GREAT-MSF User Guide

Contents

Chapter 1 Overview

GREAT (GNSS+ Research, Application and Teaching) is a comprehensive software platform designed and developed by the School of Geodesy and Geomatics at Wuhan University for space-geodesy data processing, precise positioning and orbit determination, and multi-source integrated navigation. In the software stack, core computation modules are implemented in C++ (C++17), while auxiliary scripting modules use Python 3 and C shell to automate data processing. All C++ modules follow the Google C++ Style Guide, and Git is used for version control. GREAT uses CMake for build management, allowing users to flexibly select mainstream C++ compilers such as GCC, Clang, and MSVC. Command-line applications are currently provided for both Windows and Linux platforms.

GREAT-MSF is a key module within the GREAT software suite, dedicated to multi-sensor fusion (MSF) navigation solutions. The official repository is: https://github.com/GREAT-WHU/GREAT-MSF.git. GREAT-MSF is an extension of **GREAT-PVT** (https://github.com/GREAT-WHU/GREAT-PVT.git). In GREAT-MSF, core computation modules are implemented in C++, and auxiliary scripts in Python 3 are provided for results plotting. GREAT-MSF uses CMake for build management, enabling flexible choice among GCC, Clang, and MSVC. The software consists of two portable libraries, **LibGREAT** and **LibGnut**. Beyond the GNSS positioning solutions available in GREAT-PVT, LibGREAT further integrates multi-sensor fusion navigation capabilities, including data decoding, storage, and sensor-fusion algorithms used in filtering and estimation. Building upon GREAT-PVT, GREAT-MSF adds the following major features:

- Support for multiple GNSS constellations: GPS, GLONASS, Galileo and BDS-2/3.

- Support for PPP/INS loose and tight coupling, including PPP models such as the ionosphere-free combination and un-differenced/un-combined formulations.
- Support for RTK/INS loose and tight coupling, with carrier-phase ambiguity fixing.
- Support for rapid initialization of the integrated system, including displacement-vector and velocity-vector–aided alignment.
- Support for custom IMU data formats and noise models.
- Support for zero-velocity updates (ZUPT) and nonholonomic constraints (NHC), as well as wheel-speed odometry(ODO).

In addition, the package provides plotting scripts for positioning results to facilitate users' data analysis.

Chapter 2 Principles of Integrated Navigation

2.1 Inertial-sensor errors and inertial navigation

2.1.1 Error sources of inertial instruments

Inertial instruments—gyroscopes and accelerometers—exhibit bias, scale-factor error, cross-axis coupling, and random measurement noise. Each error source typically comprises four components: a stable constant term, a temperature-dependent term, a turn-on term, and an in-run varying term. The IMU measurement model can be written as (2-1):

$$\begin{cases} \tilde{\boldsymbol{\omega}}_{ib}^{b} = \left(\boldsymbol{I} + \boldsymbol{S}_{\omega}\right)\cdot \boldsymbol{\omega}_{ib}^{b} + \delta\boldsymbol{\omega}_{ib}^{b} + \boldsymbol{\varepsilon}_{\omega} \\ \tilde{\boldsymbol{f}}_{ib}^{b} = \left(\boldsymbol{I} + \boldsymbol{S}_{f}\right)\cdot \boldsymbol{f}_{ib}^{b} + \delta\boldsymbol{f}_{ib}^{b} + \boldsymbol{\varepsilon}_{f} \end{cases} \tag{2-1}$$

where $\tilde{\boldsymbol{\omega}}_{ib}^{b}$ and $\boldsymbol{\omega}_{ib}^{b}$ denote the measured and true angular rates (gyroscope), respectively; $\tilde{\boldsymbol{f}}_{ib}^{b}$ and $\boldsymbol{f}_{ib}^{b}$ denote the measured and true specific force (accelerometer), respectively; $\boldsymbol{I}$ is the identity matrix; $\boldsymbol{S}_{\omega}$ and $\boldsymbol{S}_{f}$ represent multiplicative errors (scale-factor and cross-axis coupling) for the gyroscope and accelerometer; $\delta\boldsymbol{\omega}_{ib}^{b}$ and $\delta\boldsymbol{f}_{ib}^{b}$ are the gyroscope and accelerometer measurement errors, which in common practice may be modeled using only constant biases ($\boldsymbol{b}_g$ and $\boldsymbol{b}_a$); $\boldsymbol{\varepsilon}_{\omega}$ and $\boldsymbol{\varepsilon}_{f}$ are the angular-rate and specific-force measurement noises.

Bias error refers to the intrinsic offset of the sensor and is the dominant component of inertial-instrument error, comprising static and dynamic parts. The static bias remains constant within a single run but changes from run to run at power-up; the dynamic part varies over time and is commonly referred to as bias stability/instability. For most low-cost MEMS IMUs, the constant bias is large and poorly stable, which fundamentally drives the rapid error growth of low-cost IMU navigation. In navigation data processing, the composite bias is commonly modeled as a first-order Gauss–Markov process.

A scale-factor error is a multiplicative error proportional to the input: gyro output error scales with the true angular rate, and accelerometer output error scales with the true specific force. Cross-axis coupling arises from non-orthogonality among the IMU sensitive axes and is typically limited by manufacturing tolerances. Both terms are constant within a single power-up session and can be treated as constants for estimation; however, in many ground-vehicle scenarios their observability is limited, so these parameters should be estimated with care.

IMU measurement noise is affected by multiple factors, such as mechanical vibration and signal-processing electronics. The high-frequency portion is smoothed by the integration inherent to inertial navigation, while sub-1 Hz components can often be approximated as white noise and are characterized by power spectral density (PSD); the product of the PSD and the squared sampling interval yields the noise variance. Angular-rate noise integrates to attitude random walk, and specific-force noise integrates to velocity random walk.

2.1.2 Inertial-navigation initialization

IMU measurements are relative increments between successive epochs, and each mechanization update is computed from the previous epoch. Therefore, an inertial navigation system (INS) must be initialized before service—namely, position, velocity, and attitude initialization. Initial position and velocity are usually supplied by external navigation systems (e.g., GNSS positioning, map matching, Doppler radar). Attitude initialization—commonly called initial alignment—includes leveling (roll and pitch) and heading (azimuth) and is the primary concern.

When the vehicle is stationary, accelerometer outputs of specific force are dominated by gravity; roll and pitch tilt gravity into the horizontal axes, so the horizontal attitude can be obtained by smoothing a few seconds of accelerometer data. Heading initialization can be categorized by the presence of aiding as self-alignment versus aided alignment, and by the vehicle state as static-base alignment versus dynamic alignment. Self-alignment depends on the gyro’s north-seeking performance and includes coarse and fine alignment; it is generally accepted that gyros with bias an order of magnitude smaller than the Earth rotation rate (< 1 deg/h) are required for self-alignment. In practice, most vehicle-grade gyros used in integrated navigation lack this capability, so heading is typically initialized with external aiding. Common methods include GNSS dual-antenna attitude determination, GNSS motion-vector-aided initialization, and magnetic heading.

GNSS motion-vector-aided initialization uses the direction vector between two GNSS solutions; the motion vector can be either a displacement vector or a velocity vector. The displacement-vector method exploits the physical property that the heading remains nearly constant during straight-line driving, estimating the course angle from the trajectory. The velocity-vector method computes heading in a single epoch from the projection of the current velocity direction in the local-level frame. The basic formulas are (2-2):

$$\begin{aligned} \psi &= \arctan\left(\left|\boldsymbol{dr}^{l}_{(t_i,t_j),E}\right| \Big/ \left|\boldsymbol{dr}^{l}_{(t_i,t_j),N}\right|\right) \\ \psi &= \arctan\left(\left|\boldsymbol{v}^{l}_{t,E}\right| \Big/ \left|\boldsymbol{v}^{l}_{t,N}\right|\right) \end{aligned} \tag{2-2}$$

where $\boldsymbol{dr}^{l}_{(t_i,t_j)}$ is the displacement vector from t_i to t_j; and $\boldsymbol{v}^{l}_{t}$ is the velocity vector at time t. The heading accuracy of the two methods depends on the accuracies of the position- and velocity-derived vectors, respectively.

2.1.3 Inertial mechanization

Mechanization refers to obtaining position, velocity, and attitude by time-integrating raw IMU observations. After initializing position, velocity, and attitude, raw measurements are corrected for errors; angular rates are integrated to update attitude; specific force is resolved into the chosen navigation frame using the attitude; and double integration yields velocity and position. The INS differential equations in the Earth-centered, Earth-fixed (ECEF) frame can be written as (2-3):

$$\begin{bmatrix} \dot{\boldsymbol{C}}_b^e \\ \dot{\boldsymbol{v}}^e \\ \dot{\boldsymbol{r}}^e \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_b^e \cdot (\boldsymbol{\omega}_{eb}^b \times) \\ \boldsymbol{C}_b^e \boldsymbol{f}^b - 2\boldsymbol{\omega}_{ie}^e \times \boldsymbol{v}^e + \boldsymbol{g}^e \\ \boldsymbol{v}^e \end{bmatrix} \tag{2-3}$$

where superscripts b and i denote the body and inertial frames, respectively; $\boldsymbol{C}_b^e$ is the rotation matrix from the body frame to the ECEF frame; $\boldsymbol{\omega}_{eb}^b$ is the skew-symmetric matrix of the Earth rotation rate expressed in ECEF; $\boldsymbol{\omega}_{ie}^e$ is the transport-rate term due to motion over the rotating Earth; $\boldsymbol{g}^e$ is the gravity vector expressed in ECEF; $\boldsymbol{r}^e$ and $\boldsymbol{v}^e$ are the ECEF position and velocity, respectively.

Mechanization comprises attitude update, specific-force transformation, velocity update, and position update. For epoch t_k, the attitude update can be expressed as (2-4):

$$\boldsymbol{C}_{b(t)}^{e(t)} = \boldsymbol{C}_{e(t-1)}^{e(t)} \boldsymbol{C}_{b(t-1)}^{e(t-1)} \boldsymbol{C}_{b(t)}^{b(t-1)} \tag{2-4}$$

where $\boldsymbol{C}_{e(t-1)}^{e(t)}$ is the rotation of the Earth-fixed frame relative to the inertial frame from t_{k-1} to t_k (derivable from the Earth rotation rate $\boldsymbol{\omega}_{ie}^e$); and $\boldsymbol{C}_{b(t-1)}^{b(t)}$ is the body rotation over the interval, obtained from the corrected angular-rate increments.

