The GNSS-Transceiver: Using vector-tracking approach to convert a GNSS receiver to a simulator; implementation and verification for signal authentication

Daniel S. Maier, *Universität der Bundeswehr München*
Kathrin Frankl, *Universität der Bundeswehr München*
Prof. Thomas Pany, *Universität der Bundeswehr München*

BIOGRAPHIES

Daniel Simon Maier has a professional training as a technical draftsman and received his bachelor in Physics in 2015 and his master in Applied and Engineering Physics in 2017 from the Technical University of Munich (TUM), Germany. Since 2017 he is a research associate at the Institute of Space Technology and Space Applications of the Universität der Bundeswehr München. His current research interests include GNSS signal generation, signal authentication and signal performance analysis.

Kathrin Frankl received her German Diploma in Mathematics from Technical University of Munich. Since 2014, she is working at the Institute of Space Technology and Space Applications at Universität der Bundeswehr München, Germany. Her research focus is on integrity and authentication algorithms for terrestrial and space applications.

Prof. Thomas Pany is professor at the Universität der Bundeswehr München at the faculty of aerospace engineering where he teaches satellite navigation. His research includes all aspects of navigation ranging from deep space navigation over new algorithms and assembly code optimization. Currently, he focuses on GNSS signal processing for Galileo second generation, GNSS receiver design and GNSS/INS/LiDAR/camera fusion. To support these activities, he is developing a modular GNSS testbed for advanced navigation research. Previously, he worked for IFEN GmbH and IGASPIN GmbH and is the architect of the ipexSR and SX3 software receiver. He has around 200 publications including patents and one monography.

ABSTRACT

Studying and testing new and possible future GNSS signals and navigation messages require a signal generator which is flexible and fully modifiable. To overcome the need of implementing a signal generator from scratch, we present a way to modify an excising GNSS software receiver (SR) into a software transceiver (ST). The ST reuses the SR modules and the infrastructure for the signal generation. The modification approach is based on exploiting the vector tracking feature of the software receiver. Due to the replacement of the position in the vector tracking loop, it is possible to manipulate the numerical controlled oscillator (NCO) and thereby force the code and carrier generator to generate a signal replica which fits to the induced position. Multiplying the replica with the desired symbol value and the desired amplitude yields an entire line of sight signal. The replica signals of all satellites in tracking match the predefined user trajectory. Saving the added replica signals results in a signal stream of intermediate frequency (IF). The ST is able to track, generate or re-generate tracked signals. The concept and an implementation approach is presented. The signal quality is compared to real signals. The possibilities of the ST are shown by generating Galileo OS signals with navigation message authentication (NMA) and generating a secure code estimation (SCE) attack signal.

INTRODUCTION

Research in the field of GNSS signal performance under spoofing, jamming and multipath comes along with the need of reproducing this channels, signals, and scenarios as good as possible. The performance of currently used signals can be analyzed quite easily as it is possible to use the genuine transmitted signals or use state of the art commercial of the shelf (COTS) signal generators for the recreation of the setup under investigation. This becomes, however, much more difficult if new signals, channel structures or

navigation massages are under test. Some commercial signal generators give the possibility to implement new signals and use its own navigation message but in a very limited and restricted extent.
To overcome these restrictions, it would be necessary to implement an own signal generator [1] to have all the possibilities in creation and testing signals in all desired scenarios. However, implementing a sophisticated signal generator from scratch is a huge, difficult and time consuming task.
To avoid this task, we present a way to convert a software receiver (SR) into a GNSS software transceiver (ST), to track and generate GNSS signals. With this approach it is possible to reuse the sophisticated and optimized infrastructure of the SR for the signal generator. We exploit the fact that each SR has to create an estimated replica of code and carrier for the correlation. The key element in our approach is the usage of the software receiver vector-tracking architecture to create the desired line-of-sight parameters for updating the numerically controlled oscillator (NCO) and therefore the code and carrier replica generation.
Feeding the Line-of-Sight module with the position, velocity and time (PVT) of a pre-defined receiver trajectory gives an easy opportunity to manipulate the vector tracking loop to generate replicas as needed to recreate signals which represent the receiver movements. In addition, the desired symbols and amplitudes have to be provided to the tracking loops. Multiplying the replica signal with amplitude and symbol gets a new channel IF signal patch. Adding up all channels results in an IF signal stream for a total or even multiple constellations. If required additional Gaussian white noise can be added before the IF signal patch is written into the output file. Ionospheric and tropospheric influences are simulated as the line-of-sight parameters are calculated using ionospheric- and tropospheric-models. The ST is able to track, generate or re-generate tracked signals. A similar approach for re-generating tracked signals to upgrade existing receivers was presented by Humphreys et al.[2].
First, the SR and the vector tracking architecture is briefly presented and explained. Thereafter, the modifications are described to convert a software receiver into a software transceiver, using the vector tracking approach. The implementation as well as error sources are described, using the MuSNAT SR as foundation. At the end, we compare the generated signals with real signals and show the power of the ST by generating Galileo OS signals with navigation message authentication (NMA) as well as generating secure code estimation (SCE) attacker signals.

SOFTWARE RECEIVER AND VECTOR TRACKING ARCHITECTURE

Vector tracking is an old topic in the GNSS community and was first proposed in 1980 [3] by Copps et al. Since then, many papers [4, 5], articles [6] and books [7,8] where published, describing and studying the vector tracking topic in its full complexity and broadness with its advantages and drawbacks. The given references are by far not complete and can only be a starting point for interested readers.
To understand the necessary modifications for transforming a vector tracking based SR into a signal generator (software transceiver, ST), a brief description of conventional independent channel tracking architecture and the used vector tracking architecture follows.
All SRs have two main modules defining the processing workflow betweenthe IF sample stream as receiver input and the receiver position, velocity and timing (PVT) information as receiver output: (i) the signal processing unit with the tracking loops as core part and (ii) the navigation processor.. The signal processing unit, specially the tracking loops, process sample batches of the IF sample stream in sequential order, and extract the pseudorange ρ, pseudorange-rate $\dot{\rho}$ and possibly the pseudorange-rate-rate $\ddot{\rho}$ for all tracked satellite signals. This parameters are passed to the navigation processor which then determines the receiver's PVT.
In a conventional SR, separate and independent tracking loops are used to track each satellite signal standalone. The signal processing is schematically shown in Figure 1. Each tracking loop consists of a correlator, integrator, discriminator, loop filter, NCO as well as a code and carrier generator. The code and carrier generator creates a replica of the received satellite signal. The replica generation parameters are controlled by the NCO. An early (E), prompt (P) and late (L) version of the generated replica signal is then correlated with IF sample batch. The values for the integrated correlation signals are dumped and handed over to the discriminator. The discriminator evaluates the E, P and L values for in-phase (I) and quadrature-phase (Q) and estimate the code time delay error $\delta\tau$ as well as the carrier frequency error δf and the carrier phase error $\delta\phi$ of the replica signal. Over the loop filter these estimated error values are used to update, i.e., speed up or slow down the NCO so that the replica signal follows the satellite signal. A detailed description of the signal processing is presented in [14]. With the replica values, a good estimation of the satellite signal parameters (code delay τ, carrier Doppler f_d and carrier phase ϕ) can be determined. The satellite signal parameters are used to calculate the pseudorange ρ, pseudorange-rate $\dot{\rho}$ and the pseudorange-rate-rate $\ddot{\rho}$ which are needed for the PVT determination in the navigation module. The update rate of the tracking loops is in the order of 50 to 1000 Hz, whereas the navigation processor works with a common update rate in the order of 1 to 10 Hz. The Loop Filter with its bandwidth is used to smooth the high rate values for the NCO update. Therefore the current smoothed output values of the Loop Filter are used by the Navigation processor at dump time.

