Chapter 69
Design Paradigms for Multi-Constellation Multi-Frequency Software GNSS Receivers

James T. Curran, Mark Petovello and Gérard Lachapelle

Abstract This study addresses the challenges of designing a tracking architecture for a multi-constellation, multi-frequency GNSS receiver. Using a C++ software defined receiver platform, the design of a receiver which operates on the BeiDou, Galileo, GLONASS and GPS civil signals is explored and a modular, reconfigurable GNSS tracking architecture is presented. The component parts of this architecture are tested in a pedestrian navigation scenario examining a range of different code and carrier tracking algorithms. The challenges of developing of a flexible, universal design is addressed while also highlighting some of the benefits of signal and application specific designs. In particular, the improvements in both performance and reliability that can be leveraged by the use of dedicated multi-signal tracking strategies when operating under certain challenging conditions, is shown. Results illustrate the various trade-offs that can be made between performance and receiver flexibility when applied to a diverse set of GNSS signals.

Keywords BeiDou · DLL · FLL · Galileo · GLONASS · GNSS · GPS · Kalman Filter · PLL · SDR

69.1 Introduction

Modern GNSS receivers are becoming increasingly complex devices. In the near future these receivers will have four GNSS constellations at their disposal, each transmitting on a minimum of two frequencies with at least as many signals and modulation schemes. Furthermore, depending on the receiver application, be it survey, outdoor/indoor pedestrian, vehicular, aviation or timing, a receiver may

J. T. Curran (✉) · M. Petovello · G. Lachapelle
Location and Navigation Group, University of Calgary,
Calgary, Canada
e-mail: James.T.Curran@ucalgary.ca

J. Sun et al. (eds.), *China Satellite Navigation Conference (CSNC) 2013 Proceedings*, Lecture Notes in Electrical Engineering 243,
DOI: 10.1007/978-3-642-37398-5_69,

comprise of many different acquisition and tracking strategies, each fulfilling a different functionality. Key benefits of a software receiver platform are its flexibility and reconfigurability and, so, it is naturally suited to the multi-signal, multi-application problem. The design of a receiver to suit all needs, however, is not without its challenges.

As the number of combinations of ranging signals and processing strategies has become immense, it is perhaps desirable to strive for some signal independence in the receiver design. Doing so promotes code reuse and results in a receiver which is more readily adaptable to new signals. Of course, basic receiver operation requires, at the very least, knowledge of the signal modulation scheme, the navigation message structure and the satellite orbit and, so, a receiver cannot be completely signal independent. Nonetheless, the majority of the receiver modules can, through appropriate design, be rendered independent of the particular signal they process. Key modules amongst these are the acquisition and tracking strategies.

Tempting as a one-size-fits-all receiver may be, given the diverseness of the available signals, conventional wisdom holds that there is some merit in tuning acquisition and tracking strategies and some receiver operations to the specific strengths and weaknesses of each individual signal structure. Such a receiver architecture, while rigid and possibly more labour intensive to develop, has potential to elicit more accurate signal observations and, thus, yield enhanced navigation performance.

Against the backdrop of an ever-growing set of broadcast ranging signals, this paper presents an investigation into the merits of these two opposing design principles via implementation in GSNRx^{TM}, a GNSS software receiver developed and maintained by the PLAN Group of The University of Calgary. The study explores challenges of developing a flexible, universal design and identifies and quantifies the performance benefits that can be gained by sacrificing some flexibility through tailoring strategies to specific signals. A novel receiver architecture is presented, which facilitates significant code reuse without sacrificing receiver performance. The design is based on decomposing the receiver processing strategies in such a way as to maximize the commonality of its constituent elements. With this structure, receiver modification for new signals or different applications is reduced to simple build-time configuration.

Utilizing a C++ software receiver platform, Sect. 69.2 presents a signal- and system-independent tracking architecture, capable of hosting an arbitrary collection correlator-based tracking algorithms. An illustrative example, exhibiting the basic functionality and design of a specific tracking strategy is presented in Sect. 69.3. A number of receiver implementations are tested using simulated signals from the GPS, Galileo, GLONASS and BeiDou constellations, discussed in Sect. 69.4, revealing the relative performance of different signals using different architectures. Results of this experiment, discussed in Sect. 69.5, illustrate how some signals can benefit significantly from customized receiver design.

69.2 Tracking Architecture

The general hierarchy of the GSNRx^{TM} software receiver is depicted in Fig. 69.1. The receiver is primarily composed of a list of satellites, a list of sample sources, a Doppler removal and correlation (DRC) object, and a navigation solution. As can be seen, each satellite contains one or more channels, each of which operates on one or more signals. A channel can contain a selection of processing strategies which may include, for example, cold- and warm-acquisition strategies and a variety of tracking strategies. Which to use at any given time is at the discretion of the channel. This paper will focus on the tracking strategies and their implementation and will consider the design of a flexible, modular multi-signal tracking architecture. Depicted in Fig. 69.2 is the proposed architecture, which implements a compartmentalized design, allowing for effective code reuse and flexibility in algorithm design.

