APPENDIX J

Simulation of Navigation Systems

Paul D. Groves

This appendix discusses the software simulation of all types of navigation system, with an emphasis on GNSS and inertial navigation. It provides an overview of the issues to consider when designing a navigation simulation, but does not provide detailed mathematical models. Section J.1 summarizes the reasons for simulation. Section J.2 describes the basic components of a simulation then compares the different architectures that may be used to simulate different types of navigation system, trading off fidelity against complexity. Sections J.3 and J.4 then discuss the truth model and error generation, respectively. Finally, Section J.5 discusses the use of Monte Carlo simulation to generate performance statistics.

In addition, a basic MATLAB simulation of inertial navigation, GNSS, and their integration is included on the CD. This is described in the separate document, *Guide to MATLAB Simulation Software*.

J.1 Reasons for Simulation

There are two main reasons for simulating navigation systems: testing and performance prediction. To test new or modified algorithms and software, representative inputs are required, together with knowledge of what the outputs should be. Software simulation provides a controlled environment within which a wide range of different test scenarios may be run, including those which are difficult to implement in a physical environment. Other testing methods are discussed in Section 17.7.

Performance prediction may be used to determine navigation system specifications to meet the requirements of a particular application. It may be used as part of a wider application simulation, such as a guidance and control system, air traffic management system, or mission

planning tool. Simulation may also be used as part of a navigation system's formal certification process for a safety-critical application.

J.2 Simulation Architecture and Fidelity

Figure J.1 depicts a generic simulation architecture. The truth model determines the true position of the user equipment, together with velocity, attitude, acceleration, angular rate, and gravity where required. This information may be required for multiple bodies, such as an IMU and a GNSS antenna. Transmitter positions may also be required.

The error generation function produces error sources with the required statistical distributions and correlation properties, both in time and across different error sources.

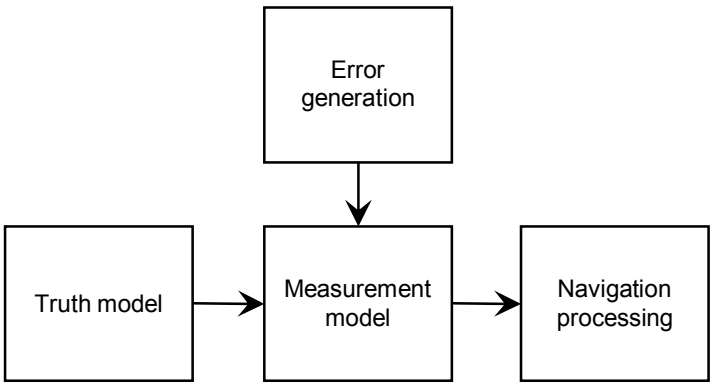

Figure J.1 Generic simulation architecture.

The measurement model uses the truth model outputs and the error sources to produce navigation system measurements or signals. For example, on the CD, the MATLAB function Generate_GNSS_measurements produces pseudo-range and pseudo-range rate measurements, while IMU_model produces specific force and angular rate measurements.

The navigation processing function uses the outputs of the measurement model to compute the simulated navigation solution. Where the simulation is being used for testing, this function will include the algorithms and/or software under test. For performance prediction, either real or simplified navigation processing may be implemented, depending on the requirements. The MATLAB functions on the CD include precision inertial navigation equations; single-epoch and filtered GNSS positioning algorithms; and loosely and tightly coupled INS/GNSS integration algorithms.

The simplest possible navigation system simulation simply adds errors directly to the position generated by the truth model. However, this typically produces unrealistic results. For example, inertial navigation errors depend on the maneuvers performed as well as the underlying error sources, while GNSS positioning errors depend on the signal geometry. Better results are thus obtained by introducing the errors at an earlier stage in the processing chain. Generally, the earlier the errors are introduced, the more realistic the simulation results are, but the greater the computational complexity. A more complex simulation requires more processing resources and greater development effort. There is thus a tradeoff to be made.

The remainder of this section considers the simulation of different classes of navigation system in more detail. Dead-reckoning systems, including inertial navigation, are discussed first; followed by ranging-based positioning systems, such as GNSS; and then the other position-fixing techniques.

J.2.1 Inertial Navigation and Other Dead-Reckoning Systems

Dead-reckoning systems exhibit position errors that grow with time as a function of the host vehicle maneuvers. Inertial navigation error behavior is particularly complex. Therefore, it is generally best to simulate these systems at the sensor output level. For inertial navigation, accelerometer specific force and gyro angular rate measurements are simulated. These are also

used for a PDR or AHRS simulation. For a magnetic compass, the magnetometer's magnetic flux density measurements are simulated. Other examples include the air pressure measurement of a barometric altimeter, wheel or transmission shaft rotation for odometry, and individual-beam frequency shifts for Doppler radar and sonar.