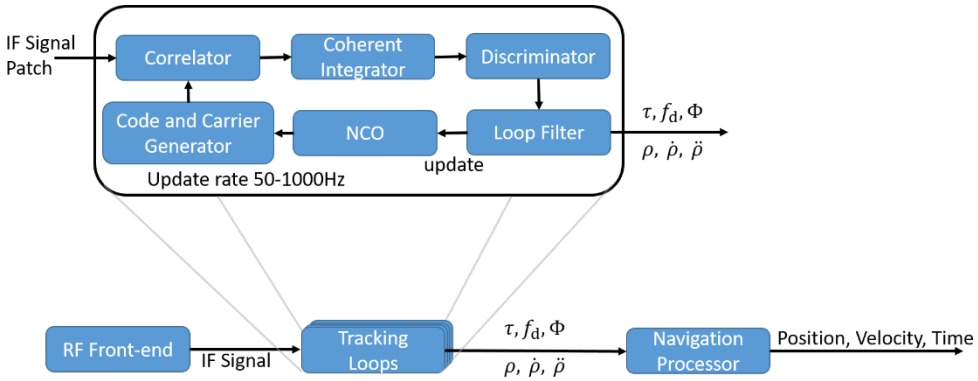

Figure 1: Sketch of independent channel tracking.

In a conventional SR all tracking loops work independently as mentioned above. This leads to the drawback that the tracking loop can easily loose the satellite signal during short periods of signal blockage or signal fading. Even if the signal outage is short, the receiver has to start with a re-acquisition of the lost signal which is time and processing power consuming. The basic idea of vector tracking is to support the single tracking loop with redundant information of the other tracking loops, so the single tracking loop is able to bridge periods of a weak signal or signal blockage. All satellite signal parameters are defined with respect to the position and velocity of receiver and satellite. Therefore, the PVT solution of the navigation processor represents the compressed information of the redundant tracking loop outputs. The vector tracking task is to feedback this redundant information to each single tracking loop, or more precise, to feedback the best estimation of it. This task can be realized in various forms, some are more and some are less complex. The more sophisticated approaches usually use some kind of extended Kalman filter (EKF) for PVT estimation (if this filter includes also inertial measurements, it is called a deep GNSS/IMU integration). The implementation used in this work is sketched in Figure 2. Here, the PVT algorithm is intentionally left unspecified. An additional module to determine the line-of-sight (LOS) parameters (ρ_i, $\dot{\rho}_i$ and $\ddot{\rho}_i$at time t_i) for all tracked satellites is needed. As already mentioned, the LOS parameters are specified with respect to the position and the dynamic of satellite and receiver, and can easily be determined with geometrical considerations. The satellite position is known from the ephemeris in the navigation message. The navigation module will update the LOS parameter with a rate of 10 Hz (100ms). The operational update rate of the tracking loop, however, is around 1000Hz (1ms). To overcome this lack of sampling points, we perform a quadratic extrapolation of the LOS parameters, assuming a constant acceleration until the next LOS update. The equations for the code extrapolation look like:

$$\rho_{i+1} = \rho_i + \dot{\rho}_i \Delta t + \frac{1}{2}\ddot{\rho}_i \Delta t^2 \quad (1)$$

$$\dot{\rho}_{i+1} = \dot{\rho}_i + \ddot{\rho}_i \Delta t \quad (2)$$

$$\ddot{\rho}_{i+1} = \ddot{\rho}_i \quad (3)$$

With $\Delta t = t_{i+1} - t_i$. This extrapolation is obviously wrong in the long term but for the short time span of 0.1s the approximation works quiet well for systems with moderate dynamics. These extrapolated LOS parameters are used to update the NCO in a hard reset fashion. This means that the replica signal changes during the 0.1s in a smooth fashion, i.e., determined by Equation (1)-(3). After the update of the LOS parameters, however, a jump occurs in the time domain of the replica signal parameters. The error of the extrapolation and the jumps of the replica signal are compensated in pseudorange calculation, as the discriminator determines $\delta\tau$, δf and $\delta\phi$. The carrier NCO is updated in a different way. Here, the Doppler values are integrated to determine the carrier phase. This means that also the Doppler and phase values are error-prone but continues.

Due to the circumstance that no Loop Filter is needed any more to determine the smoothed update values for the NCO, a new decimation filter is needed to provide smoothed pseudorange estimation values to the navigation processor. This can be a Kalman filter but we use a polynomial fit method, where a vector of discriminator values is fitted with a polynomial function to determine the signal parameters at dump time.

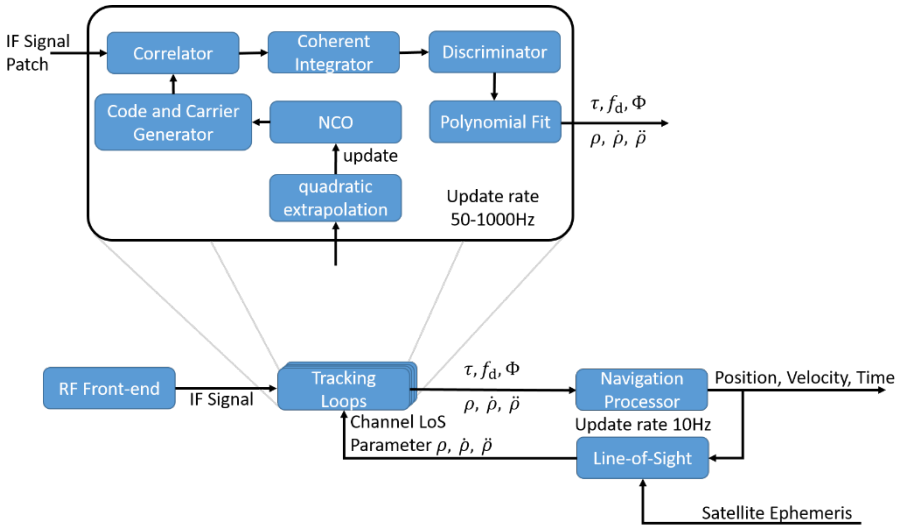

Figure 2: Vector Tracking Receiver Architecture.

SOFTWARE TRANSCEIVER

A SR needs to mimic and create the satellite signals as good as possible to be able to track the signal which is hidden under the noise floor. So, all SR are already signal generators. The only difference is that a SR reproduces an existing signal and is not creating a new one. To transform the SR vector tracking architecture into an actual signal generator, two main modifications (shown in Figure 3) are needed to be carried out.

First, one needs access to the created replica signal which consists of code $C(t)$ and carrier $\cos[2\pi(f_L + f_d)t + \phi]$ in the tracking loop. Multiplying the replica signal $\hat{r}_{IF}$ with the estimated amplitude $A(t)$ and the navigation symbol $D(t)$ for this batch, one gets a renewed version of the received signal:

$$\hat{r}_{IF}(t, \tau, f_d, \phi) = A(t-\tau)D(t-\tau)C(t-\tau)\cos[2\pi(f_L + f_d)t + \phi], \quad (4)$$

where τ is the code delay and t the receiving time. Summing up all replica signal from each tracking loop gives a renewed version of the IF signal without noise. Appending and saving the IF signal batches in sequential order in a file, yields a renewed version of the IF stream file. Before saving the batches one can add additive white Gaussian noise (AWGN) to mimic the true record as good as possible.