A key feature of this design is the separation of the housekeeping of maintaining a tracking strategy; the strategy implementation; and the fundamental tracking operations. Specifically, the design of a practical strategy has been partitioned into a three layers, referred to herein as a 'Tracker' layer, a 'Composite Strategy' layer and an 'Strategy-Interface' layer.

A 'Tracker' is a simple tool which dictates the correlator request, directly observes correlator values and predicts signal parameters. Trackers conform to a standard interface, depicted in Fig. 69.3, and estimate carrier phase, carrier frequency, code phase and code rate. The interface includes a facility for the tracker to report its internal estimates of lock or loss-of-lock. Example implementations include an FLL-DLL pair, a PLL-DLL pair, a code-frequency fine-search or a Kalman filter, although a tracker could implement any correlator-based tracking

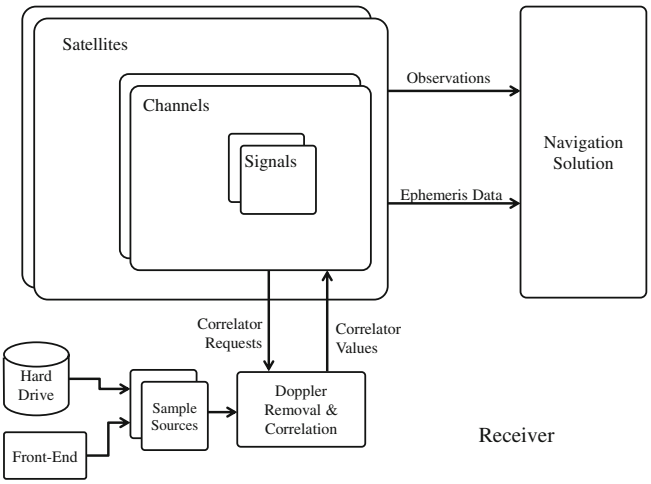

Fig. 69.1 A simplified view of the GSNRx^{TM}receiver architecture

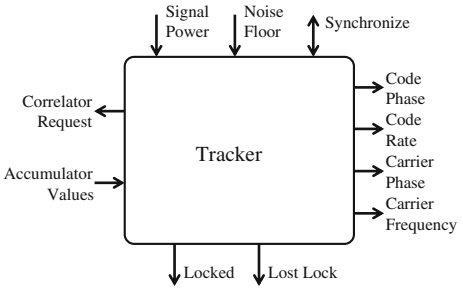

Fig. 69.2 Block diagram the standardized 'Tracker' interface

algorithm. Compliance with a standard interface allows the strategy to operate without requiring any knowledge of the particular tracking algorithm. Users can, therefore, freely design tracking algorithms (trackers) and readily incorporate them into a fully receiver implementation.

A 'Composite Strategy' is defined by a set of trackers and a scheduling state-machine. Within GSNRx^{TM}, a unique strategy can be defined at the 'composite strategy' layer by, firstly, instantiating a specific set of trackers to be used by the strategy and, secondly, implementing a state-machine which schedules the use of each tracker. This schedule may be based on time, signal-strength, bit-synchronization status or by observing a variety of lock indicators, at the discretion of the designer. At any given time, only one tracker is declared active by the state-machine, while all other trackers operate passively. To avoid transients when the state-machine switches from one tracker to another, signal parameter estimates within all passive trackers are continuously updated from those of the active one. When tracking multiple signals, it is likely that a different coherent integration period will be used on different signals and that the desired tracking loop update period may be longer than the shortest coherent integration period. To handle this eventuality, the 'Composite Strategy' layer maintains a buffer of accumulator values for use by the active tracker at each loop update epoch.

The 'Strategy-Interface' layer co-ordinates the interface between the tracking strategy and the receiver. Its tasks include driving the DRC engine to produce correlator values for all tracked signals; building signal observations; operating a set of signal monitoring utilities. Notably, these utilities, which include a C/N_0 estimator, phase and frequency lock monitors, a secondary-modulation syncher and a navigation decoder [1, 2], are maintained in this interfacing layer, as they are utilized by both the state-machine within the 'Composite Strategy' layer and are also (optionally) logged to the hard-drive by the interface layer for post-processing. In general, functionality that is deemed universal to all tracking strategies, is contained within the strategy interface layer, as it serves as the foundation for all strategy implementations.