The truth model is first used to determine the measurement that a perfect sensor would make. An error model is then applied to account for both random and systematic errors. Examples include (4.16) for accelerometers, (4.17) for gyros, and (6.9) for magnetometers. Bandlimiting and quantization may also be applied as described in Section J.4.4.

In principle, a higher fidelity may be achieved by simulating individual components within the navigation sensors.

J.2.2 GNSS and Other Radio Ranging Systems

Figure J.2 shows the GNSS user-equipment processing chain. This is also applicable to other signal-based position-fixing systems that use ranging. The simulation may begin at any point in this chain with the later processing stages then duplicating those of real user equipment. For performance prediction and testing of positioning and integration algorithms under good conditions, the simulation may begin after the signal tracking function where the pseudo-range and carrier-derived measurements are produced.

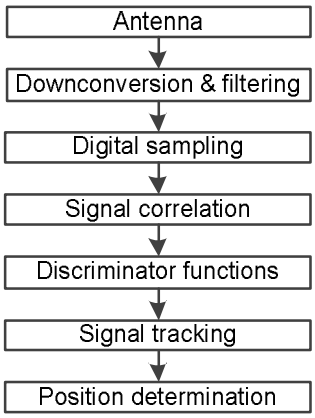

Figure J.2 GNSS user-equipment processing train (tracking mode).

Where dynamics response lags are significant, it is necessary to simulate the signal tracking, beginning the simulation at the discriminator outputs. Under poor signal-to-noise conditions, the discriminator error distributions are non-Gaussian, so the simulation must begin at the accumulated correlator outputs. It is also easier to simulate multipath interference at this stage. Clock and atmospheric propagation errors may be added to the ranges at either the inputs to or the outputs from the signal tracking functions. Ephemeris errors may also be simulated this way. Beginning the simulation at an earlier stage than the correlator outputs is only necessary where testing of the correlators or earlier processing stages is required.

For simulations of a few minutes, the satellite clock, ephemeris, ionosphere, and troposphere errors may be treated as constant. For longer simulations, their time variation must be modeled. Spatial variation of the ionosphere and troposphere errors is also significant for high-speed applications. It is simplest to treat the ionosphere and troposphere errors of signals from different satellites as independent. However, in reality, there is significant spatial correlation which should be considered if higher fidelity is required. Any simulation should model the variation in the magnitude of the ionosphere and troposphere propagation errors with elevation (see Section 9.3.2).

The partial correction of the satellite clock, ionosphere, and troposphere errors may be simulated separately within the position determination module. Alternatively, depending on the purpose of the simulation, simulation of the corrections may be omitted and the residual errors, following the correction process, directly simulated instead.

Signal blockage, attenuation, reflection, and diffraction can have a significant impact on GNSS performance, particularly in urban areas. The simplest way of accounting for these is to add ranging errors with a suitable statistical distribution. Higher fidelity may be achieved by simulating the path delays and attenuations at the correlator outputs, summing multiple signal components where appropriate. However, generating these randomly from statistical distributions does not properly model the spatial variation of these errors as the user equipment moves through the environment. It also fails to model the blockage or reflection of multiple signals by the same object, which can impact both the ensuing position errors and the performance of error mitigation techniques. Therefore, for the highest fidelity, a 3D model of the environment should be used to determine realistic path delays and attenuations [1].

For simulating other ranging-based positioning systems using radio or other signals, such as acoustics, the same approach as for GNSS may be adopted. It may also be used for Doppler positioning simply by omitting the ranging measurement generation and producing only Doppler shift measurements. For terrestrial positioning systems, it is important to simulate the variation in signal strength between different signals as this will result in variations in the ranging precision that can be achieved. Transmitter positioning errors should also be simulated by using different transmitter positions in the truth model and positioning algorithms. They should not be simulated as ranging errors.

J.2.3 Angular Radio Positioning

Angular radio positioning may be achieved using either directionally-varying signals or direction finding. In both cases, a simple simulation may be constructed by adding a signal-independent bias, a signal-dependent bias, and a random error to angles derived from the true line of sight between transmitter and receiver.

For a higher-fidelity simulation, receiver measurement errors and signal propagation errors must be considered separately; these are different for the two angular positioning techniques. With directionally-varying signals, such as VOR (Section 11.1.2), the direction is deduced from the signal modulation. Therefore, simulation of the user equipment is similar to that of ranging systems such as GNSS.

