Real time implementation of Vector Delay Lock Loop on a GNSS receiver hardware with an open software interface

Katrin Dietmayer, Muhammad Saad, Christian Strobel, Dr. Fabio Garzia,
Matthias Overbeck and Dr. Wolfgang Felber
Fraunhofer Institute for Integrated Circuits (IIS)
Nuremberg, Germany
katrin.dietmayer@iis.fhg.de

***Abstract*—In standard GNSS receivers scalar tracking is used to track the satellite signals. In this approach each signal and satellite is tracked independently. The vector tracking (VT) algorithm uses information about the movement of the antenna to calculate a combined tracking solution for all signals. This makes the tracking more stable and robust against shadowing and multipath effects. It delivers an improved navigation solution, which is less noisy. Present developments in VT have already shown good results in post processing with software-defined radio (SDR). This paper presents the results of the real-time VT developed on a GNSS hardware platform. The theory and the implementation of the used Kalman filter (KF) is described. The hardware setup, using a GNSS receiver with an open software interface is presented. First results using vector delay locked loop (VDLL) in a signal generator scenario with short-shadowed satellites are shown. It is described how the VDLL with an extended Kalman filter (EKF) calculates a navigation solution. This output is then used to compute the code numerical correlation oscillator (NCO) values for multiple space vehicles simultaneously and to steer the hardware NCOs in the receiver to close the tracking loops. The code tracking loops have to be closed simultaneously and as fast as possible to avoid timing issues.**

***Index Terms*—vector tracking (VT), vector delay locked loop (VDLL), tracking loops**

I. Introduction

GNSS receivers use tracking loops to lock onto satellite signals by synchronizing the satellite signal with a local-generated replica [1]. In a standard receiver each signal is tracked independently, hence this is referred to as scalar tracking. The GNSS receiver hardware provides correlator values. Independent code and carrier tracking loops calculate new setting values to adjust the estimates of the code phase and carrier frequency for a single tracking channel. The new estimates steer the code and carrier numerical correlation oscillator (NCO) to obtain a stable reference for the received signal and to close the loop. If at least four satellites are tracked and their pseudorange and pseudorange-rate measurements obtained, a navigation solution can be calculated. This navigation solution does not have a direct influence on the tracking loops. In weak signal conditions, e.g. caused by shadowing of the satellite, scalar tracking loops are likely unstable and can loose the signal tracking. For availability critical applications, like autonomous driving, a robust navigation solution is one of the most important aspects. Weak signal conditions often occur in urban environments, e.g. passing by high buildings.

978-1-5386-9473-2/19/$31.00 ©2019 IEEE

The approach presented in this paper controls multiple tracking loops simultaneously and in relation to each other. This is called vector tracking (VT) [2]. The signal tracking and navigation solution calculation are combined together to a single estimation algorithm. Hence they are highly related on each other. In challenging environments, VT architectures with SDRs have shown to provide better performance over scalar tracking [3]. This is mainly because the channels tracking strong signals can support the channels with weak signals.

VT uses the movement information of the antenna phase center to calculate a combined tracking solution for all signals collectively. Therefore a Kalman filter (KF) is used to estimate a navigation solution which is adjusted with the correlator values from the GNSS hardware. From the estimated navigation solution, new NCO control values for multiple satellite signals are determined simultaneously. These are then passed back to steer the hardware NCOs in the GNSS receiver to close the tracking loop.

The real-time development of VT on a GNSS hardware receiver has many challenges over the often used software-defined radio (SDR). SDRs mainly operate in post processing, hence they have no timing requirements. The VT loop in GNSS hardware has to be closed simultaneously for all channels and as fast as possible to avoid timing issues. It is important to take the whole system into account when developing VT in real-time. Effects on processing delays can contribute to timing issues. System and measurement noise have to be modelled correctly according to the used hardware and environment.

II. Vector Tracking Delay Lock Loop

The vector delay locked loop (VDLL) is one variant of VT, where the carrier tracking loops are still scalar and the code tracking loops are combined to a single tracking algorithm, as shown in Fig. 1. The navigation solution is estimated by a KF,

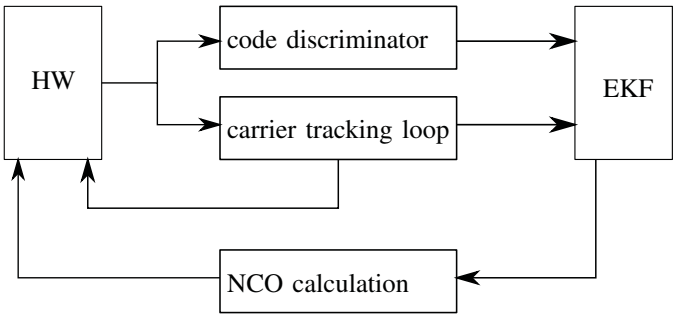

Fig. 1. Vector Delay Lock Loop.