In the implementation of a multi-signal architecture, synchronization is an important consideration as, if observations of two or more signals are to be combined, a common epoch should be defined at which to update the tracking loops. To do so, the concept of a master- and slave-signals is employed. The

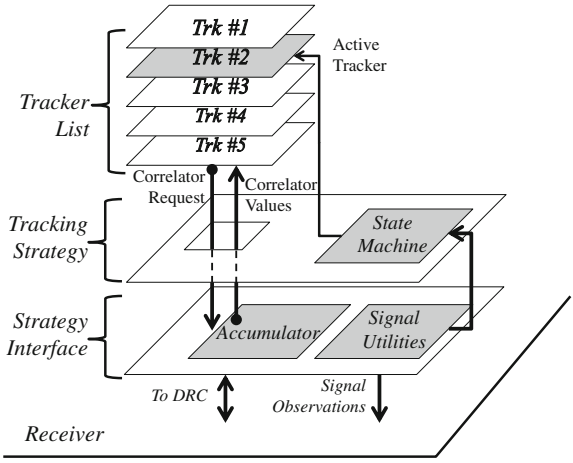

Fig. 69.3 A block diagram of the three-layer architecture of the composite tracking strategy illustrating containment of trackers (*Trk #1–5*) within tracking strategy definition and use of strategy interface with the receiver

choice of which of the tracked signals is the master signal is at the discretion of the composite strategy implementation but, typically, it will base the decision on the signal type and the status of the signal utilities. For example, the strategy may opt for a pilot-only signal, or one for which its secondary code has been synchronized, also, preference may be given to a signal which can provide the longer coherent integration period. Updates of the current active tracker are then aligned with epochs of the master signal, at which point the buffer of all accumulator values (corresponding to both master and slave signals) accrued since the last loop update is passed to the active tracker for processing. An example of this, for the Galileo E1 BC signals, is depicted in Fig. 69.4.

Another important feature of this tracking architecture is its focus on a multi-signal support. The design leverages resources such as the *vector*, *list* and *map* features of the C++ ISO standard library to effect an architecture which operates on an ensemble of quasi-synchronous[1] signals broadcast from a single transmitter. This multi-signal architecture can then be tailored to manifest any single- or multi-signal strategy (for example, data-pilot or dual-frequency), by specifying at the 'Composite Strategy' layer, the signals on which the active tracker operates. This approach confers many distinct advantages over architectures which augment, extend, or dither a single-signal strategy to achieve data-pilot operation, prominent among which being ease of use and re-usability.

[1] Although signals may have different centre frequencies and symbol (chipping) rates, they are synchronous in the sense that they experience the same time dilation induced by line-of-sight dynamics.

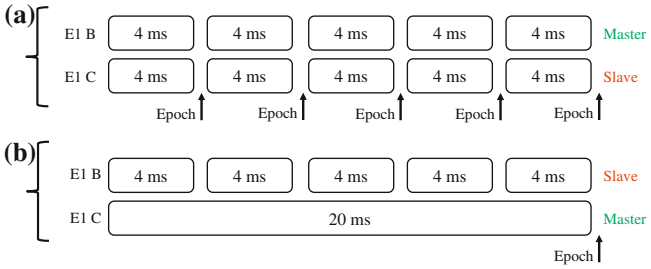

Fig. 69.4 Scheduling of tracking loop updates for a multi-signal tracking strategy operating on the Galileo E1 BC data-pilot pair before (**a**), and after (**b**), synchronization with the secondary code on E1 C

69.3 Tracking Strategies

The principle of the tracking strategy presented here is that the tracking strategy is only responsible for activating or de-activating its member tracking algorithms while each tracking algorithm (tracker) is be self-contained and build-time configurable only. By enforcing this principle, the tracking strategy need not be aware of any of the internal workings of the tracker, thus promoting simplified code and re-usability. Rather than re-tune the tracker as the prevailing conditions change, the strategy can simply select a more appropriately tuned tracker. This approach preserves the benefits of adaptive tracking algorithms without incurring the increased complexity.

The design of tracking strategies within GSNRx^{TM} involves two steps: first a list of available trackers must be populated with suitable tracker instances and, secondly, a state machine, defining when each tracker is to be used, must be defined. When populating a list of trackers, a designer may avail of a library of predefined tracker objects, or opt to design a custom algorithm. For example, the tracker objects used within this work include a selection of standard tracking algorithms [1, 3–5]:

- Fine Search: a continuous re-acquisition scheme involving a refined search in the code and frequency domains.
- pull-in FLL+DLL: a standard FLL and DLL incorporating a false-frequency lock detection algorithm.
- FLL+DLL: a standard FLL and DLL with configurable loop filters, code-aiding, and integration periods.
- PLL+DLL: a standard PLL and DLL with configurable loop filters, code-aiding, and integration periods.
- Kalman Filter: an iterated extended Kalman filter approach.

All implementations, excluding the Kalman filter,[2] support both single-signal and data-pilot operation. A given tracking strategy may contain many instances of the same tracker but each configured with differently, for example, stipulating various filter bandwidths or integration periods.

