%                       observation data from the tracking loops. All are
%                       saved every millisecond.
%--------------------------------------------------------------------
% Copyright (C) Dennis M. Akos
% Written by Darius Plausinaitis and Dennis M. Akos
```

```
% Based on code by DMAkos Oct-1999
% Modified by Mwenjie @2014.03
%--------------------------------------------------------------------
%% Initialize result structure ====================================
% Length of trackResults
trackLen = floor(msToProcess / msOfPDI);
% Channel status
trackResults.status       = '-';     % No tracked signal, or lost lock
% The absolute sample in the record of the C/A code start:
trackResults.absoluteSample = zeros(1, msToProcess);
% Freq of the C/A code:
trackResults.codeFreq      = inf(1, msToProcess);
% Frequency of the tracked carrier wave:
trackResults.carrFreq      = inf(1, msToProcess);
% Outputs from the correlators (In-phase):
trackResults.I_P          = zeros(1, msToProcess);
trackResults.I_E          = zeros(1, msToProcess);
trackResults.I_L          = zeros(1, msToProcess);
% Outputs from the correlators (Quadrature-phase):
trackResults.Q_E          = zeros(1, msToProcess);
trackResults.Q_P          = zeros(1, msToProcess);
trackResults.Q_L          = zeros(1, msToProcess);
% Outputs from the correlators (In-phase):
trackResults.I_P_pdi          = zeros(1, trackLen);
trackResults.I_E_pdi          = zeros(1, trackLen);
trackResults.I_L_pdi          = zeros(1, trackLen);
% Outputs from the correlators (Quadrature-phase):
trackResults.Q_E_pdi          = zeros(1, trackLen);
trackResults.Q_P_pdi          = zeros(1, trackLen);
trackResults.Q_L_pdi          = zeros(1, trackLen);
% Loop discriminators
trackResults.dllDiscr      = inf(1, trackLen);
trackResults.dllDiscrFilt   = inf(1, trackLen);
trackResults.pllDiscr      = inf(1, trackLen);
trackResults.pllDiscrFilt   = inf(1, trackLen);
trackResults.fllDiscr      = inf(1, trackLen);
% C/No
trackResults.CNo.VSMValue = ...
   zeros(1,floor(msToProcess/settings.CNo.VSMinterval));
trackResults.CNo.VSMIndex = ...
   zeros(1,floor(msToProcess/settings.CNo.VSMinterval));
trackResults.CNo.PRMValue=0; %To avoid error message when
```

```
trackResults.CNo.PRMIndex=0; %tracking window is closed before
completion.
%--- Copy initial settings for all channels -----------------------
trackResults = repmat(trackResults, 1, settings.numberOfChannels);

%% Initialize tracking variables ====================================
codePeriods = msToProcess;     % For GPS one C/A code is one ms
%--- DLL variables ------------------------------------------------
% Define early-late offset (in chips)
earlyLateSpc = settings.dllCorrelatorSpacing;
% Summation interval
PDIcode = 0.001*msOfPDI;    % [s]
% Calculate filter coefficient values
[tau1code, tau2code] =
calcLoopCoef(settings.dllNoiseBandwidth/sqrt(msOfPDI), ...
                              settings.dllDampingRatio, ...
                              1.0);
%--- PLL variables ------------------------------------------------
% Summation interval
PDIcarr = 0.001*msOfPDI;    % [s]
% Calculate filter coefficient values
[tau1carr, tau2carr] =
calcLoopCoef(settings.pllNoiseBandwidth/sqrt(msOfPDI), ...
                              settings.pllDampingRatio, ...
                              0.25);
hwb = waitbar(0,'Tracking...','Visible','off');
% Adjust the size of the waitbar to insert text
CNoPos = get(hwb,'Position');
set(hwb,'Position',[CNoPos(1),CNoPos(2),CNoPos(3),90],'Visible','on')
;
if (settings.fileType==1)
    dataAdaptCoeff=1;
else
    dataAdaptCoeff=2;
end
%% Start processing channels ========================================
for channelNr = 1:settings.numberOfChannels
    % Only process if PRN is non zero (acquisition was successful)
    if (channel(channelNr).PRN == 0)
        continue;
    end
```

```
    % Save additional information - each channel's tracked PRN
    trackResults(channelNr).PRN     = channel(channelNr).PRN;
    % Move the starting point of processing. Can be used to start the
    % signal processing at any point in the data record (e.g. for long
    % records). In addition skip through that data file to start at the
    % appropriate sample (corresponding to code phase). Assumes sample
    % type is schar (or 1 byte per sample)
    fseek(fid, dataAdaptCoeff*channel(channelNr).startSample, 'bof');
    % Get a vector with the C/A code sampled 1x/chip
%     caCode0 = generateCAcode(channel(channelNr).PRN);
    caCode0 = generateCode(channel(channelNr).PRN, settings.system);
    % Then make it possible to do early and late versions
    caCode = [caCode0(settings.codeLength) caCode0 caCode0(1)];
    %--- Perform various initializations ------------------------------
    % define initial code frequency basis of NCO
    codeFreq     = channel(channelNr).codeFreq;
    % define residual code phase (in chips)
    remCodePhase  = channel(channelNr).remCodePhase;
    % define carrier frequency which is used over whole tracking period
    carrFreq     = channel(channelNr).carrFreq;
    carrFreqBasis = channel(channelNr).carrFreq;    %
channel(channelNr).acquiredFreq;
    % define residual carrier phase
    remCarrPhase  = channel(channelNr).remCarrPhase;
    %code tracking loop parameters
    oldCodeNco   = 0.0;    % channel(channelNr).codeNco
    oldCodeError = 0.0;    % channel(channelNr).codeError;
    %carrier/Costas loop parameters
    oldCarrNco   = 0.0;    % channel(channelNr).carrNco
    oldCarrError = 0.0;    % channel(channelNr).carrError
    % early, late, and prompt parameters
    I_Es = 0.0;
    Q_Es = 0.0;
    I_Ps = 0.0;
    Q_Ps = 0.0;
    I_Ls = 0.0;
    Q_Ls = 0.0;
    I_Ps1 = 0.0;
    Q_Ps1 = 0.0;
    I_Ps2 = 0.0;
    Q_Ps2 = 0.0;
    %C/No computation
```

```
    vsmCnt  = 0;
    if (settings.CNo.enableVSM==1)
        CNo='Calculating...';
    else
        CNo='Disabled';
    end

    trackCnt = 1;
    pdiCnt = 1;

    %=== Process the number of specified code periods ================
    for loopCnt =  1:codePeriods

%% GUI update
-------------------------------------------------------------
       % The GUI is updated every 50ms. This way Matlab GUI is still
       % responsive enough. At the same time Matlab is not occupied
       % all the time with GUI task.
       Ln=sprintf('\n');
       trackingStatus=['Tracking: Ch ', int2str(channelNr), ...
           ' of ', int2str(settings.numberOfChannels),Ln ...
           'PRN: ', int2str(channel(channelNr).PRN),Ln ...
           'Completed ',int2str(loopCnt), ...
           ' of ', int2str(codePeriods), ' msec',Ln...
           'C/No: ',CNo,' (dB-Hz)'];

       if (rem(loopCnt, 50) == 0)
           try
               waitbar(loopCnt/codePeriods, ...
                   hwb, ...
                   trackingStatus);
           catch
               % The progress bar was closed. It is used as a signal
               % to stop, "cancel" processing. Exit.
               disp('Progress bar closed, exiting...');
               return
           end
       end

%% Read next block of data ----------------------------------------
       % Find the size of a "block" or code period in whole samples
       % Update the phasestep based on code freq (variable) and
```

```
        % sampling frequency (fixed)
        codePhaseStep = codeFreq / settings.samplingFreq;
        blksize = ceil((settings.codeLength-remCodePhase) /
codePhaseStep);
        % Read in the appropriate number of samples to process this
        % interation
        [rawSignal, samplesRead] = fread(fid, ...
            dataAdaptCoeff*blksize, settings.dataType);
        rawSignal = rawSignal';    % transpose vector
        if (dataAdaptCoeff==2)
            rawSignal1=rawSignal(1:2:end);
            rawSignal2=rawSignal(2:2:end);
            rawSignal = rawSignal1 + 1i .* rawSignal2;  %transpose vector
        end
        % If did not read in enough samples, then could be out of
        % data - better exit
        if (samplesRead ~= dataAdaptCoeff*blksize)
            disp('Not able to read the specified number of samples  for
tracking, exiting!')
            fclose(fid);
            return
        end