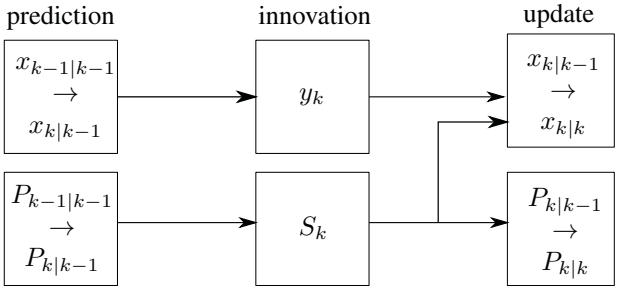

Fig. 2. Extended Kalman filter work flow.

called the states $\hat{x}$. The following parts denote estimates with "ˆ". A KF has two main phases, mostly called the prediction and update phase. For these phases a system and measurement model needs to be defined. The system model describes how the state propagate with time. The measurement model shows the relationship between the measurements and the states of interest. For a GNSS navigation filter, the measurements are not a linear function of the states, therefore an extended Kalman filter (EKF) is used. The measurements are available at discrete time intervals. The subscript k is used to denote the epoch or iteration, therefore $x(t_k) = x_k$.

A. Kalman filter algorithm

During the prediction phase, the EKF estimates a new state vector $\hat{x}_{k|k-1}$ based on the updated state $\hat{x}_{k-1|k-1}$ of the previous iteration. Therefore a transition matrix ϕ_{k-1} has to be defined. Additionally an error covariance matrix $P_{k|k-1}$ tracks the variances and covariances of each state estimate. To estimate $P_{k|k-1}$ the previous matrix $P_{k|k}$ and a system noise covariance matrix Q_{k-1} is used. After this phase, the states and error covariances are propagated from the iteration $k-1$ to k. The system is then updated with measurements during the update phase. Therefore the interaction of the measurements and the states is defined in a measurement matrix H_k. The measurement noise is taken into account with a measurement noise covariance matrix R_k. The states $\hat{x}_{k|k-1}$ are then updated by the measurement innovation, defined as $\delta z_{k|k-1}$. In the last step of the update phase, the updated error covariance matrix $P_{k|k}$ is calculated. Fig. 2 shows the work flow of an EKF. A more detailed description of the KF states includes [2].

1) *Prediction phase*
   - Propagate the state vector
   $$\hat{x}_{k|k-1} = \Phi_{k-1}\hat{x}_{k-1|k-1}$$
   - Propagate the error covariance matrix
   $$P_{k|k-1} = \Phi_{k-1}P_{k-1|k-1}\Phi_{k-1}^t + Q_{k-1}$$
2) *Update phase*
   - Calculate the measurement innovation
   $$\delta z_{k|k-1} = z_k - H_k\hat{x}_{k|k-1}$$
   - Calculate the error covariance innovation
   $$S_k = H_k P_{k|k-1} H_k^t + R_k$$
   - Calculate the Kalman gain
   $$K_k = P_{k|k-1}H_k^t S_k^{-1}$$
   - Update the state vector
   $$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\delta z_{k|k-1}$$
   - Update the error covariance matrix
   $$P_{k|k} = (I - K_k H_k)P_{k|k-1}$$

B. System Model

The following paragraph describes the tracked states and the system model, everything is described detailed in [2]. The states of interest are the users position, velocity, clock bias and drift. The user position and velocity are observed in the Earth-centred Earth-fixed frame with the corresponding x, y and z axes. The states are given as:

- position $p = (p_x, p_y, p_z)^t$,
- velocity $v = (v_x, v_y, v_z)^t$,
- clock bias b, and
- clock drift d.

