Development of Kalman Filter Algorithms for a Low Dynamics GNSS Receiver

Shreeram Murali
School of Electrical Engineering
Aalto University
Espoo, Finland
Email: shreeram.murali@aalto.fi

***Abstract*—This report presents the implementation of an Extended Kalman Filter (EKF) based estimator that can be utilized in a GNSS receiver to convert pseudoranges into position estimates. It discusses methods employed to determine instantaneous estimates of receiver position and goes on to develop an EKF algorithm to demonstrate the localization of a low dynamics receiver. This scenario is selected because it exemplifies a wide range of GNSS motion in the real world, and matches the dynamic models that are commonly utilized in actual GNSS receivers. The performance of the EKF algorithm is depicted using simulated GNSS data.**

I. Introduction

Global Navigation Satellite Systems (GNSS) is a generic term for systems that enable worldwide localization based on positioning and timing data. They are more commonly known as the Global Positioning System (GPS), a system that was developed by the US Air Force and first became operational in 1993. In this report, GNSS and GPS are used in a synonymous manner.

GNSS functions on the basis of signal triangulation. Each GNSS receiver, equipped with an internal clock, obtains timestamped signals from multiple satellites. Since light travels at a constant speed c, this allows them to find their range to each satellite based on the time it took for the signal to arrive from each satellite. Theoretically, having 3 such range measurements would be adequate to triangulate a receiver's position. This assumes that the internal clock of a GNSS receiver is synchronous with the satellite's clocks. However, in practice, these distances measurements are biased by receiver and satellite clock ranges, and are hence called *pseudoranges*. Furthermore, pseudorange measurements are subject to errors from atmospheric delays, errors in broadcast data, multipath errors, and other factors that introduce noise.

The objective of this report is to present the mathematics of determining low dynamics receiver positions based on pseudorange measurements using the Extended Kalman Filter (EKF). The low-dynamics here refers to an assumption that these receivers are experiencing slow accelerations. In more mathematical terms, the state vector considers the positions and the velocities of this receiver, but neglects the acceleration. This approach is consistent with what is found in actual GNSS receiver software, where they have separate (Kalman) filters for stationary single-point positions (zero velocity and acceleration), low-dynamics cases (nonzero velocity, low acceleration), and high-dynamics cases (high acceleration). The Extended Kalman Filter is considered for this problem and is presented as a solution for the low dynamics receiver. Numerical simulations are performed on synthetic GPS data to demonstrate how the EKF functions as an effective estimator for GNSS receivers.

II. Background

Kalman Filtering is a technique that aids in estimation by combining the system dynamics with the statistical nature of the measurement noise [1]. The system dynamics is represented by a state-space model, which can include any number of unknowns. In a low dynamics GNSS estimator, we are concerned with the positions and velocities of the receiver along with the bias and drift of its internal clock. The state estimate uses a weighting function, called the Kalman gain, which is optimized to produce a minimum error variance. For linear system models, the Kalman filter is an optimal filter that provides a closed-form unbiased estimate. For nonlinear system models, the EKF is used, which is based on the first-order Taylor series linearization of the system and measurement dynamics.

A. Coordinate Systems

Coordinates on Earth are represented through *latitude, longitude, and altitude*, which together make up the spherical coordinate system. Positive latitudes are in the Northern Hemisphere, positive longitudes are to the west of the Prime Meridian. However, spherical coordinates are nonlinear; for example, 1 degree of latitude spans 111 $\mathtt{km}$ at the equator, but reducing to 79 $\mathtt{km}$ at 45 degrees north or south.

This problem is solved by Earth-centred Earth-fixed (ECEF) coordinates. It is a Cartesian coordinate system (x, y, z), which rotates with the Earth. Today, satellite systems use ECEF coordinate systems to perform calculations before converting them to latitudes and longitudes for easy output and visualization. For computing relative orientation, the East-North-Up (ENU) convention is used, which is part of a system called local tangent plane coordinates.

B. Related Work

The usage of EKF for pseudorange estimation is standard practice in the industry. There is extensive work in existing

literature detailing the application of Kalman filters for GNSS receiver estimation problem.