%% Set up all the code phase tracking information -------------------
        % Define index into early code vector
        tcode      = (remCodePhase-earlyLateSpc) : ...
                     codePhaseStep : ...
 ((blksize-1)*codePhaseStep+remCodePhase-earlyLateSpc);
        tcode2     = ceil(tcode) + 1;
        earlyCode   = caCode(tcode2);
        % Define index into late code vector
        tcode      = (remCodePhase+earlyLateSpc) : ...
                     codePhaseStep : ...
 ((blksize-1)*codePhaseStep+remCodePhase+earlyLateSpc);
        tcode2     = ceil(tcode) + 1;
        lateCode    = caCode(tcode2);
        % Define index into prompt code vector
        tcode      = remCodePhase : ...
                     codePhaseStep : ...
                     ((blksize-1)*codePhaseStep+remCodePhase);
        tcode2     = ceil(tcode) + 1;
        promptCode  = caCode(tcode2);
```

```
        remCodePhase = (tcode(blksize) + codePhaseStep) -
settings.codeLength;

%% Generate the carrier frequency to mix the signal to baseband -----------
        time    = (0:blksize) ./ settings.samplingFreq;
        % Get the argument to sin/cos functions
        trigarg = ((carrFreq * 2.0 * pi) .* time) + remCarrPhase;
        remCarrPhase = rem(trigarg(blksize+1), (2 * pi));
        % Finally compute the signal to mix the collected data to bandband
        carrsig = exp(1i .* trigarg(1:blksize));

%% Generate the six standard accumulated values --------------------
        % First mix to baseband
        qBasebandSignal = real(carrsig .* rawSignal);
        iBasebandSignal = imag(carrsig .* rawSignal);
        % Now get early, late, and prompt values for each
        I_E = sum(earlyCode  .* iBasebandSignal);
        Q_E = sum(earlyCode  .* qBasebandSignal);
        I_P = sum(promptCode .* iBasebandSignal);
        Q_P = sum(promptCode .* qBasebandSignal);
        I_L = sum(lateCode   .* iBasebandSignal);
        Q_L = sum(lateCode   .* qBasebandSignal);

        trackResults(channelNr).I_E(loopCnt) = I_E;
        trackResults(channelNr).I_P(loopCnt) = I_P;
        trackResults(channelNr).I_L(loopCnt) = I_L;
        trackResults(channelNr).Q_E(loopCnt) = Q_E;
        trackResults(channelNr).Q_P(loopCnt) = Q_P;
        trackResults(channelNr).Q_L(loopCnt) = Q_L;

        trackResults(channelNr).absoluteSample(loopCnt) =
(ftell(fid))/dataAdaptCoeff - remCodePhase/codePhaseStep;

        signP = sign(I_P);
        I_Es = I_Es + I_E*signP;
        Q_Es = Q_Es + Q_E*signP;
        I_Ps = I_Ps + I_P*signP;
        Q_Ps = Q_Ps + Q_P*signP;
        I_Ls = I_Ls + I_L*signP;
        Q_Ls = Q_Ls + Q_L*signP;

        if( pdiCnt <= ceil(msOfPDI/2) )
```

```
            I_Ps1 = I_Ps1 + I_P*signP;
            Q_Ps1 = Q_Ps1 + Q_P*signP;
        end
        if( pdiCnt > msOfPDI - ceil(msOfPDI/2) )
            I_Ps2 = I_Ps2 + I_P*signP;
            Q_Ps2 = Q_Ps2 + Q_P*signP;
        end
        pdiCnt = pdiCnt + 1;

        %=== Prefilter ==========================================
        if (rem(loopCnt, msOfPDI) ~= 0)

        else

%% Find PLL error and update carrier NCO ------------------------------
            % Implement carrier loop discriminator (phase detector)
            carrError = atan(Q_Ps / I_Ps) / (2.0 * pi);
            % Implement carrier loop filter and generate NCO command
            carrNco = oldCarrNco + (tau2carr/tau1carr) * ...
                (carrError - oldCarrError) + carrError *
(PDIcarr/tau1carr);
            oldCarrNco   = carrNco;
            oldCarrError = carrError;
            % Modify carrier freq based on NCO command
            carrFreq = carrFreqBasis + carrNco;


%% Find DLL error and update code NCO ------------------------------
            codeError = (sqrt(I_Es * I_Es + Q_Es * Q_Es) - sqrt(I_Ls * I_Ls
+ Q_Ls * Q_Ls)) / ...
                (sqrt(I_Es * I_Es + Q_Es * Q_Es) + sqrt(I_Ls * I_Ls + Q_Ls
* Q_Ls));
            % Implement code loop filter and generate NCO command
            codeNco = oldCodeNco + (tau2code/tau1code) * ...
                (codeError - oldCodeError) + codeError *
(PDIcode/tau1code);
            oldCodeNco   = codeNco;
            oldCodeError = codeError;
            % Modify code freq based on NCO command
            codeFreq = settings.codeFreqBasis - codeNco;

%% Find FLL error
```

```
            Dot = I_Ps1 * I_Ps2 + Q_Ps1 * Q_Ps2;
            Cross = I_Ps1 * Q_Ps2 - I_Ps2 * Q_Ps1;
            freqError = atan2(Cross, Dot) / (0.001*floor(msOfPDI/2)) / (2.0
* pi);

%% Record various measures to show in postprocessing ----------------
            % Record sample number (based on 8bit samples)
            trackResults(channelNr).dllDiscr(trackCnt)      =
codeError;
            trackResults(channelNr).dllDiscrFilt(trackCnt)   = codeNco;
            trackResults(channelNr).pllDiscr(trackCnt)      =
carrError;
            trackResults(channelNr).pllDiscrFilt(trackCnt)   = carrNco;

            trackResults(channelNr).fllDiscr(trackCnt)      =
freqError;
            trackResults(channelNr).I_E_pdi(trackCnt) = I_Es;
            trackResults(channelNr).I_P_pdi(trackCnt) = I_Ps;
            trackResults(channelNr).I_L_pdi(trackCnt) = I_Ls;
            trackResults(channelNr).Q_E_pdi(trackCnt) = Q_Es;
            trackResults(channelNr).Q_P_pdi(trackCnt) = Q_Ps;
            trackResults(channelNr).Q_L_pdi(trackCnt) = Q_Ls;

            if (settings.CNo.enableVSM==1)
                if (rem(loopCnt,settings.CNo.VSMinterval)==0)
                    vsmCnt=vsmCnt+1;

CNoValue=CNoVSM(trackResults(channelNr).I_P(loopCnt-settings.CNo.VSMi
nterval+1:loopCnt),...
trackResults(channelNr).Q_P(loopCnt-settings.CNo.VSMinterval+1:loopCn
t),settings.CNo.accTime);
trackResults(channelNr).CNo.VSMValue(vsmCnt)=CNoValue;
trackResults(channelNr).CNo.VSMIndex(vsmCnt)=loopCnt;
                    CNo=int2str(CNoValue);
                end
            end
            % Evaluate the tracking results status here to ensure the
            % plotTracking to plot the results tracked so far
            % (In case the tracking update window is closed)
            trackResults(channelNr).status  = channel(channelNr).status;

            I_Es = 0.0;
```

```
            Q_Es = 0.0;
            I_Ps = 0.0;
            Q_Ps = 0.0;
            I_Ls = 0.0;
            Q_Ls = 0.0;

            I_Ps1 = 0.0;
            Q_Ps1 = 0.0;
            I_Ps2 = 0.0;
            Q_Ps2 = 0.0;

            trackCnt = trackCnt + 1;
            pdiCnt = 1;
        end % if (rem(loopCnt, msOfPDI) == 0)
        trackResults(channelNr).carrFreq(loopCnt) = carrFreq;
%         trackResults(channelNr).remCarrPhase(loopCnt) = remCarrPhase;
        trackResults(channelNr).codeFreq(loopCnt) = codeFreq;
%         trackResults(channelNr).remCodePhase(loopCnt) = remCodePhase;
    end % for loopCnt
    % If we got so far, this means that the tracking was successful
    % Now we only copy status, but it can be update by a lock detector
    % if implemented
    %trackResults(channelNr).status  = channel(channelNr).status;

    channel(channelNr).startSample = (ftell(fid))/dataAdaptCoeff;
    channel(channelNr).codePhase = 0;
    channel(channelNr).carrFreq = carrFreq;
    channel(channelNr).codeFreq = codeFreq;
    channel(channelNr).carrError      = carrError;
    channel(channelNr).carrNco        = carrNco;
    channel(channelNr).codeError      = codeError;
    channel(channelNr).codeNco        = codeNco;
    channel(channelNr).remCarrPhase    = remCarrPhase;
    channel(channelNr).remCodePhase    = remCodePhase;

    channel(channelNr).x(3) = -(channel(channelNr).carrFreq -
settings.IF) * ...
        settings.c / settings.carrFreqBasis;
end % for channelNr