With respect to time t we get the state vector:

$$x(t) = (p(t), v(t), b(t), d(t))^t$$

and in discrete time:

$$x_k = (p_k, v_k, b_k, d_k)^t$$

It is necessary to know how the states x_k change with time. Therefore, the derivation of the states with respect to time t are calculated. Initially a constant velocity model is chosen, hence all accelerations are modelled as white gaussian noise. One assumption of the standard KF is that the time derivative of each state is a linear function of the other states with additional white noise. The EKF on the other hand can handle non-linear systems, and the dynamics of the state are described by a non-linear system function f instead of a system matrix F, hence:

$$\dot{x}(t) = f(x(t), t)x(t) + G(t)w_s(t)$$

where G is the system noise matrix and w_s the system noise vector. For the error covariance matrix propagation, the system matrix is linearised by calculating the Jacobian of the system matrix with:

$$F_k = \left.\frac{\partial f(x_k, t_k)}{\partial x}\right|_{x=\hat{x}_{k-1|k-1}}$$

From the system matrix, the transition matrix Φ can be derived. The time between the iteration $k-1$ and k is denoted with δt, so that Φ becomes:

$$\Phi_k = I_8 + F_k \delta t$$

The system model with the constant velocity model assumption is simple to define. The derivation of the position with respect to time is the velocity, and for the clock bias the clock drift respectively. As the velocity and clock drift are independent of all components of the state vector, the expectations of the time derivative is zero. Hence the system matrix is:

$$F_k = \begin{pmatrix} 0_{3\times3} & I_3 & 0_{3\times1} & 0_{3\times1} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times1} & 0_{3\times1} \\ 0_{1\times3} & 0_{1\times3} & 0 & 1 \\ 0_{1\times3} & 0_{1\times3} & 0 & 0 \end{pmatrix}$$

and the system transition matrix is:

$$\Phi_k = \begin{pmatrix} I_3 & I_3\delta t & 0_{3\times1} & 0_{3\times1} \\ 0_{3\times3} & I_3 & 0_{3\times1} & 0_{3\times1} \\ 0_{1\times3} & 0_{1\times3} & 1 & \delta t \\ 0_{1\times3} & 0_{1\times3} & 0 & 1 \end{pmatrix}$$

The error covariant matrix Q shows how the noise changes in the system with respect to time. The largest impact of uncertainties in the system are the changes in velocity and clock drift. For velocity it is the untracked impact of acceleration a, that causes an error. The receiver clock frequency drift f causes a clock drift error and the receiver clock phase drift ϕ on the clock offset. Therefore the system noise matrix G is:

$$G_k = \begin{pmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times2} \\ 0_{3\times3} & I_3 & 0_{3\times2} \\ 0_{2\times3} & 0_{2\times3} & I_2 \end{pmatrix}$$

with the noise vector $w_s = (0_{1\times3}, e_a, e_\phi, e_f)^t$, where e_f describes the frequency-drift, e_ϕ the clock phase-drift uncertainty and e_a the acceleration uncertainties in each direction. We assume to have white system noise, therefore we can use the single sided PSD matrix of the components of the system noise vector w_s, S which is a diagonal matrix, so that:

$$S = \begin{pmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times1} & 0_{3\times1} \\ 0_{3\times3} & S_a & 0_{3\times1} & 0_{3\times1} \\ 0_{1\times3} & 0_{1\times3} & S_\phi & 0 \\ 0_{1\times3} & 0_{1\times3} & 0 & S_f \end{pmatrix}$$

and

$$S_a = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{pmatrix}$$

The system noise covariance matrix can be defined in different ways. One is to define the noise matrix in terms of the continuous system noise. Together with the white noise assumption and the PSD matrix S, Q is calculated as:

$$\begin{aligned} Q(t) &= \int \Phi(t) * G * S * G^t * \Phi^t(t) dt \\ &= \begin{pmatrix} \frac{1}{3}S_a t^3 & \frac{1}{2}S_a t^2 & 0_{3\times1} & 0_{3\times1} \\ \frac{1}{2}S_a t^2 & S_a t & 0_{3\times1} & 0_{3\times1} \\ 0_{1\times3} & 0_{1\times3} & S_\phi t + \frac{1}{3}S_f t^3 & \frac{1}{2}S_f t^2 \\ 0_{1\times3} & 0_{1\times3} & \frac{1}{2}S_f t^2 & S_f t \end{pmatrix} \end{aligned}$$

C. Measurement Model

The measurement model shows the relationship between the measurements and the states. A detailed derivation can be found in [2] again. In a standard KF measurement vector $z(t)$ is modelled as a linear function of the true states. In discrete time z_k is defined as:

$$z_k = H_k x_k + w_{m,k}$$

where H is the measurement matrix and w_m the measurement noise, normally assumed to be white noise. In a GNSS application the model is highly nonlinear, therefore the matrix H is replaced with a measurement function $h(x_k, t_k)$, so that z_k becomes:

$$z_k = h(x_k, t_k) + w_{m,k}$$

The innovation δz is then:

$$\begin{aligned} \delta z_{k|k-1} &= z_k - h(\hat{x}_{k|k-1}, t_k) \\ &= h(x_k, t_k) - h(\hat{x}_{k|k-1}, t_k) + w_{m,k} \end{aligned} \quad (1)$$

where z_k denotes the real measurements and $h(\hat{x}_{k|k-1}, t_k)$ are the predicted measurements derived from the predicted state vector $\hat{x}_{k|k-1}$.