As mentioned previously, within the 'Composite Strategy' one tracker is active at any given time. This active tracker performs signal parameter estimation (code and carrier tracking) while the remainder are continuously synchronized with it. The active tracker is chosen by a synchronous, finite Mealy state machine containing a unique state corresponding to each of the available trackers and two further states, representing the 'Initialized' state at which the strategy begins and the 'Lost-Lock' state which indicates to the receiver that a re-acquisition may be necessary. The input to the state machine are a set of Boolean values given by conditional operations applied to the signal utilities and the trackers internal indicators. The inputs include: C/N_0, a phase-lock indicator (PLI), a frequency-lock indicator (FLI), bit and secondary code-synch status, navigation synch status, tracking time and, tracker internal lock and loss-of-lock indicators.

An example of such a state machine is depicted in Fig. 69.5, representing that of the GSNRx^{TM} weak-signal tracking strategy. The two fine-search trackers differ only in cell size, the second having both a smaller search space and smaller cells. The second of the two FLL+DLL trackers has the narrower filter bandwidths and, unlike the first, employs carrier aiding of the code loop. The variables ζ_i represent the various conditions which dictate changes of state and are defined in Table 69.1, note, however, the subtle difference between the tracker 'Lost Lock' status and the strategy 'Lost Lock' state. The flow rational is as follows: the Kalman filter tracker provides excellent steady-state tracking performance but is unreliable during the initial transient or pull-in stage. To reach this steady state, and to hand-over to the Kalman filter tracker, either a series of fine-search trackers, or a series of FLL and DLL trackers can be used. The fine search trackers are reliable under weak-signal conditions, but are relatively slow. In contrast, the FLL and DLL trackers are relatively fast, but are only suitable when the received C/N_0 is moderate or high. This system, therefore defaults to the fine-search trackers and, should it detect a sufficiently high C/N_0, will transition to the FLL and DLL branch of the state-machine.

Of course, the use of this architecture is not restricted to the above design and can be utilized to implement any multi-algorithm strategy. For example, one tracking algorithm (tracker) may be employed to absorb the initial transient, before a transition to another tracking algorithm more suited to steady state operation. Likewise, strategies which implement phase tracking with a PLL may first apply an FLL to the received signal, transitioning to the PLL only when the frequency uncertainty had reduced sufficiently. Indeed, context-aware strategies, which are becoming increasingly popular, can be readily implemented within this proposed

[2] This is not due to a fundamental incompatibility between the Kalman filter and data-pilot signals, but rather lack of development time.

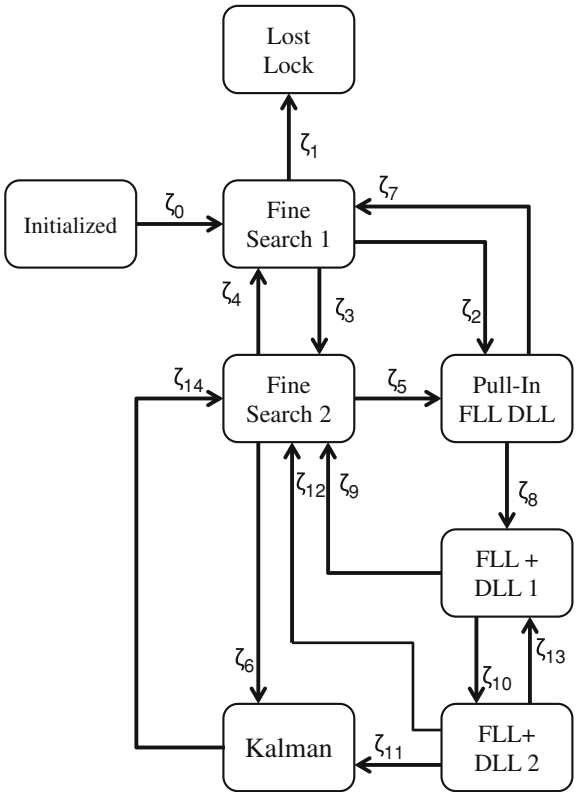

Fig. 69.5 An example of a state machine for a weak-signal tracking strategy

Table 69.1 Weak-signal tracking strategy conditional statements

ζ_0	–
ζ_1	$C/N_0 < 15.0$
ζ_2	$C/N_0 > 35.0$
ζ_3	Locked
ζ_4	$C/N_0 < 15.0$
ζ_5	$C/N_0 > 35.0$
ζ_6	Locked
ζ_7	(Lost lock) OR ($C/N_0 < 15.0$)
ζ_8	Locked
ζ_9	(Lost lock) OR ($C/N_0 < 15.0$)
ζ_{10}	FLI > 0.6
ζ_{11}	FLI > 0.85
ζ_{12}	$C/N_0 < 15.0$
ζ_{13}	FLI < 0.2
ζ_{14}	$C/N_0 < 15.0$

architecture. In all of these cases, the desired strategy can be effected by populating a list of suitably configured trackers and defining a protocol, via the state machine, to transition from one to the next.