% Close the waitbar
close(hwb)
```

53) tracking_sll

```
function [trackResults, channel]= tracking_sll(fid, channel, settings,
msToProcess)
% Performs code and carrier tracking for all channels.
%
%[trackResults, channel] = tracking(fid, channel, settings)
%
%  Inputs:
%      fid           - file identifier of the signal record for I
%      channel        - PRN, carrier frequencies and code phases of all
%                    satellites to be tracked (prepared by preRun.m from
%                    acquisition results).
%      settings       - receiver settings.
%  Outputs:
%      trackResults    - tracking results (structure array). Contains
%                    in-phase prompt outputs and absolute spreading
%                    code's starting positions, together with other
%                    observation data from the tracking loops. All are
%                    saved every millisecond.
%--------------------------------------------------------------------
% Copyright (C) Dennis M. Akos
% Written by Mwenjie @2014.03
% Based on SoftGNSS v3.0
%--------------------------------------------------------------------
%% Initialize result structure ====================================

% ms of Predection Integration time
msOfPDI = settings.msOfPDI;
% Predection Integration time
timeOfPDI = 0.001*msOfPDI;
% half of Predection Integration time
halfPDI = 0.001*floor(msOfPDI/2);
% Length of trackResults
trackLen = floor(msToProcess / msOfPDI);

% Channel status
trackResults.status        = '-';     % No tracked signal, or lost lock
% The absolute sample in the record of the C/A code start:
trackResults.absoluteSample = zeros(1, msToProcess);
% Freq of the C/A code:
trackResults.codeFreq      = inf(1, msToProcess);
```

```
% Frequency of the tracked carrier wave:
trackResults.carrFreq      = inf(1, msToProcess);
% Outputs from the correlators (In-phase):
trackResults.I_P           = zeros(1, msToProcess);
trackResults.I_E           = zeros(1, msToProcess);
trackResults.I_L           = zeros(1, msToProcess);
% Outputs from the correlators (Quadrature-phase):
trackResults.Q_E           = zeros(1, msToProcess);
trackResults.Q_P           = zeros(1, msToProcess);
trackResults.Q_L           = zeros(1, msToProcess);
% Loop discriminators
trackResults.dllDiscr      = inf(1, trackLen);
trackResults.dllDiscrFilt   = inf(1, trackLen);
trackResults.pllDiscr      = inf(1, trackLen);
trackResults.pllDiscrFilt   = inf(1, trackLen);
trackResults.fllDiscr      = inf(1, trackLen);
% C/No
trackResults.CNo = ...
    zeros(1,floor(msToProcess/settings.CNo.VSMinterval));
%--- Copy initial settings for all channels -------------------------
trackResults = repmat(trackResults, 1, settings.numberOfChannels);

%% Initialize tracking variables ==================================
codePeriods = msToProcess;    % For GPS one C/A code is one ms
%--- DLL variables --------------------------------------------------
% Define early-late offset (in chips)
earlyLateSpc = settings.dllCorrelatorSpacing;

%--- Kalman filter variables ----------------------------------------
dTf = timeOfPDI;    % kalman filter updating interval
% beta = settings.codeFreqBasis / settings.carrFreqBasis;
lambda_code = settings.c / settings.codeFreqBasis;
lambda_carr = settings.c / settings.carrFreqBasis;
% Process transition matrix
F = [ 1, 0, -2*pi*dTf/lambda_carr;
    0, 1, dTf/lambda_code;
    0, 0, 1 ];
% Covariance of process noises
Q = diag( ([-pi*dTf^2/lambda_carr, dTf^2/2/lambda_code, dTf]*5).^2 );
% Input matrix
G = [ -2*pi*dTf, 0;
    0, -dTf;
```

```
    0, 0 ];
% Measurement matrix
H = [ 1, 0, pi*dTf/lambda_carr;
    0, 1, -0.5*dTf/lambda_code;
    0, 0, -1/lambda_carr ];
D = [ pi*dTf, 0;
    0, 0.5*dTf;
    -1, 0 ];
% Covariance of measurement noises
% R = diag( [0.25, 0.5, 100].^2 );
% R = diag( [0.2, 0.1, 25].^2 );

%--- Ohters variables -------------------------------------------

hwb = waitbar(0,'Tracking...','Visible','off');

% Adjust the size of the waitbar to insert text
CNoPos = get(hwb,'Position');
set(hwb,'Position',[CNoPos(1),CNoPos(2),CNoPos(3),90],'Visible','on')
;
if (settings.fileType==1)
    dataAdaptCoeff=1;
else
    dataAdaptCoeff=2;
end

%% Start processing channels ========================================
for channelNr = 1:settings.numberOfChannels
    % Only process if PRN is non zero (acquisition was successful)
    if (channel(channelNr).PRN == 0)
        continue;
    end
    % Save additional information - each channel's tracked PRN
    trackResults(channelNr).PRN    = channel(channelNr).PRN;
    % Move the starting point of processing. Can be used to start the
    % signal processing at any point in the data record (e.g. for long
    % records). In addition skip through that data file to start at the
    % appropriate sample (corresponding to code phase). Assumes sample
    % type is schar (or 1 byte per sample)
    fseek(fid, dataAdaptCoeff*channel(channelNr).startSample, 'bof');
    % Get a vector with the C/A code sampled 1x/chip
    caCode0 = generateCode(channel(channelNr).PRN, settings.system);
```

```
    % Then make it possible to do early and late versions
    caCode = [caCode0(settings.codeLength) caCode0 caCode0(1)];
    %--- Perform various initializations ------------------------------
    % define initial code frequency basis of NCO
    codeFreq     = channel(channelNr).codeFreq;
    % define residual code phase (in chips)
    remCodePhase  = channel(channelNr).remCodePhase;
    % define carrier frequency which is used over whole tracking period
    carrFreq     = channel(channelNr).carrFreq;
    % define residual carrier phase
    remCarrPhase  = channel(channelNr).remCarrPhase;
    %code tracking loop parameters
    codeNco = settings.codeFreqBasis - codeFreq;    %
channel(channelNr).codeNco;
%    codeError = channel(channelNr).codeError;

    %carrier/Costas loop parameters
    carrNco = carrFreq - settings.IF;            %
channel(channelNr).carrNco;
%    carrError = channel(channelNr).carrError;

    %--- Kalman filter variables --------------------------------
    x = channel(channelNr).x;      % [ 0; 0; -carrNco*lambda_carr ];
    P = channel(channelNr).P;      % diag( ([10, 1, 100]).^2 );
    % Covariance of measurement noises
    [varCarrPhase, varCodePhase, varCarrFreq] =
varEstimate( channel(channelNr).CNo, ...
        settings.dllCorrelatorSpacing, settings.msOfPDI*0.001 );
    R = diag( [varCarrPhase, varCodePhase, varCarrFreq]*5^2 );

    %--- Phase lock detector
-------------------------------------------
    phaseLockDetector = channel(channelNr).phaseLockDetector;

    % early, late, and prompt parameters
    I_Es = 0.0;
    Q_Es = 0.0;
    I_Ps = 0.0;
    Q_Ps = 0.0;
    I_Ls = 0.0;
    Q_Ls = 0.0;
```

```
    I_Ps1 = 0.0;
    Q_Ps1 = 0.0;
    I_Ps2 = 0.0;
    Q_Ps2 = 0.0;

    % C/No computation
    vsmCnt  = 0;
    if (settings.CNo.enableVSM==1)
        CNo='Calculating...';
    else
        CNo='Disabled';
    end

    trackCnt = 1;
    pdiCnt = 1;

    %=== Process the number of specified code periods ================
    for loopCnt =  1:codePeriods

%% GUI update -------------------------------------------------------
        % The GUI is updated every 50ms. This way Matlab GUI is still
        % responsive enough. At the same time Matlab is not occupied
        % all the time with GUI task.
        Ln = sprintf('\n');
        trackingStatus=['Tracking: Ch ', int2str(channelNr), ...
            ' of ', int2str(settings.numberOfChannels),Ln ...
            'PRN: ', int2str(channel(channelNr).PRN),Ln ...
            'Completed ',int2str(loopCnt), ...
            ' of ', int2str(codePeriods), ' msec',Ln...
            'C/No: ',CNo,' (dB-Hz)'];

        if (rem(loopCnt, 50) == 0)
            try
                waitbar(loopCnt/codePeriods, ...
                    hwb, ...
                    trackingStatus);
            catch
                % The progress bar was closed. It is used as a signal
                % to stop, "cancel" processing. Exit.
                disp('Progress bar closed, exiting...');
                return
            end
```

```
        end

%% Read next block of data
------------------------------------------------
        % Find the size of a "block" or code period in whole samples

        % Update the phasestep based on code freq (variable) and
        % sampling frequency (fixed)
        codePhaseStep = codeFreq / settings.samplingFreq;

        blksize = ceil((settings.codeLength-remCodePhase) /
codePhaseStep);
        % Read in the appropriate number of samples to process this
        % interation
        [rawSignal, samplesRead] = fread(fid, ...
            dataAdaptCoeff*blksize, settings.dataType);

        rawSignal = rawSignal';    % transpose vector

        if (dataAdaptCoeff==2)
            rawSignal1 = rawSignal(1:2:end);
            rawSignal2 = rawSignal(2:2:end);
            rawSignal = rawSignal1 + 1i .* rawSignal2;  %transpose vector
        end