In a GNSS receiver, for each visible and tracked satellite signal j the pseudorange ρ and the pseudorange-rate $\dot{\rho}$ is commonly measured. Therefore the predicted pseudorange and pseudorange-rate with the EKF states are calculated as:

$$\hat{\rho}^j_{k|k-1} = |p^j_{s,k} - \hat{p}_{k|k-1}| + \hat{b}_{k|k-1}$$

$$\dot{\hat{\rho}}^j_{k|k-1} = \hat{u}^t_{k|k-1}|v^j_{s,k} - \hat{v}_{k|k-1}| + \hat{d}_{k|k-1}$$

with the satellite position $p^j_{s,k}$ and velocity $v^j_{s,k}$. Hence the measurement function for N satellites is:

$$h(\hat{x}_{k|k-1}, t_k) = \begin{pmatrix} \hat{\rho}^1_{k|k-1} \\ \vdots \\ \hat{\rho}^N_{k|k-1} \\ \dot{\hat{\rho}}^1_{k|k-1} \\ \vdots \\ \dot{\hat{\rho}}^N_{k|k-1} \end{pmatrix}$$

As the system matrix, the measurement matrix H needs to be linearised. Therefore the Jacobian derivatives have to be calculated as:

$$H_k = \left.\frac{\partial h(x, t_k)}{\partial x}\right|_{x=\hat{x}_{k|k-1}}$$

$$H_{\rho p} = \frac{\partial \rho_k}{\partial p_k} = \frac{(p_{s,k} - p_k)}{|p_{s,k} - p_k|} \tag{2}$$

$$H_{\dot{\rho} p} = \frac{\partial \dot{\rho}_k}{\partial p_k} = -\frac{2}{|p_{s,k} - p_k|}(\dot{p}_{s,k} - \dot{p}_k) \tag{3}$$

$$H_{\dot{\rho} v} = \frac{\partial \dot{\rho}_k}{\partial v_k} = -\frac{p_{s,k} - p_k}{|p_{s,k} - p_k|} \tag{4}$$

$$\frac{\partial \rho_k}{\partial v_k} = 0 \tag{5}$$

$$\frac{\partial \rho_k}{\partial b_k} = 1 \tag{6}$$

$$\frac{\partial \rho_k}{\partial d_k} = 0 \tag{7}$$

$$\frac{\partial \dot{\rho}_k}{\delta b_k} = 0 \tag{8}$$

$$\frac{\partial \dot{\rho}_k}{\partial d_k} = 1 \tag{9}$$

With equations (2) – (9) the Jacobian measurement matrix H_k is defined as:

$$H_k = \begin{pmatrix} H_{\rho p} & 0 & 1 & 0 \\ H_{\dot{\rho} p} & H_{\dot{\rho} v} & 0 & 1 \end{pmatrix}\Bigg|_{x=\hat{x}_{k|k-1}}$$

As the pseudorange-rate has a weak dependency on the position, the $H_{\dot{\rho} p}$ terms can be neglected as shown in [2]. For the real-time implementation, the measurements given from the GNSS hardware should be used to update the EKF. In the VDLL, a separate scalar carrier frequency loop is running for each satellite. Hence inputs for the EKF are the code phase and pseudorange-rate measurements. The innovation vector $\delta z_{k|k-1}$ for the pseudorange can directly be replaced with the discriminator function of the code phase. A non-coherent normalised early minus late envelope discriminator, like in [4], is used to calculate the code tracking error φ_k for each satellite j as:

$$\begin{aligned} \delta z^j_{k|k-1,\rho} &= \rho^j_k - \hat{\rho}^j_k + w^j_{m,k} \\ &\approx -\frac{c}{f_{co}} \varphi^j_k \end{aligned}$$

with the code frequency f_{co} and the speed of light c. Therefore the measurement function $h(\hat{x}_{k|k-1}, t_k)$ is zero for the pseudorange, because $\delta z^j_{k|k-1,\rho}$ already contains the pseudorange error.