As discussed in Sect. 69.2, the scheduling of correlator values from the DRC is handled by the 'Composite Strategy' layer. At any given tracking epoch, the buffer of available correlator values is presented to the active tracker. The tracker itself is responsible for both dictating the correlator request and processing these resultant correlator values and, so, the decision to perform a multi-signal algorithm (for example, a data-pilot, or dual-frequency tracking scheme) is at the discretion of the current active tracker. This architecture, therefore, not only supports these multi-signal algorithms, but can provide a blend of single- and multi-signal algorithms, by appropriately populating 'Composite Strategy' with the appropriate trackers.

69.4 Test Scenario

While the previous sections have detailed the design of a flexible tracking strategy architecture, and the means by which it can be configured, the question of which tracking algorithms to use, when they should be used, and to which signals they should be applied, remains. To explore this question, this section presents a simple pedestrian navigation scenario within which the performance of a selection of trackers can be examined.

Tracking performance was examined by processing a set of simulated IF data, representing a pedestrian navigation scenario. The data contained three front-end channels, the first was configured with a centre frequency of 1569.0 MHz to receive BeiDou B1I signals [6], the second was configured with a centre frequency of 1575.0 MHz to receive GPS L1 C/A and Galileo E1bc signals [7, 8], and the third was configured with a centre frequency of 1602.0 MHz to receive the GLONASS L1 OF signals [9]. A complex sample rate of 8.0 MHz was used on each front-end channel. The simulation was designed to provide one satellite from each of the four systems at the same elevation and azimuth, of 52°and 90°, respectively, relative to the receiver. Although not necessarily realistic from a constellation perspective, this ensured that the receiver observed the same line-of-sight (LOS) dynamics on each signal, facilitating a more fair comparative analysis, from a signal processing perspective.

In total, eleven minutes of data were generated, during which time the user traversed a rectangular trajectory measuring 50 m east-west and 100 m north-south with a cornering radius of 5 m. The received signal strength on all satellites was set to an initial value of 48 dBHz, at which it was held constant for a period of one minute and, subsequently, reduced at a rate of 0.05 dB/s reaching a final value of 18 dBHz at the end of the eleventh minute. A second set of data was collected as a reference, using the same simulation configuration but without applying attenuation to the received signal. This data-set served as a reference against which

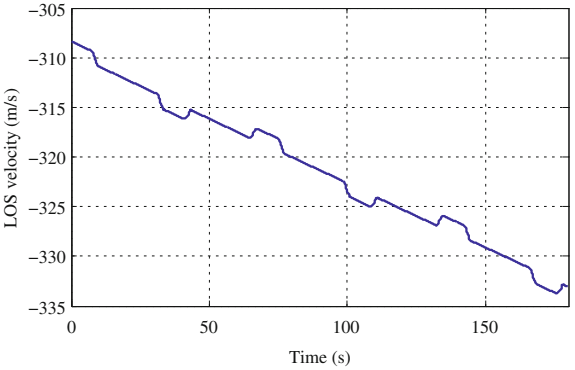

Fig. 69.6 Line of sight velocity between the satellites and the receiver fore the first three minutes of the simulated dataset

tracking performance was measured. The LOS velocity observed on these received signals is depicted in Fig. 69.6.

Rather than testing full tracking strategies, the focus of this experiment was to assess the relative tracking performance and efficiency of some key tracker objects. In particular, the FLL-DLL, the PLL-DLL and the Kalman filter trackers were examined. To observe the tracking performance, the receiver was executed and, following successful acquisition, the strategy was stalled in the tracking state of interest for the duration of the data-set. For the GPS, GLONASS and BeiDou signals, a single-signal strategy was employed, while for the Galileo signals a single-signal strategy was applied to the E1b signal, another single-signal strategy was applied to E1c and a data-pilot strategy was applied to the E1bc pair. In this way, the benefits of employing a signal specific custom tracking strategy can be directly measured. In this case, a the difference between employing a single-signal, pilot-only or data-pilot strategy on the Galileo E1bc pair is examined.

Table 69.2 details some of the characteristics of the received signals, notable amongst which are the slope and the coherent integration period. The absolute slope of the ranging code correlation function (denoted Abs. Slope), evaluated at a code offset of one half of the corresponding code spacing (denoted E-L Spacing), is a key contributor to discriminator-based code tracking performance. The maximum attainable coherent integration period (denoted Coh. Int. Period),

Table 69.2 Signal details

	GPS L1 C/A	GLONASS L1 OF	BeiDou E1BI	Galileo E1 B	Galileo E1 C
Chip rate (Mcps)	1.023	0.511	2.046	1.023	1.023
Code period (ms)	1	1	1	4	4
Data modulated	y	y	y	y	n
Coherent integration period (ms)	20	20	20	4	∞
Wavelength (cm)	19.0	18.7	19.1	19.0	19.0
E-L spacing (chip)	0.4	0.4	0.4	0.4	0.4
Absolute slope (m^{-1})	3.4e-3	1.7e-3	6.8e3	13.2e-3	50e-3

restricted here by the presence of data modulation, has implications for tracking accuracy and sensitivity. In all receiver configurations, the coherent integration period adjusted to 20 ms where possible (but, of course, was limited to 4 ms on the Galileo E1b signal). For data-modulated signals, Costas-style carrier phase and frequency discriminators were employed where appropriate and coherent discriminators applied otherwise. Similarly, when the carrier phase was tracked, where appropriate, an early-minus-late in-phase discriminator was used while an early-minus-late envelope discriminator was used when only the carrier frequency was tracked.