        % If did not read in enough samples, then could be out of
        % data - better exit
        if (samplesRead ~= dataAdaptCoeff*blksize)
            disp('Not able to read the specified number of samples  for
tracking, exiting!')
            fclose(fid);
            return
        end

%% Set up all the code phase tracking information
-------------------------
        % Define index into early code vector
        tcode      = (remCodePhase-earlyLateSpc) : codePhaseStep : ...
 ((blksize-1)*codePhaseStep+remCodePhase-earlyLateSpc);
        tcode2     = ceil(tcode) + 1;
        earlyCode   = caCode(tcode2);
        % Define index into late code vector
```

```
        tcode      = (remCodePhase+earlyLateSpc) : codePhaseStep : ...
 ((blksize-1)*codePhaseStep+remCodePhase+earlyLateSpc);
        tcode2     = ceil(tcode) + 1;
        lateCode    = caCode(tcode2);
        % Define index into prompt code vector
        tcode      = remCodePhase : codePhaseStep : ...
                     ((blksize-1)*codePhaseStep+remCodePhase);
        tcode2     = ceil(tcode) + 1;
        promptCode  = caCode(tcode2);
        remCodePhase = (tcode(blksize) + codePhaseStep) -
settings.codeLength;

%% Generate the carrier frequency to mix the signal to baseband -----------
        time    = (0:blksize) ./ settings.samplingFreq;
        % Get the argument to sin/cos functions
        trigarg = ((carrFreq * 2.0 * pi) .* time) + remCarrPhase;
        remCarrPhase = rem(trigarg(blksize+1), (2 * pi));
        % Finally compute the signal to mix the collected data to bandband
        carrsig = exp(1i .* trigarg(1:blksize));

%% Generate the six standard accumulated values ---------------------
        % First mix to baseband
        qBasebandSignal = real(carrsig .* rawSignal);
        iBasebandSignal = imag(carrsig .* rawSignal);
        % Now get early, late, and prompt values for each
        I_E = sum(earlyCode  .* iBasebandSignal);
        Q_E = sum(earlyCode  .* qBasebandSignal);
        I_P = sum(promptCode .* iBasebandSignal);
        Q_P = sum(promptCode .* qBasebandSignal);
        I_L = sum(lateCode   .* iBasebandSignal);
        Q_L = sum(lateCode   .* qBasebandSignal);
        %
        signP = sign(I_P);
        I_Es = I_Es + I_E*signP;
        Q_Es = Q_Es + Q_E*signP;
        I_Ps = I_Ps + I_P*signP;
        Q_Ps = Q_Ps + Q_P*signP;
        I_Ls = I_Ls + I_L*signP;
        Q_Ls = Q_Ls + Q_L*signP;
        % I and Q output for FLL
        if( pdiCnt <= ceil(msOfPDI/2) )
            I_Ps1 = I_Ps1 + I_P*signP;
```

```
            Q_Ps1 = Q_Ps1 + Q_P*signP;
        end
        if( pdiCnt > msOfPDI - ceil(msOfPDI/2) )
            I_Ps2 = I_Ps2 + I_P*signP;
            Q_Ps2 = Q_Ps2 + Q_P*signP;
        end
        pdiCnt = pdiCnt + 1;

%%       %=== Prefilter ==========================================
        if (rem(loopCnt, msOfPDI) == 0)
            % PLL
            carrError = atan(Q_Ps / I_Ps);
            % DLL
            codeError = (1-earlyLateSpc) * ...
                ( sqrt(I_Es * I_Es + Q_Es * Q_Es) - ...
                sqrt(I_Ls * I_Ls + Q_Ls * Q_Ls) ) / ...
                ( sqrt(I_Es * I_Es + Q_Es * Q_Es) + ...
                sqrt(I_Ls * I_Ls + Q_Ls * Q_Ls) );
            % FLL
            Dot = I_Ps1 * I_Ps2 + Q_Ps1 * Q_Ps2;
            Cross = I_Ps1 * Q_Ps2 - I_Ps2 * Q_Ps1;
            freqError = atan2(Cross, Dot) / halfPDI / (2.0 * pi);
            %% Prefilter
            u = [carrNco; codeNco];
            z = [carrError; codeError; freqError];
            [x,P] = kf(x,P,F,Q,G,u,z,H,D,R);
            carrNco = -x(3) / lambda_carr + 1*x(1)/(2.0*pi)/dTf;  %
            codeNco = x(3) / lambda_code + 1*x(2)/dTf;   %

            carrFreq = settings.IF + carrNco;
            codeFreq = settings.codeFreqBasis - codeNco;

%% Record various measures to show in postprocessing
----------------------
            % Record sample number (based on 8bit samples)
            trackResults(channelNr).dllDiscr(trackCnt)      =
codeError;
            trackResults(channelNr).dllDiscrFilt(trackCnt)   =
codeNco; % x(2);
            trackResults(channelNr).pllDiscr(trackCnt)      =
carrError;
            trackResults(channelNr).pllDiscrFilt(trackCnt)   =
```

```
carrNco; % x(1);

          trackResults(channelNr).fllDiscr(trackCnt)      =
freqError;

          % record channel information
          channel(channelNr).startSample  =
(ftell(fid))/dataAdaptCoeff;
%           channel(channelNr).codePhase    = 0;
          channel(channelNr).carrFreq    = carrFreq;
          channel(channelNr).codeFreq    = codeFreq;
          %    channel(channelNr).carrError      = carrError;
          %    channel(channelNr).carrNco        = carrNco;
          %    channel(channelNr).codeError      = codeError;
          %    channel(channelNr).codeNco        = codeNco;
          channel(channelNr).remCarrPhase    = remCarrPhase;
          channel(channelNr).remCodePhase    = remCodePhase;

          channel(channelNr).x = x;
          channel(channelNr).P = P;

          channel(channelNr).phaseLockDetector = phaseLockDetector;

          I_Es = 0.0;
          Q_Es = 0.0;
          I_Ps = 0.0;
          Q_Ps = 0.0;
          I_Ls = 0.0;
          Q_Ls = 0.0;

          I_Ps1 = 0.0;
          I_Ps2 = 0.0;
          Q_Ps1 = 0.0;
          Q_Ps2 = 0.0;

          trackCnt = trackCnt + 1;
          pdiCnt = 1;

       end % if (rem(loopCnt, msOfPDI) == 0)

       % Evaluate the tracking results status here to ensure the
       % plotTracking to plot the results tracked so far
```

```
        % (In case the tracking update window is closed)
        trackResults(channelNr).status  = channel(channelNr).status;

        trackResults(channelNr).I_E(loopCnt) = I_E;
        trackResults(channelNr).I_P(loopCnt) = I_P;
        trackResults(channelNr).I_L(loopCnt) = I_L;
        trackResults(channelNr).Q_E(loopCnt) = Q_E;
        trackResults(channelNr).Q_P(loopCnt) = Q_P;
        trackResults(channelNr).Q_L(loopCnt) = Q_L;

        trackResults(channelNr).absoluteSample(loopCnt) = ...
            (ftell(fid))/dataAdaptCoeff - remCodePhase/codePhaseStep;

        trackResults(channelNr).carrFreq(loopCnt) = carrFreq;
%         trackResults(channelNr).remCarrPhase(loopCnt) = remCarrPhase;

        trackResults(channelNr).codeFreq(loopCnt) = codeFreq;
%         trackResults(channelNr).remCodePhase(loopCnt) = remCodePhase;

%% Compute C/No
        if (settings.CNo.enableVSM == 1)
            if (rem(loopCnt,settings.CNo.VSMinterval) == 0)
                vsmCnt = vsmCnt + 1;

                CNoValue = CNoEstimator( ...

trackResults(channelNr).I_P(loopCnt-settings.CNo.VSMinterval+1:loopCn
t),...

trackResults(channelNr).Q_P(loopCnt-settings.CNo.VSMinterval+1:loopCn
t), settings.CNo.accTime );
                trackResults(channelNr).CNo(vsmCnt) = CNoValue;
                channel(channelNr).CNo = CNoValue;
                CNo = num2str(CNoValue);

                channel(channelNr).CNo = CNoValue;

                % set R
                [varCarrPhase, varCodePhase, varCarrFreq] =
varEstimate( CNoValue, ...
                    settings.dllCorrelatorSpacing,
settings.msOfPDI*0.001 );
```

