**保密等级： 公开**

**文档编号：**

**发放范围：**

**MobileGNSS-SPP**

**技术文档**

**文档版权声明**

本文档由Winchell编写，并享有版权。

任何人或组织不得违反「版权法」，在未经同意的情况下，以任何形式（包括但不限于电子版、印刷版、微缩版、复印、录制等）复制本文件、将其储存于可读取的系统或发送出去。

本文件中出现的产品或公司名称是其各自拥有者的商标或注册商标。

**目录**

[1. 摘要 4](#_Toc200586245)

[2. 数据分析 5](#_Toc200586246)

[2.1 数据采集 5](#_Toc200586247)

[2.2 C/N0（载波噪声密度比） 6](#_Toc200586248)

[2.3 观测间的相关性 6](#_Toc200586249)

[2.4 伪距和多普勒之间的相关性 8](#_Toc200586250)

[2.5 TDCP（历元间差分载波相位） 9](#_Toc200586251)

[3. RWLS 11](#_Toc200586252)

[3.1 SNR加权模型 11](#_Toc200586253)

[3.2 速度估计中的加权 12](#_Toc200586254)

[3.3 M估计（稳健估计） 13](#_Toc200586255)

[4. EKF 18](#_Toc200586256)

[4.1 xxx 18](#_Toc200586257)

[4.1.1 xxx 18](#_Toc200586258)

[5. 代价最小化 19](#_Toc200586259)

[5.1 xxx 19](#_Toc200586260)

[5.1.1 xxx 19](#_Toc200586261)

[附录 20](#_Toc200586262)

[A.1 C/N0、SNR与CNR 20](#_Toc200586263)

[A.2 MGEX-AGNSS服务 21](#_Toc200586264)

[参考资料 22](#_Toc200586265)

# 1. 摘要

待完善。

# 2. 数据分析

算法调试前的数据分析是一项很容易被忽视的工作，可能更多人为了省事喜欢直接将算法先跑起来，然后再进行一系列修改。我认为对需要优化的对象保持一个宏观的认识总归是不会错的。

MobileGNSS-SPP主要针对客户提供的某一款GNSS芯片（其型号暂时无从得知）进行了优化，该芯片数据是L1单频的，不过它提供了丰富的星座数据（GPS、BDS、GLO、GAL、QZS）。本章节还会对其中一些具体的数据指标进行分析。

## 2.1 数据采集

|  |  |
| --- | --- |
|  |  |
| (a) 测试平台实景图 | (b) 平台连接示意图 |

图2-1 测试平台示意图

测试平台如上所示，测试设备为目标安卓手机、和芯星通UM982（RTK），测试软件为GEO++、GNSSLogger。数据分析和后续算法的提升都是基于上述平台所录制的数据进行的。

经过实测发现GEO++录制的数据更完整，GNSSLogger录制的数据中某些卫星经常会缺失伪距观测，不过GEO++偶尔也会异常，导致中间记录中断，并缺失较长一段时间的数据。整体上原始数据还是以GEO++为主，异常的时候就多录制几组，而GNSSLogger用来记录结果数据（NMEA）。

至于星历数据的采集方式参考附录A.2。

## 2.2 C/N0（载波噪声密度比）

C/N0是GNSS领域中最标准的信号质量指标。但是由于叫法上的习惯，大家也会称作SNR（信噪比），其具体区别参考附录A.1。

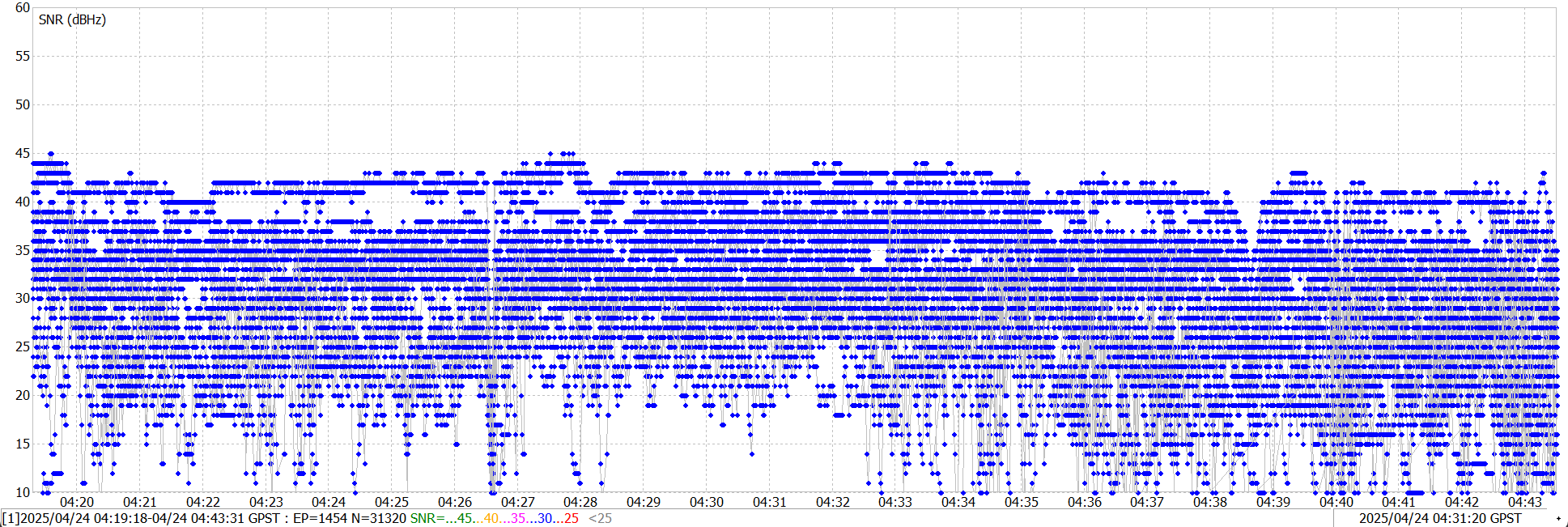
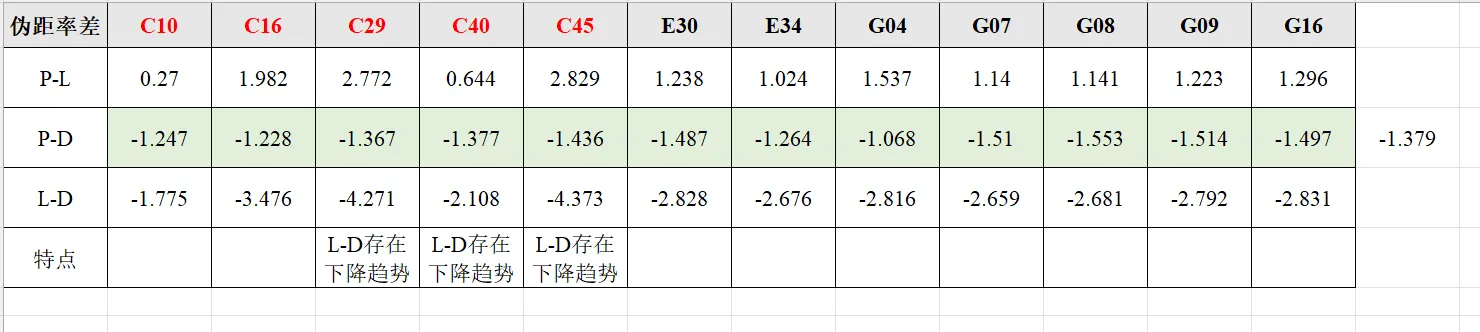


图2-2 手机C/N0的数值水平

根据一组开阔数据的表现来看，C/N0一般在45左右，相应的RTKLIB中的**stats-snrmax**配置需要设置为45，配置文件位于**MobileGNSS-SPP\app\rnx2rtkp\conf**。

stats-snrmax       =45         # (dB.Hz)