Sarunic presented EKF algorithms for stationary, low dynamics, and high dynamics receivers [3]. The document also details a special case where numerical round-off errors are observed, and presents a solution to address that problem. A similar thesis by [2] presents least-squares and Kalman filter based solutions under a similar context. There have also been personal projects devoted to finding least-squares solution for the estimator using raw GNSS data from Android devices [4]. Other related work also covers the usage of Kalman filter based estimators for fusing inertial sensor data along with GNSS pseudoranges in order to arrive at a better estimate [5].

C. GNSS Positioning

GNSS works on the principle of signal triangulation. A timestamped signal is sent from each satellite to a receiver on the ground. Since the signal travels at the speed of light, the time that it takes to arrive can be measured by comparing the timestamp of the receiver's clock and that of the satellite clock. This scenario introduces three unknown positional variables and one unknown variable for the receiver's clock bias, which may not synchronize with the satellite time.

The pseudorange of the i-th receiver ρ_i is a function of the known satellite position (X_1, Y_1, Z_1), the unknown receiver clock bias ct_r, and the unknown receiver position (x, y, z).

$$\rho_i = [(X_i - x)^2 + (Y_i - y)^2 + (Z_i - z)^2]^{1/2} + ct_r \tag{1}$$

Solving for the 4 unknowns $i = 1 \dots 4$ requires pseudorange measurements from at least 4 satellites. There can be other sources of noises and bias in the pseudorange measurement, but the most promiment one is the receiver clock bias. Other sources are neglected for the formulation of this problem.

For 4 satellites, this problem yields 4 equations:

$$\begin{aligned}
\rho_1 &= [(X_1 - x)^2 + (Y_1 - y)^2 + (Z_1 - z)^2]^{1/2} + ct_r \\
\rho_2 &= [(X_2 - x)^2 + (Y_2 - y)^2 + (Z_2 - z)^2]^{1/2} + ct_r \\
\rho_3 &= [(X_3 - x)^2 + (Y_3 - y)^2 + (Z_3 - z)^2]^{1/2} + ct_r \\
\rho_4 &= [(X_4 - x)^2 + (Y_4 - y)^2 + (Z_4 - z)^2]^{1/2} + ct_r
\end{aligned} \tag{2}$$

For brevity, let (1) be written as $\rho_i = f(x, y, z, b)$, where the clock bias $b = ct_r$. The function f can be linearized about a nominal trajectory $(\hat{x}, \hat{y}, \hat{z}, \hat{b})$ using the Taylor series expansion results in:

$$\begin{aligned}
&f(x, y, z, b) = \\
&f(\hat{x}, \hat{y}, \hat{z}, \hat{b}) + (x - \hat{x}, y - \hat{y}, z - \hat{z}, b - \hat{b}) f'(x, y, z, b)
\end{aligned} \tag{3}$$

The first order linearization results in:

$$\rho_i = \hat{\rho}_i - \frac{X_i - \hat{x}}{\hat{r}_i}\delta x - \frac{Y_i - \hat{y}}{\hat{r}_i}\delta y - \frac{Z_i - \hat{z}}{\hat{r}_i}\delta z \tag{4}$$

Where $\hat{r}_i = \hat{\rho}_i - \hat{b}$.

These equations can be arranged into a concise matrix formulation, where the residual between the observed and computed pseudoranges from i satellites are determined as:

$$\begin{bmatrix} \delta\rho_1 \\ \vdots \\ \delta\rho_i \end{bmatrix} = \begin{bmatrix} a_{x,1} & a_{y,1} & a_{z,1} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ a_{x,i} & a_{y,i} & a_{z,i} & 1 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta z \\ \delta b \end{bmatrix} \tag{5}$$

Where $a_{x,i}$ is the coefficient of δx in (4). Equation (5) can be written as:

$$\Delta\rho = \mathbf{H}\Delta\mathbf{x} \tag{6}$$

It can be noted that (6) can be solved as $\Delta\mathbf{x} = \mathbf{H}^{-1}\Delta\rho$ if $i = 4$, or with the least squares solution $\Delta x = (\mathbf{H}^\top\mathbf{H})^{-1}\mathbf{H}^\top\Delta\rho$ when $i > 4$.

III. Extended Kalman Filter

The Extended Kalman Filter (EKF) is a more suitable method than the least-squares solution for this task. The EKF is inherently recursive in nature, and it uses the previous measurements (posterior) to arrive at a new estimate, as opposed to least squares, which only provides for a point solution.