Direction finding constantly varies the gain pattern of the antenna (see Section 7.1.5). Signal direction is deduced by determining when the received signal strength was minimized and comparing this with the direction of the antenna's gain pattern at the time. Random errors in the timing of the minimization will occur with standard deviations that depend on both the signal to noise ratio and the sharpness of the antenna gain pattern. Systematic errors will arise from imperfections in the antenna gain pattern and the timing process. Biases will also arise if there is significant variation in the signal direction over a measurement cycle. This may occur if the host vehicle is rotating. As for GNSS, a tracking loop will typically be employed to smooth out the noise over successive cycles at the expense of increased sensitivity to motion lag. As with GNSS, the simulation may begin before or after the tracking function.

Signal propagation errors affect both types of angular positioning. Refraction causes the signal path to curve. It affects LF and MF signals more than VHF and UHF signals, though the latter experience significant vertical curvature due to variation of the troposphere with height. Reflection breaks the link between the signal direction and the transmitter-receiver line of sight and affects higher frequencies more. Directionally-varying signals are most sensitive to reflectors near the transmitter as it is the signal direction at the transmitter which is measured, whereas direction finding is most sensitive to reflectors near the receiver. A reflector near the middle of the signal path disrupts both methods. Propagation errors may be simulated using an environmental model or a statistical model. If a statistical model is used, it is important to account for the spatial correlation of the errors and ensure that they are reproduced if the mobile user equipment visits the same location more than once.

J.2.4 Radio Proximity Positioning

The simplest way of simulating proximity positioning is just to switch the position solution between the positions of different transmitters or base stations. The simplest switching criteria is to select the nearest base station. A slightly more sophisticated approach would account for the variation of coverage radii according to transmission power, height, and possibly user demand, selecting the base station with the maximum coverage radius to distance ratio. To avoid oscillation at boundaries, it could be required that the new coverage radius to distance ratio exceeds the old by a certain factor before switching occurs. Finally, to avoid switching to a base station out of range, the distance must be less than the coverage radius for switching to occur.

Advanced proximity positioning techniques use multiple transmitters or base stations. To simulate these, a list of receivable transmitters must be generated and input to the proximity positioning algorithm. The simplest approach is to simulate a constant coverage radius for each transmitter and mark this as receivable if the mobile user is within this distance of the transmitter. However, range can vary with distance due to azimuthal variations in transmission power, variations in terrain, and obstructions, such as buildings. This can be simulated by varying the coverage radius with direction. However, real coverage areas have holes within them and islands of coverage outside. A grid can be used to indicate where a signal is and is not receivable, while for the greatest fidelity, a model of the terrain and buildings can be used.

Signal reception depends on the receiver as well as the transmitter. The antenna gain of the mobile user equipment may vary with direction. Reception may be masked or attenuated in certain directions by a person or vehicle body. Furthermore, a higher signal strength is sometimes needed for initial acquisition of a signal than for subsequent tracking. Finally, some receivers are simply more sensitive than others. Therefore, realistic determination of whether or not a signal is receivable requires modeling of both the signal strength from the transmitter and the sensitivity of the receiver.

J.2.5 Pattern Matching

Pattern matching positioning techniques include map matching (Section 13.1), terrain-referenced navigation (Section 13.2), and radio received signal strength fingerprinting (Section 7.1.6). As with any positioning technique, the simplest way of simulating pattern matching is simply to add errors to the true position. Errors due to discrepancy between the matching database and the real world will be the same each time the mobile user visits a particular location, while sensor measurement errors will not be correlated with the position. Further complications are that pattern matching can sometimes produce a completely wrong position or be unable to determine position at all.

A higher-fidelity pattern-matching simulation will generate measurements of the parameter(s) to be matched with the database, such as terrain height, and input these to a pattern-matching positioning algorithm. Measurements are generated using a second database that gives the “true” values of the matching parameters as a function of position. Interpolation to the required position is performed and errors added according to a model of the sensor used. The sensor errors will typically be both systematic and random.

Ideally, the truth database will be significantly more accurate than the database used by the pattern-matching positioning algorithm itself. However, in practice, the positioning algorithm database may be the best available. In this case the truth database must be generated by subtracting simulated errors from the positioning algorithm database. These simulated errors will typically be spatially correlated.

J.2.6 Image-Based Positioning

Image-based positioning (see Section 13.3) generates proximity, range, and/or angular measurements in the same manner as radio positioning. However, it must also correctly identify the environmental features that it is using as positioning landmarks and this can

sometimes go wrong. A low-fidelity simulation simply adds errors representative of the sensor used to the true position. It also produces an appropriate number of wrong or absent fixes.