Accelerometer outputs are specific force in the body frame and must be projected into the selected navigation frame using the attitude before integration to obtain velocity increments. One must subtract gravity and kinematic (Coriolis/transport) accelerations—collectively referred to as "harmful" accelerations. Thus, the velocity at t_k can be written as (2-5):

$$\begin{aligned} \boldsymbol{v}^e(t_k) &= \boldsymbol{v}^e(t_{k-1}) + \Delta\boldsymbol{v}_{sf}^e + \Delta\boldsymbol{v}_{hf}^e \\ \Delta\boldsymbol{v}_{sf}^e &= \int_{t_{k-1}}^{t_k} \boldsymbol{C}_b^e(t_k)\, \boldsymbol{f}^b(t_k) dt \\ \Delta\boldsymbol{v}_{hf}^e &= \int_{t_{k-1}}^{t_k} \left[-2\boldsymbol{\omega}_{ie}^e(t_k) \times \boldsymbol{v}^e(t_k) + \boldsymbol{g}^e(t_k)\right] dt \end{aligned} \tag{2-5}$$

where $\Delta\boldsymbol{v}_{sf}^e$ is the velocity increment due to specific force, and $\Delta\boldsymbol{v}_{hf}^e$ is the increment due to harmful accelerations. Position is then updated using the trapezoidal rule:

$$\boldsymbol{r}^e(t_k)=\boldsymbol{r}^e(t_{k-1})+\left[\boldsymbol{v}^e(t_k)+\boldsymbol{v}^e(t_{k-1})\right]/2 \tag{2-6}$$

2.2 GNSS/INS Integrated Models

2.2.1 INS error-state model

The vehicle's navigation state is time-propagated by INS mechanization. In the Earth-fixed frame, the differential equations for the navigation-state errors can be written as (2-7):

$$\begin{bmatrix}\dot{\boldsymbol{\phi}}\\ \delta\dot{\boldsymbol{v}}^e\\ \delta\dot{\boldsymbol{r}}^e\end{bmatrix}=\begin{bmatrix}-\boldsymbol{\omega}_{ie}^e\times\boldsymbol{\phi}-\boldsymbol{C}_b^e\delta\boldsymbol{\omega}_{ib}^b+\boldsymbol{\xi}_{\boldsymbol{\phi}}\\ \boldsymbol{C}_b^e\boldsymbol{f}^b\times\boldsymbol{\phi}-2\boldsymbol{\omega}_{ie}^e\times\delta\boldsymbol{v}^e+\boldsymbol{C}_b^e\delta\boldsymbol{f}^b+\boldsymbol{\xi}_v\\ \delta\boldsymbol{v}^e+\boldsymbol{\xi}_{\boldsymbol{r}}\end{bmatrix} \tag{2-7}$$

where $\boldsymbol{\phi}$, $\delta\boldsymbol{v}^e$ and $\delta\boldsymbol{r}^e$ are the attitude misalignment angles, velocity error, and position error, respectively. The attitude misalignment is defined as the difference between the computed and true attitude matrices, e.g., $\boldsymbol{C}_b^e=(1+\boldsymbol{\phi}\times)\tilde{\boldsymbol{C}}_b^e$; The processes $\xi_{\boldsymbol{\phi}}$, ξ_v and $\xi_{\boldsymbol{r}}$ are the system driving white-noise terms for attitude, velocity, and position, each assumed zero-mean Gaussian. Note that "attitude in the Earth-fixed frame" lacks direct physical interpretation; one must convert to the local-level frame's attitude matrix before extracting the conventional Euler angles (pitch, roll, and heading).

IMU-related states include (but are not limited to) sensor bias, scale-factor error, gravity-model error, and cross-axis coupling. State selection depends on factors such as sensor grade and vehicle dynamics. Generally, IMU errors tied to vehicle dynamics can be observable, but not all are; therefore, simulation studies are commonly used to assess observability under different motion profiles. In most ground-vehicle scenarios, it suffices to estimate only the gyroscope and accelerometer biases as IMU-related parameters, typically modeled as first-order Gauss–Markov processes (2-8):

$$\begin{bmatrix} \dot{\boldsymbol{b}}_g \\ \dot{\boldsymbol{b}}_a \end{bmatrix} = \begin{bmatrix} -\frac{1}{\boldsymbol{\tau}_g}\boldsymbol{b}_g + \boldsymbol{w}_g \\ -\frac{1}{\boldsymbol{\tau}_a}\boldsymbol{b}_a + \boldsymbol{w}_a \end{bmatrix} \tag{2-8}$$

where $\boldsymbol{w}_g$ and $\boldsymbol{w}_a$ are the gyro and accelerometer driving noises; $\boldsymbol{\tau}_g$ and $\boldsymbol{\tau}_a$ are the autocorrelation times of the Gauss–Markov processes. Rough values may be obtained via Allan-variance-based power spectral analysis.

2.2.2 GNSS-related parameter state model

GNSS-related states depend on the adopted processing strategy. In RTK mode, the GNSS states to be estimated include residual atmospheric delays and double-difference ambiguities. In un-differenced ionosphere-free processing, GNSS-related states include receiver clock offset, ionosphere-free ambiguities, etc. Regardless of the chosen GNSS processing strategy, GNSS states in a tightly coupled estimator can still be modeled and estimated according to their standard formulations; see **GREAT-PVT_1.0** for details.

2.2.3 Loosely coupled GNSS/INS measurement model

Using the known spatial relationship (lever arm) between the GNSS antenna phase center and the IMU center, one performs center compensation, i.e.,

$$\boldsymbol{r}_{ant}^e = \boldsymbol{r}_{imu}^e + \boldsymbol{C}_b^e \cdot \boldsymbol{l}_{ant}^b \tag{2-9}$$

where $\boldsymbol{r}_{ant}^e$ and $\boldsymbol{r}_{imu}^e$ are the ECEF positions of the GNSS antenna phase center and the IMU center, respectively; $\boldsymbol{l}_{ant}^b$ is the antenna position vector expressed in the IMU body frame and can be measured accurately in advance. Accounting for the errors of the involved quantities yields

$$\begin{aligned} &\tilde{\boldsymbol{r}}_{ant}^e - \boldsymbol{\delta r}_{ant}^e = \tilde{\boldsymbol{r}}_{imu}^e - \boldsymbol{\delta r}_{imu}^e + (1+\boldsymbol{\phi}\times)\tilde{\boldsymbol{C}}_b^e \cdot \boldsymbol{l}_{ant}^b \\ &\Rightarrow \boldsymbol{\delta}\tilde{\boldsymbol{r}}_{ant}^e = \boldsymbol{\delta r}_{imu}^e + \left(\tilde{\boldsymbol{C}}_b^e \cdot \boldsymbol{l}_{ant}^b\right)\times\boldsymbol{\phi} \end{aligned} \tag{2-10}$$

which, together with (2-9), forms the spatial lever-arm model for GNSS/IMU fusion. Combined with the GNSS models in **GREAT-PVT_1.0**, one can derive loosely/tightly coupled measurement models based on raw pseudorange and carrier-phase observations. Note that tight coupling of raw GNSS observations relies on a Taylor expansion about the INS-mechanized position transformed into the appropriate frame.

The PPP/INS loose-integration measurement equations are：

$$\tilde{\boldsymbol{p}}_b^e - \tilde{\boldsymbol{p}}_{ant}^e + \tilde{\boldsymbol{C}}_b^e \cdot \boldsymbol{p}_{ant}^b + \boldsymbol{\varepsilon}_p = \left(\tilde{\boldsymbol{C}}_b^e \cdot \boldsymbol{l}_{ant}^b\right) \times \phi_b^e + \delta \boldsymbol{p}_b^e \tag{2-11}$$

$$\tilde{\boldsymbol{v}}_b^e - \tilde{\boldsymbol{v}}_{ant}^e + \tilde{\boldsymbol{v}}_{trans}^e + \boldsymbol{\varepsilon}_v = \tilde{\boldsymbol{v}}_{trans}^e \times \delta\phi_b^e + \delta \boldsymbol{v}_b^e - \boldsymbol{C}_b^e \cdot \left(\boldsymbol{l}_{ant}^b \times\right) \cdot \delta\boldsymbol{\omega}_{ib}^b \tag{2-12}$$

with，

$$\tilde{\boldsymbol{v}}_{trans}^e = \left[\tilde{\boldsymbol{C}}_b^e \cdot \left(\tilde{\boldsymbol{\omega}}_{ib}^b \times\right) - \left(\tilde{\boldsymbol{\omega}}_{ie}^e \times\right) \cdot \tilde{\boldsymbol{C}}_b^e\right] \cdot l_{gnss}^b \tag{2-13}$$

where $\tilde{\boldsymbol{p}}_{gnss}^e$ and $\tilde{\boldsymbol{v}}_{gnss}^e$ are the GNSS-derived position and velocity; $\boldsymbol{\varepsilon}_p$ and $\boldsymbol{\varepsilon}_v$ are the corresponding measurement noises. The term $\tilde{\boldsymbol{v}}_{trans}^e$ is the velocity lever-arm compensation arising from the sensor-center offset.

2.2.4 Tightly coupled GNSS/INS measurement model

2.2.4.1 PPP/INS tight coupling

Taking the ionosphere-free combination as an example, the tightly coupled PPP/INS observation equations can be written as (2-14):

$$\begin{cases} v_{P,IF}^s = \boldsymbol{g}^s\left(\boldsymbol{l}^e \times\right) \cdot \boldsymbol{\phi} + \boldsymbol{g}^s \cdot \delta \boldsymbol{r}^e + c \cdot \delta t_r + mf^s \cdot \delta T_w + e_P^s \\ v_{L,IF}^s = \boldsymbol{g}^s\left(\boldsymbol{l}^e \times\right) \cdot \boldsymbol{\phi} + \boldsymbol{g}^s \cdot \delta \boldsymbol{r}^e + c \cdot \delta t_r + mf^s \cdot \delta T_w + \lambda_{IF} \cdot \delta N_{IF} + \varepsilon_L^s \end{cases} \tag{2-14}$$

where $v_{P,IF}^s$ and $v_{L,IF}^s$ are the ionosphere-free pseudorange and carrier-phase residuals (observed minus computed); δT_w is the zenith wet tropospheric delay; mf^s is the tropospheric mapping function; $\boldsymbol{l}^e = \boldsymbol{C}_b^e \cdot \boldsymbol{l}_{ant}^b$; c is the speed of light in vacuum; $\boldsymbol{g}^s$ is the satellite position; δt_r is the receiver clock offset; λ is the ionosphere-free wavelength; N is the (real-valued) carrier-phase ambiguity (in cycles);and e_P^s, ε_L^s are the pseudorange and phase noises. Because of hardware delays at both the satellite and receiver, the corresponding delays are absorbed by the carrier-phase ambiguities, which therefore lose their integer nature. The above gives the un-differenced PPP/SINS tight-coupling model for the ionosphere-free combination; the un-combined PPP model is analogous and is omitted here.