A. State Space Modelling

For a stationary receiver, the system can be modelled as:

$$\underbrace{\begin{bmatrix} \delta x_{k+1} \\ \delta y_{k+1} \\ \delta z_{k+1} \\ \delta b_{k+1} \\ \delta \dot{b}_{k+1} \end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{F}} \underbrace{\begin{bmatrix} x_k \\ y_k \\ z_k \\ b_k \\ \dot{b}_k \end{bmatrix}}_{\mathbf{x}}$$

Where T_s is the sampling interval between two measurements from the GNSS. However, for a low dynamics moving receiver, the state is chosen to be a combination of the positions and velocities, resulting in $\mathbf{x} = (x, \dot{x}, y, \dot{y}, z, \dot{z}, b, \dot{b})$. The state transition matrix is then given by:

$$\begin{aligned}
\mathbf{F} &= \begin{bmatrix} \mathbf{F}_s & 0 & 0 & 0 \\ 0 & \mathbf{F}_s & 0 & 0 \\ 0 & 0 & \mathbf{F}_s & 0 \\ 0 & 0 & 0 & \mathbf{F}_s \end{bmatrix}_{8\times 8} \\
\mathbf{F}_s &= \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}
\end{aligned} \tag{7}$$

It is noted that $\mathbf{F}$ is an 8×8 matrix since the state vector now consists of 3 positions, 3 velocities, the clock bias, and the clock drift.

The measurement model is nonlinear, as can be inferred from (1). It can be written as:

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{8}$$

Here, $\mathbf{y}$ is a vector containing all the pseudorange measurements $\rho_{1\dots i}$. The nonlinear matrix $\mathbf{h}(\mathbf{x})$ can be formed from (2). However, the Jacobian of $\mathbf{h}$ is of higher relevance to this application than $\mathbf{h}$ itself.

The Jacobian of $\mathbf{h}(\mathbf{x})$, $\mathbf{H_x}$ can be found as:

$$\mathbf{H_x} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \rho_1}{\partial x} & 0 & \frac{\partial \rho_1}{\partial y} & 0 & \frac{\partial \rho_1}{\partial z} & 0 & 1 & 0 \\ \frac{\partial \rho_2}{\partial x} & 0 & \frac{\partial \rho_2}{\partial y} & 0 & \frac{\partial \rho_2}{\partial z} & 0 & 1 & 0 \\ \frac{\partial \rho_3}{\partial x} & 0 & \frac{\partial \rho_3}{\partial y} & 0 & \frac{\partial \rho_3}{\partial z} & 0 & 1 & 0 \\ \frac{\partial \rho_4}{\partial x} & 0 & \frac{\partial \rho_4}{\partial y} & 0 & \frac{\partial \rho_4}{\partial z} & 0 & 1 & 0 \end{bmatrix} \tag{9}$$

Where, for measurements $i = 1, 2, 3, 4$, the partial derivatives of the pseudorange can be found by differentiating (1) analytically.

Furthermore, the process covariance matrix $\mathbf{Q}$ is a function of the sampling interval T_s and the satellite clock frequency noise S_f and satellite clock frequency drift noise S_g (white noise spectral density).

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_s & 0 & 0 & 0 \\ 0 & \mathbf{Q}_s & 0 & 0 \\ 0 & 0 & \mathbf{Q}_s & 0 \\ 0 & 0 & 0 & \mathbf{Q}_c \end{bmatrix}_{8\times 8} \tag{10}$$

$$\mathbf{Q}_s = \begin{bmatrix} \frac{T_s^3}{3} & \frac{T_s^2}{2} \\ \frac{T_s^2}{2} & T_s \end{bmatrix}$$

$$\mathbf{Q}_c = \begin{bmatrix} S_f T_s + \frac{S_g T_s^3}{3} & \frac{S_g T_s^2}{2} \\ \frac{S_g T_s^2}{2} & S_g T_s \end{bmatrix}$$

Putting this together, we obtain the state space formulation of the dynamics as:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}\mathbf{x}_k + \mathbf{q}_k \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k) + \mathbf{r}_k \end{aligned} \tag{11}$$

The Gaussian process noise $\mathbf{q}_k \sim N(\mathbf{0}, \mathbf{Q})$, and the measurement noise $\mathbf{r}_k \sim N(\mathbf{0}, \mathbf{R}_k)$.