A medium-fidelity simulation generates range and/or angular measurements using an error model of the imaging sensor, which could be based on a camera, radar, sonar, a laser scanner, or a combination of these. These are input to the positioning algorithm along with the position of the landmark that has been measured. Three cases must be considered: correct feature identification, in which case a small landmark position error is simulated; false feature identification, in which case a large landmark position error is simulated; and no feature identification, in which case no measurements are supplied to the positioning algorithm. To support this, a database is required containing the following information for each feature:

- Its true position;
- Its reported position;
- The probability of correct identification, which may vary with distance from the feature;
- The position reported when the feature is wrongly identified;
- The area within which the feature may be observed.

A high-fidelity simulation simulates the actual images observed by the sensor and inputs these to the full image-based positioning algorithm. This will require considerably more processing capacity and development effort than the medium-fidelity approach.

J.3 Truth Model

The truth model determines the true position of the user equipment, together with other quantities, such as velocity and attitude. This is used both as a baseline for simulating the navigation system(s) and to provide a reference against which the navigation solution may be compared in order to determine its errors. Figure J.3 depicts the components of a truth model.

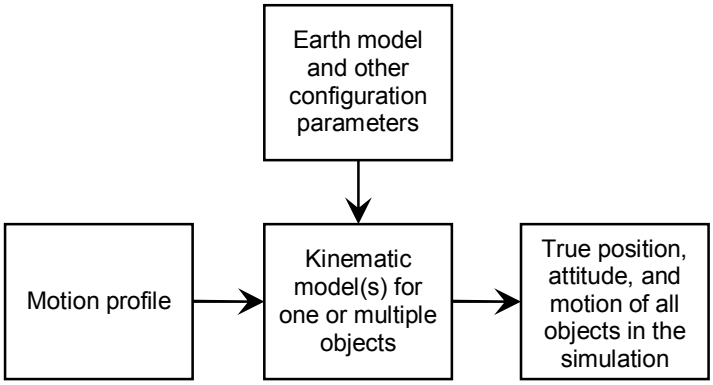

Figure J.3 The components of a simulation truth model.

This section first describes the motion profile, which contains the minimum necessary information to describe the motion of the object or objects being simulated. The kinematic model, which determines the additional motion parameters required by the simulation, is described next. Finally, the extension of the simulation to multiple objects, including GNSS satellites, and assumptions about the Earth are discussed.

J.3.1 Motion Profile

The motion profile is a series of records describing the motion of one or more objects. Forms this may take include

- Time and position;
- Time and velocity (plus initial position);
- Time, position, and attitude;
- Time, velocity, and attitude (plus initial position);

- Time, position, velocity, and attitude;
- Time, acceleration, and angular rate (plus initial position, velocity, and attitude);
- Time, specific force, and angular rate (plus initial position, velocity, and attitude).

Different position and attitude representations, coordinate frames, and axes conventions may be used in the motion profile. Therefore, the format must be specified explicitly.

A motion profile may be entirely simulated or obtained from recorded trials data. A simulated profile is typically generated in advance and read into the navigation simulation from a file. However, it may also be generated as the simulation is run. Different simulation applications require different fidelities of motion. In some cases, an unrealistically smooth trajectory (which is relatively easy to generate), is sufficient. In other cases, phenomena such as vehicle flexure and vibration, air turbulence, sea states, wheel slippage, and human body motion must be simulated.

Where the motion profile is obtained from trials data, a higher perfomance navigation system than that simulated is typically used. It may also be necessary to edit the recorded data to correct for errors, interpolate it to increase the time resolution or regularize the data rate, and/or translate the motion from one part of a vehicle to another.

J.3.2 Kinematic Model

The kinematic model integrates and/or differentiates the motion profile to determine additional motion parameters. It may also perform interpolation to enable the simulation to run at a higher update rate. The precision inertial navigation equations described in Chapter 5 may be used to update attitude using angular rate, transform specific force resolving axes, update velocity using specific force or acceleration, and update position using velocity.