To take the noise of the measurements into account, the measurement noise covariance matrix R is defined. Depending on the implementation level and the filter, the R matrix can be modelled as diagonal and constant or as a function of known dynamics of the system. It is often necessary to determine some coefficients for the R matrix empirically. If the carrier-to-noise density ratio (C/N_0), of each signal is used, the measurement noise covariance matrix is optimally weighted in the VDLL, see [2]. More Information about C/N_0 can be found e.g. in [1].

D. NCO Control Algorithm

To steer the hardware and closing the loop, new NCO have to be calculated. The independent scalar carrier loop calculates new carrier NCO values and provides these to the hardware, and the VDLL new code NCO values. The best feed back time for the code NCO values is directly after the EKF measurement update. To calculate new code NCO values the state vector is used. The new code NCO for the jth signal is

$$\hat{f}^j_{co,k+1} = f_{co}\left(1 - \frac{\hat{\rho}^j_{k+1|k} - \hat{\rho}^j_{k|k-1}}{c\tau}\right) \tag{10}$$

with the code frequency f_{co}, the update interval τ and c the speed of light. The pseudorange $\hat{\rho}^j_k$ for the signal j at the iteration k is calculated with the position and clock bias estimation from the predicted EKF states

$$\hat{\rho}^j_{k|k-1} = |p^j_{s,k} - \hat{p}_{k|k1}| + \hat{b}_{k|k-1} + e_{I,k} + e_{T,k} - e_{c,k}$$

where the satellite position $p^j_{s,k}$ and clock offset $e_{c,k}$ is obtained from the ephemeris data, the Ionospheric error $e_{I,k}$ and the Tropospheric error $e_{T,k}$ can be calculated using models.

Depending on the system there might be a significant time lag between calculation and validity of the new NCO values. This includes the time taken to get the GNSS measurements, update interval, time to calculate the new NCO values and to steer the hardware. Therefore, it has to be considered during the implementation in order not to lose data or get timing issues.

III. System architecture and implementation

A. GOOSE

In order to test the VDLL algorithm, the multi-signal and multi-constellation GNSS receiver with an open interface "GOOSE", developed by Fraunhofer Institute (IIS) [5], is used. It is based on a modular approach and the idea is that different modules can be used with the same receiver setup. It is composed of a geodetic antenna with a stable phase center as well as an analog front-end which allows L1, L2 and L5 GNSS signal bands to be processed. The baseband board equipped with a Xilinx 7-series field-programmable gate array (FPGA) supports 90 parallel hardware channels and uses a dual core ARM Cortex A9 processor having a PCIe interface.

The correlation data from the baseband board is sent via an open interface to the processing system, where the VDLL and KF algorithm is developed. Furthermore, the calculated NCO values from the VDLL are sent back to the baseband via the same open interface. For more information see [6] and [7].

B. Real-time implementation

For real-time implementation and testing the GOOSE receiver is used. To start with a VDLL the EKF has to be initialized with a valid navigation solution. The position, velocity, clock bias and drift are calculated by scalar tracking loops and a single epoch position, velocity, and time (PVT) algorithm. As a total state EKF is used the error covariance matrix P has to be initialised with values, which should reflect the variance of the initialisation process. These values could be determined empirically or by evaluating the receiver navigation solution with a standard approach. It is valid to

initialize the clock drift at zero, which then needs to be reflected in the initial uncertainty with a higher variance. This approach is also chosen for the implementation on the GOOSE receiver.

For the tests, GPS L1 C/A signals are simulated with a Spirent GSS9000 GPS/GNSS Signal Generator. The GPS satellite signal characteristics are described in [1]. After a successful acquisition of the signal the hardware provides the correlation values with 50 Hz as an integration time of 20 ms is selected. The correlation values of each signal are provided in every 20 ms burst. The GPS satellites are synchronised, hence if their internal clock errors are neglected, all satellites transmit their signals at the same time, the time of transmission (tot). However, the actual transmission time is not generally the same because of the satellite clock errors. With real signals this could lead to some timing issues. Here only simulated signals with no satellite clock errors are used, hence this error is negligible for now.

The signals are received at different times. In one burst the GOOSE hardware provides the correlation values for each signal at the specific time of arrival (toa). As the EKF processes a full set of measurements, the correlation values in one burst have to be saved until the information of all satellites is received, normally from the furthest tracked satellite, shown in Fig. 3. Therefore the update interval τ from equation (10) is different for each channel dependent on the signal arrival times. It also needs to be mentioned, that it is possible to perform a scalar measurement update but therefore the EKF design has to be changed. As the code tracking is more relaxed with time correlated errors, the vector measurement update is used.