2.2.4.2 RTK/INS tight coupling

For classical RTK, the tightly coupled observables are double-difference pseudorange and carrier phase, yielding the measurement model (2-15):

$$\begin{cases} v_{P,DD} = \boldsymbol{g}^{m,n}\left(\boldsymbol{l}^{e}\times\right)\cdot\boldsymbol{\phi}+\boldsymbol{g}^{m,n}\cdot\boldsymbol{\delta r}+e_{DD} \\ v_{L,DD} = \boldsymbol{g}^{m,n}\left(\boldsymbol{l}^{e}\times\right)\cdot\boldsymbol{\phi}+\boldsymbol{g}^{m,n}\cdot\boldsymbol{\delta r}+\delta N_{DD}+\boldsymbol{\varepsilon}_{DD} \end{cases} \tag{2-15}$$

with symbols consistent with the foregoing. For medium- to long-baseline cases, residual atmospheric delays must also be included. The above is a tightly coupled model with explicit ambiguity parameters, i.e., the ambiguities are jointly estimated within the integrated filter.

In summary, the detailed GNSS/SINS tight-coupling workflow is as follows: after system initialization (including estimator initialization and INS initial alignment), the integrated filter enters the navigation phase. Raw IMU outputs are first corrected for sensor errors and non-commutative effects; attitude, velocity, and position are then updated in sequence, followed by state prediction using the INS error-propagation model. In parallel, raw GNSS observations are pre-processed with INS-aided information—e.g., cycle-slip detection and gross-error pre-screening. The INS position is then center-compensated (lever-arm) and used to build the measurement model and perform the measurement update. Post-fit quality control is applied to detect gross errors; measurement updates are iterated until outliers are removed. Finally, the state estimates are fed back in closed loop for correction.

2.3 Zero-velocity and nonholonomic constraint models

2.3.1 Zero-velocity constraint (ZUPT)

The zero-velocity constraint is a salient and highly effective source of reliable information. When a vehicle is stationary, the rate quantities that reflect changes in the vehicle state—velocity, angular rate, and specific force—can ideally be regarded as zero. Accordingly, constraint models can be constructed for different physical quantities. Taking the zero-velocity measurement model as an example:

$$\boldsymbol{0} = \boldsymbol{v}_{imu}^{e} \Rightarrow \tilde{\boldsymbol{v}}_{imu}^{e} + \boldsymbol{\varepsilon}_{v} = \delta \boldsymbol{v}_{imu}^{e} \tag{2.16}$$

where $\tilde{\boldsymbol{v}}_{imu}^{e}$ is the mechanized INS velocity, and $\boldsymbol{\varepsilon}_{v}$ is white measurement noise affected by vehicle vibration and sensor noise.

The crux of zero-velocity updating lies in detecting the vehicle's motion state. Because INS errors accumulate over time, the velocity from pure IMU mechanization alone is not sufficiently reliable for stationarity detection. External aiding is typically employed—for example, GNSS speed, wheel-odometer pulses, camera/LiDAR frame-to-frame matching. One may also use IMU data after bias correction and denoising to detect motion states. Once stationarity is accurately identified, the associated measurements exert very strong constraints and can be assigned small measurement-noise variances.

2.3.2 Nonholonomic constraint (NHC) model

Due to the kinematics of ground vehicles, motion is predominantly along the longitudinal axis; the lateral and vertical (normal) velocities in the vehicle frame vvv are effectively zero. This implicit information can be introduced into the integrated filter as virtual measurements. Because only the two transverse components of velocity are involved, this is termed a nonholonomic constraint. The measurement model (with quantities restricted to the lateral and normal components) is:

$$\begin{aligned}
&\boldsymbol{v}_{nhc}^{v}=\boldsymbol{C}_{b}^{v}\left[\boldsymbol{C}_{e}^{b}\cdot\boldsymbol{v}_{imu}^{e}+\left(\boldsymbol{\omega}_{ib}^{b}-\boldsymbol{\omega}_{ie}^{b}\right)\times\boldsymbol{l}_{v}^{b}\right]\\
&\Rightarrow\boldsymbol{C}_{b}^{v}\tilde{\boldsymbol{v}}_{imu}^{e}-\tilde{\boldsymbol{v}}_{nhc}^{v}+\boldsymbol{C}_{b}^{v}\left(\tilde{\boldsymbol{\omega}}_{eb}^{b}\times\boldsymbol{l}_{v}^{b}\right)+\boldsymbol{\varepsilon}=-\boldsymbol{C}_{b}^{v}\boldsymbol{M}\cdot\boldsymbol{\phi}+\boldsymbol{C}_{b}^{v}\tilde{\boldsymbol{C}}_{e}^{b}\cdot\boldsymbol{\delta v}_{imu}^{e}-\boldsymbol{C}_{b}^{v}\left(\boldsymbol{l}_{v}^{b}\times\right)\boldsymbol{\delta\omega}_{ib}^{b}
\end{aligned}\tag{2.17}$$

where $\boldsymbol{M}=\tilde{\boldsymbol{C}}_{e}^{b}\cdot\left(\tilde{\boldsymbol{v}}_{imu}^{e}\times\right)+\left(\boldsymbol{l}_{v}^{b}\times\right)\tilde{\boldsymbol{C}}_{e}^{b}(\tilde{\omega}_{ie}^{e}\times)$; $\boldsymbol{v}_{nhc}^{v}$ extracts the lateral and normal components in the vehicle frame; $\boldsymbol{l}_{v}^{b}$ is the position vector (lever arm) of the NHC reference point relative to the IMU, expressed in the IMU body frame—the reference point is the hub center of a non-steering wheel; $\boldsymbol{C}_{b}^{v}$ is the rotation matrix from body to vehicle frame. Other symbols are as defined earlier.

Note that IMU installation misalignment can induce significant lateral and normal velocity components. Suppose the vehicle travels at 10 m/s and the IMU has a 5 deg yaw misalignment and a 2 deg pitch misalignment; then the induced lateral and normal speed components are approximately 0.87 m/s and 0.34 m/s, respectively, rendering the NHC unusable. Therefore, IMU installation biases should be calibrated in advance.

2.4 Wheel-odometer (ODO) observation model

Most ground vehicles are equipped with wheel-speed sensors accessible via the CAN bus, or with external wheel odometers whose pulses can be converted to forward speed. When converting pulses to speed, the wheel radius must be introduced. Because wheels deform to some extent during driving, this deformation is commonly modeled as a radius-related scale-factor error, typically treated as a random-walk process. Considering these factors, the wheel-odometer measurement model (restricted to the longitudinal component) can be written as:

$$\begin{aligned}
&\left(1+s\right)v_{odo}^{v}=\boldsymbol{C}_{e}^{b}\cdot\boldsymbol{v}_{imu}^{e}+\left(\boldsymbol{\omega}_{ib}^{b}-\boldsymbol{\omega}_{ie}^{b}\right)\times\boldsymbol{l}_{v}^{b}\\
&\Rightarrow\tilde{\boldsymbol{v}}_{imu}^{b}-\left(1+\tilde{s}\right)\tilde{v}_{odo}^{v}+\tilde{\boldsymbol{\omega}}_{eb}^{b}\times\boldsymbol{l}_{v}^{b}+\varepsilon=-\boldsymbol{M}\cdot\boldsymbol{\phi}+\tilde{\boldsymbol{C}}_{e}^{b}\cdot\boldsymbol{\delta v}_{imu}^{e}\\
&-\tilde{v}_{odo}^{v}\cdot\boldsymbol{\delta}s-\boldsymbol{l}_{v}^{b}\times\boldsymbol{\delta\omega}_{ib}^{b}
\end{aligned}\tag{2.18}$$

where v_{odo}^{v} is the forward speed obtained from pulse conversion; *s* is the scale-factor parameter in the wheel-speed model; and other symbols follow the preceding definitions. External wheel odometers are usually mounted on non-steering wheels, so their reference point coincides with that used for the NHC—the hub center of a non-steering wheel. The two models are thus consistent, except that the wheel-speed constraint explicitly includes the scale-factor parameter. The scale factor typically converges rapidly when the vehicle speed varies; installation angles can be viewed as multiplicative factors (the projection of the forward speed into the IMU body frame) and can therefore be estimated jointly with the scale factor.

Chapter 3 System Requirements and License

3.1 System requirements

The Windows CUI (command-line user interface) executable included in the installer was built with Microsoft Visual Studio (VS) on Windows 11 (64-bit). The folder contains all required dynamic-link libraries (DLLs). In addition, the CUI app and Linux shared libraries were built and tested on CentOS Linux release 7.7.1908 with x64 CPUs. For Macintosh, the CUI app and dynamic libraries were built on macOS 10.15.3 using AppleClang 11.0.3.11030032 with kernel version Darwin 19.3.0.

Users may also build executable binaries on their own operating systems (Windows, Linux, or macOS) using CMake, an open-source, cross-platform build system.

3.2 License

GREAT-MSF is open-source software distributed under the GNU General Public License, version 3 (GPL-3) (https://www.gnu.org/licenses/gpl-3.0.html).

3.3 Copyright

Developers:

- GREAT Team, Wuhan University

Third-party libraries:

- GREAT-MSF uses the G-Nut library (http://www.pecny.cz/) — Copyright (C) 2011–2016 GOP – Geodetic Observatory Pecny, RIGTC
- GREAT-MSF uses the pugixml library (http://pugixml.org) — Copyright (C) 2006–2014 Arseny Kapoulkine
- GREAT-MSF uses the Newmat library (http://www.robertnz.net/nm_intro.htm) — Copyright (C) 2008 R. B. Davies

- GREAT-MSF uses the spdlog library (https://github.com/gabime/spdlog) — Copyright (c) 2015–present, Gabi Melman & spdlog contributors
- GREAT-MSF uses the GLFW library (https://www.glfw.org/) — Copyright (C) 2002–2006 Marcus Geelnard; Copyright (C) 2006–2019 Camilla Löwy
- GREAT-MSF uses the Eigen library (https://eigen.tuxfamily.org) — Copyright (C) 2008–2011 Gael Guennebaud
- GREAT-MSF uses the PSINS library (https://psins.org.cn) — Copyright (c) 2015–2025 Gongmin Yan

Chapter 4 Building and Installing on Windows

The software package is available at https://github.com/GREAT-WHU/GREAT-MSF.

You can clone it via Git, or directly download the archive **GREAT-MSF_<ver>.zip**.

The **GREAT-MSF** directory structure is as follows.