## 2.3 观测间的相关性

图2-3 原始观测数据间的相关性  
其中的P-L表示来自伪距的伪距率与来自载波的伪距率之差的平均值，其他类似

参考GSDC 2023-2024第二名Suzuki的分析[01]，可以通过计算来自不同物理量的伪距率，并两两相减，从而判断物理量之间是否相关（差值为0则相关，不为0则不相关）。

通俗来理解，伪距率即代表了卫星与接收机间的相对速度，这个相对速度除了通过多普勒计算得到，也可以通过伪距或载波在单位时间上的距离变化来得到。

通过对伪距率的研究发现：

* 该芯片物理量中的观测量两两不匹配；
* 发现伪距和多普勒之间的关系比较稳定（平均在-1.379左右），不过暂时没有发现其背后的含义；
* 北斗的载波测量非常不稳定，系统误差比较大。

至于判断是否相关之后又能做什么，这里参考Suzuki的做法。

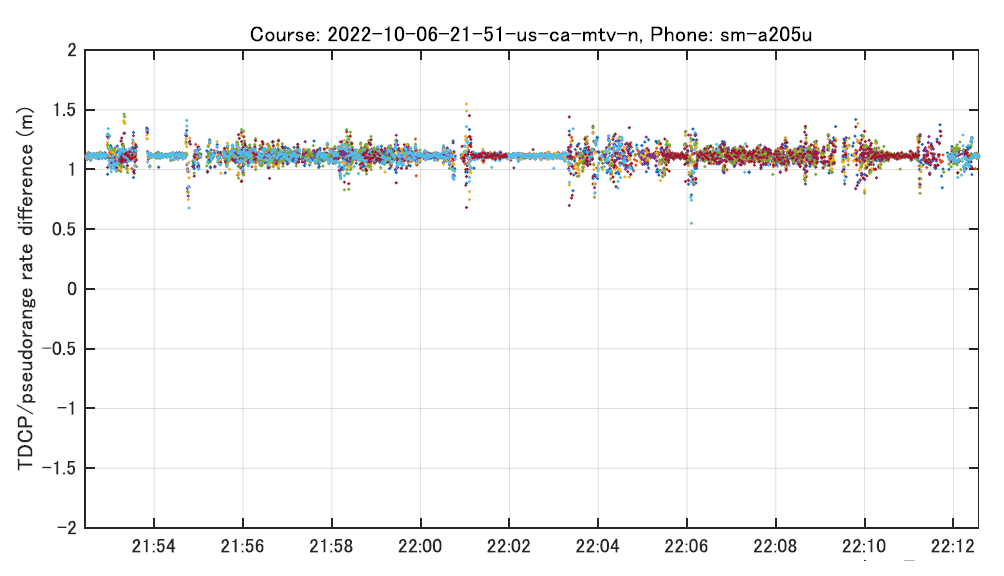


图2-4 TDCP与伪距率之间的差异（同图2-3中的L-D）

图2-4为Suzuki计算的TDCP与伪距率之间的差值（L-D）。通常，这种差异应为零。然而，如图2-5所示，某些智能手机的TDCP中包含大约1.1米的固定偏差误差。对于这些智能手机，预先校正了偏差误差，并将其用作TDCP因子。

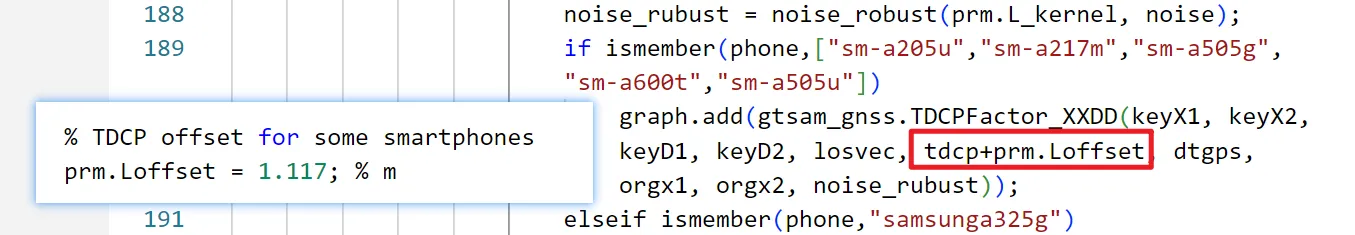


图2-5 TDCP系统误差补偿

通过查看Suzuki的代码[02]，发现他直接将固定偏差补偿到TDCP中了，并认为多普勒数据是正确的。不过由于我们的数据中物理量之间两两不相关，所以并不知道哪个物理量是完全正确的。

## 2.4 伪距和多普勒之间的相关性

根据Suzuki论文中的介绍，伪距和多普勒相关性的比较主要还是看：利用伪距钟差算出的钟漂和利用多普勒算出的钟漂之间的相关性。不过我认为此前在2.3节中的分析也能判断出伪距与多普勒之间的相关性，图2-3中的P-D显示两者存在一个相对稳定的偏差（平均在-1.379左右）。