```
                R = diag( [varCarrPhase, varCodePhase, varCarrFreq]*5^2 );
            end
        end

    end % for loopCnt

    % If we got so far, this means that the tracking was successful
    % Now we only copy status, but it can be update by a lock detector
    % if implemented
    %trackResults(channelNr).status  = channel(channelNr).status;

end % for channelNr

% Close the waitbar
close(hwb)
```

54) tracking_vll

```
function [trackResultsVll, navSolutionsVll, channel, vllSettings] =
tracking_vll(fid, ...
    channel, chnList, navSolutions, eph, svTimeTable, settings)
% Performs code and carrier tracking for all channels.
%
%[trackResults, channel] = tracking(fid, channel, settings)
%
%   Inputs:
%       fid             - file identifier of the signal record for I
%       channel         - PRN, carrier frequencies and code phases of all
%                       satellites to be tracked (prepared by preRum.m from
%                       acquisition results).
%       settings        - receiver settings.
%   Outputs:
%       trackResults    - tracking results (structure array). Contains
%                       in-phase prompt outputs and absolute spreading
%                       code's starting positions, together with other
%                       observation data from the tracking loops. All are
%                       saved every millisecond.
%--------------------------------------------------------------------
% Copyright (C) Dennis M. Akos
% Written by Mwenjie @2014.03
% Based on SoftGNSS v3.0
%--------------------------------------------------------------------
%% Initialize result structure ====================================
```

```
% ms to Process
msToProcess = settings.msToVll;
% Predection Integration time
timeOfPDI = 0.001*settings.msOfPDI;
% Channel PRN
trackResultsVll.PRN          = 0;
% Channel status
trackResultsVll.status       = zeros(1, msToProcess);     % No tracked
signal, or lost lock
% The absolute sample in the record of the C/A code start:
trackResultsVll.absoluteSample = zeros(1, msToProcess);
% Freq of the C/A code:
trackResultsVll.codeFreq     = inf(1, msToProcess);
% Frequency of the tracked carrier wave:
trackResultsVll.carrFreq     = inf(1, msToProcess);
% Outputs from the correlators (In-phase):
trackResultsVll.I_P          = zeros(1, msToProcess);
trackResultsVll.I_E          = zeros(1, msToProcess);
trackResultsVll.I_L          = zeros(1, msToProcess);
% Outputs from the correlators (Quadrature-phase):
trackResultsVll.Q_E          = zeros(1, msToProcess);
trackResultsVll.Q_P          = zeros(1, msToProcess);
trackResultsVll.Q_L          = zeros(1, msToProcess);
% C/No of each channel
trackResultsVll.CNo    = zeros(1,
floor(settings.msToVll/settings.navSolPeriod));

%--- Copy initial settings for all channels -------------------------
trackResultsVll = repmat(trackResultsVll, 1,
settings.numberOfChannels);

for jj = 1:settings.numberOfChannels
    trackResultsVll(jj).PRN = channel(jj).PRN;
end

%% Initialize tracking variables ==================================
lambda_code = settings.c / settings.codeFreqBasis;
lambda_carr = settings.c / settings.carrFreqBasis;
rxTime = navSolutions.rxTime(end);

%--- Kalman filter variables ----------------------------------------
dTf = 1/settings.navSolRate;    % kalman filter updating interval
```

```
x = [ navSolutions.X(end); navSolutions.Y(end); navSolutions.Z(end);
    navSolutions.Vx(end); navSolutions.Vy(end); navSolutions.Vz(end);
    0; 0 ];
P = diag( ([ones(1,3)*10, ones(1,3)*1, 10, 100]).^2 );
Q = diag( ([ones(1,3)*0.5*dTf^2, ones(1,3)*dTf, 0.5*dTf^2, dTf]*2).^2 );

%--- GUI -------------------------------------------------------------
hwb = waitbar(0,'Vector tracking...','Visible','on');

%% Start vector tracking ==========================================
for channelNr = chnList
    channel(channelNr).sumP = 0;
    channel(channelNr).bitBoundary =
rem( svTimeTable(channelNr).subFrameStart + ...
ceil( (settings.msToSll-svTimeTable(channelNr).subFrameStart)/20 )*20
- settings.msToSll, 20 ) - 1;
    if channel(channelNr).bitBoundary <= 0
        channel(channelNr).bitBoundary = channel(channelNr).bitBoundary
+ 20;
    end
end

msCnt = 1;
vllCntOffset = floor( settings.msToSll/settings.navSolPeriod );
sllCntOffset = floor( settings.msToPreSll/settings.navSolPeriod );

for vllCnt = 1:floor(settings.msToVll/settings.navSolPeriod)

%% Vector tracking ================================================
    [trackResults_temp, channel] = trackingv_ms(fid, channel, settings,
settings.navSolPeriod+1);
    channelStatus = zeros(1,settings.numberOfChannels);
    % append trackResults_temp to trackResults
    for channelNr = chnList

        if channel(channelNr).status == 'T'
            channelStatus(channelNr) = 1;
        end
trackResultsVll(channelNr).status(msCnt:msCnt+settings.navSolPeriod)
= trackResults_temp(channelNr).status;
trackResultsVll(channelNr).absoluteSample(msCnt:msCnt+settings.navSol
Period) =  trackResults_temp(channelNr).absoluteSample;
```

```
trackResultsVll(channelNr).codeFreq(msCnt:msCnt+settings.navSolPeriod)
=  trackResults_temp(channelNr).codeFreq;
trackResultsVll(channelNr).carrFreq(msCnt:msCnt+settings.navSolPeriod)
= trackResults_temp(channelNr).carrFreq;
trackResultsVll(channelNr).I_P(msCnt:msCnt+settings.navSolPeriod) =
trackResults_temp(channelNr).I_P;
trackResultsVll(channelNr).I_E(msCnt:msCnt+settings.navSolPeriod) =
trackResults_temp(channelNr).I_E;
trackResultsVll(channelNr).I_L(msCnt:msCnt+settings.navSolPeriod) =
 trackResults_temp(channelNr).I_L;
trackResultsVll(channelNr).Q_P(msCnt:msCnt+settings.navSolPeriod) =
trackResults_temp(channelNr).Q_P;
trackResultsVll(channelNr).Q_E(msCnt:msCnt+settings.navSolPeriod) =
trackResults_temp(channelNr).Q_E;
trackResultsVll(channelNr).Q_L(msCnt:msCnt+settings.navSolPeriod) =
trackResults_temp(channelNr).Q_L;
       %% Compute C/No
       if (settings.CNo.enableVSM == 1)
           channel(channelNr).CNo = CNoEstimator( ...
               trackResults_temp(channelNr).I_P, ...
               trackResults_temp(channelNr).Q_P, ...
               settings.CNo.accTime );
       end
       trackResultsVll(channelNr).CNo(vllCnt) =
channel(channelNr).CNo;
    end
    activeChnList = find(channelStatus > 0);

%% GUI update ========================================================

    Ln = sprintf('\n');
    trackingStatus = [ 'Vector Tracking ... ', Ln, ...
       'Complete ',num2str(vllCnt*settings.navSolPeriod/1000), ...
       ' of ', int2str(settings.msToVll/1000), ' sec' ];

    try
       waitbar(vllCnt*settings.navSolPeriod/settings.msToVll, ...
           hwb, trackingStatus);
    catch
       % The progress bar was closed. It is used as a signal
       % to stop, "cancel" processing. Exit.
       disp('Progress bar closed, exiting...');
```

```
        return
    end


%% Calculate the current sample number, corresponding satellites ======
%% transmitting time and raw receiver time ============================
    sampleNum =
(vllCnt+vllCntOffset+sllCntOffset)*settings.samplingFreq/settings.nav
SolRate...
        + settings.skipNumberOfSamples;
    transmitTime = findTransTime(sampleNum, chnList, svTimeTable,...
        trackResults_temp, settings,
vllCntOffset*settings.navSolPeriod+msCnt-1);

    rxTime = rxTime + 1/settings.navSolRate;

%% Find satellites positions and clocks corrections =================

    [satPos, satVel, satClkCorr] = satPosVel(transmitTime, ...
        [trackResults_temp(chnList).PRN], eph, settings.system);

%% Find pseudoranges and pseudorange rates ==========================
    pos = x(1:3);

    % Calculate tropospheric and ionospheric correction
    tropErr = zeros(1,settings.numberOfChannels);
    ionoErr = zeros(1,settings.numberOfChannels);
    for jj = chnList
        %--- Find the elevation angel of the satellite ----------------
        [~, el, ~] = topocent(pos, satPos(:,jj) - pos);

        %--- Calculate tropospheric correction --------------------
        tropErr(jj) = 2.47 / (sin(el * pi/180) + 0.0121);

        %--- Calculate ionospheric correction --------------------

    end

    rawP = calculatePseudoranges( ...
        transmitTime, rxTime, activeChnList, settings) + ...
        satClkCorr * settings.c - tropErr - ionoErr;
```

```
    for jj = activeChnList
        % add code errors to pseudoranges
        rawP(jj) = rawP(jj) + lambda_code * channel(jj).x(2);
    end

    rawF = zeros(1,settings.numberOfChannels);
    for jj = activeChnList
        rawF(jj) = channel(jj).x(3);
    end

    % correct satPos error due to the earth rotation
    for jj = chnList
        satPos(:,jj) = e_r_corr( norm(satPos(:,jj)-pos)/settings.c,
satPos(:,jj) );
    end