Table 4.1 Directory structure of GREAT-MSF

Directory/File	Subdirectory/File	Description
./src		Source code
	./app	Main programs of GREAT-MSF and GREAT-PVT
	./LibGREAT	Core algorithms library for GNSS and multi-sensor fusion
	./LibGnut	Gnut library
	./third-party	Third-party libraries
	CMakeLists.txt	CMake Lists file
./ sample_data		Example datasets
	./MSF_20201027	GNSS challenge-scenario example
	./MSF_20201029	GNSS open-sky scenario example
	./ MSF_20211013	GNSS challenge-scenario example
	./ MSF_20211012	Example with wheel-odometer measurements
./plot		Plotting scripts
./doc		Documentation files
	GREAT-MSF_1.0.pdf	User Guide

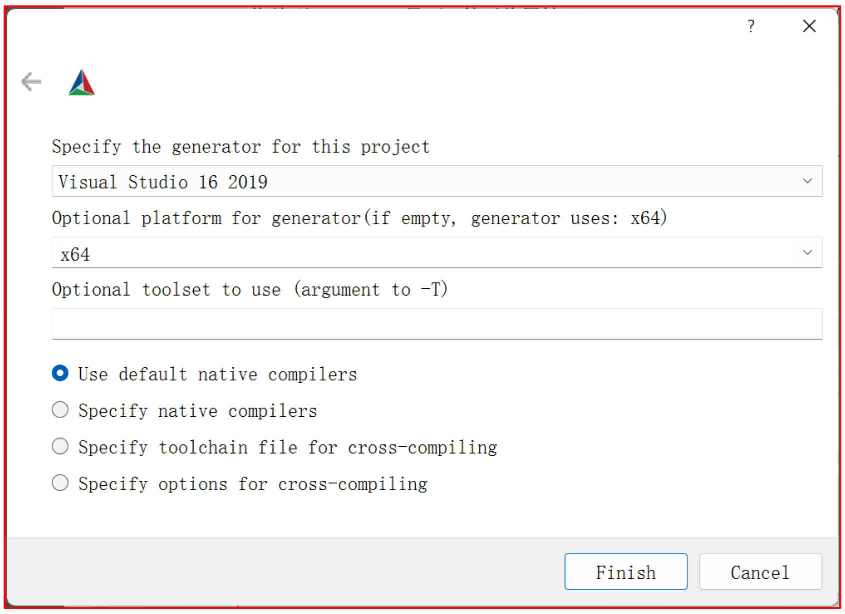
The following steps show how to build the GREAT-MSF executables on Windows.

(1) Obtain and install CMake from https://cmake.org/download/. **Note:** the minimum required CMake version is **3.0.0**.

(2) Launch **cmake-gui**.

(3) Click **Browse Source…**, then select the directory <install_dir>/GREAT-MSF_<ver>/src, or drag and drop the CMakeLists.txt from <install_dir>/GREAT-MSF_<ver>/src into the CMake GUI. Then set **Where to build the binaries** to <install_dir>/GREAT-MSF_<ver>/src/build.

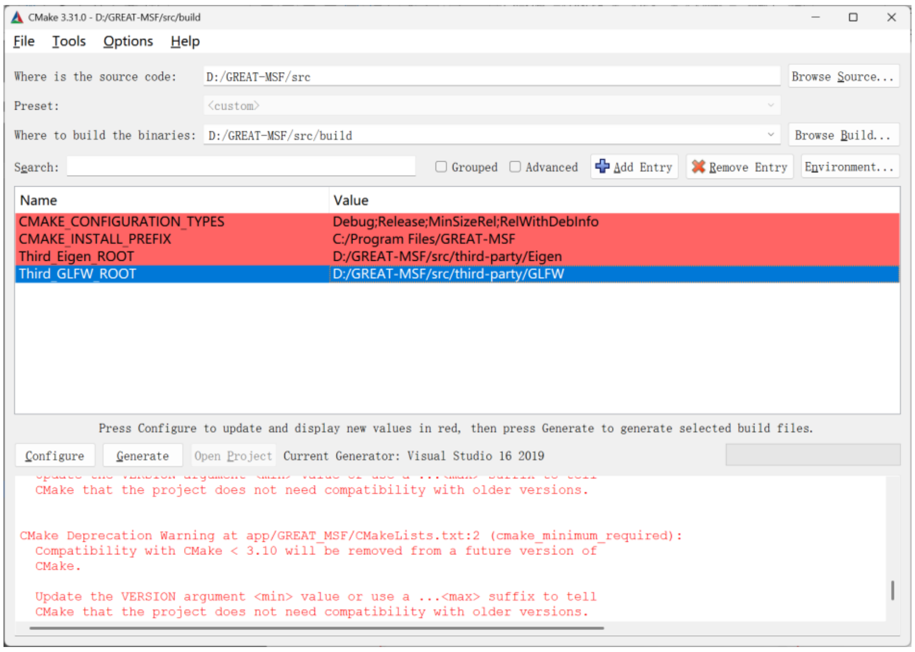
(4) Click **Configure** and select an Integrated Development Environment (IDE) for the project (this dialog appears only the first time you click **Configure**).


Specify the generator for this project
Visual Studio 16 2019
Optional platform for generator(if empty, generator uses: x64)
x64
Optional toolset to use (argument to -T)

(● Use default native compilers)
Specify native compilers
Specify toolchain file for cross-compiling
Specify options for cross-compiling

Figure 4.1 Example of IDE selection

(5) Configure paths to third-party libraries.



Where is the source code: D:/GREAT-MSF/src

Preset: <custom>

Where to build the binaries: D:/GREAT-MSF/src/build

Search:

Name	Value
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX	C:/Program Files/GREAT-MSF
Third_Eigen_ROOT	D:/GREAT-MSF/src/third-party/Eigen
Third_GLFW_ROOT	D:/GREAT-MSF/src/third-party/GLFW

Press Configure to update and display new values in red, then press Generate to generate selected build files.

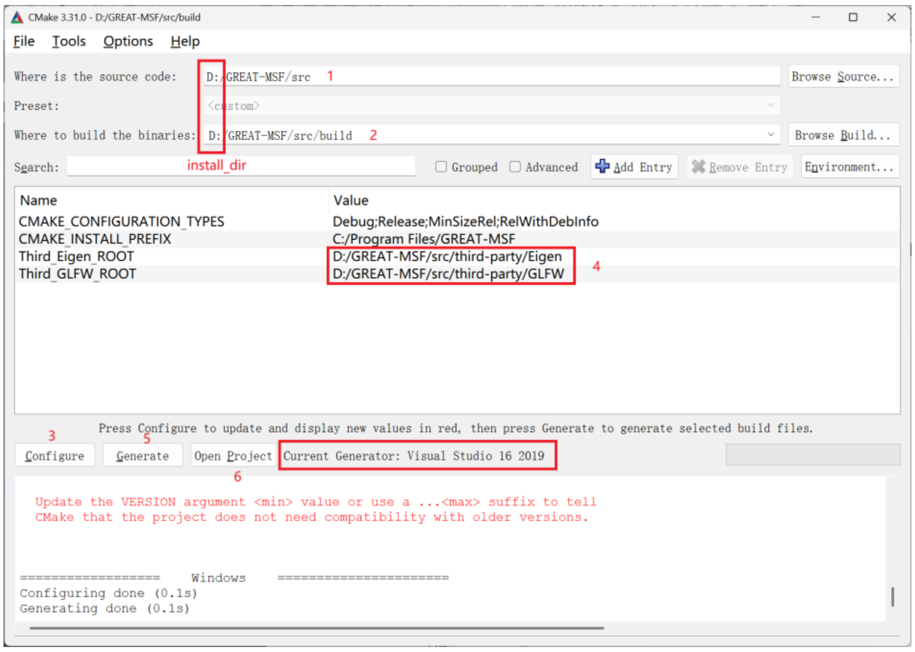
Current Generator: Visual Studio 16 2019

```
  CMake that the project does not need compatibility with older versions.


CMake Deprecation Warning at app/GREAT_MSF/CMakeLists.txt:2 (cmake_minimum_required):
  Compatibility with CMake < 3.10 will be removed from a future version of
  CMake.

  Update the VERSION argument <min> value or use a ...<max> suffix to tell
  CMake that the project does not need compatibility with older versions.
```

Figure 4.2 Configuring third-party library paths

(6) Click **Generate** to write the build files into <install_dir>/GREAT-MSF_<ver>/src/build.

(7) Click **Open Project**, then compile the source code in the chosen IDE.



Where is the source code: D:/GREAT-MSF/src

Preset: <custom>

Where to build the binaries: D:/GREAT-MSF/src/build

Search:

Name	Value
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX	C:/Program Files/GREAT-MSF
Third_Eigen_ROOT	D:/GREAT-MSF/src/third-party/Eigen
Third_GLFW_ROOT	D:/GREAT-MSF/src/third-party/GLFW

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 16 2019

```
  Update the VERSION argument <min> value or use a ...<max> suffix to tell
  CMake that the project does not need compatibility with older versions.



==================      Windows     ======================
Configuring done (0.1s)
Generating done (0.1s)
```

Figure 4.3 Building GREAT-MSF on Windows

Chapter 5 Example Data Processing

5.1 Example file directory structure

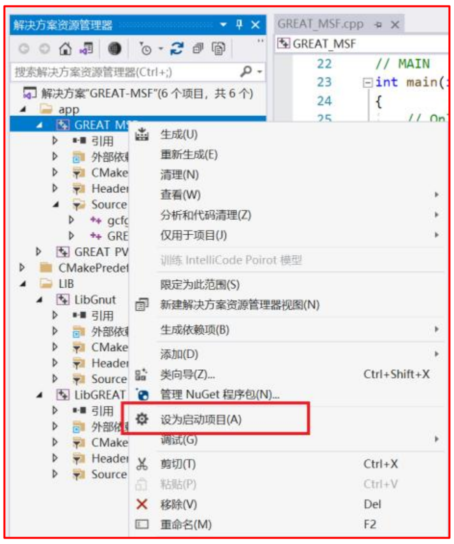
The example datasets are located in <install_dir>/ GREAT-MSF_<ver>/ sample_data. Several example datasets are provided: **MSF_20201029** (GNSS open-sky scenario), **MSF_20201027** and **MSF_20211013** (GNSS challenge scenarios). In addition, we provide **MSF_20211012**, an example that includes odometer data. For each dataset, XML configuration files are available for PPP/INS and RTK/INS in both loose and tight coupling modes. See the appendix for details on the configuration files. The directory structure of the example-data folder is as follows:

Table 5.1 Directory structure of the example datasets

目录/文件	说明
./GNSS	GNSS data folder containing base/rover observation folders and corresponding product folders (ephemerides, clocks, etc.)
./IMU	IMU data folder
./model	System-model files
./xml	XML configuration files
./ref	Reference results computed by the IE solver
./result	Output results