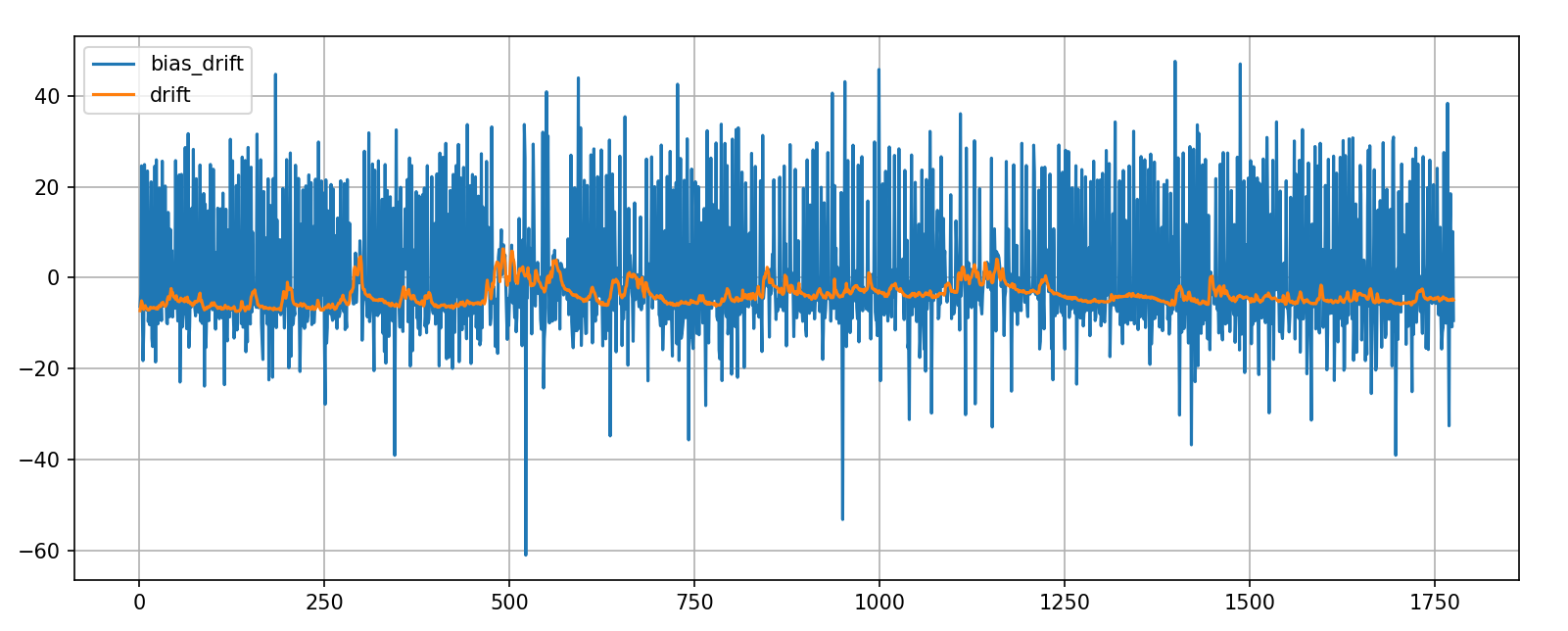


图2-6(a) 伪距和多普勒之间的相关性（整体）

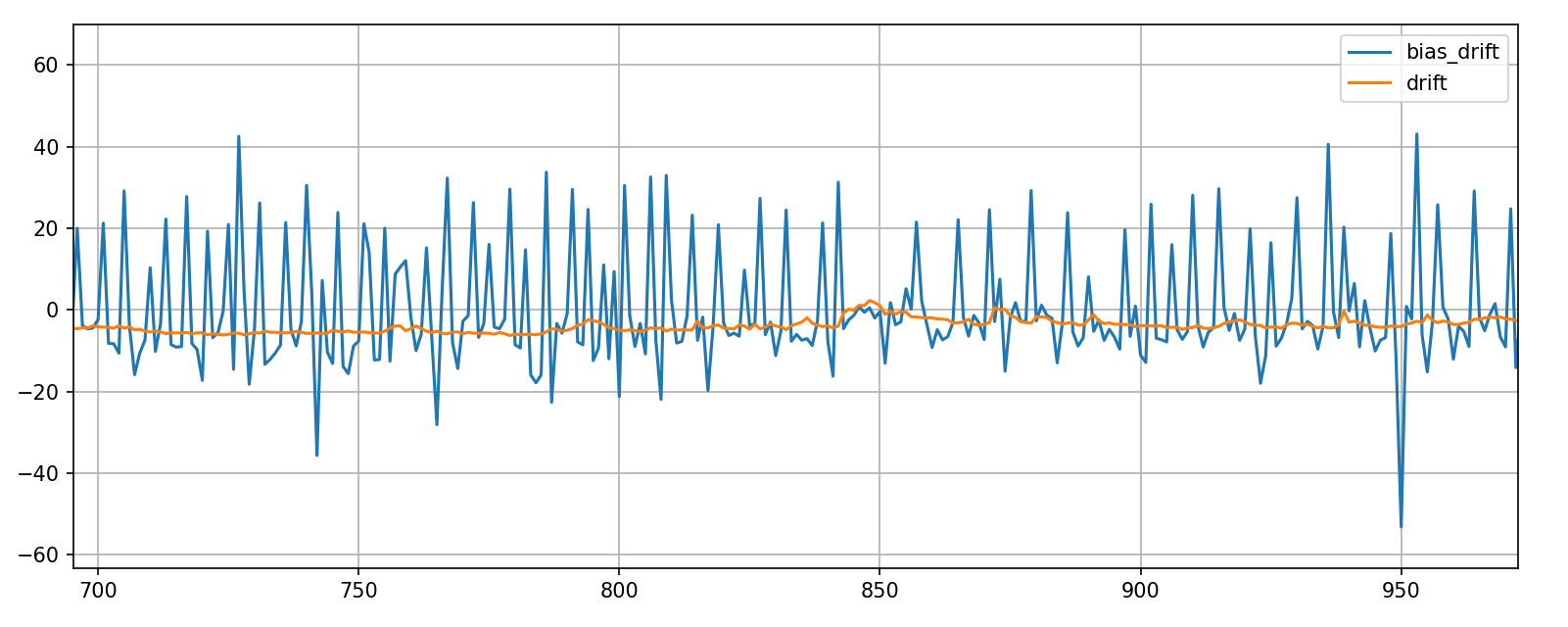


图2-6(b) 伪距和多普勒之间的相关性（局部）

图2-6 伪距和多普勒之间的相关性

按照Suzuki的方法，从钟漂的角度来看伪距与多普勒之间的相关性。从2-6(a)发现，两种钟漂似乎中心并没有在一起；不过由2-6(b)发现，钟差里面吸纳了很多不规则的系统误差，它可能间隔一些历元出现，导致图中出现了很多毛刺。如果去掉这些毛刺的话，似乎两者的中心线还是比较接近的。

由于我相信2.3节中P-D的结论，因此我暂且认为伪距与多普勒不相关，那么给我的启示是在EKF估计器里面，我最好不要用钟漂递推钟差。

## 2.5 TDCP（历元间差分载波相位）

在RINEX原始观测数据的语境下，ADR代表的就是载波相位。GSDC赛事中，高分选手几乎都用到了ADR，并用其进行速度估计。

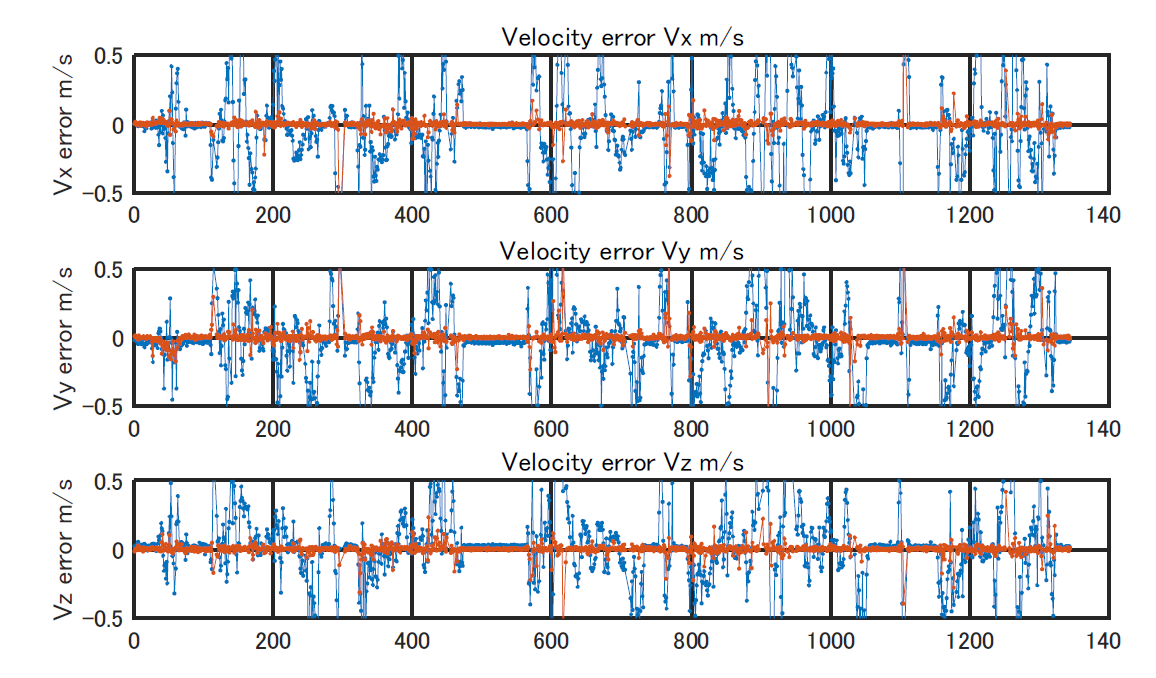
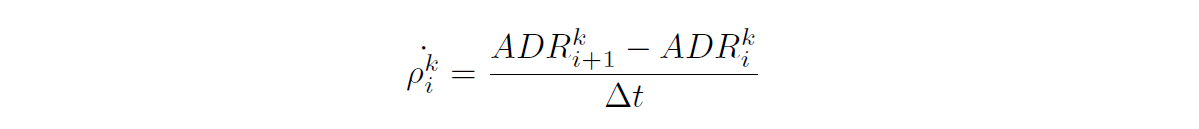


图2-7 多普勒与ADR速度估计之间的误差比较。ADR能比多普勒更准确地估计速度，但由于周跳现象，其可用性较低。

多普勒和ADR都可以用来估计速度，而基于ADR计算的速度TDCP（来自载波的伪距率）比基于多普勒计算的速度更加精确（如图2-7所示）。基于ADR的伪距率TDCP计算如下：



TDCP的优点是可以实现相对开阔地段的亚米级定位。但是TDCP的准确性低于多普勒（因为需要考虑周跳和半周模糊度问题）。特别的，ADR的缺失通常发生在突然加速和减速中，此时的速度状态也会出现缺失[3]。