%% Find receiver position and veloctiy ============================
    fkfData.flag = zeros(1,settings.numberOfChannels);
    fkfData.flag(activeChnList) = 1;
    fkfData.r_sat = satPos;
    fkfData.v_sat = satVel;
    fkfData.z = [rawP; rawF];
    fkfData.R = ones(2,2,settings.numberOfChannels);
    for jj = activeChnList
%       fkfData.R(:,:,jj) = channel(jj).P(2:3,2:3);
%       fkfData.R(:,:,jj) = diag( [2, 0.1].^2 );
        fkfData.R(:,:,jj) = diag( [channel(jj).P(2,2)*lambda_code^2 +
2^2, ...
            channel(jj).P(3,3)] );
    end

    [x,P,nFilters] = fkf(x,P,Q,fkfData,dTf);

%% Correct
    if nFilters > 0
        rxTime = rxTime - x(7)/settings.c;
        x(7) = 0;
    end

    % vector tracking feedback
    for jj = chnList
```

```
        rho = x(1:3) - satPos(:,jj);
        correctedP = norm(rho);

        los = rho / correctedP;
        correctedF = dot( x(4:6) - satVel(:,jj), los ) + x(8);

%         if channel(jj).status == 'T'
           p2 = P(4,4) + P(5,5) + P(6,6);
           p1 = channel(jj).P(3,3);
           channel(jj).x(3) = (channel(jj).x(3)/p1 +
correctedF/p2)/(1/p1+1/p2);

           channel(jj).P = (channel(jj).P + channel(jj).P')/2;

           carrNco = -channel(jj).x(3) / lambda_carr +
channel(jj).x(1)/(2.0*pi)/timeOfPDI;
           codeNco = channel(jj).x(3) / lambda_code +
channel(jj).x(2)/timeOfPDI;

           channel(jj).carrFreq = settings.IF + carrNco;
           channel(jj).codeFreq = settings.codeFreqBasis - codeNco;

           % pseudorange
           txTime = rxTime - (correctedP + tropErr(jj) + ionoErr(jj)) / ...
               settings.c + satClkCorr(jj);
           startSample = sampleNum + ...
               settings.codeLength * (1 - rem(txTime*1e3,1)) * ...
               settings.samplingFreq/channel(jj).codeFreq;
           p2 = P(1,1) + P(2,2) + P(3,3);
           p1 = channel(jj).P(2,2);
           channel(jj).startSample = round(channel(jj).startSample + ...
               p1/(p1+p2)*(startSample-channel(jj).startSample));

%         else
% %           channel(jj).x(1) = 0;
% %           channel(jj).x(2) = 0;
%           channel(jj).x(3) = correctedF;
% %           channel(jj).P = diag( ([10, 1, 100]).^2 );
% %           channel(jj).CNo = settings.initCNo;
%
%           carrNco = -channel(jj).x(3) / lambda_carr;
%           codeNco = channel(jj).x(3) / lambda_code;
```

```
%
%           channel(jj).carrFreq = settings.IF + carrNco;
%           channel(jj).codeFreq = settings.codeFreqBasis - codeNco;
%
% %           channel(jj).remCarrPhase = 0.0;
% %           channel(jj).remCodePhase = 0.0;
%
%           % pseudorange
%           txTime = rxTime - (correctedP + tropErr(jj) + ionoErr(jj))
/ ...
%               settings.c + satClkCorr(jj);
%           channel(jj).startSample = sampleNum + ...
%               round( settings.codeLength * (1 - rem(txTime*1e3,1)) * ...
%               settings.samplingFreq/channel(jj).codeFreq );
%       end

    end

%% Record
    navSolutionsVll.rxTime(vllCnt) = rxTime;

    navSolutionsVll.X(vllCnt)  = x(1);
    navSolutionsVll.Y(vllCnt)  = x(2);
    navSolutionsVll.Z(vllCnt)  = x(3);
    navSolutionsVll.dt(vllCnt) = x(7);

    navSolutionsVll.Vx(vllCnt)  = x(4);
    navSolutionsVll.Vy(vllCnt)  = x(5);
    navSolutionsVll.Vz(vllCnt)  = x(6);
    navSolutionsVll.ddt(vllCnt) = x(8);

    %=== Convert to geodetic coordinates ==============================
    [navSolutionsVll.latitude(vllCnt),
navSolutionsVll.longitude(vllCnt), ...
        navSolutionsVll.height(vllCnt)] = ...
        cart2geo( navSolutionsVll.X(vllCnt), ...
        navSolutionsVll.Y(vllCnt), ...
        navSolutionsVll.Z(vllCnt), ...
        5);

    %=== Convert to UTM coordinate system =============================
    navSolutionsVll.utmZone =
```

```
findUtmZone(navSolutionsVll.latitude(vllCnt), ...
       navSolutionsVll.longitude(vllCnt));

   [navSolutionsVll.E(vllCnt),  navSolutionsVll.N(vllCnt),
navSolutionsVll.U(vllCnt)] = ...
       cart2utm(x(1), x(2), x(3), navSolutionsVll.utmZone);

%% do something else

   msCnt = msCnt + settings.navSolPeriod;

end  % for vllCnt

vllSettings.x = x;
vllSettings.P = P;
vllSettings.rxTime = rxTime;
vllSettings.msOffset = settings.msToSll + settings.msToVll;

% Close the waitbar
close(hwb)
```

55) tracking_utc

```
function [trackResultsUtc, navSolutionsUtc, channel, utcSettings] =
tracking_utc(fid, ...
   channel, chnList, navSolutions, eph, svTimeTable, settings)
% Performs code and carrier tracking for all channels.
%
%[trackResults, channel] = tracking(fid, channel, settings)
%
%  Inputs:
%      fid           - file identifier of the signal record for I
%      channel        - PRN, carrier frequencies and code phases of all
%                    satellites to be tracked (prepared by preRum.m from
%                    acquisition results).
%      settings       - receiver settings.
%  Outputs:
%      trackResults    - tracking results (structure array). Contains
%                    in-phase prompt outputs and absolute spreading
%                    code's starting positions, together with other
%                    observation data from the tracking loops. All are
%                    saved every millisecond.
%--------------------------------------------------------------------
```

```
% Copyright (C) Dennis M. Akos
% Written by Mwenjie @2014.03
% Based on SoftGNSS v3.0
%--------------------------------------------------------------------
%% Initialize result structure ========================================

% ms to Process
msToProcess = settings.msToVll;
% Predection Integration time
timeOfPDI = 0.001*settings.msOfPDI;

% Channel PRN
trackResultsUtc.PRN          = 0;

% Channel status
trackResultsUtc.status       = zeros(1, msToProcess);     % No tracked
signal, or lost lock

% The absolute sample in the record of the C/A code start:
trackResultsUtc.absoluteSample = zeros(1, msToProcess);

% Freq of the C/A code:
trackResultsUtc.codeFreq     = inf(1, msToProcess);

% Frequency of the tracked carrier wave:
trackResultsUtc.carrFreq     = inf(1, msToProcess);

% Outputs from the correlators (In-phase):
trackResultsUtc.I_P          = zeros(1, msToProcess);
trackResultsUtc.I_E          = zeros(1, msToProcess);
trackResultsUtc.I_L          = zeros(1, msToProcess);

% Outputs from the correlators (Quadrature-phase):
trackResultsUtc.Q_E          = zeros(1, msToProcess);
trackResultsUtc.Q_P          = zeros(1, msToProcess);
trackResultsUtc.Q_L          = zeros(1, msToProcess);

% C/No of each channel
trackResultsUtc.CNo    = zeros(1,
floor(settings.msToVll/settings.navSolPeriod));

%--- Copy initial settings for all channels -------------------------
```

```
trackResultsUtc = repmat(trackResultsUtc, 1,
settings.numberOfChannels);

for jj = 1:settings.numberOfChannels
    trackResultsUtc(jj).PRN = channel(jj).PRN;
end

%% Initialize tracking variables ====================================

lambda_code = settings.c / settings.codeFreqBasis;
lambda_carr = settings.c / settings.carrFreqBasis;

rxTime = navSolutions.rxTime(end);

%--- Kalman filter variables -----------------------------------------
dTf = 1/settings.navSolRate;    % kalman filter updating interval

% x = [ navSolutions.X(end); navSolutions.Y(end); navSolutions.Z(end);
%     navSolutions.Vx(end); navSolutions.Vy(end); navSolutions.Vz(end);
%     0; 0 ];
x = zeros(8,1);

P = diag( ([ones(1,3)*10, ones(1,3)*1, 10, 100]).^2 );

Q = diag( ([ones(1,3)*0.5*dTf^2, ones(1,3)*dTf, 0.5*dTf^2, dTf]*2).^2 );

%% Initialize Dead Reckoning variables ============================

%% Initialize AHRS kalman filter
dT = 1/100;
F_ahrs = eye(3);
G_ahrs = eye(3)*dT;
H_ahrs = eye(3);