The reference data-set was tracked using the weak-signal tracking strategy described in Sect. 69.3. The test data-set was tracked with each of the tracking algorithms under examination. In each case a log of the estimated signal parameters was recorded, at a rate of 1.0 kHz. Estimates of tracking errors were made by analyzing the difference between the log corresponding to the reference data and that of the attenuated data. The results are discussed in Sect. 69.5.

69.5 Tracking Performance

The results of the experiment described in Sect. 69.4 are discussed here. Results are presented for FLL+DLL, PLL+DLL, and Kalman filter tracking algorithms, using single-signal implementations for all signals and, also, using a data-pilot combining algorithm for the Galileo E1bc pair. One important point to consider here is that the different tracking algorithms have not been tuned to provide compatible tracking results and, so, a direct comparison between, for example, the FLL+DLL tracking error and that of the PLL+DLL, is not valid. Rather, these results are intended to provide insight into the relative performance of different signals for a given tracking algorithm. For example, a comparison of the frequency tracking error observed when using either the PLL+DLL or FLL+DLL trackers on the GPS L1 C/A signal is not meaningful whereas a comparison of the frequency tracking error observed when using the FLL+DLL tracker on either the GPS L1 C/A or BeiDou B1I is quite useful.

Figure 69.7a, b and c show the root-mean square (rms) code phase tracking error versus received signal C/N_0 for each of the three tracking algorithms. In both Fig. 69.7a, b, the code tracking is implemented using a standard DLL algorithm, the difference being that the former employs an early–minus-late envelope discriminator as it tracks only carrier phase, while the latter can avail of an early-minus-late in-phase discriminator as the phase is tracked by a PLL. The results clearly confirm the well known result that the tracking error is inversely proportional to the slope of the ranging code auto correlation function, whereby GLONASS exhibits the poorest tracking precision and strategies which include the Galileo E1c signal exhibit the best. Evident also are the effects of integration period for both the PLL and FLL algorithms, whereby the Galileo E1b signal looses lock at a higher C/N_0 in all cases, due to the 4 ms limit imposed by data-

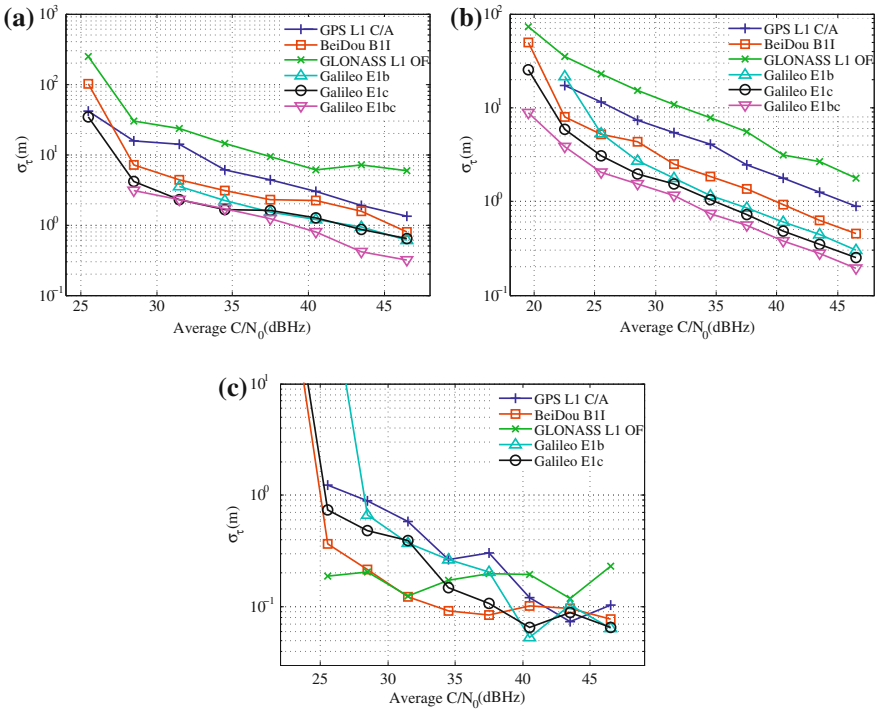

Fig. 69.7 Code phase tracking error, in metres, for the FLL+DLL, PLL+DLL and Kalman filter tracking architectures. **a**, **b** The PLL+DLL tracking architecture. **c** The Kalman filter tracking architecture

modulation. Interestingly, for the PLL+DLL algorithm, the Galileo E1bc data-pilot algorithm provides the best performance, owing to the lack of data modulation on the pilot channel and the sharpness of the ranging code correlation function, while for the FLL+DLL algorithm, the Galileo E1c algorithm performs best. This is perhaps the result of the carrier aiding applied to the code loop which, for the data-pilot combining FLL, as illustrated in Fig. 69.8a, performs more poorly than the pilot-only implementation.