5.2 Parameter configuration and execution in Visual Studio

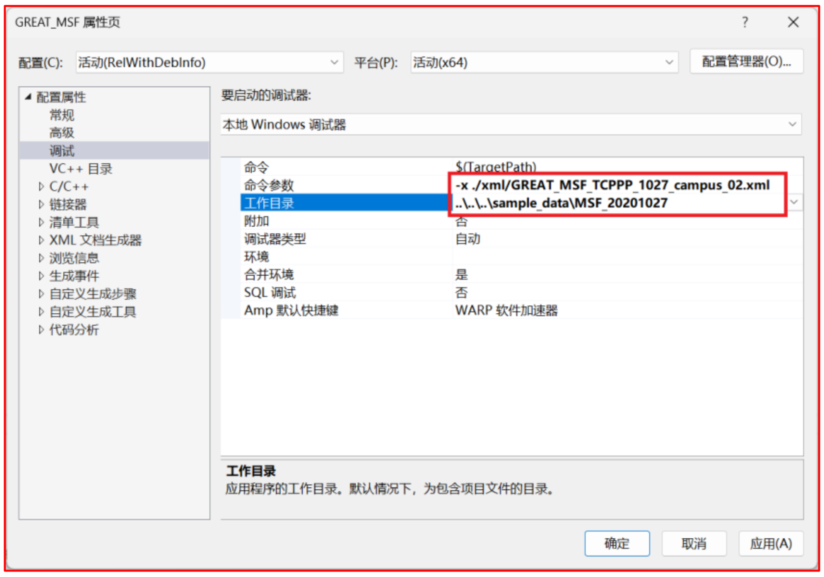
（1） After a successful build, open the app folder in the Visual Studio Solution Explorer. Right-click **GREAT_MSF** and set it as the **Startup Project**.



```
// MAIN
int main(
{
```

Figure 5.1 Setting **GREAT_MSF** as the startup project in VS

（2） Right-click **GREAT_MSF** and open **Properties**. Set **Configuration** to **RelWithDebInfo**. Set the example’s directory as the **Working Directory**, and specify the **Command Arguments**.


GREAT_MSF 属性页

配置(C): 活动(RelWithDebInfo)　平台(P): 活动(x64)

要启动的调试器: 本地 Windows 调试器

命令	$(TargetPath)
命令参数	-x ./xml/GREAT_MSF_TCPPP_1027_campus_02.xml
工作目录	..\..\sample_data\MSF_20201027
附加	否
调试器类型	自动
环境	
合并环境	是
SQL 调试	否
Amp 默认快捷键	WARP 软件加速器

工作目录
应用程序的工作目录。默认情况下，为包含项目文件的目录。

Figure 5.2 Example of solver command arguments and working-directory settings

（3） Run under **RelWithDebInfo**.

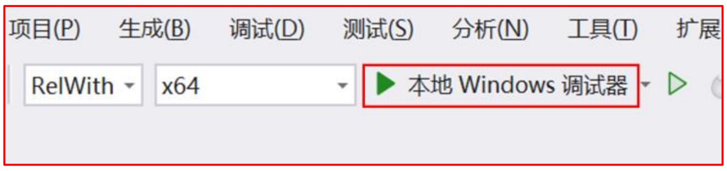
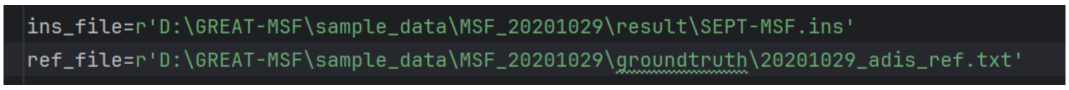


Figure 5.3Running GREAT-MSF

5.3 Result evaluation with plotting scripts

GREAT-MSF provides Python scripts to plot and analyze PPP and RTK solutions. These scripts are located in the plot folder. The main script is **Evaluation.py**. After the program produces results, modify the paths in the script as shown and run it.



```
ins_file=r'D:\GREAT-MSF\sample_data\MSF_20201029\result\SEPT-MSF.ins'
ref_file=r'D:\GREAT-MSF\sample_data\MSF_20201029\groundtruth\20201029_adis_ref.txt'
```

Figure 5.4 Editing paths in Evaluation.py

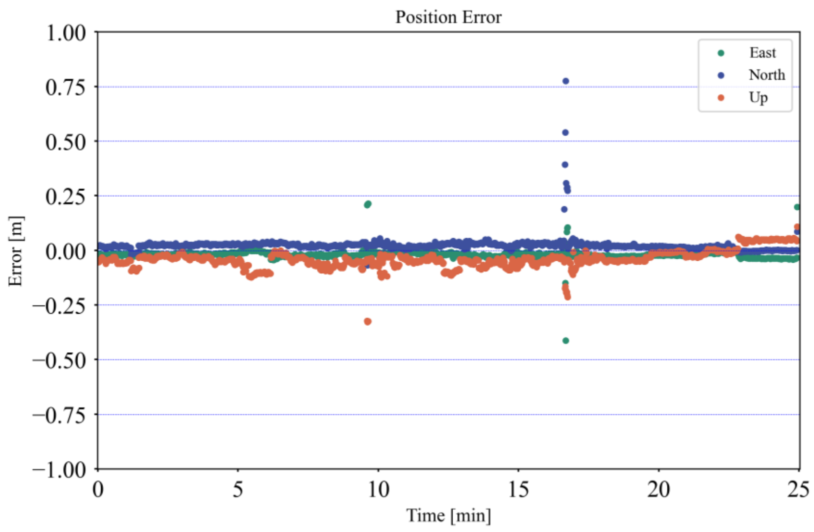


Figure 5.5 Plotting results using Evaluation.py (a)

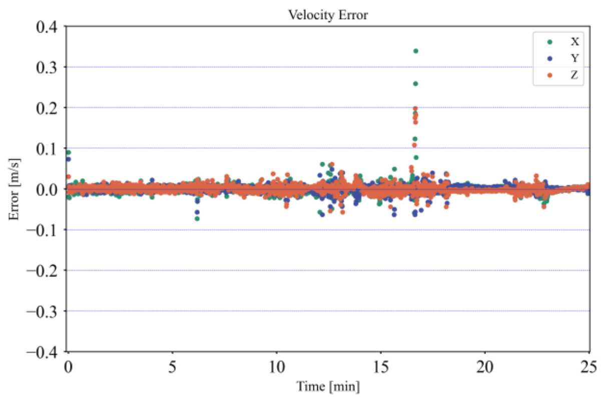


Figure 5.6 Plotting results using Evaluation.py (b)

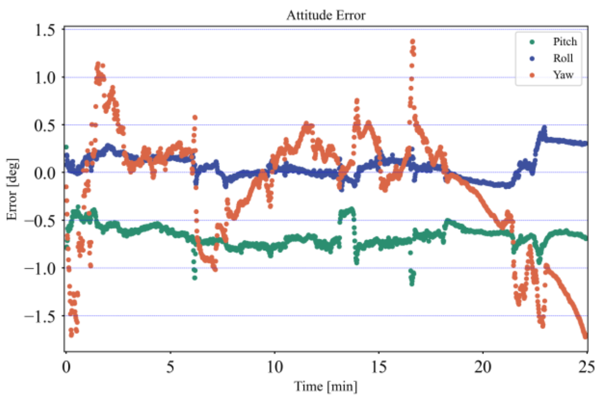


Figure 5.7 Plotting results using Evaluation.py (c)

5.4 Enabling motion constraints or odometer in the configuration file

In the new version, the NHC, ZUPT, and ODO measurement models have been updated. This section demonstrates how to configure the XML to properly enable these measurements.

5.4.1 Enabling the nonholonomic constraint (NHC)

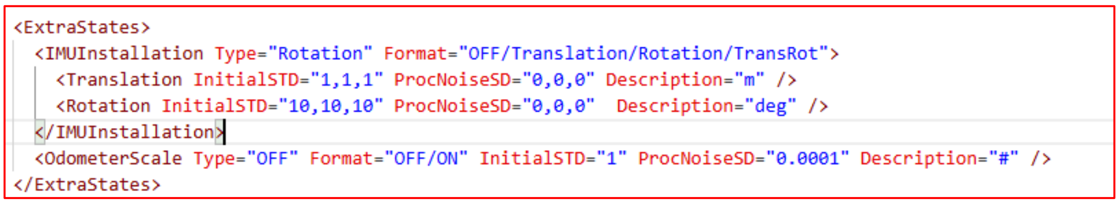
As noted in §2.3.2, IMU installation angles can induce sizable lateral and normal velocity components. Therefore, to enable NHC you must either pre-calibrate IMU installation biases or estimate them as states within the filter. Enable the Installation node under <ins> by setting it to ON.

If the installation angles have been calibrated in advance, set them under the Rotation node below; the lever arm has minor impact on NHC.

```
<Installation Type="ON" Format="OFF/ON" Description="Take the left rear wheel as the origin">
  <Rotation Pitch="0.9231" Roll="0" Yaw="0.0205"  Description="deg" />
  <Lever X="0" Y="0"  Z="0" Description="m" />
</Installation>
```

Figure 5.8 Setting initial installation angles

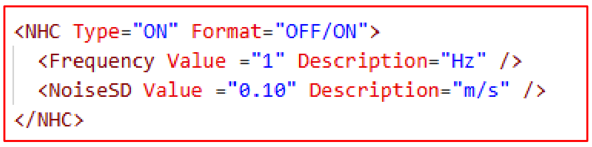
Alternatively, to estimate installation angles within the filter, set ExtraStates-IMUInstallation under the <integration> node to Rotation, and set reasonable initial variances and process noises in the Rotation node.



```xml
<ExtraStates>
  <IMUInstallation Type="Rotation" Format="OFF/Translation/Rotation/TransRot">
    <Translation InitialSTD="1,1,1" ProcNoiseSD="0,0,0" Description="m" />
    <Rotation InitialSTD="10,10,10" ProcNoiseSD="0,0,0"  Description="deg" />
  </IMUInstallation>
  <OdometerScale Type="OFF" Format="OFF/ON" InitialSTD="1" ProcNoiseSD="0.0001" Description="#" />
</ExtraStates>
```

Figure 5.9 Estimating installation angles as filter states

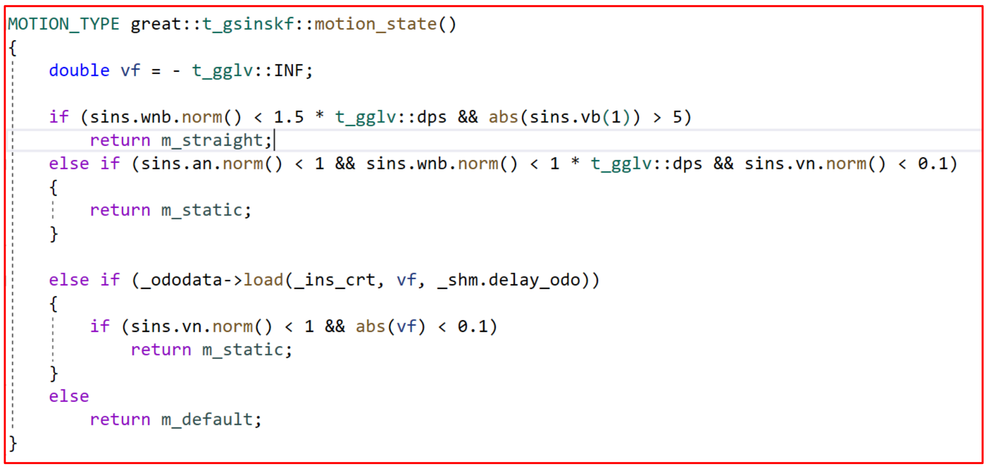
Finally, enable <NHC> under <integration> and set appropriate update frequency and noise parameters.



```xml
<NHC Type="ON" Format="OFF/ON">
  <Frequency Value ="1" Description="Hz" />
  <NoiseSD Value ="0.10" Description="m/s" />
</NHC>
```

Figure 5.10 Enabling the NHC node

The thresholds for straight-line/stationary detection are located in the motion_state function.