From (5.23) and (5.38), the velocity may be determined from Cartesian position using

$$\mathbf{v}^{\gamma}_{\gamma b}(+) = \frac{2}{\tau_i}\left[\mathbf{r}^{\gamma}_{\gamma b}(+) - \mathbf{r}^{\gamma}_{\gamma b}(-)\right] - \mathbf{v}^{\gamma}_{\gamma b}(-) \qquad \gamma \in i, e\,, \tag{J.1}$$

where the suffixes (+) and (–) denote the current and previous values, respectively, and τ_i is the update time interval. Where the previous velocity is unknown, it should be assumed equal to the current velocity. Similarly, from (5.56), the velocity may be determined from curvilinear position using

$$\begin{aligned}
v^n_{eb,N}(+) &\approx \left[R_N\left(L_b(-)\right) + h_b(+)\right]\left[\frac{2}{\tau_i}\left(L_b(+) - L_b(-)\right) - \frac{v^n_{eb,N}(-)}{R_N\left(L_b(-)\right) + h_b(-)}\right] \\
v^n_{eb,E}(+) &= \left[R_E\left(L_b(+)\right) + h_b(+)\right]\cos L_b(+)\left[\frac{2}{\tau_i}\left(\lambda_b(+) - \lambda_b(-)\right) - \frac{v^n_{eb,E}(-)}{\left[R_E\left(L_b(-)\right) + h_b(-)\right]\cos L_b(-)}\right] \\
v^n_{eb,D}(+) &= -\frac{2}{\tau_i}\left(h_b(+) - h_b(-)\right) - v^n_{eb,D}(-)
\end{aligned} \tag{J.2}$$

From (5.20), (5.36), and (5.54), the average acceleration over the update interval may be determined from velocity using

$$\begin{aligned}
\mathbf{a}^i_{ib} &= \left(\mathbf{v}^i_{ib}(+) - \mathbf{v}^i_{ib}(-)\right)\big/\tau_i \\
\mathbf{a}^e_{eb} &= \left(\mathbf{v}^e_{eb}(+) - \mathbf{v}^e_{eb}(-)\right)\big/\tau_i + 2\boldsymbol{\Omega}^e_{ie}\mathbf{v}^e_{eb}(-) \\
\mathbf{a}^n_{eb} &= \left(\mathbf{v}^n_{eb}(+) - \mathbf{v}^n_{eb}(-)\right)\big/\tau_i + \left(\boldsymbol{\Omega}^n_{en}(-) + 2\boldsymbol{\Omega}^n_{ie}\right)\mathbf{v}^n_{eb}(-)
\end{aligned}\,, \tag{J.3}$$

where $\boldsymbol{\Omega}^e_{ie}$, $\boldsymbol{\Omega}^n_{ie}$, and $\boldsymbol{\Omega}^n_{en}$ are given by (5.25), (5.41), and (5.44), respectively. Specific force is obtained using

$$\begin{aligned} \mathbf{f}_{ib}^{i} &= \mathbf{a}_{ib}^{i} - \boldsymbol{\gamma}_{ib}^{i} \\ \mathbf{f}_{ib}^{\gamma} &= \mathbf{a}_{eb}^{\gamma} - \mathbf{g}_{b}^{\gamma} \qquad \gamma \in e, n \end{aligned}, \tag{J.4}$$

where gravity and gravitational models are described in Section 2.4.7.

The specific force resolving axes may be transformed to the body frame using

$$\mathbf{f}_{ib}^{b} = \left(\overline{\mathbf{C}}_{b}^{\gamma}\right)^{-1} \mathbf{f}_{ib}^{\gamma} \qquad \gamma \in i, e, n, \tag{J.5}$$

where $\overline{\mathbf{C}}_{b}^{i}$, $\overline{\mathbf{C}}_{b}^{e}$, and $\overline{\mathbf{C}}_{b}^{n}$ are given by (5.84), (5.85), and (5.86), respectively.

The coordinate transformation matrix describing the rotation from the previous to current body frames may be obtained from the current and previous attitude solutions using

$$\begin{aligned} \mathbf{C}_{b-}^{b+} &= \mathbf{C}_{i}^{b}(+)\mathbf{C}_{b}^{i}(-) \\ \mathbf{C}_{b-}^{b+} &= \mathbf{C}_{e}^{b}(+)\begin{pmatrix} \cos\omega_{ie}\tau_i & \sin\omega_{ie}\tau_i & 0 \\ -\sin\omega_{ie}\tau_i & \cos\omega_{ie}\tau_i & 0 \\ 0 & 0 & 1 \end{pmatrix}\mathbf{C}_{b}^{e}(-) \\ \mathbf{C}_{b-}^{b+} &= \mathbf{C}_{n}^{b}(+)\left[\mathbf{I}_3 - \left(\boldsymbol{\Omega}_{ie}^{n} + \tfrac{1}{2}\boldsymbol{\Omega}_{en}^{n}(+) + \tfrac{1}{2}\boldsymbol{\Omega}_{en}^{n}(-)\right)\tau_i\right]\mathbf{C}_{b}^{n}(-) \end{aligned}. \tag{J.6}$$

