ASEN 6519: State Estimation

# A trilateration algorithm for localized positioning systems

Ian Thomas
University of Colorado, Boulder
5/6/2021

## Contents

# 1 Introduction

In the next decade, NASA hopes to send many manned missions to the moon, including plans to set up a permanent base on the moon where astronauts can inhabit for extended periods of time. With people comes the need for a reliable positioning and communication network that will allow astronauts to know where they are and talk to each other in a localized area (∼10 km radius). This has driven the need to develop systems such as JPL's LunaNet, for which my senior project team (P4LO — Positioning for Lunar Operations) is developing a prototype network of pseudolites (stationary satellites, basically cell towers) to provide a first run positioning and communications network. I want to focus on the positioning aspect of the project (and the ranging/signal processing going on behind the scenes). The concept of operations for the mission is shown below:

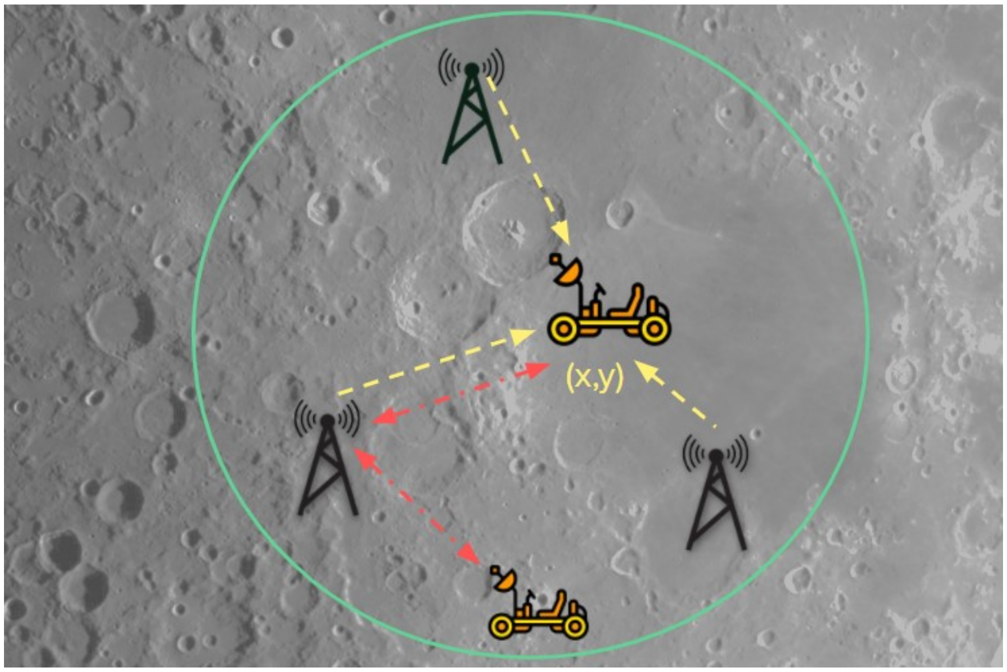


Figure 1: P4LO Lunar CONOPS. Yellow arrows depict the range measurements.

## 1.1 Ranging

Team P4LO has opted for a one-way ranging system, for which a pseudolite sends a CDMA (code-division multiple access) signal for a receiver to decode and determine the code delay, and then the initial range measurement after that [1]. To get within 10 m accuracy, the combined error between the pseudolite clock, the receiver clock, and the algorithm to determine the time of arrival of the signal must be at most 30 ns. The main factors affecting this time are the stability/Allen deviation of the clocks on both the pseudolite and the receiver [2], as well as the quality of the signal received by the receiver (captured by the signal to noise ratio, or SNR) [1]. If the pseudolites and receivers are calibrated properly and there isn't any interference, individual range measurements can be made to within 10 meter 1-$\sigma$ accuracy.

The 10 meter accuracy is under ideal conditions though, and can be thrown off due to clock drifts and interferences from other sources, such as other pseudolites. On the lunar surface, there aren't many of the interferences or environmental factors that affect the GPS signals here on earth (other cellular interference, nonlinear atmospheric loss, GPS satellite position uncertainty, or buildings/trees, having to transmit signals over 20,000 km), which in many ways simplifies the ranging problem. However, the P4LO local trilateration concept suffers from one major problem that GPS doesn't suffer from: the near-far problem.