With the second modification, it is possible to influence the replica generation in a way that a desired user trajectory can be reproduced. To do so, the working principle of the vector tracking is exploited. Not the PVT solution of the navigation processor is feeded back but a new PVT solution originates of a before defined user trajectory. Due to the replacement of the PVT solution, the correlation, integration, discrimination, polynomial fit and PVT determination processes are obsolete and can be bridged or turned off to save computing power. A more detailed discussion of the modifications follows.

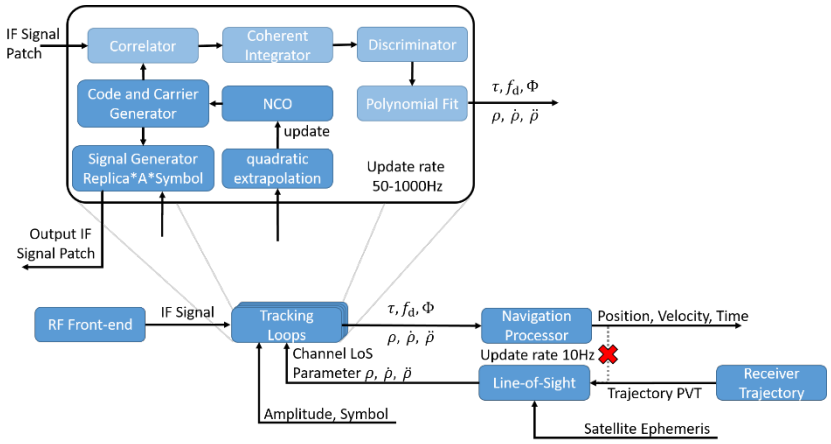

Figure 3: Sketch of signal generation architecture, using the vector tracking infrastructure.

IMPLEMENTATION

The following section gives insight in the GNSS software transceiver implementation following the concept described above. The implementation is based on the ipexSR software receiver [9], the software packet was renamed into Multi Sensor Navigation Analyzing Tool (MuSNAT) [10]. The section 'Signal generation' describes the implementation steps needed to modify the replicas with amplitude, data bit and noise. Section 'User PVT trajectory feedback' shows the modification steps needed to replace the original PVT solution with a user defined trajectory.

Signal generation

The MuSNAT SR processes IF sample packages of ~30 ms. This IF sample package is divided into IF sample frames with a length of ~ 0.8ms. The sample frame packages are processed in sequential order. In each processing step, the sample frame is distributed to all Master Channels. Each Master Channel processes one satellite signal and is composed of one or two tracking channels depending on whether the satellite signal has one (data) or two (data and pilot) components. All these channels process the sample frames successively. Due to this sequential workflow, we are able to allocate memory in the receiver with the same size as the IF sample package and handover memory pointer to the tracking loops. So all tracking loops can add their locally created satellite signal to this memory segment. When the IF sample package processing is finished, the generated IF sample package is finally processed in total. In this final processing step, AWGN can be added and output conversions can be performed. In the conversion step, the generated IF stream output format is set. Internally, the replica signals are stored as double float values, to ensure a minimal quantization errors in the signal addition process. For the saving process, the double float values can be converted to the desired output format. In this work, either a real-valued 8bit or an I-Q-8bit conversion is used.

For the signal amplitude, we use predefined $C/N0$ values for each satellite in tracking. The amplitude parameter A_s for each satellite s is defined as a relative amplitude. The strongest satellite is assigned an amplitude of 1, weaker signals are assigned a corresponding smaller amplitude. The calculations were done with the equation

$$A_s = 10^{\frac{C/N0_s - C/N0_{max}}{20}}. \tag{5}$$

The standard deviation σ_{AWGN} of the AWGN was calculated with

$$\sigma_{AWGN} = \sqrt{\frac{A^2}{2} * \frac{f_S}{2} * 10^{-\frac{C/N0_{max}}{10}}}, \tag{6}$$

where $A = 1$, as the AWGN is adapted to the strongest signal. The equations above were derived from [12]

$$C/N0 = 10\log(SNR * BW) = 10\log\left(\frac{P}{N} * \frac{f_S}{2}\right), \tag{7}$$

where power $P = \frac{A^2}{2}$, $N = \sigma^2_{AWGN}$ and f_S equals the sampling rate. SNR and BW denote the signal-to-noise ratio and the bandwidth. The amplitude value $A_s(t-\tau)$ can also be used to implement a land mobile satellite (LMS) channel model [11] to simulate realistic environments.

The last missing part for the signal generation represents the navigation symbol D. The navigation message has to be pre-produced and is stored in a receiver internal data base. $D(t-\tau)$ is selected for each sample frame accordingly to the satellite transmitting time.

User PVT trajectory feedback

For the user trajectory input we use a text file of the format:

t_{traj}	p_x	p_y	p_z	v_x	v_y	v_z
0.0	4115730.103	800016.890	4790936.102	0.000	0.000	0.000
0.5	4115730.082	800016.886	4790936.121	-0.093	-0.018	0.082
0.1	4115730.010	800016.872	4790936.185	-0.196	-0.038	0.173
⋮						

Where t_{traj} is the trajectory time, p_x, p_y and p_z represent the user positon in the WGS 84 system in meter and v_x, v_y and v_z represent the user velocity in the WGS 84 system in m/s. The trajectory points (2 Hz) need to be interpolated to the PVT update time steps (10 Hz). To do so, a 4th degree spline interpolation approach was chosen. The 4th degree spline interpolation is defined with the following three equations:

$$S_j(t) = a_j^4 \left(t - t_j\right)^4 + a_j^3 \left(t - t_j\right)^3 + a_j^2 \left(t - t_j\right)^2 + a_j^1 \left(t - t_j\right) + a_j^0 \tag{8}$$

$$\dot{S}_j(t) = 4a_j^4 \left(t - t_j\right)^3 + 3a_j^3 \left(t - t_j\right)^2 + 2a_j^2 \left(t - t_j\right) + a_j^1 \tag{9}$$

$$\ddot{S}_j(t) = 12a_j^3 \left(t - t_j\right)^2 + 6a_j^3 (t - t_j) + 2a_j^2 \tag{10}$$

Where $S_j(t)$, $\dot{S}_j(t)$ and $\ddot{S}_j(t)$ are the position, velocity and acceleration at time t, j indicates the current fix point. The five parameters $a_j^4, a_j^3, a_j^2, a_j^1$ and a_j^0 define the interpolation function between the fix points j and $j+1$. There are four static conditions in position and velocity and one continuity condition in the acceleration:

$$S_j\left(t_j\right) = a_j^0 = p_{x_j} \tag{11}$$

$$S_j\left(t_{j+1}\right) = a_j^4 \, \Delta t^4 + a_j^3 \, \Delta t^3 + a_j^2 \Delta t^2 + a_j^1 \Delta t + a_j^0 = p_{x_{j+1}} \tag{12}$$

$$\dot{S}_j\left(t_j\right) = a_j^1 = v_{x_j} \tag{13}$$

$$\dot{S}_j\left(t_{j+1}\right) = 4a_j^4 \, \Delta t^3 + 3a_j^3 \, \Delta t^2 + 2a_j^2 \Delta t + a_j^1 = v_{x_{j+1}} \tag{14}$$

$$\ddot{S}_j\left(t_j\right) = 2a_j^2 = \ddot{S}_{j-1}\left(t_j\right) = 12a_{j-1}^4 \Delta t + 6a_{j-1}^3 \Delta t + 2a_{j-1}^2 \tag{15}$$

With $\Delta t = (t_{j+1} - t_j)$. Following equations are used to update the parameters.

$$a_j^0 = p_{x_j} \tag{16}$$

$$a_j^1 = v_{x_j} \tag{17}$$

$$a_j^2 = \frac{1}{2}\ddot{S}_{j-1}\left(t_j\right) \tag{18}$$

$$a_j^3 = 4\frac{p_{x_{j+1}} - p_{x_j}}{\Delta t^3} - \frac{v_{x_{j+1}} + 3v_{x_j}}{\Delta t^2} - \frac{\ddot{S}_{j-1}(t_j)}{\Delta t} \tag{19}$$

$$a_j^4 = -3\frac{p_{x_{j+1}} - p_{x_j}}{\Delta t^4} + \frac{v_{x_{j+1}} + 2v_{x_j}}{\Delta t^3} + \frac{\ddot{S}_{j-1}(t_j)}{2\Delta t^2} \tag{20}$$

The interpolations in x, y and z dimension are done independent from each other. The PVT solution also includes values for clock error and clock drift as well as accuracy estimations (mean radial spherical error, MRSE) for positon and velocity. Clock error and clock drift are set to zero, position and velocity accuracy is set to $0.001\ m$ and $0.001\ m/s$. These parameter can be defined in the trajectory input file and set like needed. The interpolated PVT solution is handed over to the LOS module.