In contrast to the differences in code tracking performance between the various signals when using FLL+DLL or PLL+DLL algorithms, Fig. 69.7c suggests that the Kalman filter can provide a similar level of code tracking accuracy for all signals, irrespective of the modulation scheme. The only exception being the tracking of Galileo E1b which, due to a reduced coherent integration period of 4 ms, loses lock at a higher C/N_0 than the other signals.

Figure 69.8a and b show the root-mean square (rms) carrier frequency tracking error versus received signal C/N_0 for FLL+DLL and the Kalman filter tracking algorithms. Examining Fig. 69.8a, it is clear that the signals which can offer a coherent integration period of 20 ms perform the best, with the 4 ms integration period of the Galileo E1b signal resulting in significantly poorer tracking

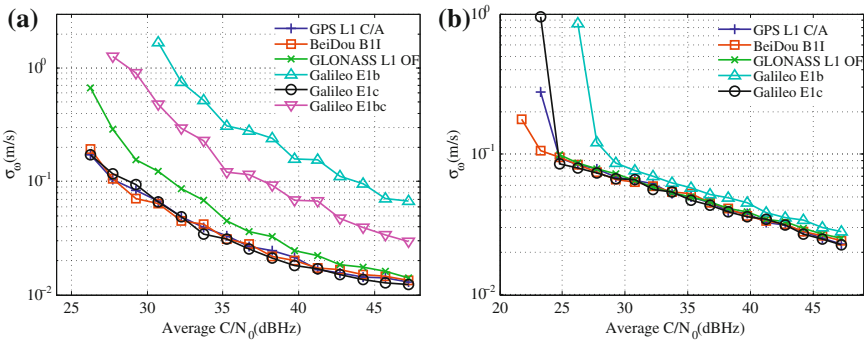

Fig. 69.8 Carrier frequency tracking error, in metres per second, for the FLL+DLL and Kalman filter tracking architectures. **a** The FLL+DLL tracking architecture. **b** The Kalman filter tracking architecture

performance. Surprisingly, the algorithm which combines the Galileo E1bc data-pilot pair, performs by almost as poorly. This observation betrays a sub-optimal data-pilot combining, indicating that, perhaps, data-pilot combining is not as straightforward for carrier frequency tracking as it is for carrier phase or code phase. Similar to the FLL+DLL algorithm, the carrier frequency tracking performance of the Kalman filter, presented in Fig. 69.8b, is similar for all signals which can support a 20 ms coherent integration period, the Galileo E1b signal, once again, suffering from the effects of a 250 bps data modulation. Another curious feature of these results is the relationship between received signal strength and carrier frequency tracking error. For the Kalman filter algorithm, the rms tracking error exhibits a linear inversely proportional relationship with the received C/N_0. The FLL, in contrast, incurs a much more rapid degradation in tracking performance with reducing C/N_0.

Finally, the carrier phase tracking performance of the PLL+DLL and Kalman filter tracking algorithms is presented in Fig. 69.9a and b. Here the benefits of the

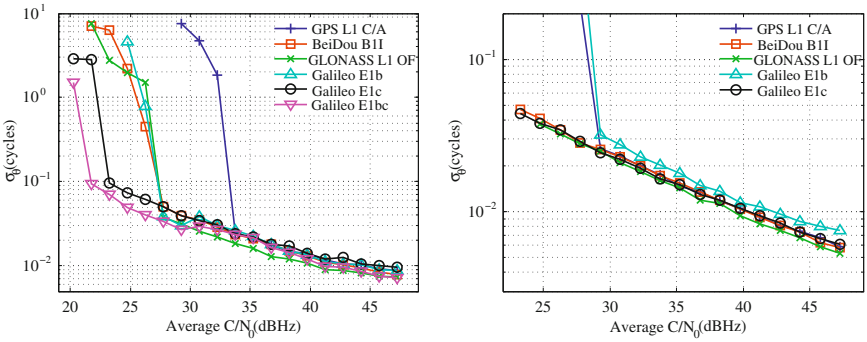

Fig. 69.9 Carrier phase tracking error, in cycles, for the PLL+DLL and Kalman filter tracking architectures. **a** The PLL+DLL tracking architecture. **b** The Kalman filter tracking architecture

pilot signal become pronounced as the Galileo E1c and Galileo E1bc data-pilot pair significantly outperform the other signals when tracked by a PLL+DLL algorithm. For the Kalman filter tracking algorithm, the limited coherent integration period afforded by the Galileo E1b signal, once again results in a reduced tracking precision, relative to the other signals.