The near-far problem for CDMA happens when one transmitter is much closer to the receiver than other transmitters, so if they are all transmitting at the same power, the signal from the closer transmitter will appear much stronger than the other signals and drown them out. Possible solutions to this problem include power scaling each transmitter for an initial estimate of receiver location, but this can be only optimized for one receiver, or implementing a form of adaptive gain control. The near-far problem has the effect of decreasing the SNR for every signal coming from further away, while the closer signal appears clearer (higher SNR).

Extremely noisy range measurements can also be obtained from the power of the signal arriving at the receiver: the more power, the closer the transmitter. Transmit power drops by an inverse square law with distance, so if the transmitters are all transmitting the same power, it is possible to roughly approximate the distance to each of them.

## 1.2 Positioning

The receiver has to determine its own position and clock bias based on the noisy range measurements it receives from pseudolites in a process called trilateration. There are many ways to implement the trilateration algorithm, such as least squares, but one of the challenges of implementing the algorithm is in the case of near-far interference, as team P4LO's concept will encounter.

# 2 Problem Definition

## 2.1 Pseudolite Geometry

Let three pseudolites be placed in an equilateral triangle of side length 5 km on a flat plane (assuming no curvature of the moon). Each pseudolite $i$ is defined by its coordinates in meters $p_i = [\xi_i, \eta_i]^{\mathrm{T}}$. The receiver position at timestep $k$ is given by $x^k = \left[\xi^k, \eta^k\right]^{\mathrm{T}}$. The pseudolites will be arranged such that pseudolite 1 is at $[0, 0]$ m, pseudolite 2 is at $[5000, 0]$ m, and pseudolite 3 is at $\left[2500, 2500\sqrt{3}\right]$ m:

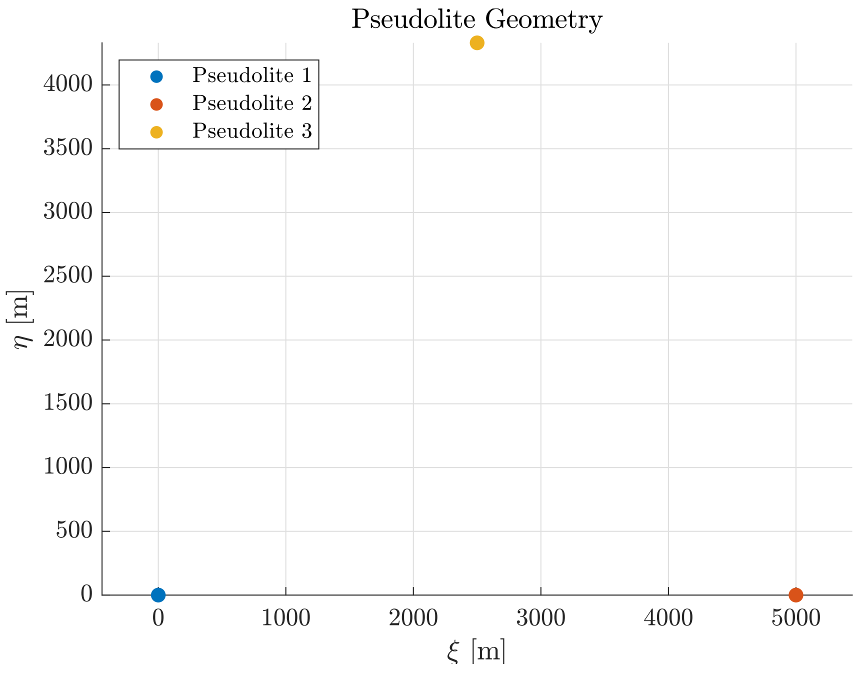


Figure 2: Pseudolite Geometry

## 2.2 Ranging and Trilateration

The range equation at timestep $k$, assuming no noise is present, is given by:

$$\begin{aligned} r_i^2 &= \left(p_i - x^k\right)^{\mathrm{T}} \left(p_i - x^k\right) \\ &= \left(\xi_i - \xi^k\right)^2 + \left(\eta_i - \eta^k\right)^2 \end{aligned} \tag{1}$$