In this module, the LOS parameters are determined for all satellites in tracking. Therefore, the satellite position at the transmitting time is determined. Thereafter, the distance between satellite and receiver is calculated. Additionally to earth rotation and relativistic effects, also ionospheric and tropospheric models are applied. The derivatives of the pseudorange ($\dot{\rho}$, $\ddot{\rho}$) are approximated by linear considerations.

In the next step, the LOS parameter are transferred to the tracking loops. Here a second extrapolation step is needed to adapt the sampling rate from the 10 Hz of the navigation process to the 1000 Hz of the tracking loops. The quadratic extrapolation is already described above in the vector tracking section, compare Equation (1)-(3). In the normal vector tracking process, this approximation is acceptable because the error is compensated by the discriminator value. However, this approximation becomes an issue for the signal generation. The replica is created, following the extrapolated values, and a replica jump occurs at the edge of the last extrapolated point of $S_j(t_{j+1})$ to the first point of the new extrapolation $S_{j+1}(t_{j+1})$. These jumps could make it more difficult to track the generated signal, especially for the phase tracking. The interpolation and extrapolation process is visualized in Figure 4. As 'Real trajectory' a sinusoidal movement in x direction is used. The movement amplitude is 10 m with an absolute velocity of 3 m/s. Only the x position was fitted, not the pseudorange, that means that Figure 4 is just for illustration purposes and shows not the real inter- and extrapolation processes. The generated signals are, despite the error in the extrapolation, nice and easy to track for, see 'Result'

section. In future it is foreseen that the quadratic extrapolation step is replaced with an interpolation, very similar to the trajectory interpolation. Therefore, not only the current LOS parameters ($\rho_i, \dot{\rho}_i, \ddot{\rho}_i$) but also the future LOS parameters ($\rho_{i+1}, \dot{\rho}_{i+1}, \ddot{\rho}_{i+1}$) need to be passed to the tracking loop. That is possible, as the entire trajectory is predefined. This interpolation allows us then to create a continuous satellite signals.

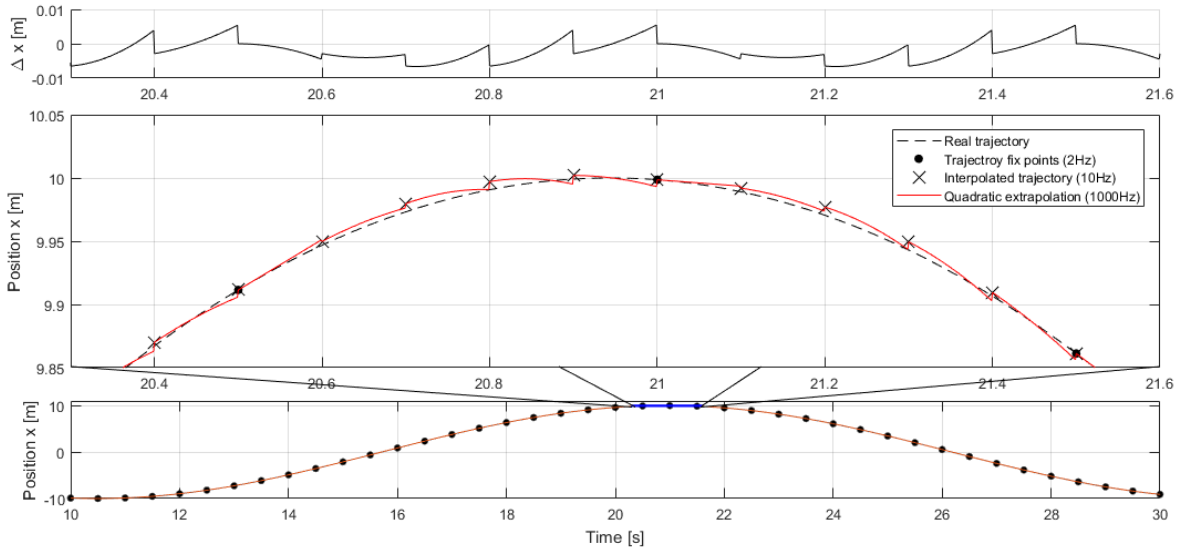

Figure 4: Visualization of the interpolation and extrapolation process. The real trajectory is a sinusoidal movement with an amplitude of 10 m and absolute velocity of 3 m/s. Only the x position was fitted, not the pseudorange.

RESULTS

Pseudorange verification

To verify the correct implementation and to check the influences of interpolation and extrapolation process to the LOS parameter creation, the values for pseudorange ρ, pseudorange -rate $\dot{\rho}$, pseudorange-rate-rate $\ddot{\rho}$ and the carrier phase were plotted and evaluated. The extrapolated LOS parameter for a circular user trajectory is plotted in Figure 5. In the pseudorange plot (top) only the satellite movement is visible as the pseudorange is governed by that. In pseudorange-rate and the pseudorange-rate-rate, however, the velocity and acceleration changes caused by the user movements are clearly visible. Figure 6 shows a zoom in of Figure 5. In the zoomed plot a constant behavior of $\ddot{\rho}$ and a linear behavior of $\dot{\rho}$ is observed as expected between the quadratic extrapolation jumps. For the pseudorange, however, there is no clear jump visible. That is encouraging, as it means that the extrapolation process does not induce a large error in the pseudorange creation. But it has to be mentioned, that this lack of jump may also be explained by the small $\ddot{\rho}$ values in this scenario. The validation for bigger $\ddot{\rho}$ values is in progress. If bigger $\ddot{\rho}$ values lead to a non-negligible jump in the pseudorange creation, the substitution of the extrapolation process with an interpolation process is consider. As the carrier phase is calculated via an interpolation of the Doppler value, no jump is expected and is also not observed.