% Q = diag( ([dt, dt, dt]*1e-2).^2 );
Q_ahrs = diag( ([dT, dT, dT]*1e-1).^2 );
% R_ahrs = diag( [0.1, 0.1, 0.2].^2 );

x_ahrs = zeros(3,1);
P_ahrs = diag( [pi/2, pi/2, pi/2].^2 );

%% Initialize postion, velocity and attitude
```

```
global acc mag gyro speed acceleration

drLen = floor(dTf/dT);

pos = zeros(3,drLen);
vel = zeros(3,drLen);
att = zeros(3,drLen);

C_n2e = posMat(navSolutions.latitude(end)*pi/180, ...
    navSolutions.longitude(end)*pi/180, 0);
hgt = navSolutions.height(end);
vel_1 = zeros(3,1);
vel_2 = zeros(3,1);

imuCnt = 50.7e2;
navCnt = 1;

% Initialize attitude
dAzimuth = 3*pi/180;
C_b2n = attFromAccMag(mean(acc(:,imuCnt+1:imuCnt+500),2), ...
    mean(mag(:,imuCnt+1:imuCnt+500),2), dAzimuth);

%--- GUI
-------------------------------------------------------------------
hwb = waitbar(0,'Vector tracking...','Visible','on');

%% Start vector tracking ========================================

msCnt = 1;
vllCntOffset = floor( settings.msToSll/settings.navSolPeriod );
sllCntOffset = floor( settings.msToPreSll/settings.navSolPeriod );

for vllCnt = 1:floor(settings.msToVll/settings.navSolPeriod)

%% Dead Reckoning

for jj = 1:drLen
    u_ahrs = gyro(:,imuCnt)*pi/180;
%     C_b2n_est = attFromAccMag(acc(:,imuCnt),mag(:,imuCnt));
    grav_b = acc(:,imuCnt);
    grav_b(1) = grav_b(1)+speed(imuCnt)*u_ahrs(3);
    grav_b(2) = grav_b(2) - acceleration(imuCnt);
```

```
%    grav_b(3) = sqrt(9.7934^2-grav_b(1).^2-grav_b(2).^2);
    C_b2n_est = attFromAccMag(grav_b,mag(:,imuCnt),dAzimuth);
    dAtt_cross = C_b2n'*C_b2n_est - eye(3);
    z = [ dAtt_cross(3,2); dAtt_cross(1,3); dAtt_cross(2,1) ];

    % Adaptive
    dAcc = abs(norm(acc(:,imuCnt)) - 9.7934);
    if dAcc < 0.1
        dAcc = 0.1;
    end
    dMag = abs(norm(mag(:,imuCnt)) - 0.45);
    if dMag < 0.2
        dMag = 0.2;
    elseif dMag > 1.0
        dMag = 100.0;
    end
%    dAcc = 0.1;
%    dMag = 0.2;
    R_ahrs = diag( [dAcc, dAcc, 0.2+10*(dMag-0.2)].^2 );

    % AHRS Kalman Filter
    [x_ahrs,P_ahrs] =
kf_ahrs(x_ahrs,P_ahrs,F_ahrs,Q_ahrs,G_ahrs,u_ahrs,z,H_ahrs,R_ahrs);

    % Îó²îÐÞÕý
    dAtt = x_ahrs(1:3);
    x_ahrs(1:3) = zeros(3,1);
    dAtt_cross = crossForm(dAtt);
    C_b2n = C_b2n*(eye(3)+dAtt_cross); % +1/2*dAtt_cross*dAtt_cross

    % ¼ÆËã×ËÌ¬½Ç
    [att(1,jj), att(2,jj), att(3,jj)] = attCalc(C_b2n);

    % velocity
    vel(:,jj) = C_b2n*[0;speed(imuCnt);0];

    % position
    dX = (vel_2 + 4*vel_1 + vel(:,jj))/6*dT;
    vel_2 = vel_1;
    vel_1 = vel(:,jj);
    F_curv = curvMat(C_n2e,hgt,1);
    zeta = F_curv*dX;
```

```
    zeta_cross = crossForm(zeta);
    C_n2e = C_n2e*(eye(3)+zeta_cross);
    hgt = hgt + dX(3);
    [pos(1,jj),pos(2,jj),~] = posCalc(C_n2e);
    pos(3,jj) = hgt;

    imuCnt = imuCnt + 1;
end
C_b2n = ( 1.5*eye(3) - 0.5*C_b2n*(C_b2n') )*C_b2n;

%% Vector tracking
============================================================

    [trackResults_temp, channel] = trackingv_ms(fid, channel, settings,
settings.navSolPeriod+1);

    channelStatus = zeros(1,settings.numberOfChannels);

    % append trackResults_temp to trackResults
    for channelNr = chnList

        if channel(channelNr).status == 'T'
            channelStatus(channelNr) = 1;
        end


trackResultsUtc(channelNr).status(msCnt:msCnt+settings.navSolPeriod)
= trackResults_temp(channelNr).status;
trackResultsUtc(channelNr).absoluteSample(msCnt:msCnt+settings.navSol
Period) = trackResults_temp(channelNr).absoluteSample;
trackResultsUtc(channelNr).codeFreq(msCnt:msCnt+settings.navSolPeriod)
= trackResults_temp(channelNr).codeFreq;
trackResultsUtc(channelNr).carrFreq(msCnt:msCnt+settings.navSolPeriod)
= trackResults_temp(channelNr).carrFreq;
trackResultsUtc(channelNr).I_P(msCnt:msCnt+settings.navSolPeriod) =
trackResults_temp(channelNr).I_P;
trackResultsUtc(channelNr).I_E(msCnt:msCnt+settings.navSolPeriod) =
trackResults_temp(channelNr).I_E;
trackResultsUtc(channelNr).I_L(msCnt:msCnt+settings.navSolPeriod) =
trackResults_temp(channelNr).I_L;
trackResultsUtc(channelNr).Q_P(msCnt:msCnt+settings.navSolPeriod) =
trackResults_temp(channelNr).Q_P;
```

```
trackResultsUtc(channelNr).Q_E(msCnt:msCnt+settings.navSolPeriod) =
trackResults_temp(channelNr).Q_E;
trackResultsUtc(channelNr).Q_L(msCnt:msCnt+settings.navSolPeriod) =
trackResults_temp(channelNr).Q_L;

       %% Compute C/No
       if (settings.CNo.enableVSM == 1)
           channel(channelNr).CNo = CNoEstimator( ...
              trackResults_temp(channelNr).I_P, ...
              trackResults_temp(channelNr).Q_P, ...
              settings.CNo.accTime );
       end

       trackResultsUtc(channelNr).CNo(vllCnt) =
channel(channelNr).CNo;

   end

   activeChnList = find(channelStatus > 0);

%% GUI update
=========================================================================

   Ln = sprintf('\n');
   trackingStatus = [ 'Vector Tracking ... ', Ln, ...
       'Complete ',num2str(vllCnt*settings.navSolPeriod/1000), ...
       ' of ', int2str(settings.msToVll/1000), ' sec' ];

   try
       waitbar(vllCnt*settings.navSolPeriod/settings.msToVll, ...
           hwb, trackingStatus);
   catch
       % The progress bar was closed. It is used as a signal
       % to stop, "cancel" processing. Exit.
       disp('Progress bar closed, exiting...');
       return
   end


%% Calculate the current sample number, corresponding satellites ======
%% transmitting time and raw receiver time ============================
   sampleNum =
```

```
(vllCnt+vllCntOffset+sllCntOffset)*settings.samplingFreq/settings.nav
SolRate...
       + settings.skipNumberOfSamples;
    transmitTime = findTransTime(sampleNum, chnList, svTimeTable,...
       trackResults_temp, settings,
vllCntOffset*settings.navSolPeriod+msCnt-1);

    rxTime = rxTime + 1/settings.navSolRate;

%% Find satellites positions and clocks corrections ================
    [satPos, satVel, satClkCorr] = satPosVel(transmitTime, ...
       [trackResults_temp(chnList).PRN],eph, settings.system);

%% Find pseudoranges and pseudorange rates ========================
    posE_temp = llh2ecef(C_n2e, hgt);

    % Calculate tropospheric and ionospheric correction
    tropErr = zeros(1,settings.numberOfChannels);
    ionoErr = zeros(1,settings.numberOfChannels);
    for jj = chnList
       %--- Find the elevation angel of the satellite ----------------
       [az, el, ~] = topocent(posE_temp, satPos(:,jj) - posE_temp);

       %--- Calculate tropospheric correction --------------------
       tropErr(jj) = 2.47 / (sin(el * pi/180) + 0.0121);