The attitude increment is then given by

$$\boldsymbol{\alpha}_{ib}^{b} = \begin{cases} \dfrac{\mu}{2\sin\mu}\begin{pmatrix} C_{b-2,3}^{b+} - C_{b-3,2}^{b+} \\ C_{b-3,1}^{b+} - C_{b-1,3}^{b+} \\ C_{b-1,2}^{b+} - C_{b-2,1}^{b+} \end{pmatrix} & \mu > 2\times10^{-5} \\ \dfrac{1}{2}\begin{pmatrix} C_{b-2,3}^{b+} - C_{b-3,2}^{b+} \\ C_{b-3,1}^{b+} - C_{b-1,3}^{b+} \\ C_{b-1,2}^{b+} - C_{b-2,1}^{b+} \end{pmatrix} & \mu \le 2\times10^{-5} \end{cases}, \tag{J.7}$$

where

$$\mu = \arccos\left[\tfrac{1}{2}\left(C_{b-1,1}^{b+} + C_{b-2,2}^{b+} + C_{b-3,3}^{b+} - 1\right)\right]. \tag{J.8}$$

Finally, the angular rate of the body frame with respect to inertial space, resolved about body-frame axes is

$$\boldsymbol{\omega}_{ib}^{b} = \boldsymbol{\alpha}_{ib}^{b} / \tau_i . \tag{J.9}$$

The preceding equations are implemented in the MATLAB functions Kinematics_ECI, Kinematics_ECEF, and Kinematics_NED on the CD.

Sometimes the motion profile may be inconsistent. For example, the integral of the velocity may not equal the change in position. This can happen where the motion profile update interval is larger than the interval used to generate the profile or if the profile is poorly interpolated. If the simulation uses inconsistent profile data for its truth model, inexplicable errors in the simulated navigation solution can arise due to the reference used to calculate the errors not being consistent. Therefore, it is better to take only one linear quantity (i.e., position or velocity or acceleration) and one angular quantity (i.e., attitude or angular rate) from the motion profile and to generate the others using the kinematic model.

Similarly, where the kinematic model is used to interpolate the motion profile, only one linear quantity and one angular quantity should be interpolated, with the others obtained by integration or differentiation to ensure consistency. Furthermore, the interpolation algorithm should not alter the values in the motion profile, assuming these are consistent with each other.

J.3.3 Multiple Objects and GNSS Satellites

Different components of an integrated navigation system, such as an IMU or a GNSS antenna, are often located on different parts of the host vehicle (or pedestrian). For some simulation applications, it is also necessary to model the individual locations of each inertial sensor. A truth model is thus often required for multiple objects. There can be a separate motion profile and kinematic model for each location on a body that is required. Alternatively, a common motion profile may be used with the motion of the individual locations then generated from it as the simulation runs. Similarly, the motion profile may describe a different part of the body from that where the navigation sensors are located (e.g., the center of gravity). Where the body is rigid, the attitude, position, velocity, and acceleration may be transposed from one part of the body to another as described in Section 2.5.5, while the angular rate is the same throughout the body. Where the body flexes, a more complex model is required. An extreme example is the human body motion.

As well as modeling multiple objects on a common body, a navigation simulation may also be required to model independent objects. For simulations of relative or cooperative positioning involving multiple vehicles or people, the full motion model must be duplicated. However, most radio transmitters, sonar transponders, and other features used as landmarks may simply be modeled as fixed with respect to the Earth. Their positions may be entered as part of the simulation configuration process or loaded from a separate database.

GNSS satellite positions and velocities may be generated using the same orbital models used to calculate them in user equipment. These are described in Section 8.5.2 and Section G.4 of Appendix G on CD. For some simulation applications, simplified models may be used that assume the orbits are circular and that satellites precisely follow their nominal orbits. However, for performance certification and for testing user equipment software that calculates satellite orbits, realistic models must be used. Using realistic models also enables real satellite orbit data, downloaded from the Internet or from the broadcast navigation data, to be used in the simulation.

J.3.4 Modeling the Earth

For basic performance prediction simulations or for testing new concepts, simplified Earth models may be used. The shape of the Earth may be approximated to a sphere, or even a flat Earth assumed; Earth rotation may be neglected and gravity may be assumed constant. However, these simplifications must be applied across the whole simulation: the truth model, the measurement model, and the navigation processing. Consequently, simplified Earth modeling is unsuited to the testing of algorithms and software that must work with real data, either recorded or live. Similarly, for realistic Earth modeling, it should be ensured that datum parameters and gravity models are consistent across all stages of the simulation. Otherwise, unexpected navigation solution errors may occur.

J.4 Error Generation

This section describes the generation of the error sources used in navigation system simulation. Random number generation is discussed first, followed by the simulation of random sequences, correlated error sources, and finally, bandlimiting and quantization.