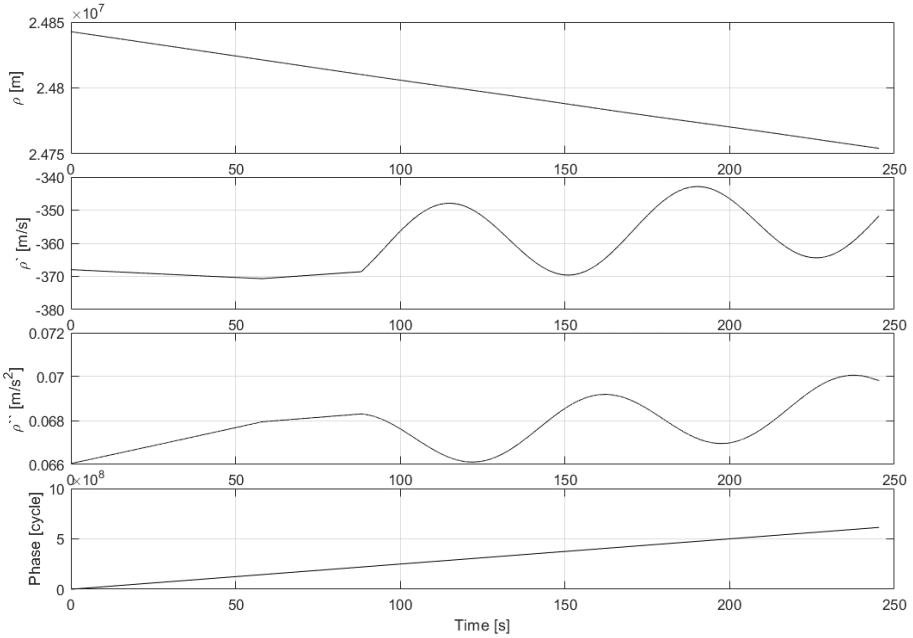

Figure 5: Generated LOS parameter after the quadratic extrapolation step. User trajectory is a circular movement. From top to bottom: pseudorange ρ, pseudorange -rate ρ̇, pseudorange-rate-rate ρ̈ and the carrier phase.

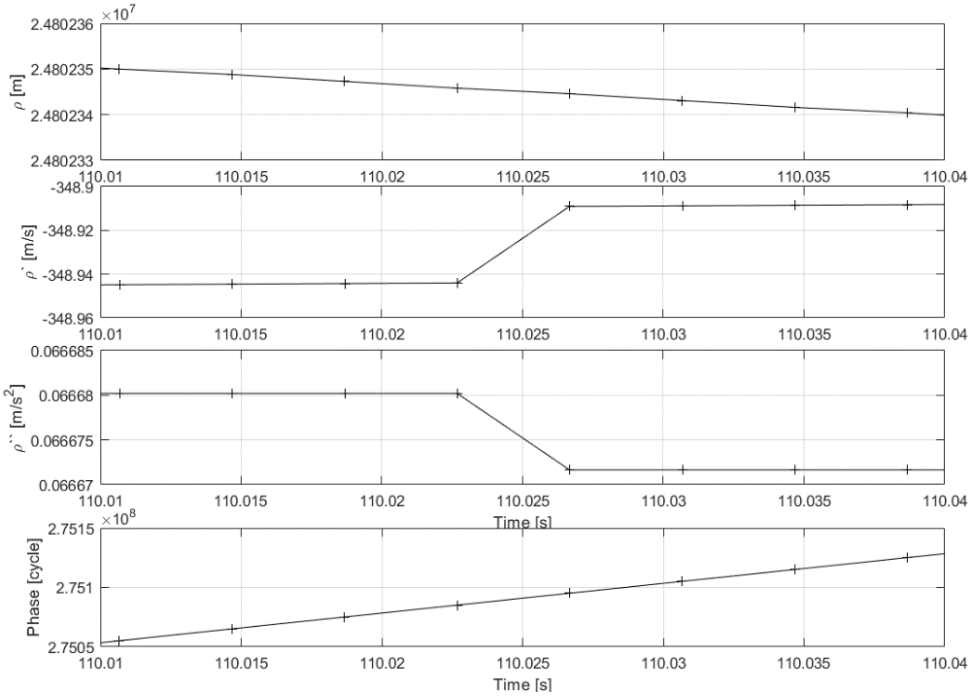

Figure 6: Generated LOS parameter after the quadratic extrapolation step, zoom of Figure 5. User trajectory is a circular movement. From top to bottom: pseudorange ρ, pseudorange -rate ρ̇, pseudorange-rate-rate ρ̈ and the carrier phase.

Tracking performance: genuine vs. generated Galileo OS signal

To check the quality of the generated signal, we compare the tracking results of a genuine recorded signal with a generated signal. The Galileo OS E1 signal was recorded with static receiver positon and open sky conditions. For the generated signal, we used a

static trajectory with the same position as the recording antenna. Also, we generated the signal for the same day and time. The $C/N0$ ratio for the signal generation was set to 52 dB-Hz, to match the genuine signal strength. The strong signal power ensure that errors in the signal generation are easily visible. Both IF sample files were processed with the same configurations. The only difference was that the genuine IF file has a sampling rate of 20 MHz and the generated IF file has a sampling rate of 10 MHz. For processing, the MuSNAT software receiver was used. In *Figure 7* the tracking performances are compared, on the left side the genuine signal and on the right side the generated signal. *Figure 7* shows the plots (from top to bottom) for the signal power (C/N0), code discriminator (e_{DLL}), phase discriminator (e_{PLL}), frequency discriminator (e_{FLL}) and code minus carrier (CMC). The signal power of the generated signal is slightly higher than it was defined in the generation process, a mean signal strength of 52.4 dB-Hz was measured. The tracking performance of e_{DLL}, e_{PLL} *and* e_{FLL} are in the same order for genuine and generated signal. The tracking performance of the generated signal is slightly better, this may result from the stronger signal strength and the fact that the genuine signal faced more disturbances. This can also be observed when comparing the CMC plots.
Anyway, it is visible that the generated signal matches the original signal well in terms of tracking performance and that the replica jumps due to the extrapolation process are negligible.