It is important to note that while the process model is linear, the nonlinearity of this system stems solely from the measurement equation. The EKF equations for these system can now be formulated.

B. Extended Kalman Filter Algorithm

The EKF algorithms are an extension of the Kalman filter algorithms to nonlinear dynamic models [6][7].

The General EKF algorithm can be adapted to this case as follows:

1) Set initial values:
$$\hat{\mathbf{x}}, \mathbf{P}_0$$
2) Predict state and error covariance:
$$\begin{aligned} \hat{\mathbf{x}}_k^- &= \mathbf{F}\hat{\mathbf{x}}_{k-1}^- \\ \mathbf{P}_k^- &= \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^\top + \mathbf{Q} \end{aligned}$$
3) Compute the Kalman gain:
$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_\mathbf{x}^\top (\mathbf{H}_\mathbf{x} \mathbf{P}_k^- \mathbf{H}_\mathbf{x}^\top + \mathbf{R})^{-1}$$
4) Update the state estimate and the error covariance
$$\begin{aligned} \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{h}(\hat{\mathbf{x}}_k^-)) \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k \mathbf{H}_\mathbf{x} \mathbf{P}_k^- \end{aligned}$$

Steps 2–4 are recursive in nature. This implies that after one step, the updated values of state and error covariance can be used for the next iterations.

The main advanage of EKF is its simplicity. Linearization is a common technique employed in various applications to account for model nonlinearities. It is also highly versatile for processess affected by Gaussian noise. In this case, since the measurement model function is differentiable, it is possible to apply the EKF to this model. It is also common in many modern numerical applications to use automatic differentiation for the computation of Jacobians, rather than computing the derivatives by hand.

IV. Implementation

Numerical simulations are conducted to demonstrate the efficacy of the EKF algorithm for GNSS receiver localization. The codes for this project can be accessed on GitHub[1]. The repository contains the library `gnss` which was created for this project, along with a Jupyter Notebook implementation of EKF for localizing a GPS receiver.

A. Libraries

The Python library `gps_helper` is used extensively for the simulation of synthetic GPS data. For this project, a custom library `gnss` was created as a wrapped around the library easier implementation. The wrapper contains a class `SimulatePseudorange` which has methods useful for generating pseudorange measurement data.

Apart from these, the standard Python libraries (mainly `NumPy`) are used for matrix algebra.

B. Modelling

The newer version of `NumPy` has improved methods of generating random noise. The seed values are randomly sampled from the OS, ensuring that added noise is non-repeatable and random.

The initial position and velocity of the receiver are set in the Notebook, along with the number of satellites. In the example provided in this report, the sampling interval is considered to be `1s`, and the number of satellites are 4.

The trajectory used here is a combination of line segments that travel east, north, west or south, since they are defined in the ENU coordinate frame. In order to simulate GPS data based on these line segments, there is a text file that contains details of 4 real GPS satellites. Using the methods created in the library, the trajectory of the satellites and the receiver can be visualised. This can be seen in (1).

C. Extended Kalman Filter

To implement EKF, some useful functions for computing the system model and Jacobian are defined. Using these functions, the EKF algorithm is implemented. The codes for these functins are attached in the appendix of this report.

[1]https://github.com/shreeram-murali/gps-ekf

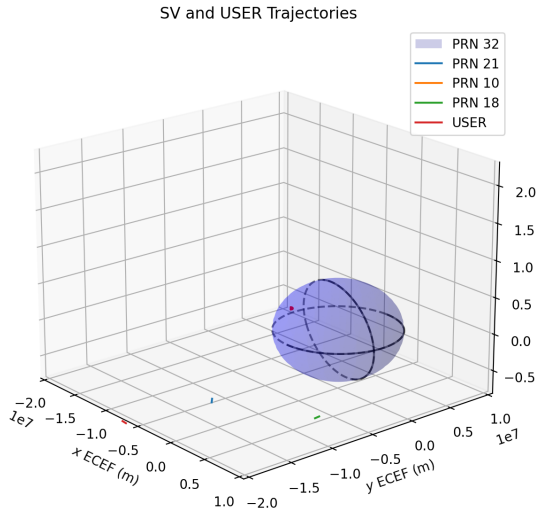

Fig. 1. Satellite and Receiver trajectory in ECEF coordinates

V. Results

The results of the simulation were obtained by setting a trajectory which moves the receiver in the following sequence, with each segment of 0.1 miles: *east, north, east, east, south, west, south, east, south, west, west, north, west*. Figure (3) shows the estimated trajectory; it can also be observed that the simulated trajectory follows the grid pattern in the graph.