       %--- Calculate ionospheric correction --------------------
       if settings.system == 2    % Beidou
          [ lat, lon, ~ ] = posCalc( C_n2e );
          ionoErr(jj) = ionocBD(lat, lon, el, az, transmitTime(jj), ...
             eph(trackResults_temp(jj).PRN).alpha,
eph(trackResults_temp(jj).PRN).beta);
       end
    end

    rawP = calculatePseudoranges( ...
       transmitTime, rxTime, activeChnList, settings) + ...
       satClkCorr * settings.c - tropErr - ionoErr;

    for jj = activeChnList
       % add code errors to pseudoranges
       rawP(jj) = rawP(jj) + lambda_code * channel(jj).x(2);
```

```
    end

    rawF = zeros(1,settings.numberOfChannels);
    for jj = activeChnList
        rawF(jj) = channel(jj).x(3);
    end

    % correct satPos error due to the earth rotation
    for jj = chnList
        satPos(:,jj) = e_r_corr( norm(satPos(:,jj)-posE_temp)/settings.c,
satPos(:,jj) );
    end

%% Find receiver position and velocity
=======================================

    % Navigation Filter
    fkfData.flag = zeros(1,settings.numberOfChannels);
    fkfData.flag(activeChnList) = 1;
    fkfData.r_sat = satPos;
    fkfData.v_sat = satVel;
    fkfData.z = [rawP; rawF];
    fkfData.R = ones(2,2,settings.numberOfChannels);
    for jj = activeChnList
%       fkfData.R(:,:,jj) = channel(jj).P(2:3,2:3);
%       fkfData.R(:,:,jj) = diag( [2, 0.1].^2 );
        fkfData.R(:,:,jj) = diag( [channel(jj).P(2,2)*lambda_code^2 +
2^2, ...
          channel(jj).P(3,3)] );
    end

    sinsBak.C_n2e = C_n2e;
    sinsBak.h_n = hgt;
    sinsBak.v_n = vel_1;
    sinsBak.C_b2n = C_b2n;
%    sinsBak.gyro = gyro(:,ii);
%    sinsBak.acc = acc(:,ii);

    [x,P,nFilters] = fukf_utc(x,P,Q,fkfData,sinsBak,dTf);


%% Correct
```

```
    if nFilters > 0
        rxTime = rxTime - x(7)/settings.c;
        x(7) = 0;

        % Dead Reckoning feedback
        Rx = 1/(F_curv(2,1)/C_n2e(3,2));
        Ry = -1/F_curv(1,2);
        %--- pos correction ----------------------------------------------
        dPos = [x(2)/Ry; x(1)/Rx; x(3)];
        dP_n = [-dPos(1); dPos(2)*C_n2e(3,2); dPos(2)*C_n2e(3,3)];
        dP_n_cross = [ 0, -dP_n(3), dP_n(2);
            dP_n(3), 0, -dP_n(1);
            -dP_n(2), dP_n(1), 0 ];
        C_n2e = C_n2e*(eye(3)-dP_n_cross);
        % Õý»¯
        C_n2e = ( 1.5*eye(3) - 0.5*(C_n2e*C_n2e') )*C_n2e;
        hgt = hgt - dPos(3);
        %--- vel correction ----------------------------------------------
        vel_1 = vel_1 - x(4:6);
        vel_2 = vel_2 - x(4:6);

        x(1:6) = 0;

    end

    % vector tracking feedback
    posE_temp = llh2ecef(C_n2e, hgt);
    velE_temp = C_n2e*vel_1;

    for jj = chnList

        rho = posE_temp - satPos(:,jj);
        correctedP = norm(rho);

        los = rho / correctedP;
        correctedF = dot( velE_temp - satVel(:,jj), los ) + x(8);

        if channel(jj).status == 'T'
            p2 = P(4,4) + P(5,5) + P(6,6);
            p1 = channel(jj).P(3,3);
            channel(jj).x(3) = (channel(jj).x(3)/p1 +
correctedF/p2)/(1/p1+1/p2);
```

```
            channel(jj).P = (channel(jj).P + channel(jj).P')/2;

            carrNco = -channel(jj).x(3) / lambda_carr +
channel(jj).x(1)/(2.0*pi)/timeOfPDI;
            codeNco = channel(jj).x(3) / lambda_code +
channel(jj).x(2)/timeOfPDI;

            channel(jj).carrFreq = settings.IF + carrNco;
            channel(jj).codeFreq = settings.codeFreqBasis - codeNco;

            % pseudorange
            txTime = rxTime - (correctedP + tropErr(jj) + ionoErr(jj)) / ...
                settings.c + satClkCorr(jj);
            startSample = sampleNum + ...
                settings.codeLength * (1 - rem(txTime*1e3,1)) * ...
                settings.samplingFreq/channel(jj).codeFreq;
            p2 = P(1,1) + P(2,2) + P(3,3);
            p1 = channel(jj).P(2,2);
            channel(jj).startSample = round(channel(jj).startSample + ...
                p1/(p1+p2)*(startSample-channel(jj).startSample));

        else
%           channel(jj).x(1) = 0;
            channel(jj).x(2) = 0;
            channel(jj).x(3) = correctedF;
%           channel(jj).P = diag( ([10, 1, 100]).^2 );
%           channel(jj).CNo = settings.initCNo;

            carrNco = -channel(jj).x(3) / lambda_carr;
            codeNco = channel(jj).x(3) / lambda_code;

            channel(jj).carrFreq = settings.IF + carrNco;
            channel(jj).codeFreq = settings.codeFreqBasis - codeNco;

%           channel(jj).remCarrPhase = 0.0;
            channel(jj).remCodePhase = 0.0;

            % pseudorange
            txTime = rxTime - (correctedP + tropErr(jj) + ionoErr(jj)) / ...
                settings.c + satClkCorr(jj);
            channel(jj).startSample = sampleNum + ...
```

```
                round( settings.codeLength * (1 - rem(txTime*1e3,1)) * ...
                settings.samplingFreq/channel(jj).codeFreq );
        end

    end

%% Record

    navSolutionsUtc.pos(:,navCnt:navCnt+drLen-1) = pos;
    navSolutionsUtc.vel(:,navCnt:navCnt+drLen-1) = vel;
    navSolutionsUtc.att(:,navCnt:navCnt+drLen-1) = att;

    navCnt = navCnt + drLen;

%    navSolutionsUtc.rxTime(vllCnt) = rxTime;
%
%    navSolutionsUtc.X(vllCnt)  = x(1);
%    navSolutionsUtc.Y(vllCnt)  = x(2);
%    navSolutionsUtc.Z(vllCnt)  = x(3);
%    navSolutionsUtc.dt(vllCnt) = x(7);
%
%    navSolutionsUtc.Vx(vllCnt)  = x(4);
%    navSolutionsUtc.Vy(vllCnt)  = x(5);
%    navSolutionsUtc.Vz(vllCnt)  = x(6);
%    navSolutionsUtc.ddt(vllCnt) = x(8);
%
%    %=== Convert to geodetic coordinates =========================
%    [navSolutionsUtc.latitude(vllCnt),
navSolutionsUtc.longitude(vllCnt), ...
%        navSolutionsUtc.height(vllCnt)] = ...
%        cart2geo( navSolutionsUtc.X(vllCnt), ...
%        navSolutionsUtc.Y(vllCnt), ...
%        navSolutionsUtc.Z(vllCnt), ...
%        5);
%
%    %=== Convert to UTM coordinate system ================================
%    navSolutionsUtc.utmZone =
findUtmZone(navSolutionsUtc.latitude(vllCnt), ...
%        navSolutionsUtc.longitude(vllCnt));
%
%    [navSolutionsUtc.E(vllCnt),  navSolutionsUtc.N(vllCnt),
navSolutionsUtc.U(vllCnt)] = ...
```

```
%       cart2utm(x(1), x(2), x(3), navSolutionsUtc.utmZone);

%% do something else

    msCnt = msCnt + settings.navSolPeriod;

end  % for vllCnt

utcSettings.x = x;
utcSettings.P = P;
utcSettings.rxTime = rxTime;
utcSettings.msOffset = settings.msToSll + settings.msToVll;

utcSettings.imuCnt = imuCnt;
utcSettings.x_ahrs = x_ahrs;
utcSettings.P_ahrs = P_ahrs;
utcSettings.C_n2e = C_n2e;
utcSettings.hgt = hgt;
utcSettings.vel_1 = vel_1;
utcSettings.vel_2 = vel_2;
utcSettings.C_b2n = C_b2n;

% Close the waitbar
close(hwb)
```

56) varEstimate

```
function [varCarrPhase, varCodePhase, varCarrFreq] = varEstimate(CNo, d,
T)
% 根据通道C/No估算PLL、DLL、FLL等鉴相器的输出噪声方差
k = 10^(CNo/10);

if CNo > 30
    F = 1;
else
    F = 2;
end

% PLL output variance
varCarrPhase = 1/(k*T)*(1+1/(2*k*T));

% DLL output variance
varCodePhase = d/(k*T)*(1+1/(k*T*(1-d)));
```

```
% FLL output variance
varCarrFreq = F/(pi^2*(T/2)^3*k)*(1+1/(k*T/2));
```