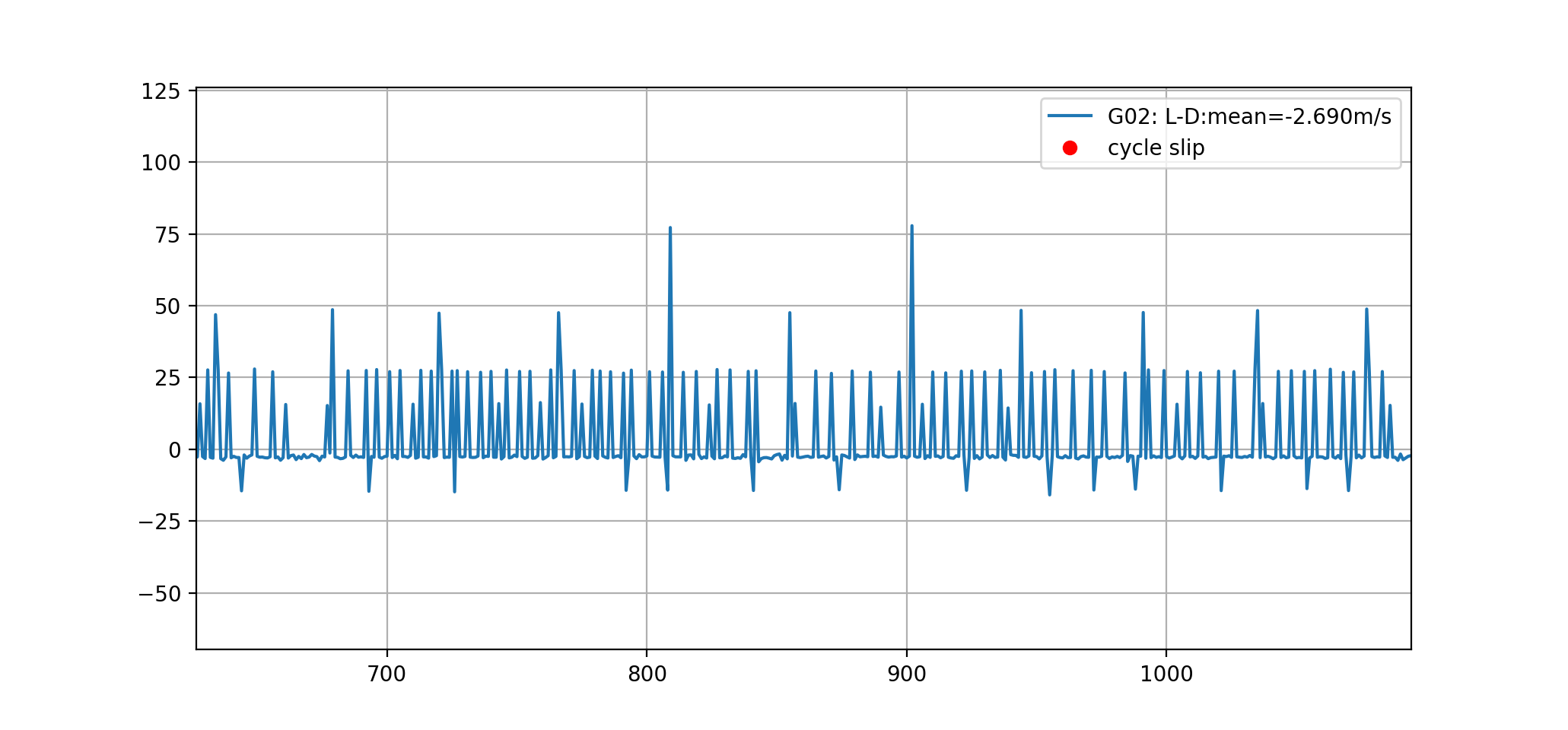


图2-8(a) TDCP与伪距率（来自多普勒）的差值（简称为L-D）

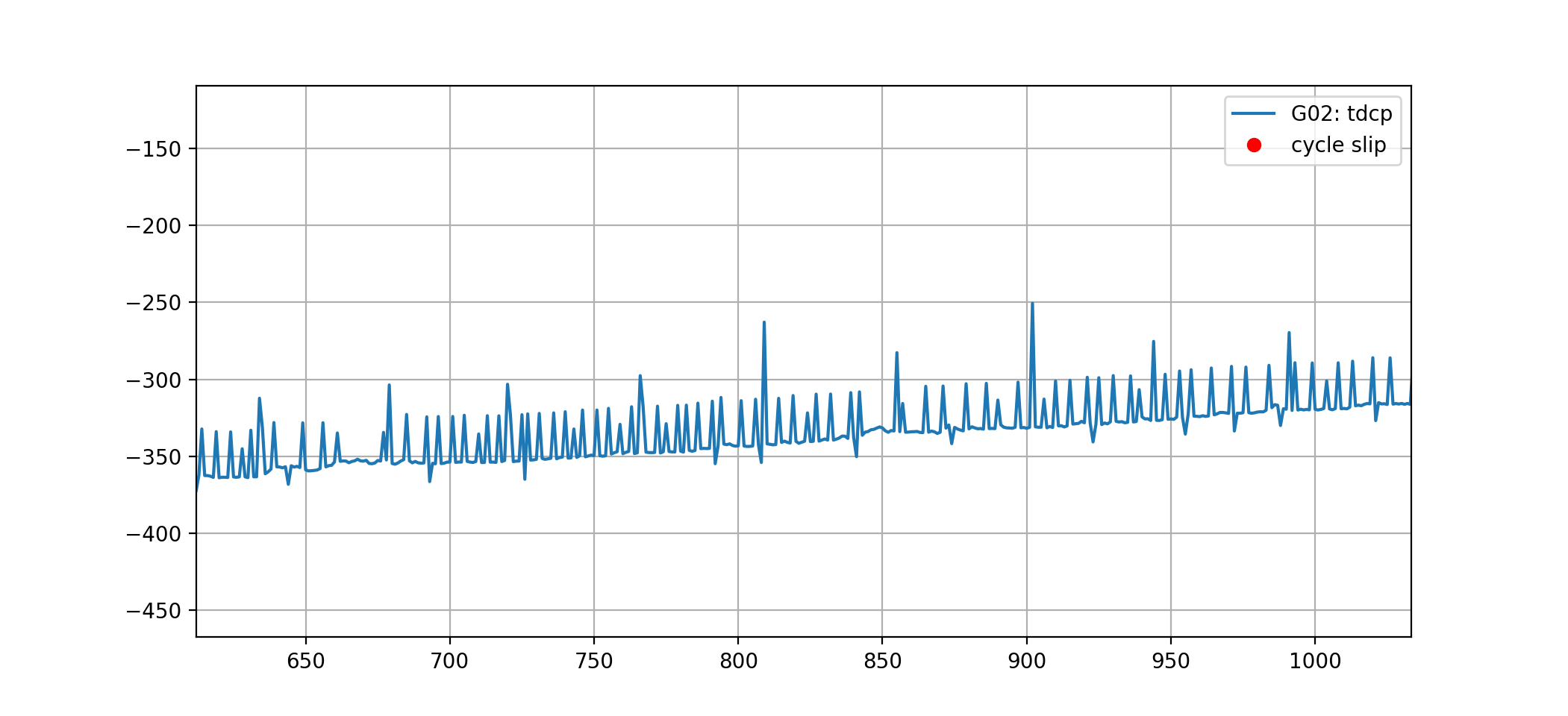


图2-8(b) TDCP

图2-8 原始观测中的系统性偏差

分析载波相位数据的时候发现，载波观测中存在一定的系统性偏差，它导致了TDCP和L-D数值中出现了很多毛刺，后续的研究发现，伪距中也存在相同的现象，并且某个历元出现问题，那么所有卫星都会存在该现象，而非某颗卫星独有。我暂时将其归因于钟跳，尽管通常不会每几秒就发生一次钟跳。