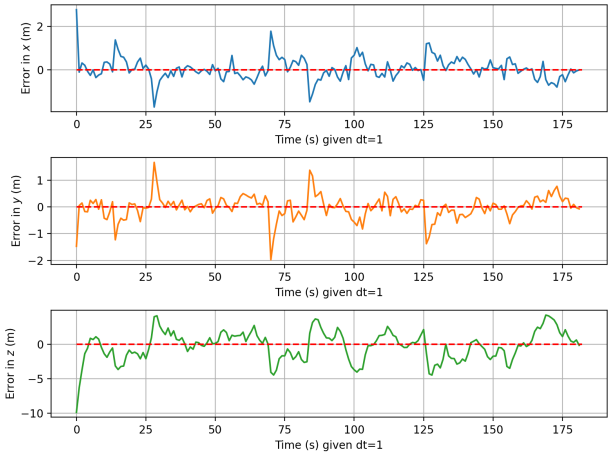

Fig. 2. EKF errors in estimates

The errors along each axis can be visualized in Figure (2). It is observed that the errors are within a few meters of the true value.

VI. Conslusion and Future Work

The position of GNSS receivers are commonly localized using EKF-based algorithms. This report presented the application of EKF for a simulated GNSS receiver trajectory. The results yield good position tracking of the receiver, with the error within the range of few meters. This is usually the case for most GPS receivers today; the error usually improves with the number of satellites. There are modern GPS receivers that have real-time kinematics (RTK) facilitating centimeter-level accuracy. However, for singular pseudorange-based measurements, GPS measurements are accurate to a few meters.

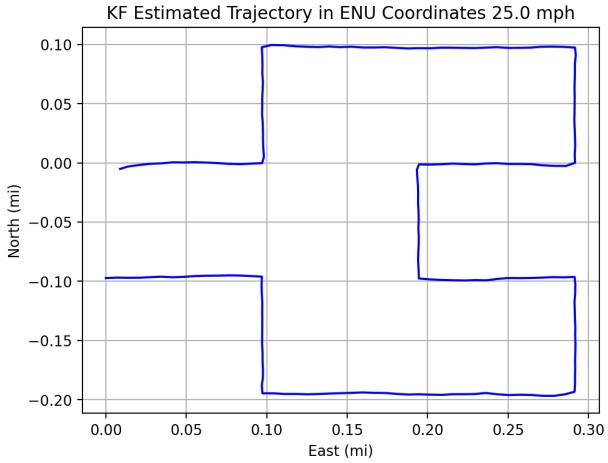

Fig. 3. Estimated trajectory of the receiver

Future work can include the implementation of tracking high dynamics receivers. Usage of the Unscented Kalman Filter, employing the Unscented Transform, is a natural next step [8][9]. Another practical implementation can be involve the usage of real-world raw GNSS data from multiple devices paired with accurate ground-truth data. Such a dataset is provided by Google recorded from Android devices[2] including GPS, GLONASS, Beidou.

References

[1] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems," Journal of Basic Engineering (ASME), Vol. 82D, March 1960, pp. 35–37.

[2] M. Lashley, 'Kalman Filter Based Tracking Algorithms For Software GPS Receivers'.

[3] P. W. Sarunic, 'Development of GPS Receiver Kalman Filter Algorithms for Stationary, Low-Dynamics, and High-Dynamics Applications'.

[4] 'Calculating Your Android's Position from Raw GNSS Pseudorange Measurements', Mitchell D Johnson. Accessed: Apr. 12, 2024. [Online]. Available: https://www.johnsonmitchelld.com/2021/03/14/least-squares-gps.html

[5] E. D. Kaplan and C. Hegarty, Eds., Understanding GPS: principles and applications, 2nd ed. in Artech House mobile communications series. Boston: Artech House, 2006.

[6] Jazwinski, A. H. 1970. Stochastic Processes and Filtering Theory.

[7] Grewal, M. S., and Andrews, A. P. 2015. Kalman Filtering: Theory and Practice Using MATLAB. 4th edn. Wiley.