Another approximation is the propagation of 20 ms ahead of the state vector. The estimated states refer to the next signal arrival times, which is not exactly 20 ms ahead because of relative motion between the satellites and the receiver and clock errors. However, this error is really small and is therefore not considered. A detailed error estimation can be found in [8].

As the GOOSE receiver needs a feedback for each correlation measurement interruption to steer the hardware, the carrier loop is directly performed after it. A scalar second order loop filter (for more information see [9]) is chosen to calculate new carrier NCO values. At the same time the pseudorange-rate and code phase error is saved for the VDLL. The EKF system and measurement model are defined as described before. The coefficients of the PSD matrix S are determined empirically for the test setup. They heavily depend on the chosen scenario. Therefore P is often modelled with respect to the velocity. The measurement noise covariance matrix is modelled with the C/N_0 measurements and optimised for the dynamics of the chosen scenario.

Before the VDLL runs, independent scalar code and carrier tracking loops are running. Therefore two second order loop filters are implemented. The carrier loop continues as before when the code loop is switched to VDLL mode. As all information is available – a valid PVT, ephemeris data and measurements – the tracking switch to the VDLL. Directly after the update, the new estimated states are used to generate new code NCO values to steer the hardware.

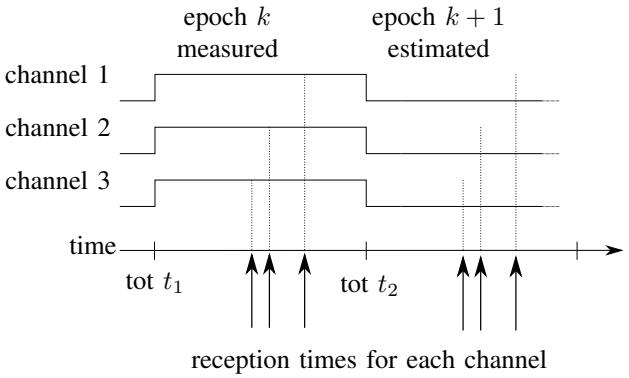

Fig. 3. Measurements in one burst for each channel.

IV. Results

The VDLL was implemented on the GOOSE receiver and tested with a simulated scenario using a Spirent Signal Generator. For the constant velocity system model, a scenario with one constant moving vehicle is simulated. The starting point is located at latitude 49° 29.800', longitude 11° 8.495' and height 391 m. A heading of 127° and constant velocity 10 m/s is set. To compare the navigation solution, the logging files from Spirent are used. The skyplot shown in Fig. 4 containing seven GPS satellites with a geometric dilution of precision (GDOP) around 2.93. The scenario runs for 10 min. During the first 30 s, standard tracking loops are running, providing an initial navigation solution to initialise the VDLL. The position error of the estimated states are shown in Fig. 5, the velocity error in Fig. 6 and the clock bias and drift in Fig. 7. The initial value of the clock drift gets really fast adjusted by the EKF to a stable one, therefore the picture was enlarged to the interesting area. Fig. 8 shows the calculated code doppler errors calculated with 10 for each signal, which are send back to steer the hardware code NCOs for a stable reference.

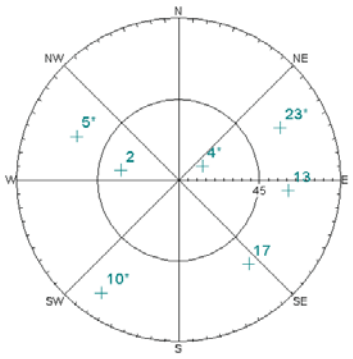

Fig. 4. Skyplot of the scenario.

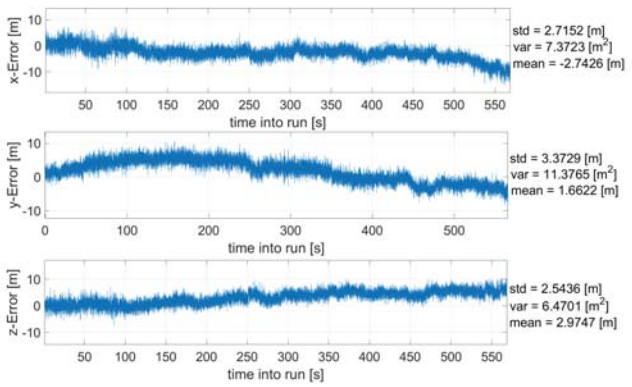

Fig. 5. Position error in ECEF frame along x-, y- and z-axes.