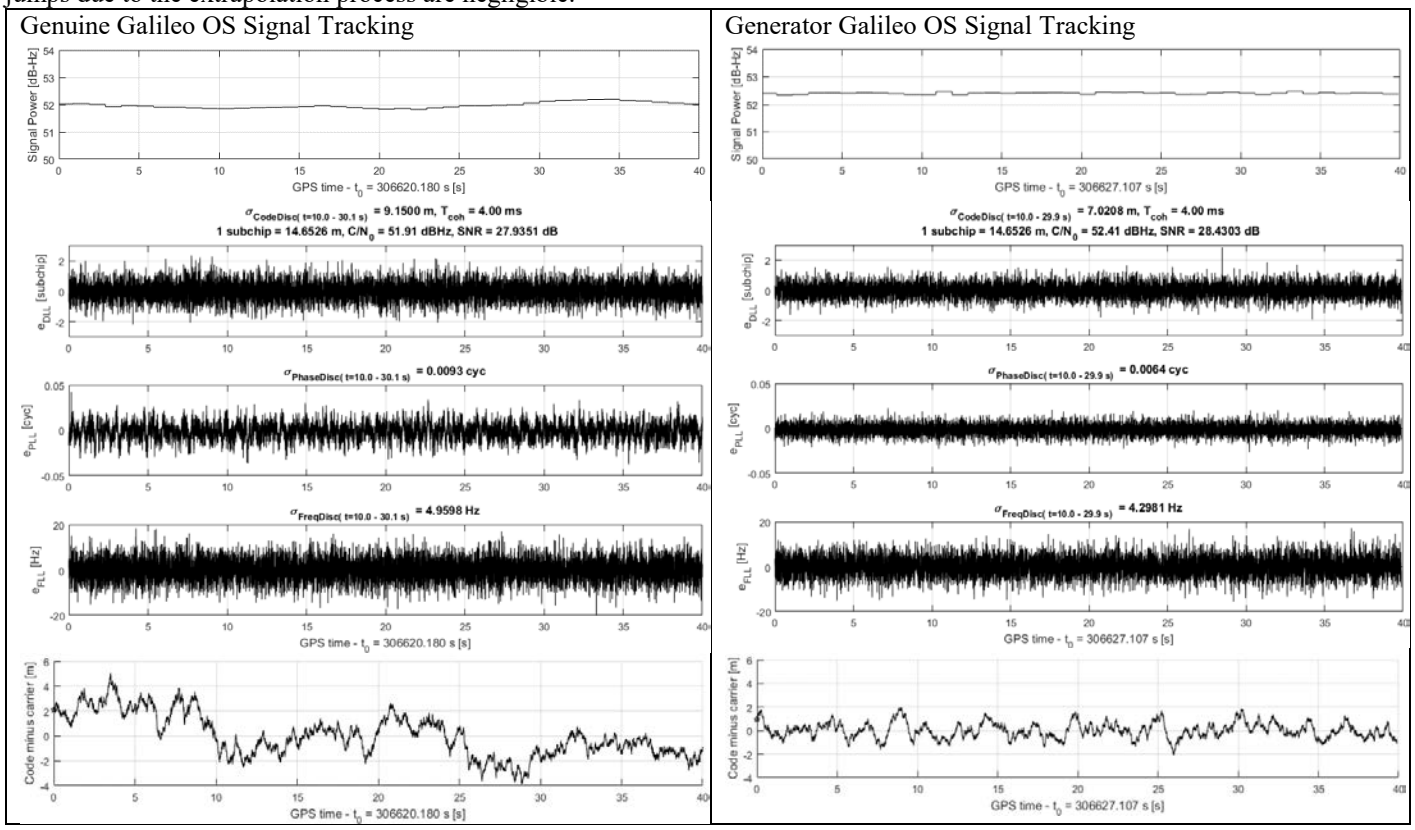

Figure 7: Comparison of the tracking performance of a genuine Galileo OS signal (left) and a ST generated Galileo OS signal (right). From top to bottom the plots show signal power ($C/N0$), code discriminator (e_{DLL}), phase discriminator (e_{PLL}), frequency discriminator (e_{FLL}) and code minus carrier (CMC).

Galileo OS signal with navigation message authentication (NMA)

The MuSNAT signal generator was used to implement and verify Navigation Message Authentication (NMA) [15, 16, 17] based on Galileo OS signals. Therefore, real Galileo satellite signals were recorded and processed by the MuSNAT to extract the symbol-stream (navigation massage) and satellite ephemeris for each satellite with in sight. In the second step, the spare bits in the Galileo E1B INAV navigation massage are replaced by the authentication bits. Thereafter, the symbol stream, the satellite ephemeris and the desired C/N0 values for the tracked satellites were read into the GNSS-Transceiver to generate the IF sample-stream file. For analysis, the IF sample file was again processed by the MuSNAT-Transceiver to extract the bits and verify the authentication. The detailed analysis of the NMA is presented by Maier et al. [18]. The signal for the authentication testing is composed of four satellites. The C/N0 values were set to 48, 50, 52 and 54 dB-Hz, so the authentication could be studied over a broader power range. In Figure 8 the signal power and the Doppler of the generated signals are plotted. As before, all signals show a higher C/N0 value as defined. Besides

that, the signals were tracked fine, the navigation data could be extract and the PVT solution could be determine right. An In- and Quadrature-Phase analysis for the generated satellite signal is plotted in Figure 9. Additionally to the OSNMA test on data level, the MuSNAT transceiver was used to create the pure satellite signals without noise. This IF stream file was used to feed a software define radio (SDR). The signals were transmitted by the SDR. The received signal was analyzed for symbol errors in the navigation message and the authentication behavior, plotted in Figure 10. A detailed evaluation was done by Maier et al. [18].

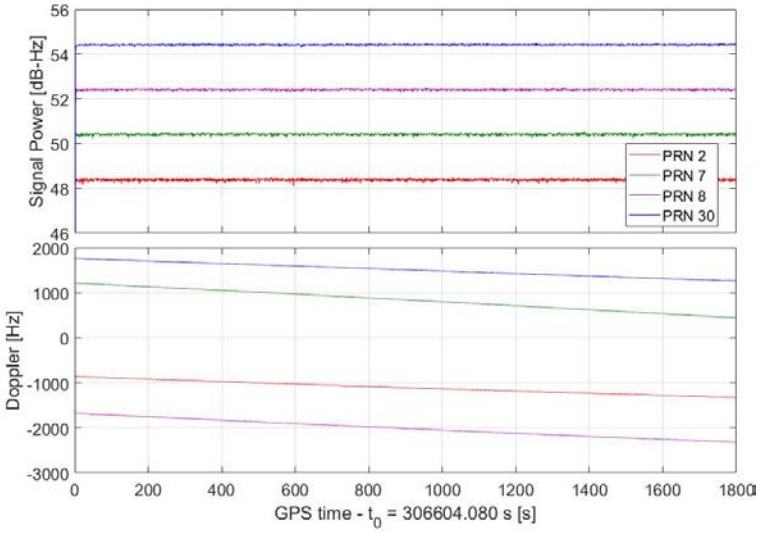

Figure 8: Generated and processed Galileo OSNMA signal. Power (top) and Doppler (bottom) plots of Galileo E1 signals of PRN 2, 7, 8 and 30.[18]

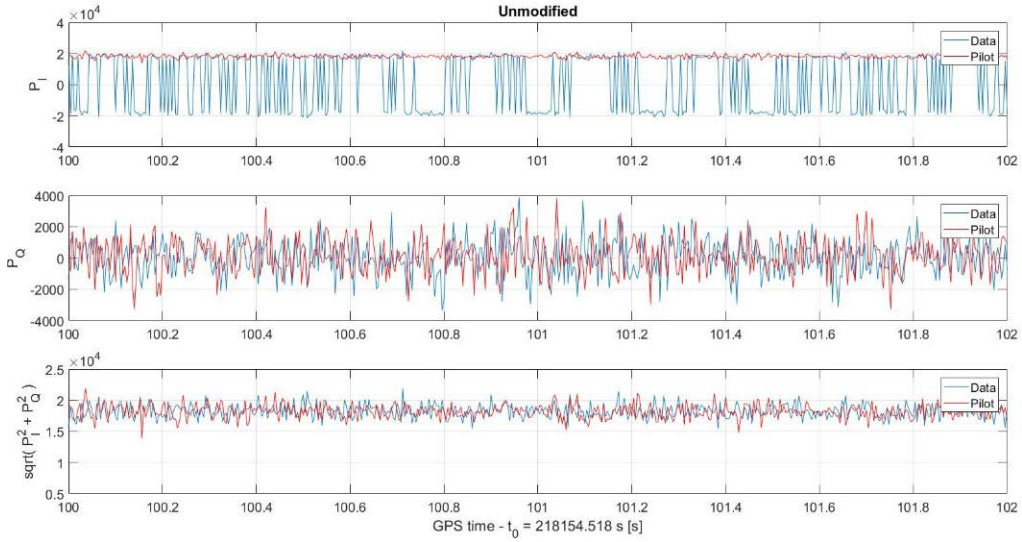

Figure 9: In- and Quadrature-Phase analysis for generated signal with authenticated bits. Generation and processing was done with MuSNAT. .[18]