J.4.1 Random Number Generation

Values of constant error sources can be entered manually as part of the simulation configuration process. This can be useful where the simulation of an individual instrument, characterized in the laboratory, is required. However, if a random sample is required, it is difficult to generate truly random numbers manually with the risk that the ensuing simulation results could be biased. Generating large numbers of random numbers manually (e.g., for a random noise process) is also impractical. Therefore, random number generation is an important part of any simulation.

Most random number generating functions output a value uniformly distributed between 0 and 1. This may be converted to another distribution, such as the Gaussian (or normal) distribution, using the inverse cumulative distribution function. The CDF is defined in Section B.2 of Appendix B on CD. Gaussian-distributed random numbers may also be obtained from uniformly distributed random numbers using the Box-Muller method [2], while some programming languages already provide functions for this (e.g., randn in MATLAB).

A random number generator produces a pseudo-random number from an input value, known as a seed, and a mathematical formula. It also generates a new seed for use next time it is called. Sometimes it is desirable to repeat a sequence of random numbers (e.g., to compare the performance of different algorithm configurations). This can be done by setting the seed to a known value at the beginning of the simulation.

If a single sequence of random numbers is used throughout a simulation, activating or deactivating a feature that uses random numbers will cause the random numbers to change throughout the simulation, which may not be desirable. Furthermore, it may be useful to change the random numbers used in one part of a simulation, such as an IMU model, while leaving other random numbers unchanged. Therefore, it can be useful to implement multiple independent random noise sequences in parallel. To achieve this, the current values of the seeds for each sequence must be stored. Whenever the random number generator is called, it must be initialized with the appropriate stored seed. The seed output by the random number generator then replaces the previous seed in the store.

J.4.2 Simulating Random Sequences

As described in Section B.4 of Appendix B, a random process (or stochastic process) is a random variable that changes continuously with time. A random sequence is a random variable that changes at discrete time intervals. As computer simulations are inherently discrete time, they use random sequences, even when simulating a random process.

The simplest random sequences are random constants and white noise sequences; they also represent the two extremes. A random constant is initialized to a random number from the appropriate distribution. The same value is then used for all subsequent epochs of the simulation. A white noise sequence takes a new random value at every epoch which is independent of all previous values. Often the single-sided PSD, S_w^{f1}, or root thereof, is specified, in which case the standard deviation is given by

$$\sigma_w = \sqrt{\frac{S_w^{f1}}{\tau_w}} = \sqrt{S_w^{f1} f_w}\,, \tag{J.10}$$

where τ_w is the sampling interval of the white noise sequence and f_w is the sampling frequency.

Random sequences are often modeled as first-order Gauss-Markov sequences. A random walk sequence, x, is simulated using

$$x_k = x_{k-1} + \sigma_w N_k(0,1), \tag{J.11}$$

where the subscript k denotes the epoch and $N_k(0,1)$ is the k^{th} sample from a zero-mean unit-variance Gaussian distribution. A much larger standard deviation is typically used to set the initial value, x_0. Note that the variance of the process grows linearly with time (i.e., the standard deviation increases with the square root of time).

A constant-variance exponentially-correlated first-order Gauss-Markov sequence is simulated using

$$x_k = \exp(-\tau/\tau_c) x_{k-1} + \sigma_x \sqrt{1-\exp(-2\tau/\tau_c)}\, N_k(0,1), \tag{J.12}$$

where τ is the update interval, τ_c is the correlation time of the Gauss-Markov sequence, and σ_x is its standard deviation. Provided that σ_x is also used as the standard deviation of the initial value of the sequence, x_0, the variance and standard deviation of x will remain constant.

An error source may also be modeled as the combination of multiple random sequences. For example, an accelerometer bias (see Section 4.4) may be represented as the sum of a fixed constant that is known, a random constant that is unknown, and a constant-variance exponentially-correlated Gauss-Markov sequence. Furthermore, each of these may be modeled as the sum of a temperature-independent term, a first-order temperature-dependent term, a second-order temperature-dependent term, and so forth.

J.4.3 Correlated Error Sources

Not all errors in a navigation system are independent. Position and velocity errors in different directions are typically partially correlated due to common underlying sources of error. The noise on GNSS early, prompt, and late correlator outputs is also correlated as described in Section 9.1.4.3.