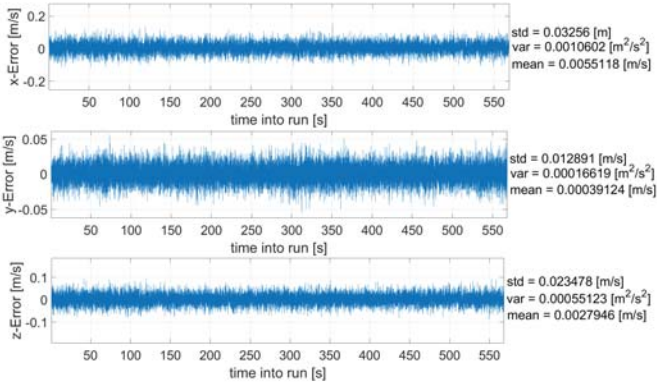

Fig. 6. Velocity error in ECEF frame along x-, y- and z-axes.

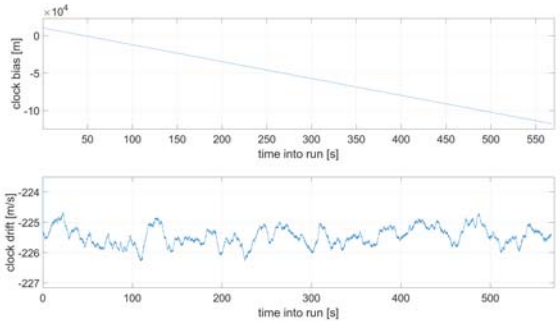

Fig. 7. Clock bias and drift estimates of EKF.

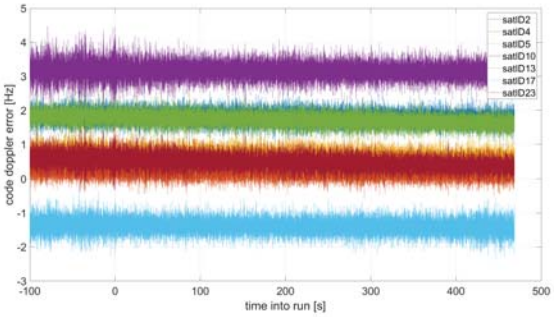

Fig. 8. Code doppler corrections to steer the hardware.

For further testing, the Spirent scenario was adjusted by simulating a satellite which gets shadowed for a short time. The same scenario settings as before are used, except for satellite 17. After 165 s satellite 17 is turned off for 10 s. With standard tracking loops, this channel would be stopped. Fig. 9 shows the position error and Fig. 10 the velocity error, regarding the Spirent logs. After 117 s into run, one can see that the tracking of satellite 17 is not impacted significantly. This is due to the measurement noise covariance matrix R limiting the influence of this channel on the others. For the simulated shadowing duration of 10 s, the values are sufficiently good enough to compensate the signal outage and calculate the code correction for satellite 17. This can be seen in Fig. 12.

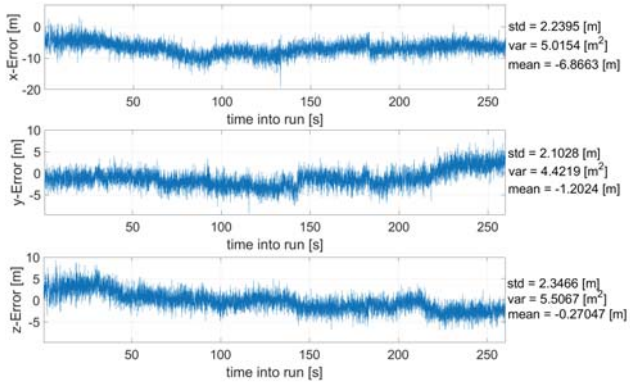

Fig. 9. Position error in ECEF frame along x-, y- and z-axes.

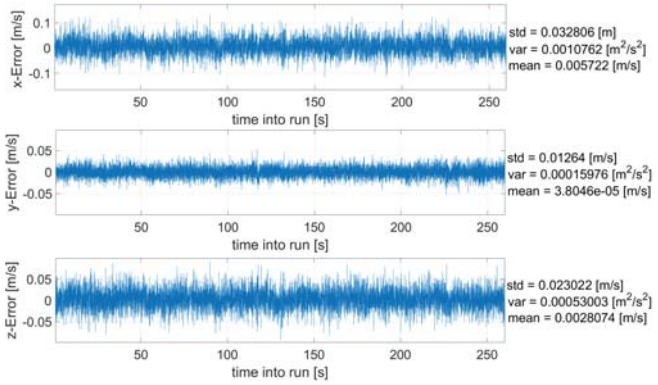

Fig. 10. Velocity error in ECEF frame along x-, y- and z-axes.