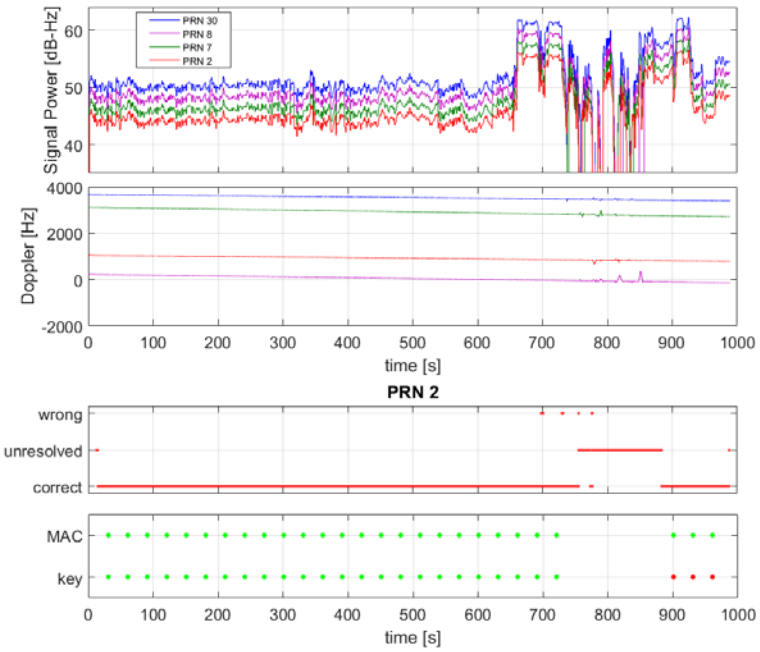

Figure 10: Power (first) and Doppler (second) plots of Galileo E1 signals of PRN 2, 7, 8 and 30 for generated and 'Over-The-Air' transmitted signals. The third plot shows the occurrence of errors in the received symbols for PRN 2. The bottom plot shows the authentication of MAC and key in the navigation massage for PRN 2. Each dot represents one subframe, where green stands for a successful authentication and red identifies not authentic subframes.[18]

Secure code estimation SCE spoofing attack

The MuSNAT signal generator was also used to study the influence of secure code estimation and replay (SCER) attacks [13,14] on software receiver. In contrast to the NMA test, these test wouldn't be able with a COTS signal generator system, because here we need to modify the generated signal on a deeper level.

In an SCER attack, the villain aims, in the first phase, to mimic the genuine signal and transmit it to the victim synchronously. For this attack, however, the villain needs to know the symbol before it is received by him or the victim to generate and transmit the spoofing signal. Therefore, the villain needs to estimates the symbol by correlating over a fraction of the beginning of the genuine symbol. In the estimation time δt_{est}, the villain transmits only noise for the symbol (symbol value equals 0). After the estimation time, it is assumed, that the villain knows the genuine symbol and transmits a correct mimicked signal. To imitate this attack, a symbol estimation time is defined, e.g. 10% of the symbol time, $\delta t_{est} = 0.10 * t_{symbol}$. During the signal generation the symbol value is set to 0 for the time δt_{est} and thereafter, the true symbol value is set.

Three different generation approaches were tested. First, only the E1-B data symbol is set to zero, compare Figure 11. In this case, the correlation values in the data channel are smaller than in the pilot channel this can easily be detected by a receiver. Therefore, in the second case, the pilot channel was also set to zero during the symbol estimation time (Figure 12). In the third case, plotted in Figure 13, only the data channel was set to zero but the second part of the symbol was generated with an increased power to compensate the correlation value reduction due to the first zero (noise) part.[18] For the last case, no difference can be observed between the unmodified signal in Figure 9 and the SCER attack in Figure 13 (symbol estimation time of 40%).

The C/N0 behavior for the three cases and the different estimation times is shown in Figure 14. The C/N0 drops significantly if data- and pilot-channel is set to zero. Also, for the case of no correlation compensation, a reduced C/N0 is visible, compared to the case with power compensation.[18] For a symbol estimation time of 50% and higher, no continues tracking was possible.

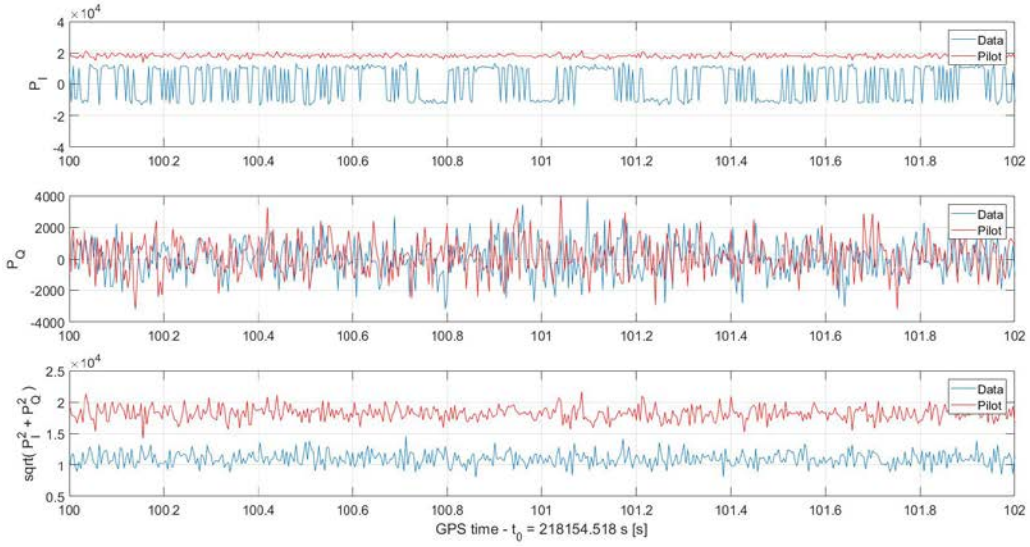

Figure 11: Power of I-component, Q-component and the absolute value of the sample file with a symbol estimation time of 40% of the total symbol time. No compensation of correlation degradation was applied.[18]

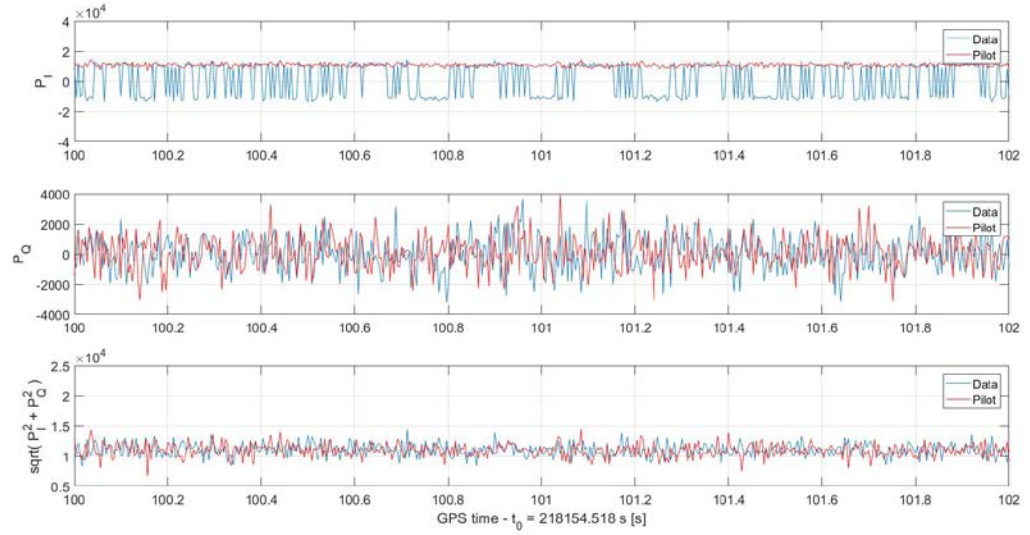

Figure 12: Power of I-component, Q-component and the absolute value of the sample file with a symbol estimation time of 40% of the total symbol time. The symbol estimation was applied to Data- and Pilot-Channel.[18]