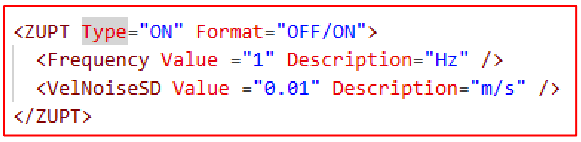


```cpp
MOTION_TYPE great::t_gsinskf::motion_state()
{
    double vf = - t_gglv::INF;

    if (sins.wnb.norm() < 1.5 * t_gglv::dps && abs(sins.vb(1)) > 5)
        return m_straight;
    else if (sins.an.norm() < 1 && sins.wnb.norm() < 1 * t_gglv::dps && sins.vn.norm() < 0.1)
    {
        return m_static;
    }

    else if (_ododata->load(_ins_crt, vf, _shm.delay_odo))
    {
        if (sins.vn.norm() < 1 && abs(vf) < 0.1)
            return m_static;
    }
    else
        return m_default;
}
```

Figure 5.11 In function motion_state, determine straight-line/stationary based on the thresholds therein

5.4.2 Enabling the zero-velocity constraint (ZUPT)

Enable <ZUPT> under <integration> and set appropriate update frequency and noise parameters. The detection thresholds are as in Figure 5.11.

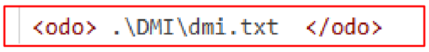


```
<ZUPT Type="ON" Format="OFF/ON">
  <Frequency Value ="1" Description="Hz" />
  <VelNoiseSD Value ="0.01" Description="m/s" />
</ZUPT>
```

Figure 5.12 Enabling the zero-velocity constraint

5.4.3 Enabling the wheel odometer (ODO)

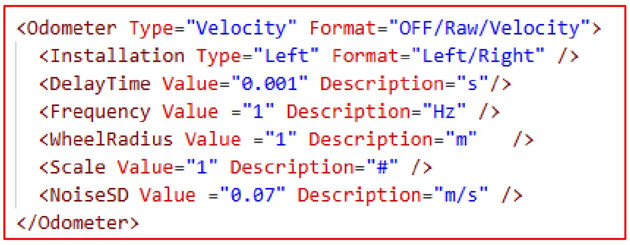
First, provide the odometer file under <inputs>:



```
<odo> .\DMI\dmi.txt  </odo>
```

Figure 5.13 Supplying the ODO file

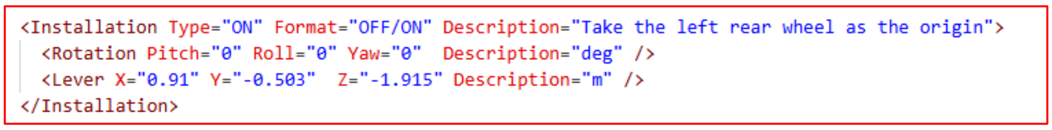
Enable the odometer under the <integration> node and set it to Velocity. For other formats, modify the decoder in odofile.cpp, function t_odofile::decode_data. See the appendix for the meaning of the nodes shown below.



```
<Odometer Type="Velocity" Format="OFF/Raw/Velocity">
  <Installation Type="Left" Format="Left/Right" />
  <DelayTime Value="0.001" Description="s"/>
  <Frequency Value ="1" Description="Hz" />
  <WheelRadius Value ="1" Description="m"   />
  <Scale Value="1" Description="#" />
  <NoiseSD Value ="0.07" Description="m/s" />
</Odometer>
```

Figure 5.13 Odometer node

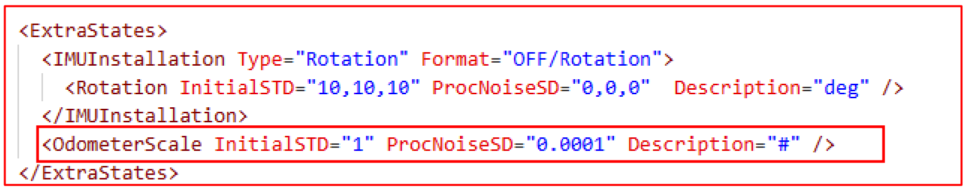
Under <ins> → <Installation>, set the lever arm between the odometer and the IMU, with X Y Z denoting right-forward-up.



```
<Installation Type="ON" Format="OFF/ON" Description="Take the left rear wheel as the origin">
  <Rotation Pitch="0" Roll="0" Yaw="0"  Description="deg" />
  <Lever X="0.91" Y="-0.503"  Z="-1.915" Description="m" />
</Installation>
```

Figure 5.14 Setting the odometer-to-IMU lever arm

In addition, when using ODO measurements it is usually necessary to estimate the scale-factor error. Set reasonable initial variances and process noise for the OdometerScale state under the <ExtraStates> node.



```
<ExtraStates>
  <IMUInstallation Type="Rotation" Format="OFF/Rotation">
    <Rotation InitialSTD="10,10,10" ProcNoiseSD="0,0,0"  Description="deg" />
  </IMUInstallation>
  <OdometerScale InitialSTD="1" ProcNoiseSD="0.0001" Description="#" />
</ExtraStates>
```

Figure 5.14 Setting the initial variance and process noise for the scale factor under <ExtraStates>

Chapter 6 Appendix

6.1 XML configuration file description

The configuration file is in XML format and contains settings for input/output files, processing schemes, and solution strategies. Any part of a line starting with "<!--" and ending with "-->" is a comment. The table below shows the XML format used by GREAT-MSF. The <ins> and <integration> nodes are newly added; all other meanings are identical to those in the GREAT-PVT XML configuration.

Table 6.1 XML configuration for GREAT-MSF

Item	Description	tag / attribute
Basic settings (level-1 node)		**<gen>**
Start time	GPS time at start, format "YYYY-MM-DD hh:mm:ss"	<beg>
End time	GPS time at end, format "YYYY-MM-DD hh:mm:ss"	<end>
Satellite systems	GNSS constellations to be used	<sys>
Station list	Stations to be processed (4-character codes)	<rec>
Sampling interval	Observation sampling interval	<int>
Estimation method	Default is filtering	<est>
Rover name	Rover and base names; used only in RTK/INS mode—must NOT be set in PPP/INS	<rover>
Base name		<base>
Input files (level-1 node)		**<inputs>**

Item	Description	tag / attribute
RINEX observation files	Files used for processing. Supports RINEX 2.10, 2.11, 2.12, 3.00–3.04. The program auto-detects by rover/base names; the first 4 characters of filenames must match the station codes.	<rinexo>
RINEX broadcast ephemerides	Supports RINEX 2.10, 2.11, 2.12, 3.00 – 3.04	<rinexn>
Precise clocks	Precise clock products; optional in RTK/INS	<rinexc>
Precise ephemerides	Precise orbit products; optional in RTK/INS	<sp3>
Antenna file	Satellite/receiver antenna phase-center corrections	<atx>
Ocean tide file	Ocean tide loading corrections	<blq>
Planetary ephemerides	Used to compute planetary parameters	<de>
EOP file	Earth orientation parameters for rotation matrices	<eop>
IMU file	IMU data file (format defined under <ins> → <DataFormat>)	<imu>
Output files (level-1 node)		**<outputs>**
Log file	Processing log output	<log>
GNSS results	Satellite positioning results	<flt>
MSF results	Integrated navigation results	<ins>
Trajectory file	KML trajectory for visualization in Google Earth, etc.	<kml>
Processing settings (level-1 node)		**<process>**

Item	Description	tag / attribute
Phase observations	Use carrier-phase observations: true/false	<phase>
Troposphere parameters	Estimate troposphere: true/false	<tropo>
Ionosphere parameters	Estimate ionosphere: true/false	<iono>
Doppler observations	Use Doppler observations: true/false	<doppler>
Troposphere model	Selected tropospheric model	<tropo_model>
A-priori sigmas of estimated parameters	Site coordinates	<sig_init_crd>
	Site velocity	<sig_init_vel>
	Troposphere	<sig_init_ztd>
	Ambiguities	<sig_init_amb>
	Galileo ISB/IFB	<sig_init_gal>
	GLONASS ISB/IFB	<sig_init_glo>
	BDS ISB/IFB	<sig_init_bds>
	Ionosphere	<sig_init_vion>
Elevation cutoff	Minimum elevation angle for usable observations	<minimum_elev >
Observation combination	Observation-combination mode for processing: IONO_FREE (ionosphere-free, dual-frequency); RAW_ALL (un-differenced, un-combined; dual/multi-frequency)	<obs_combination>
Max post-fit residual	Threshold for residual editing	<max_res_norm >

Item	Description	tag / attribute
Kinematic mode	Dynamic processing: true/false	<pos_kin>
Minimum satellites	Minimum number of satellites used	<min_sat>
Observation weighting	Method for observation weights	<obs_weight>
BDS code bias	Correct BDS code biases: true/false	<bds_code_bias_corr>
Cycle-slip detection	Slip-detection model (default)	<slip_model>
Frequencies	Number of observation frequencies used	<frequency>
Filter settings (level-1 node)		**<filter>**
Filtering algorithm	srcf (square-root cubature Kalman filter) / kalman (Kalman filter)	method_flt
A-priori process noises	Coordinate white noise	noise_crd
	Velocity white noise	noise_vel
	Receiver clock-rate white noise	noise_dclk
	Receiver clock white noise	noise_clk
	Ionosphere white noise	noise_vion
	Troposphere random walk	rndwk_ztd
	Ambiguity random walk	rndwk_amb
	GLONASS ISB/IFB random walk	rndwk_glo
	Galileo ISB/IFB random walk	rndwk_gal
	BDS ISB/IFB random walk	rndwk_bds
	GPS IFB random walk	rndwk_gps
Ambiguity fixing (level-1 node)		**<ambiguity>**
Fixing mode	NO (no fixing) / SEARCH (fix ambiguities)	<fix_mode>
UPD mode	Use UPD products for fixing	<upd_mode>