Correlated error sources must be simulated together. Suppose a vector, $\mathbf{x}$, comprising n correlated error sources is required and that $\mathbf{C}_\mathbf{x}$ is the covariance matrix of $\mathbf{x}$, defining the variances of the error sources and their correlation properties as described in Section B.2.1 of Appendix B. This may be produced by first generating a set of n independent random variables, $\mathbf{y}$, with covariance, $\mathbf{C}_\mathbf{y}$. The correlated error sources are then given by

$$\mathbf{x} = \mathbf{L}\mathbf{y}\,, \tag{J.13}$$

where $\mathbf{L}$ is a lower triangular matrix and

$$\mathbf{C}_\mathbf{x} = \mathbf{L}\mathbf{C}_\mathbf{y}\mathbf{L}^\mathrm{T}\,, \tag{J.14}$$

where $\mathbf{L}$ and $\mathbf{C}_\mathbf{y}$ are obtained by Cholesky factorization of $\mathbf{C}_\mathbf{x}$ as described in Section A.6 of Appendix A on CD. In cases where $\mathbf{C}_\mathbf{x}$ is constant, the factorization need only be performed once, at the beginning of the simulation. However, if $\mathbf{C}_\mathbf{x}$ varies, the factorization must be performed every time the error sources are updated.

J.4.4 Bandlimiting and Quantization

Bandwidth limitations of both error sources and whole sensor outputs (i.e., including the wanted signal) are often significant. For example, GNSS signal tracking loops have a limited bandwidth with the benefit that higher-bandwidth noise is attenuated and the penalty that there are lags in responding to dynamics (see Section 9.3.3).

If the double-sided bandwidth, B, is less than a quarter of the simulation update rate, bandlimiting should be simulated. This can be done using a simple fixed-gain smoothing filter. Thus, the bandlimited signal at epoch k, $x_{B,k}$, is given by

$$x_{B,k} = Kx_{U,k} + (1-K)x_{B,k-1}\,, \tag{J.15}$$

where $x_{U,k}$ is the corresponding unfiltered signal and K is the filter gain, given by

$$K = 4B\tau = 4B/f\,, \tag{J.16}$$

where τ is the update interval and f is the update frequency.

Bandlimiting is typically applied to the signal as well as the error sources, in which case, it should be applied to the relevant outputs of the measurement model (see Figure J.1) rather than to the error source directly.

All digital signals are quantized to some degree; otherwise, an infinite number of bits would be required. Some sensors are inherently quantised. For example, a wheel speed sensor records the number of times a wheel tooth passes the sensor over a particular time interval. Quantization errors should be simulated wherever they make a significant contribution to the overall navigation solution error.

Although a quantization may be simulated by generating separate quantization errors and adding them to the appropriate sensor measurements, it is generally more realistic to apply an

actual quantization process to the relevant outputs of the measurement model. Where both bandlimiting and quantization are simulated, the bandlimiting should be applied first.

Quantization may occur with or without the residuals being carried over to the next epoch. Without carry over, the quantized signal is given by

$$x_{Q,k} = q\,\text{round}\left(x_{U,k}/q\right), \tag{J.17}$$

where $x_{U,k}$, is the corresponding unquantized signal, q is the quantization level and the function round rounds its argument to the nearest integer. A quantization process may also round all arguments upward or downward.

With residual carry over, the quantized signal becomes

$$x_{Q,k} = q\,\text{round}\left[\left(x_{U,k} + r_{k-1}\right)/q\right], \tag{J.18}$$

where r_{k-1} is the quantization residual from the previous epoch. The quantization residual is updated using

$$r_k = r_{k-1} + x_{U,k} - x_{Q,k}. \tag{J.19}$$

Quantization to the nearest integer with residual carry over is implemented in the MATLAB function IMU_model.

J.5 Monte Carlo Simulation

Individual simulation runs can be used for testing software and algorithms. However, they are not suited to performance prediction because certain combinations of random errors can lead to unrealistically optimistic or pessimistic performance. In a Monte Carlo simulation, multiple runs are conducted, each with a different set of randomly chosen values of the random variables, typically error sources. Performance statistics, such as means, standard deviations, and covariances of the navigation solution errors are then computed from the set of results.

The precision of a Monte Carlo simulation is proportional to the square root of the number of runs. Thus, if 100 runs are conducted, the performance statistics should be accurate to within about 10%, whereas if there are 400 runs, the accuracy should be within 5%.

References

[1] Bradbury, J., et al., "Code Multipath Modelling in the Urban Environment Using Large Virtual Reality City Models: Determining the Local Environment," *Journal of Navigation*, Vol. 60, No. 1, 2007, pp. 95-105.

[2] Box, G. E. P., and M. E. Muller, "A Note on the Generation of Random Normal Deviates," *Annals of Mathematical Statistics*, Vol. 29, No. 2, 1958, pp. 610–611.