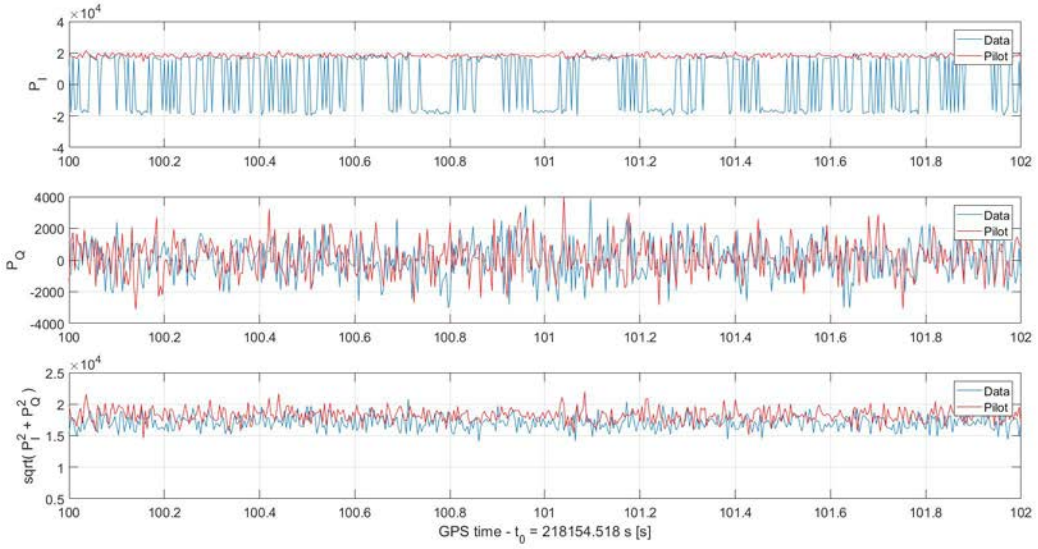

Figure 13: Power of I-component, Q-component and the absolute value of the sample file with a symbol estimation time of 40% of the total symbol time. The power of the second part of the symbol was increased to compensate the correlation degradation.[18]

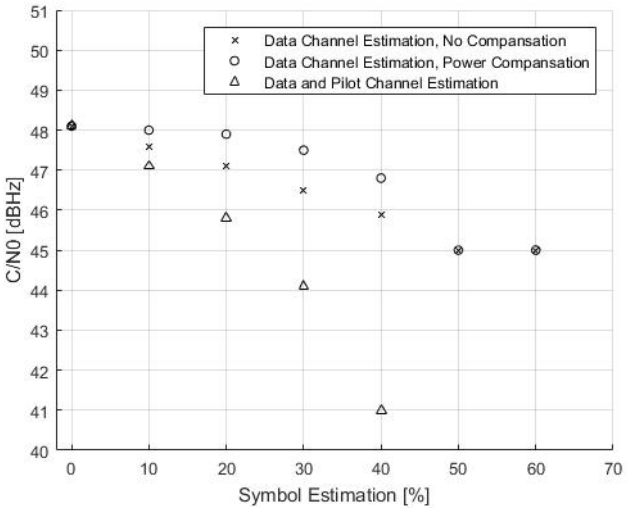

Figure 14: C/N0 behavior for Data-channel estimation without compensation, data-channel estimation with power compensation and data- and Pilot-channel estimation over the estimation time in percent of the total symbol time.[18]

CONCLUSION

In this work, it was shown that the conversion of a software receiver into a software transceiver using the vector tracking approach is feasible and easy to realize. The basic concept was presented and error sources were described. The implementation of the concept using the MuSNAT software receiver was explained. The quality of the generated signals was compared to real signals in terms of tracking performance. Also, it was shown how the ST was used to test NMA in the Galileo OS. At the end, we showed the power of the implemented ST with the creation of SCE attacker signals.

ACKNOWLEDGMENTS

This work is funded by the German Federal Ministry for Economic Affairs and Energy on the basis of a decision by the German Bundestag. It is administrated by the German Aerospace Center in Bonn, Germany, (FKZ: 50 NA 1703).

REFERENCES

1. Teunissen, P., Montenbruck O., "18. Simulators and Test Equipment," eds. *Springer handbook of global navigation satellite systems*. Springer, 2017.
2. Humphreys, T., Jahshan, B., and Brent, L., "The GPS Assimilator: a method for upgrading existing GPS user equipment to improve accuracy, robustness, and resistance to spoofing," *ION Conference 2010*, Portland, 2010.
3. Copps, E. M., Geier, G. J., Fidler, W. C., Grundy, P. A. "Optimal processing of GPS signals," *Navigation* 27.3: 171-182, 1980.
4. Pany, T., Bernd E., "Use of a vector delay lock loop receiver for GNSS signal power analysis in bad signal conditions," *Position, Location, And Navigation Symposium, 2006 IEEE/ION*. IEEE, 2006.
5. WON, J.-H., Dominik D., Bernd E., "Performance Comparison of Different Forms of Kalman Filter Approaches for a Vector-Based GNSS Signal Tracking Loop," *Navigation* 57.3 2010: 185-199.
6. Petevello, M., Lashley, M., Bevly, D.M., "What are vector tracking loops, and what are their benefits and drawbacks?," *InsideGNSS*, May/June, 2009.
7. Teunissen, P., Montenbruck O., "14 Signal Processing," eds. *Springer handbook of global navigation satellite systems*. Springer, 2017.
8. Parkinson, B. W., Enge, P., Axelrad, P., & Spilker Jr, J. J. (Eds.), *Global positioning system: Theory and applications, Volume II*. American Institute of Aeronautics and Astronautics, 1996.
9. Stöber C., Anghileri M., Sicramaz Ayaz A., Dötterböck D., Krämer I., Kropp V., Sanromà Güixens D., Won J.-H., Eissfeller B., and Pany T., "ipexSR: a real-time multi-frequency software GNSS receiver, " *52nd International Symposium ELMAR*, *IEEE Proceedings*, 2010.
10. Institute of Space Technology and Space Applications (ISTA), "MuSNAT," URL: https://www.unibw.de/lrt9/lrt-9.2/software-packages/musnat/view, September 2018.
11. Lehner, A., and Steingass, A., "A novel channel model for land mobile satellite navigation." *Institute of Navigation Conference ION GNSS*. 2005.
12. Petovello, M., Joseph, A., "Measuring GNSS Signal Strenght," *InsideGNSS*, November/December, 2010.
13. Humphreys, T. E., "Detection strategy for cryptographic GNSS anti-spoofing," *IEEE Transactions on Aerospace and Electronic Systems*, *49*(2), 1073-1090, 2013.
14. Psiaki, M. L., Humphreys, T. E. "GNSS Spoofing and Detection," *Proceedings of the IEEE*, *104*(6), 1258-1270, 2016.
15. Wesson, K. D., Rothlisberger, M. P., Humphreys, T. E., "A proposed navigation message authentication implementation for civil GPS anti-spoofing," In *Radionavigation Laboratory Conference Proceedings,* 2011.
16. Kerns, A. J., Wesson, K. D., Humphreys, T. E., "A blueprint for civil GPS navigation message authentication," In *Position, Location and Navigation Symposium-PLANS 2014, 2014 IEEE/ION* (pp. 262-269). IEEE, 2014.
17. Fernández-Hernández, I., Rijmen, V., Seco-Granados, G., Simon, J., Rodríguez, I.,Calle, J. D., "A navigation message authentication proposal for the Galileo open service, " *Navigation: Journal of the Institute of Navigation*, *63*(1), 85-102, 2016.
18. Maier, D., Frankl, K., Blum, R., Eissfeller, B., Pany, T., "Preliminary Assessment on the Vulnerability of NMA-based Galileo Signals for a Special Class of Record & Replay Spoofing Attacks," *Proceedings of IEEE/ION PLANS 2018*, Monterey, CA, April 2018, pp. 63-71.