Item	Description	tag / attribute
Partial fixing	NO / YES	<part_fix>
Min number for partial fixing	Minimum ambiguities to fix in partial-fix mode	<part_fix_num>
Ratio	Ratio threshold in LAMBDA test	<ratio>
Reference satellite	Configure reference satellite: NO/YES	<set_refsat>
Minimum common time	Minimum overlap time for fixing on the same satellite	<min_common_time>
Extra wide-lane decisions	Parameters for different combinations: alpha&maxdev (confidence), maxsig (max sigma)	<extra_widelane_decision>
Wide-lane		<widelane_decision>
Narrow-lane		<narrowlane_decision>
Satellite settings (level-1 nodes)		**<gps>/<bds>/<gal>/<glo>**
A-priori sigma of observations	Code	sigma_C
	Carrier phase	sigma_L
Frequencies	Satellite frequencies (bands), values 1/2/3/4/5	<freq>
Satellites	PRN list	<sat>

Bands	Band mapping by system:
	GPS 1→L1, 2→L2, 5→L5;
	GAL 1→E1, 5→E5a, 7→E5b, 8→E5, 6→E6;
	BDS 2→B1I, 7→B2I, 6→B3I, 1→B1C, 5→B2a, 9→B2b, 8→B2a+b;
	GLO 1→G1, 2→G2
		<band>

Item	Description	tag / attribute
INS settings (level-1 node)		**<ins>**
Processing interval	Start / End (seconds of week)	<ProcTime>
Data format	AxisOrder (garfu/gaflu/gafrd, with g=gyro, a=accel; r right, f forward, u up), GyroUnit (DPS/RPS/RAD/DPH/RPH), AcceUnit (MPS/MPS2), Frequency (Hz)	<DataFormat>
Alignment	OFF / STATIC / POS (GNSS position-vector aided) / VEL (GNSS velocity-vector aided)	<Alignment>
Initial states	Position (OFF/Cartesian/Geodetic; Cartesian in meters), Velocity (OFF/Cartesian; m/s), Attitude (OFF/ON; deg), GyroBias (deg/h), AcceBias (mg)	<InitialStates>
Installation parameters	Type (enable b-to-v installation), Rotation (deg), Lever (m)	<Installation>
Integration settings (level-1 node)		**<integration>**
GNSS	Type (OFF/LCI/TCI); AntennaLever (Type: RFU in IMU frame; set X,Y,Z); DelayTime (max GNSS–IMU time offset for a measurement epoch); LCISetting (for loose coupling: MinSat, MaxPDOP, MaxNorm)	<GNSS>
Estimator	IMUErrorModel (ADIS16470 / StarNeto GI7660 / Customize); Attitude/Velocity/Position/GyroBias/AcceBias each with InitialSTD and ProcNoiseSD (units as noted)	<Estimator>

Item	Description	tag / attribute
Odometer	Type (OFF/Raw/Velocity; for Velocity, format is “SOW + forward speed”), Installation (left/right wheel), DelayTime, Frequency, WheelRadius (not needed when Type=Velocity), Scale (initial scale factor), NoiseSD (m/s)	<Odometer>
Nonholonomic constraint	Type (OFF/ON), Frequency (Hz), NoiseSD (m/s)	<NHC>
Zero-velocity constraint	Type (OFF/ON), Frequency (Hz), VelNoiseSD (m/s)	<ZUPT>
State augmentation	IMUInstallation (Rotation to estimate installation angles in the filter, or OFF; set initial variances and process noises inside), OdometerScale (initial variance and process noise for the odo scale factor in state prediction)	<ExtraStates>

Examples of the level-1 nodes <ins> and <integration> are shown below; for other examples refer to **GREAT-PVT_1.0** or the provided XML files.

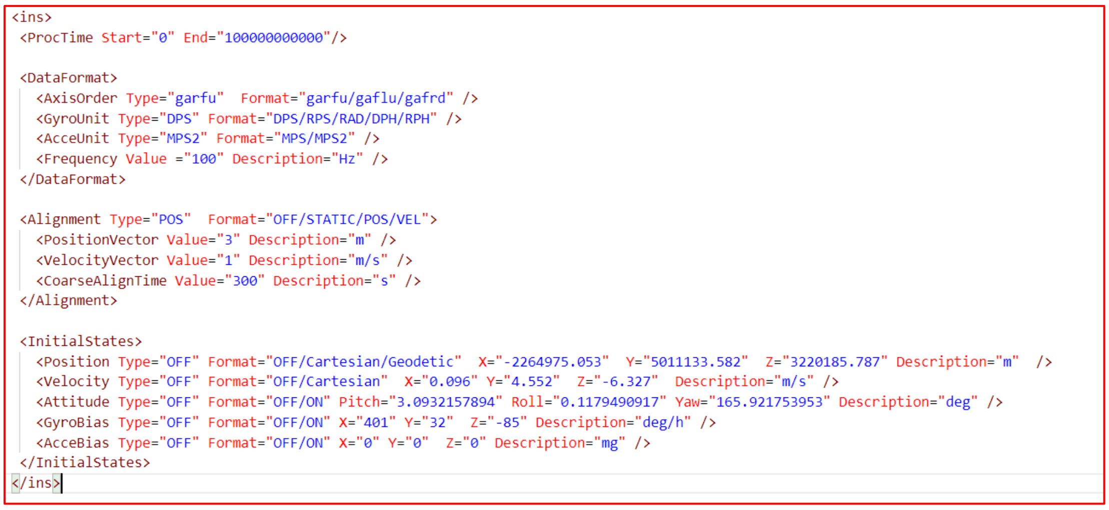

```
<ins>
 <ProcTime Start="0" End="100000000000"/>

 <DataFormat>
   <AxisOrder Type="garfu"  Format="garfu/gaflu/gafrd" />
   <GyroUnit Type="DPS" Format="DPS/RPS/RAD/DPH/RPH" />
   <AcceUnit Type="MPS2" Format="MPS/MPS2" />
   <Frequency Value ="100" Description="Hz" />
 </DataFormat>

 <Alignment Type="POS"  Format="OFF/STATIC/POS/VEL">
   <PositionVector Value="3" Description="m" />
   <VelocityVector Value="1" Description="m/s" />
   <CoarseAlignTime Value="300" Description="s" />
 </Alignment>

 <InitialStates>
   <Position Type="OFF" Format="OFF/Cartesian/Geodetic"  X="-2264975.053"  Y="5011133.582"  Z="3220185.787" Description="m"  />
   <Velocity Type="OFF" Format="OFF/Cartesian"  X="0.096" Y="4.552"  Z="-6.327"  Description="m/s" />
   <Attitude Type="OFF" Format="OFF/ON" Pitch="3.0932157894" Roll="0.1179490917" Yaw="165.921753953" Description="deg" />
   <GyroBias Type="OFF" Format="OFF/ON" X="401" Y="32"  Z="-85" Description="deg/h" />
   <AcceBias Type="OFF" Format="OFF/ON" X="0" Y="0"  Z="0" Description="mg" />
 </InitialStates>
</ins>
```

Figure 6.1 Example of <ins> node (a)

```
<Installation Type="ON" Format="OFF/ON" Description="Take the left rear wheel as the origin">
  <Rotation Pitch="0" Roll="0" Yaw="0"  Description="deg" />
  <Lever X="0.91" Y="-0.503"  Z="-1.915" Description="m" />
</Installation>
```

Figure 6.2 Example of <ins> node (b)

```
  <integration>
    <GNSS Type="LCI" Format="OFF/LCI/TCI">
        <AntennaLever Type="RFU" Format="RFU" X="-0.05" Y="-0.41" Z="0.15" Description="m" />
        <DelayTime Value="0.01" Description="s"/>
        <LCISetting MinSat="5" MaxPDOP="6" MaxNorm="15"/>

    </GNSS>

    <Estimator >
        <IMUErrorModel Type="ADIS 16470"   Format="Customize/ADIS 16470/StarNeto"   />
        <Attitude InitialSTD="1,1,10" ProcNoiseSD="1,3,3" Description="deg" />
        <Velocity InitialSTD="1,1,1" ProcNoiseSD="1,1,1" Description="m/s" />
        <Position InitialSTD="10,10,10" ProcNoiseSD="0,0,0" Description="m" />
        <GyroBias InitialSTD="100,100,100" ProcNoiseSD="0,0,0" Description="deg/h" />
        <AcceBias InitialSTD="10,10,10" ProcNoiseSD="0,0,0" Description="mg" />
    </Estimator>
</integration>
```

Figure 6.3 Example of <integration> node (a)

```
<Odometer Type="Velocity" Format="OFF/Raw/Velocity">
  <Installation Type="Left" Format="Left/Right" />
  <DelayTime Value="0.001" Description="s"/>
  <Frequency Value ="1" Description="Hz" />
  <WheelRadius Value ="1" Description="m"   />
  <Scale Value="1" Description="#" />
  <NoiseSD Value ="0.07" Description="m/s" />
</Odometer>

<NHC Type="ON" Format="OFF/ON">
  <Frequency Value ="1" Description="Hz" />
  <NoiseSD Value ="0.05" Description="m/s" />
</NHC>

<ZUPT Type="ON" Format="OFF/ON">
  <Frequency Value ="1" Description="Hz" />
  <VelNoiseSD Value ="0.01" Description="m/s" />
</ZUPT>

<ExtraStates>
  <IMUInstallation Type="Rotation" Format="OFF/Rotation">
    <Rotation InitialSTD="10,10,10" ProcNoiseSD="0,0,0"  Description="deg" />
  </IMUInstallation>
  <OdometerScale InitialSTD="1" ProcNoiseSD="0.0001" Description="#" />
</ExtraStates>
```

Figure 6.4 Example of <integration> node (b)

6.2 Result file (**.ins) description

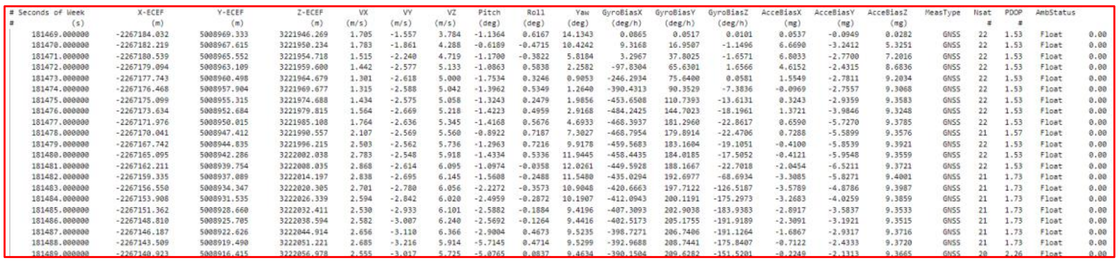
The flt file records PPP or RTK positioning solutions and their precision indicators, as detailed below.