设备产生钟跳时，该误差通常会归到钟差项中，因此理论上它不会影响算法的运算结果，不过由于残差的变化，它可能会影响到后续残差相关的操作，由于我抗差会使用迭代最小二乘，而迭代过程中会消除钟差（含钟跳）的影响，因此钟跳将不会影响后续的抗差部分。

# 3. RWLS

由于最小二乘较为简单，能比较方便和快速地对一些方法进行验证，因此初始时我将利用最小二乘进行一些基础的优化。

## 3.1 SNR加权模型

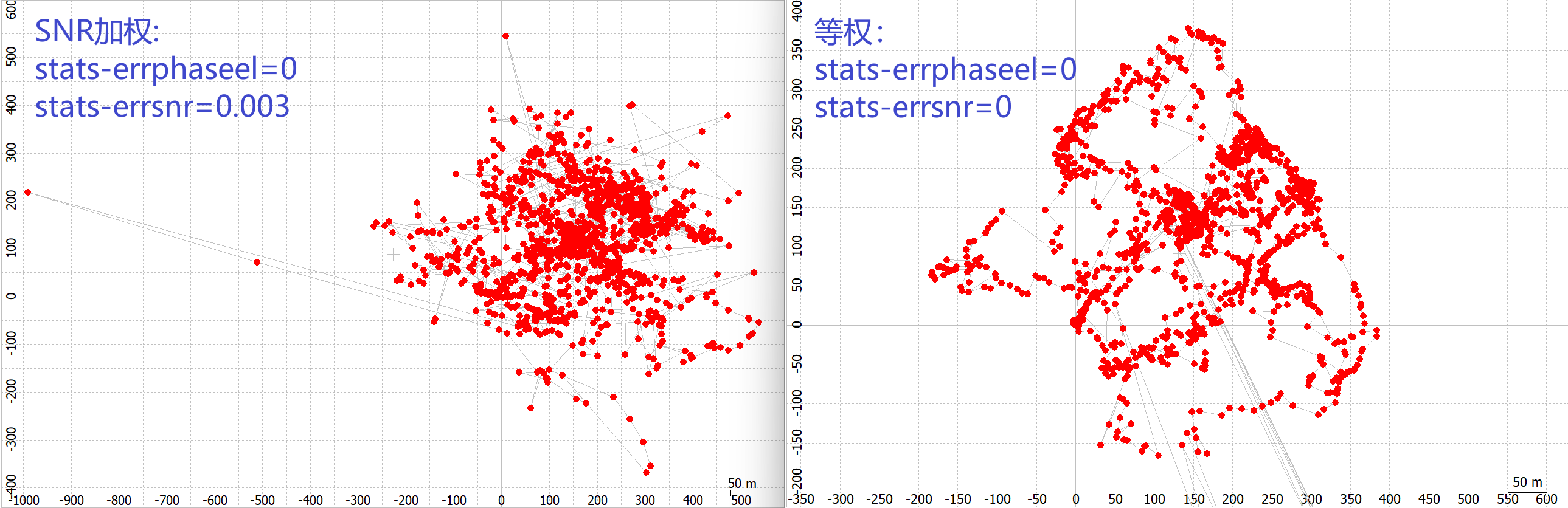


图3-1 SNR加权与等权模型对比（02-street/data01）

demo5-b34k版本RTKLIB在打开SNR加权后轨迹会崩溃，不过这并不是因为SNR加权效果不如等权，这里显然是错误，而非误差。



图3-2 SNR加权代码

将ssat[i]改成ssat[sat-1]即可解决，打开SNR加权会对性能存在提升。

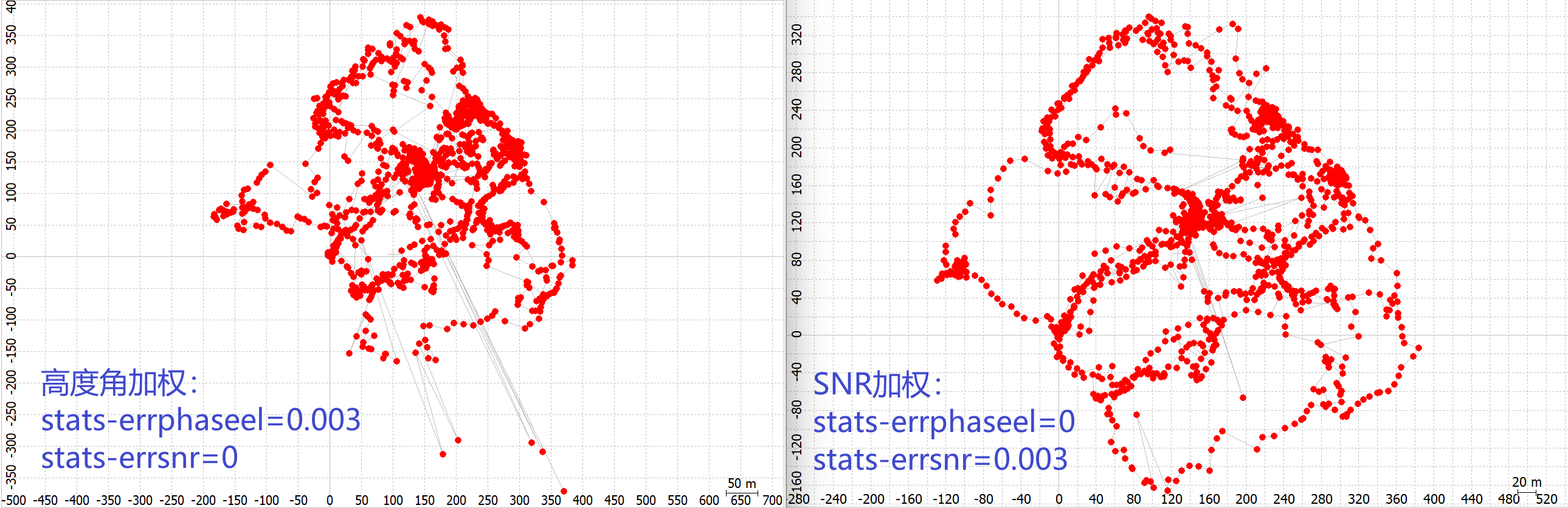


图3-3 高度角与SNR加权模型（02-street/data01）

表3-1　SNR加权性能指标

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **场景** | **高架环路** | **树荫街道** | **市中心** | **高架遮挡** |
| 组别#1（data01） | | | | |
| **原厂结果(m)** | 6.17  (4.58 7.75 30.79) | 8.63  (5.35 11.90 14.29) | 10.06  (5.64 14.48 34.47) | 27.92  (8.56 47.29 82.89) |
| **WLS结果(m)** | 16.68  (5.97 27.40 160.53) | 31.38  (12.65 50.10 190.73) | 48.00  (22.53 73.46 3193.52) | 32.59  (13.67 51.52 163.76) |
| 组别#2（data02） | | | | |
| **原厂结果(m)** | 6.13  (4.41 7.85 19.23) | 10.44  (6.55 14.33 28.74) | 13.25  (7.88 18.61 38.67) | 44.62  (8.72 80.52 93.91) |
| **WLS结果(m)** | 19.96  (7.88 32.03 134.01) | 36.16  (14.79 57.53 159.24) | 42.28  (19.47 65.10 117.21) | 58.86  (20.77 96.96 **11189.89**) |

1. 最终指标为(CEP50+CEP95)/2，其他指标为CEP50、CEP95和最大漂移（CEP100）；
2. CEP50体现轨迹的平均水平，CEP95体现误差高位数水平，最大漂移体现异常情况；
3. **绿色**表示改善明显的指标，**红色**表示未改善指标或异常指标。

以上为使用SNR加权模型WLS在不同场景上的表现，目前由于代码中没有抗差，因此最大漂移指标都比较大。

基于卫星仰角的GNSS观测模型更适用于专用接收机。然而，对于智能手机，基于信号强度（载噪比，C/N0）的模型比依赖仰角的模型更能有效地处理观测权重[4,5,6]。因为智能手机观测误差往往由信号质量问题主导，而非大气误差，而使用仰角加权观测的动机正是为了解决大气误差。