It is worth commenting on the tracking error incurred while tracking the GPS L1 C/A signal, as shown in Fig. 69.9a. The eleven minute data-set was tracked from beginning to end and the rms tracking error calculated over successive 30 s windows. Should the tracking algorithm begin to loose lock early in the data-set, the accumulated phase error will continue to contribute to the rms error for later time-points. Examining Fig. 69.9a, it appears that the receiver lost lock on the carrier phase at approximately five minutes (corresponding to a C/N_0 of 34 dBHz) thus rendering tracking error estimates for later time-points (lower C/N_0 values) invalid.

69.6 Conclusions

A novel GNSS tracking architecture has been presented in this work which address receiver design challenges which arise when implementing a multi-constellation receiver. The design features a highly modular design that facilitates fast and efficient tracking algorithm design and definition by promoting a high degree of module and code re-use. Moreover, this design lends itself to simple and rapid adaptation to new signals as the become available. The effectiveness of this architecture was demonstrated and tested in the context of a pedestrian navigation scenario using the BeiDou, Galileo, GLONASS and GPS constellations.

While the tracking performance results presented here cannot be interpreted as a commentary on the relative performance of the various tracking algorithms, they do serve to indicate the suitability and efficiency of each strategy when applied to signals from each of the constellations. That is, for a given tracking algorithm, be it the PLL-DLL, the FLL-DLL, or the Kalman filter, the relative suitability of each signal to that algorithm can be examined, in particular, the cost-benefit tradeoff of implementing a data-pilot tracking algorithm can be assessed.

In terms of ranging code tracking, results show that the Kalman filter based tracking algorithm provides the best overall performance, exhibiting the least sensitivity to modulation scheme or coherent integrations period. For the DLL-based tracking algorithms, however, the attainable performance is sensitive to the absolute slope of the ranging code auto-correlation function.

In contrast, the carrier frequency tracking performance of both the FLL-based and Kalman filter based tracking algorithms, exhibit a similar trend across the various signals. The tracking performance is most heavily influenced by the attainable coherent integration period and the only notable difference being the relationship to the prevailing C/N_0, it being linear for the Kalman filter and not so for the FLL.

The carrier phase tracking performance of both the PLL-based algorithms and the Kalman filter algorithm, also exhibits some interesting features. It is apparent that the PLL-based algorithm is relatively insensitive to the coherent integration period afforded by the tracked signal and, at least for moderate to high C/N_0 values, achieves comparable tracking performance for all of the signals. In contrast, the Kalman filter clearly achieves a poorer precision for the Galileo E1b signal, which can only support coherent integration period of 4 ms. Collectively, these observations suggest that, for a given target application, environment and set of constellations, there are some clear advantages to employing one tracking a algorithm over another. There does not, therefore, appear to be an individual algorithm which has a distinctly universal appeal.

References

1. Kaplan ED (ed) (2006) Understanding GPS: principles and applications, vol 1, chap 5, pp 179–194. Artech House Inc. ISBN 1-58053-894-0
2. VanDierendonck AJ (1996) Global positioning system: theory & applications (Volume One) (Progress in Astronautics and Aeronautics), chap 8, pp 329–408. AIAA (American Institute of Aeronautics & Ast, 1996). ISBN 1-56347-106-X
3. Misra P, Enge P (1996) Global positioning system, signals, measurements and performance, vol 2. Ganga-Jamuna Press. ISBN 0-9709544-1-7
4. Curran J, Lachapelle G, Murphy C (2012) Improving the design of frequency lock loops for gnss receivers. Aerosp Electron Syst, IEEE Trans 48:850–868
5. Gleb A (1974) Applied optimal estimation. MIT Press, Cambridge, Mass. ISBN 9780262570480
6. BeiDou Navigation Satellite System (2013) Signal in space interface control document. http://www.beidou.gov.cn/attach/2012/12/27/201212273da29c5eb8274deb8cd2b178228ba2bd.pdf. (Accessed: 01 Jan 2013)
7. GPS Navstar JPO (2013) Navstar GPS Space Segment / Navigation User Interfaces. http://www.navcen.uscg.gov/pubs/gps/icd200/default.htm. (Accessed: 01 Jan 2013)
8. Galileo Project Office (2013) Galileo OS SIS ICD http://ec.europa.eu/enterprise/policies/satnav/galileo/files/galileo_os_sis_icd_revised_3_en.pdf. (Accessed: 01 Jan 2013)
9. GLONASS Navigation Satellite System (2013) Interface Control Document. http://gauss.gge.unb.ca/GLONASS.ICD.pdf. (Accessed: 01 Jan 2013)