Table 6.2 GREAT-MSF result file description

Col.	Column name (1–19)	Description	Unit	Format
1	Seconds of Week	GPS seconds of week	s	F18.6
2	X-ECEF	ECEF X coordinate	m	F18.3
3	Y-ECEF	ECEF Y coordinate	m	F18.3
4	Z-ECEF	ECEF Z coordinate	m	F18.3
5	Vx-ECEF	ECEF X velocity	m/s	F10.3
6	Vy-ECEF	ECEF Y velocity	m/s	F10.3
7	Vz-ECEF	ECEF Z velocity	m/s	F10.3
8	Pitch	Pitch (down negative, up positive)	deg	F10.4
9	Roll	Roll (left negative, right positive)	deg	F10.4
10	Yaw	Yaw (west of north positive)	deg	F10.4
11	GyroBiasX	Estimated gyro bias, X-axis	deg/h	F12.4

12	GyroBiasY	Estimated gyro bias, Y-axis	deg/h	F12.4
13	GyroBiasZ	Estimated gyro bias, Z-axis	deg/h	F12.4
14	AcceBiasX	Estimated accel bias, X-axis	mg	F12.4
15	AcceBiasY	Estimated accel bias, Y-axis	mg	F12.4
16	AcceBiasZ	Estimated accel bias, Z-axis	mg	F12.4
17	MeasType	Measurement type at this epoch (e.g., GNSS, NHC)	-	A10
18	OdoScale	Odometer scale factor (output only when ODO is active)	#	F12.4
19	Nsat	Number of usable satellites	#	I5
20	PDOP	Position dilution of precision	-	F7.2
21	AmbStatus	Ambiguity status (e.g., Fixed, Float)	#	A8
22	Ratio	Ratio value	-	F10.2

Example:

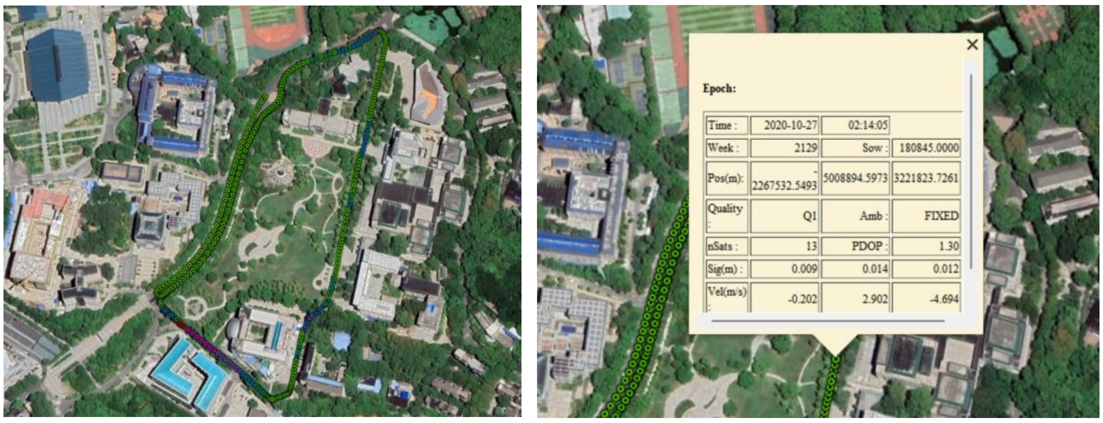


```
# Seconds of Week        X-ECEF        Y-ECEF        Z-ECEF     VX      VY     VZ    Pitch     Roll      Yaw  GyroBiasX  GyroBiasY  GyroBiasZ  AcceBiasX  AcceBiasY  AcceBiasZ  MeasType  Nsat  PDOP  AmbStatus
#              (s)           (m)           (m)           (m)  (m/s)   (m/s)  (m/s)    (deg)    (deg)    (deg)    (deg/h)    (deg/h)    (deg/h)       (mg)       (mg)       (mg)               #     #
     181469.000000  -2267184.032   5008969.333   3221946.269  1.705  -1.557  3.784  -1.1364   0.6167  14.1343     0.0865     0.0517     0.0101     0.0537    -0.0949     0.0282      GNSS    22  1.53   Float      0.00
     181470.000000  -2267182.219   5008967.615   3221950.234  1.783  -1.861  4.288  -0.6189  -0.4715  10.4242     9.3168    16.9507    -1.1496     6.6690    -3.2412     5.3251      GNSS    22  1.53   Float      0.00
     181471.000000  -2267180.539   5008965.552   3221954.718  1.515  -2.240  4.719  -1.1700  -0.3822   5.8184     3.2967    37.8025    -1.6571     6.8033    -2.7700     7.2016      GNSS    22  1.53   Float      0.00
     181472.000000  -2267179.094   5008963.109   3221959.600  1.442  -2.577  5.133  -1.0863   0.5838   2.2582   -97.8304    65.6301     1.6566     4.6152    -2.4315     8.6836      GNSS    22  1.53   Float      0.00
     181473.000000  -2267177.743   5008960.498   3221964.679  1.301  -2.618  5.000  -1.7534   0.3246   0.9053  -246.2934    75.6400     0.0581     1.5549    -2.7811     9.2034      GNSS    22  1.53   Float      0.00
     181474.000000  -2267176.468   5008957.904   3221969.677  1.315  -2.588  5.042  -1.3962   0.5349   1.2640  -390.4313    90.3529    -7.3836    -0.0969    -2.7557     9.3068      GNSS    22  1.53   Float      0.00
     181475.000000  -2267175.099   5008955.315   3221974.688  1.434  -2.575  5.058  -1.3243   0.2479   1.9856  -453.6508   110.7393   -13.6131     0.3243    -2.9359     9.3583      GNSS    22  1.53   Float      0.00
     181476.000000  -2267173.634   5008952.684   3221979.815  1.564  -2.669  5.218  -1.4223   0.4959   2.9168  -484.2425   144.7023   -18.1961     1.3721    -3.9846     9.3248      GNSS    22  1.53   Float      0.00
     181477.000000  -2267171.976   5008950.015   3221985.108  1.764  -2.636  5.345  -1.4168   0.5676   4.6933  -468.3937   181.2960   -22.8617     0.6590    -5.7270     9.3785      GNSS    22  1.53   Float      0.00
     181478.000000  -2267170.041   5008947.412   3221990.557  2.107  -2.569  5.560  -0.8922   0.7187   7.3027  -468.7954   179.8914   -22.4706     0.7288    -5.5899     9.3576      GNSS    21  1.57   Float      0.00
     181479.000000  -2267167.742   5008944.835   3221996.215  2.503  -2.562  5.736  -1.2963   0.7216   9.9178  -459.5683   183.1604   -19.1051    -0.4100    -5.8539     9.3921      GNSS    22  1.53   Float      0.00
     181480.000000  -2267165.095   5008942.286   3222002.038  2.783  -2.548  5.918  -1.4334   0.5336  11.9445  -458.4435   184.0185   -17.5052    -0.4121    -5.9548     9.3559      GNSS    22  1.53   Float      0.00
     181481.000000  -2267162.211   5008939.754   3222008.035  2.868  -2.614  6.095  -1.0974  -0.0358  12.0261  -449.5928   188.1667   -22.7018    -2.0454    -6.5211     9.3721      GNSS    22  1.53   Float      0.00
     181482.000000  -2267159.335   5008937.089   3222014.197  2.838  -2.695  6.145  -1.5608  -0.2488  11.5480  -435.0294   192.6977   -68.6934    -3.3085    -5.8271     9.4001      GNSS    21  1.73   Float      0.00
     181483.000000  -2267156.550   5008934.347   3222020.305  2.701  -2.780  6.056  -2.2272  -0.3573  10.9048  -420.6663   197.7122  -126.5187    -3.5789    -4.8786     9.3987      GNSS    21  1.73   Float      0.00
     181484.000000  -2267153.908   5008931.535   3222026.339  2.594  -2.842  6.020  -2.4959  -0.2872  10.1907  -412.0943   200.1191  -175.2973    -3.2683    -4.0259     9.3859      GNSS    21  1.73   Float      0.00
     181485.000000  -2267151.362   5008928.660   3222032.411  2.530  -2.933  6.101  -2.5882  -0.1884   9.4196  -407.3093   202.9038  -183.9383    -2.8917    -3.5837     9.3533      GNSS    21  1.73   Float      0.00
     181486.000000  -2267148.810   5008925.705   3222038.594  2.582  -3.007  6.240  -2.5692  -0.1264   9.4416  -402.5173   205.1755  -191.9189    -2.3091    -3.1921     9.3515      GNSS    21  1.73   Float      0.00
     181487.000000  -2267146.187   5008922.626   3222044.914  2.656  -3.110  6.366  -2.9004   0.4673   9.5235  -398.7271   206.7406  -191.1264    -1.6867    -2.9317     9.3716      GNSS    21  1.73   Float      0.00
     181488.000000  -2267143.509   5008919.490   3222051.221  2.685  -3.216  5.914  -5.7145   0.4714   9.5299  -392.9688   208.7441  -175.8407    -0.7122    -2.4333     9.3720      GNSS    21  1.73   Float      0.00
     181489.000000  -2267140.923   5008916.415   3222056.978  2.555  -3.017  5.725  -5.0765   0.0837   9.4634  -390.1504   209.6282  -151.5201    -0.2249    -2.1313     9.3665      GNSS    20  2.26   Float      0.00
```

Figure 6.3 Example of GREAT-MSF result output file

6.3 Trajectory file (**.kml) description

The integrated-solution trajectory file can be opened with Google Earth or similar map software to inspect estimated accuracy and the solution information at each coordinate point, e.g.:


Epoch:

Time :	2020-10-27	02:14:05	
Week :	2129	Sow :	180845.0000
Pos(m):	-2267532.5493	5008894.5973	3221823.7261
Quality :	Q1	Amb :	FIXED
nSats :	13	PDOP :	1.30
Sig(m) :	0.009	0.014	0.012
Vel(m/s) :	-0.202	2.902	-4.694

Figure 6.4 Example of opening a GREAT-MSF trajectory output file

Different point colors along the trajectory indicate different solution-accuracy levels.