## 3.2 速度估计中的加权

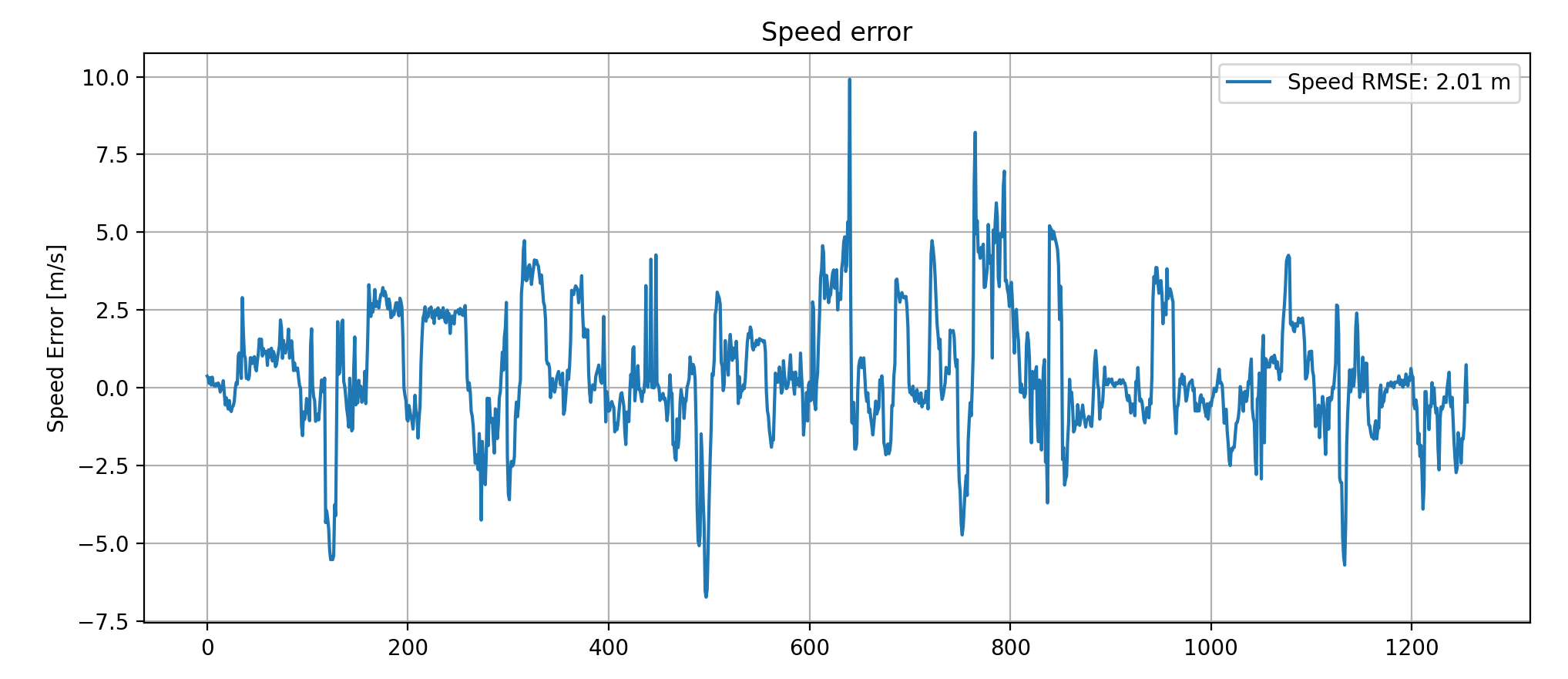


图3-4 未增加加权模型的速度误差（02-street/data01）

RTKLIB中的速度估计原本是没有进行加权的，其表现的性能如图3-4所示，基本和等权差不多，这里可以直接将伪距中的加权方法拷贝过来。

// resdop内部

opt\_.eratio[0] = 30.0; // doppler/phase error ratio

if (ssat)

    var[nv]=varerr(&opt\_,&ssat[obs[i].sat-1],&obs[i],azel[1+i\*2],sys);

else

    var[nv]=varerr(&opt\_,NULL,&obs[i],azel[1+i\*2],sys);

…

// resdop调用

for (j=0;j<nv;j++) {

    sig=sqrt(var[j]);

    v[j]/=sig;

    for (k=0;k<4;k++) H[k+j\*4]/=sig;

}

具体的，需要在resdop中增加var计算的代码，随后在resdop调用的地方进行加权操作。



图3-5 增加SNR加权的速度误差（02-street/data01）

可以看到速度的误差水平有所降低。

## 3.3 M估计（稳健估计）

M估计是一种广义的最大似然估计方法，旨在最小化一个目标函数，该函数对异常值不敏感。其核心思想是通过定义一个损失函数（或称为目标函数）来替代最小二乘法中的平方损失。

如果换一个角度来看M估计，可以认为它是一种更可靠的加权模型，该加权模型利用了观测中的验前残差，验前残差反映了预测与观测之间的差别，差别越小则给观测更大的权重，反之则给观测更小的权重。

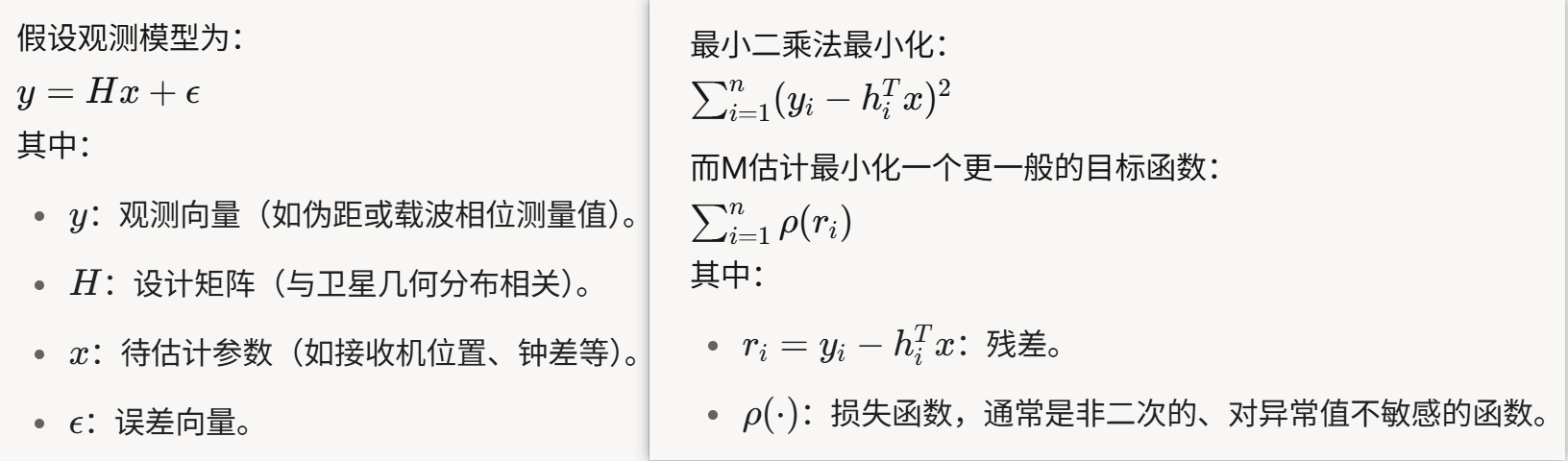


图3-6 M估计理论

M估计的数学表达如上所示。另外M估计的损失函数（或目标函数）通常包括Huber、IGG3和Tukey模型等，这里我将主要关注Huber和IGG3。

static int RobustLsq(const double \*H, const double \*v, int nx, int nv, double \*dx, double \*Q, double \*var, int mode)