[8] Julier, S. J., Uhlmann, J. K., and Durrant-Whyte, H. F. 1995. A new approach for filtering nonlinear systems. Pages 1628–1632 of: Proceedings of the 1995 American Control, Conference, Seattle, Washington.

[9] Julier, S. J., Uhlmann, J. K., and Durrant-Whyte, H. F. 2000. A new method for the nonlinear transformation of means and covariances in filters and estimators. IEEE Transactions on Automatic Control, 45(3), 477–482.

[2]https://www.kaggle.com/datasets/google/android-smartphones-high-accuracy-datasets

APPENDIX

Jacobian:

```
def Hx(x1, satpos):

    H = np.zeros((nsats, 8))
    for i in range(nsats):
        denominator = np.linalg.norm(
            x1[:6:2].flatten() - satpos[i,:]
            )
        H[i,0:6:2] = (x1[:6:2].flatten() -
            satpos[i,:]) / denominator
        H[i,6] = 1.

return H
```

Measurement model function:

```
def hx(x1, satpos):
    y_pred = np.zeros((nsats, 1))
    for i in range(nsats):
        y_pred[i] = np.linalg.norm(
            x1[:6:2].flatten() - satpos[i,:] )
            + x1[6]

    return y_pred
```

The EKF implementation:

```
samples = satpos.shape[2]
xh = np.zeros((samples, 3))
Pdiag = np.zeros((samples, 8))

x[0:6:2, 0] = userpos_ecf[0, :] + 25 *
    random_normal(0, 1, 3)


for i in range(samples):
    mygps.measurement(userpos_ecf[i, :],
        satpos[:,:,i])

    H = Hx(x, satpos[:,:,i])

    xp = F @ x
    Pp = F @ P @ F.T + Q

    # K = solve(H @ Pp @ H.T + R, Pp @ H.T)
    K = Pp @ H.T @ np.linalg.pinv(H @ Pp @ H.T
        + R)

    yp = hx(xp, satpos[:,:,i])

    x = xp + K @ (mygps.user_pseudorange - yp)
    P = Pp - K @ H @ Pp

    xh[i,:] = x[0:6:2, 0]
    Pdiag[i,:] = P.diagonal()
```

The `SimulatePseudorange` class from the library specifically created to replicate GPS measurements salted with noise.

```
class SimulatePseudorange:

def __init__(self, pr_std = 0, cdt = 0,
    nsats = 4, dt=1):

    self.pr_std = pr_std
    self.cdt = cdt
    self.nsats = nsats
    self.user_pseudorange =
        np.zeros((nsats, 1))

    self.GPS_dataset =
        self.create_gps_dataset(dt)

def measurement(self, user_pos, sat_pos)
    -> None:

    for i in range(self.nsats):
        self.user_pseudorange[i, 0] =
            np.linalg.norm(user_pos -
            sat_pos[i])
        self.user_pseudorange[i, 0] +=
            self.cdt + random_normal(0,
            self.pr_std)

def generate_usersatpos(self,
    line_segment, user_velocity,
    today=datetime.now(timezone.utc)):

    userpos_enu, userpos_ecf, satpos,
        satvel =
        self.GPS_dataset.user_traj_gen(
        route_list=line_segment,
        vmph=user_velocity,
        yr2=today.year - 2000,
        mon=today.month,
        day=today.day,
        hr=today.hour,
        minute=today.minute,
        sec=today.second
    )

    return userpos_enu, userpos_ecf,
        satpos, satvel

def traj3d_viz(self, satpos, userpos_ecf,
    ele=20, azi=-40):
    return
        GPS.sv_user_traj_3d(self.GPS_dataset,
        satpos, userpos_ecf, ele, azi)

def ecef2enu(self, pos):
    refecef, lla1, lla2 =
        self.GPS_dataset.ref_ecef,
        self.GPS_dataset.ref_lla[0],
        self.GPS_dataset.ref_lla[1]
    return GPS.ecef2enu(pos, refecef, lla1,
        lla2)

def create_gps_dataset(self, Ts):
    ds =
        GPS.GPSDataSource('GPS_tle_1_10_2018.txt',
        rx_sv_list = ('PRN 32','PRN
        21','PRN 10','PRN 18'),
                    ref_lla=(38.8454167,
                        -104.7215556,
                        1903.0), ts=Ts)
    return ds
```