V. Conclusion and Future Work

The paper introduces the principle architecture of a VDLL implemented on the GOOSE receiver and the underlying algorithm of the EKF. The implementation of the VDLL on the GOOSE receiver shows good results improving the tracking of GPS L1 C/A signals. The EKF provides a stable navigation solution for a constant velocity scenario. Adaptations for the carrier loops during the signal blocking had to be implemented but showing stable tracking for at least 10 s. Future work

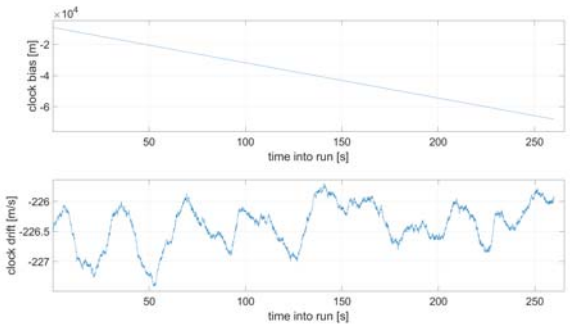

Fig. 11. Clock bias and drift estimates of EKF.

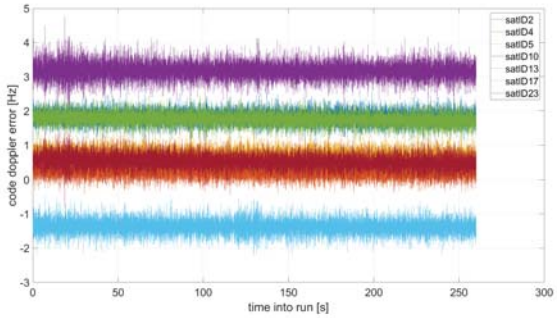

Fig. 12. Code doppler corrections to steer the hardware.

includes improving the adaptation of the carrier loop during the signal blocking. For this paper, the testing scenarios where chosen to have a constant satellite constellation. For future scenarios, a dynamic constellation with rising up and setting down satellites have to be considered in the VDLL by dynamically updating the list of actively tracked satellites. This has to be done without interrupting the VDLL. And last more error sources as ionospheric, tropospheric and satellite clock errors will bring the model even closer to real-world conditions.

VI. Acknowledgement

The work for this paper has been conducted under the PRoPART project, which has received funding from the European GNSS Agency under the European Unions Horizon 2020 research and innovation programme under grant agreement No 776307.

References

[1] E. D. Kaplan and C. J. Hegarty, “Understanding GPS: Princiles and Applications, 2nd Ed.,” Artech House, 2006.

[2] P. D. Groves, “Principles of GNSS Inertial and Multisensor Integation navigation systems, 2nd Ed.” Artech House, 2013.

[3] Y. Yang, J. Zhou and O. Loffeld, “GPS Receiver Tracking Loop Design based on a Kalman Filtering Approach,” Proceedings ELMAR-2012, pp. 121-124, 2012.

[4] S. Zhao and D. Akos, “An Open Source GPS/GNSS Vector Tracking Loop - Implementation, Filter tuning and Results,” Proceedings of the 2011 International Technical Meeting of The Institute of Navigation, San Diego, CA, January 2011, pp. 1293-1305.

[5] F. Garzia, C. Strobel, M. Overbeck, N. Kumari, S. Joshi, F. Förster and W. Felber, “A Multi-Frequency Multi-Constellation GNSS Development Platform with an Open Interface,” in: IEEE Xplore 2016, DOI 978-1-4799-8915-7.

[6] M. Overbeck, F. Garzia, A. Popugaev, O. Kurz, F. Förster, W. Felber, A. Ayaz, S. Ko, and B. Eissfeller, “GOOSE GNSS Receiver with an Open Software Interface,” Proceedings of the 28th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2015), Tampa, Florida, September 2015, pp. 3662-3670.

[7] M. Overbeck, F. Garzia, C. Strobel, C. Nickel, M. Saad, D. Meister and W. Felber, “GNSS-Receiver with Open Interface for Deeply Coupling and Vector Tracking,” Proceedings of the 29th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2016), Portland, Oegon, September 2016, pp. 1222-1229.

[8] M. Lashley, D. Bevly and M. Perovello, “Vector Delay Lock Loops,” in Inside GNSS, September 2012.

[9] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen, “A Software-Defined GPS and Galileo Receiver: A Single-frequency Approach,” Birkhäuser.