{

    double \*W = eye(nv), \*v\_new = mat(1, nv), \*H\_new = mat(nx, nv);

    double \*dx\_prev = mat(nx, 1), \*ddx = mat(nx, 1);

    double sig, weight, resid, diff;

    int info = 0, iter, i, j;

    const int max\_iter = 15;

    const double tol = 1e-2;

    matcpy(dx\_prev, dx, nx, 1);

    memset(dx, 0x00, nx \* sizeof(double));

    for (iter = 0; iter < max\_iter; iter++) {

        for (i = 0; i < nv; i++) {

            sig = sqrt(var[i]);

            weight = 1.0 / sig;

            resid = fabs(v[i] - dot(H + i \* nx, dx, nx));

            if (mode == ROBUST\_VEL) {

                weight = RobustWeightLsq(resid, sig, ROBUST\_HUBER, ROBUST\_VEL);

            } else {

                weight = RobustWeightLsq(resid, sig, ROBUST\_HUBER, ROBUST\_POS);

            }

            W[i + i \* nv] = SQR(weight);

            v\_new[i] = v[i] \* weight;

            for (j = 0; j < nx; j++) {

                H\_new[j + i \* nx] = H[j + i \* nx] \* weight;

            }

        }

        if ((info = lsq(H\_new, v\_new, nx, nv, dx, Q))) {

            trace(3, "RobustLsq: lsq failed, info=%d \n", info);

            break;

        }

        for (i = 0; i < nx; i++) ddx[i] = dx[i] - dx\_prev[i];

        matcpy(dx\_prev, dx, nx, 1);

        diff = norm(ddx, nx);

        if (diff<tol) break;

    }

    free(W); free(v\_new); free(H\_new); free(dx\_prev); free(ddx);

return info;

}

以上给出给出了M估计实现的代码，需要注意：

* 迭代流程。迭代最小二乘，先计算权，再更新H和v；
* dx初值。初始的时候需要将dx赋值为0，不然结果可能会出现毛刺，本质是增加了迭代次数，以及位置首次迭代会除以残差本身。

表3-2　抗差WLS性能指标

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **场景** | **高架环路** | **树荫街道** | **市中心** | **高架遮挡** |
| 组别#1（data01） | | | | |
| **SNR加权**  **WLS结果(m)** | 16.68  (5.97 27.40 160.53) | 31.38  (12.65 50.10 190.73) | 48.00  (22.53 73.46 3193.52) | 32.59  (13.67 51.52 163.76) |
| **抗差**  **WLS结果(m)** | **8.19**  (**3.63 12.75** 160.53) | **13.52**  **(7.01 20.03 128.81)** | **28.81**  **(11.87 45.74 619.10)** | 31.20  (12.70 49.69 87.36) |
| 组别#2（data02） | | | | |
| **SNR加权**  **WLS结果(m)** | 19.96  (7.88 32.03 134.01) | 36.16  (14.79 57.53 159.24) | 42.28  (19.47 65.10 117.21) | 58.86  (20.77 96.96 **11189.89**) |
| **抗差**  **WLS结果(m)** | **8.79**  **(5.15 12.44 59.99)** | **16.51**  **(6.69 26.33 147.37)** | **31.43**  **(12.00 50.86 167.75)** | **51.03**  (**14.18 87.88** **11189.89**) |

1. 最终指标为(CEP50+CEP95)/2，其他指标为CEP50、CEP95和最大漂移（CEP100）；
2. CEP50体现轨迹的平均水平，CEP95体现误差高位数水平，最大漂移体现异常情况；
3. **绿色**表示改善明显的指标，**红色**表示未改善指标或异常指标。

由上表可看到抗差最小二乘，比单纯的SNR加权效果要很多。

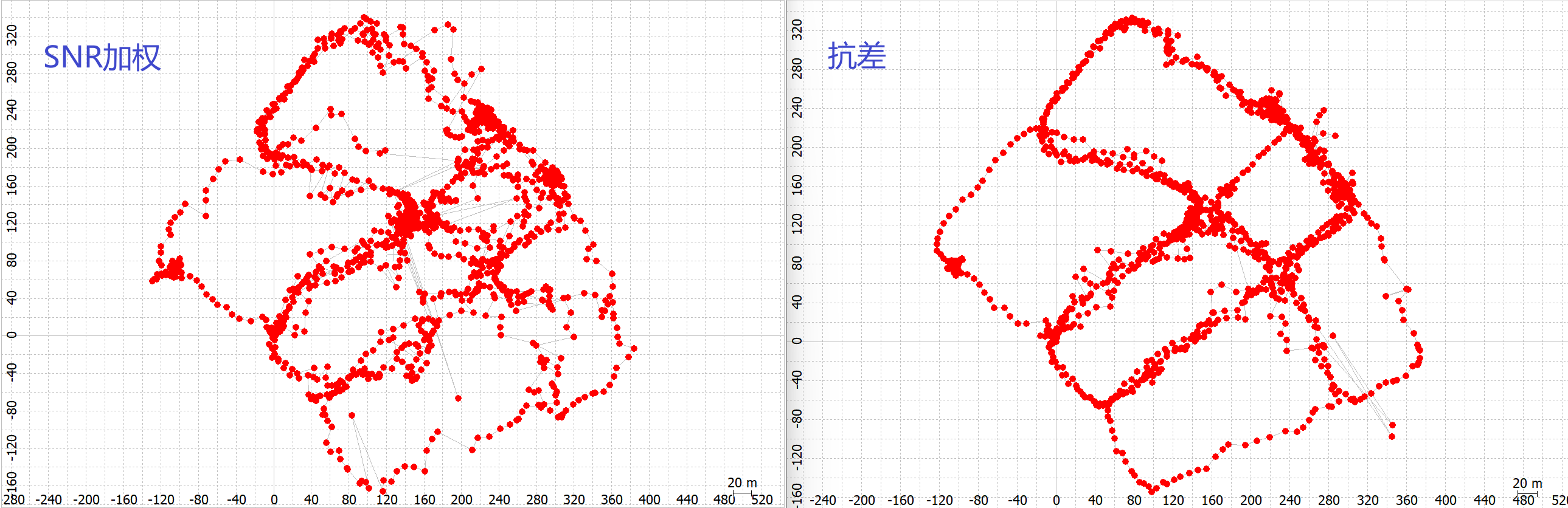


图3-7(a) SNR加权与抗差轨迹对比（02-street/data01）

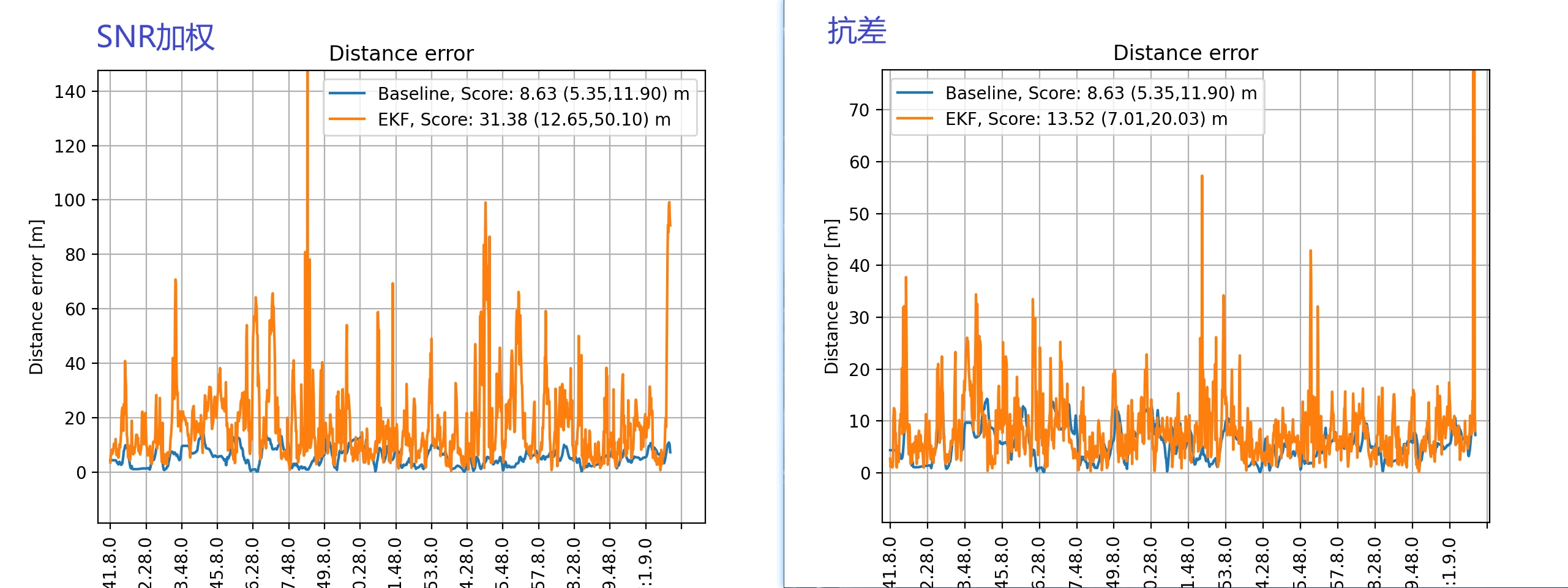


图3-7(b) SNR加权与抗差位置误差对比（02-street/data01）



图3-7(c) SNR加权与抗差速度误差对比（02-street/data01）

图3-7 SNR加权与抗差对比（02-street/data01）

以02-street/data01组别数据为例，抗差后的结果所表现的性能比不抗差提升了很多。

# 4. EKF

说明。

## 4.1 xxx

文字。

文字。

### 4.1.1 xxx

文字。

文字。

# 5. 代价最小化

说明。

## 5.1 xxx

文字。

文字。

### 5.1.1 xxx

文字。

文字。

# 附录

## A.1 C/N0、SNR与CNR

GNSS领域中常把C/N0称为“SNR”，主要是因为SNR是更广为人知的术语。但在技术上，C/N0是更准确的指标，反映了GNSS信号的载波功率与噪声密度关系。C/N0、SNR和CNR的定义、用途和表达方式有所不同。以下是它们的区别：

**1. C/N0（Carrier-to-Noise Density Ratio，载波噪声密度比）**

* **定义：**C/N0是载波功率与单位带宽内噪声功率的比值，单位通常为dB-Hz。
* **公式：**C/N0 = 10 \* log10(Pc / N0)，其中Pc是载波功率，N0是噪声功率谱密度。
* **典型值：**在GNSS中，C/N0通常在30 dB-Hz到50 dB-Hz之间，值越高表示信号质量越好。

**2. SNR（Signal-to-Noise Ratio，信噪比）**

* **定义：**SNR是信号功率与噪声功率的比值，单位通常为dB。
* **公式：**SNR = 10 \* log10(Ps / Pn)，其中Ps是信号功率，Pn是总噪声功率。
* **典型值：**GNSS接收机的SNR值可能在10 dB到30 dB之间，具体取决于设备和环境。

**3. CNR（Carrier-to-Noise Ratio，载噪比）**

* **定义：**CNR是载波功率与总噪声功率的比值，单位为dB。
* **公式：**CNR = 10 \* log10(Pc / Pn)，其中Pc是载波功率，Pn是总噪声功率。
* **典型值：**CNR值通常低于C/N0，具体数值取决于带宽和环境。

三者主要区别总结如下：

表A-1　GNSS标识符

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **指标** | **单位** | **关注点** | **与带宽关系** | **应用场景** |
| C/N0 | dB-Hz | 载波功率 vs 噪声密度 | 无关 | 信号质量评估、跟踪能力分析 |
| SNR | dB | 信号功率 vs 总噪声 | 有关 | 基带信号处理、相关器输出分析 |
| CNR | dB | 载波功率 vs 总噪声 | 有关 | 射频前端信号质量评估 |

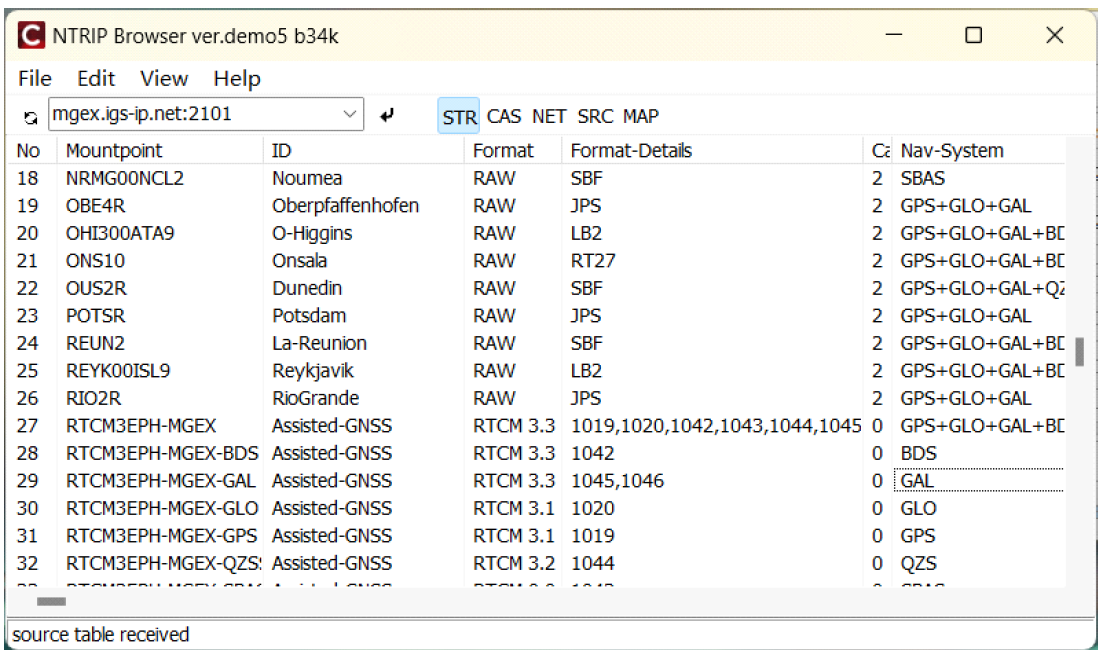
C/N0是GNSS中最常用的指标，用于判断信号是否足以被接收机捕获和跟踪，尤其在弱信号环境（如城市峡谷、室内）中非常关键。

SNR常用于接收机内部的信号处理分析，帮助优化算法或评估抗噪性能。

CNR多用于射频阶段，反映天线和前端硬件的性能。

## A.2 MGEX-AGNSS服务

服务网址：[mgex.igs-ip.net GNSS Streaming Server](https://mgex.igs-ip.net/home)



图A-1　RTKLIB NTRIP Browser

使用RTKLIB的NTRIP服务工具连接星历AGNSS服务，选择RTCM3EPH-MGEX挂载点。可以保持习惯直接挂着该服务让它每天源源不断接收RTCM格式的星历数据即可，由于并不是流媒体格式，因此并不会占据很多内存。避免有时候做实验忘记记录星历了。

后面可以使用RTKCONV将RTCM星历数据转换为RINEX格式。

# 参考资料

[01] [An Open-Source Factor Graph Optimization Package for GNSS and IMU Integration in Smartphones](https://www.ion.org/publications/abstract.cfm?articleID=19923)

[02] Source code to reproduce the results of the GSDC 2023 on Kaggle: <https://github.com/taroz/gsdc2023>

[03] [GSDC 2022 3rd Place Solution](https://www.kaggle.com/competitions/smartphone-decimeter-2022/discussion/341305)

[04] T. Suzuki, “Gnss odometry: Precise trajectory estimation based on carrier phase cycle slip estimation,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7319–7326, 2022.

[05] M. H˚akansson, “Characterization of gnss observations from a nexus 9 android tablet,” *GPS solutions*, vol. 23, no. 1, p. 21, 2019.

[06] European Global Navigation Satellite Systems Agency. (2019). PPP-RTK Market and Technology Report. Available online: <https://www.euspa.europa.eu/sites/default/files/calls_for_proposals/rd.03_-_ppp-rtk_market_and_technology_report.pdf> (accessed on 18 Aug 2022).