The trilateration equations are then given by the system of range equations:

$$\begin{aligned} r_1 &= \sqrt{(\xi_1 - \xi^k)^2 + (\eta_1 - \eta^k)^2} \\ r_2 &= \sqrt{(\xi_2 - \xi^k)^2 + (\eta_2 - \eta^k)^2} \\ r_3 &= \sqrt{(\xi_3 - \xi^k)^2 + (\eta_3 - \eta^k)^2} \end{aligned} \tag{2}$$

Range measurements, however, consist of a noise term at timestep $k$ of $v_r^k$ and the receiver clock bias $b^k$:

$$\rho_i = \sqrt{(\xi_i - \xi^k)^2 + (\eta_i - \eta^k)^2} + v_{i,r}^k + b^k c \tag{3}$$

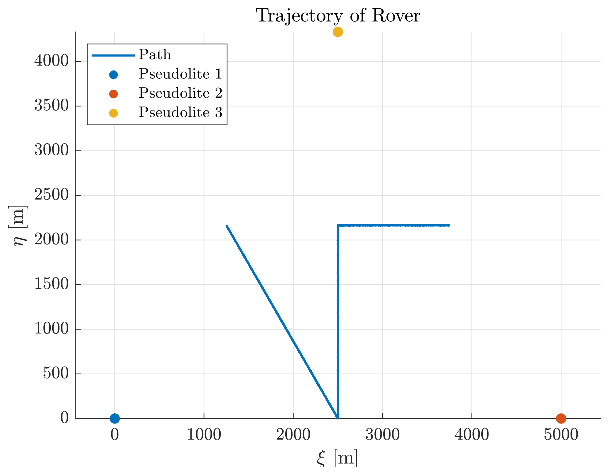
When solving the trilateration equations, one must find the best $\hat{b}^k$, $\hat{\xi}^k$, and $\hat{\eta}^k$ to satisfy the following equations:

$$\begin{aligned} \rho_1 - \hat{b}^k &= \sqrt{(\xi_1 - \hat{\xi}^k)^2 + (\eta_1 - \hat{\eta}^k)^2} \\ \rho_2 - \hat{b}^k &= \sqrt{(\xi_2 - \hat{\xi}^k)^2 + (\eta_2 - \hat{\eta}^k)^2} \\ \rho_3 - \hat{b}^k &= \sqrt{(\xi_3 - \hat{\xi}^k)^2 + (\eta_3 - \hat{\eta}^k)^2} \end{aligned} \tag{4}$$

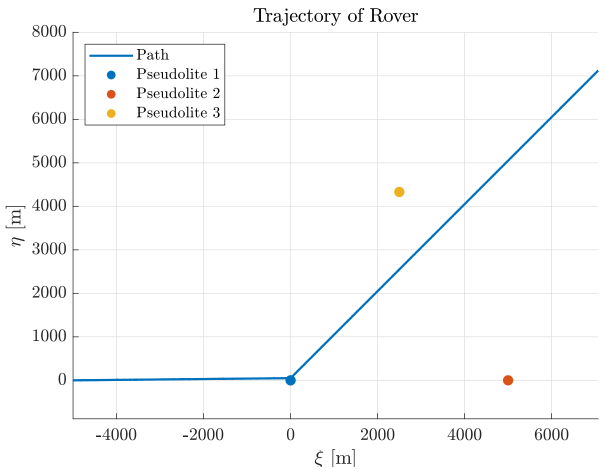
If we have the SNR of the range measurements available, it is possible to weight one equation over another in order to rely more heavily on the more accurate measurements.

## 2.3 Truth Model Simulation and Measurement Generation

Rovers on the lunar surface typically will move in straight lines from point A to point B at slow speeds, so as not to cause wear and tear. The truth paths of the rovers can really look like any path in two dimensions, so long as the speed of the rover doesn't exceed some upper bound. For this simulation, the rover will move at roughly 1-2 m/s. Four useful test cases come up: trajectory which doesn't come too close to the pseudolites and no clock drift, trajectory which does pass near the pseudolites without clock drift, trajectory that doesn't come near the pseudolites with clock drift, and finally a trajectory that passes near one of the pseudolites with clock drift. The paths were arbitrarily selected to fill each criterion, and are constant with or without the clock drift. They are discretized at $\Delta t = 5$ s:

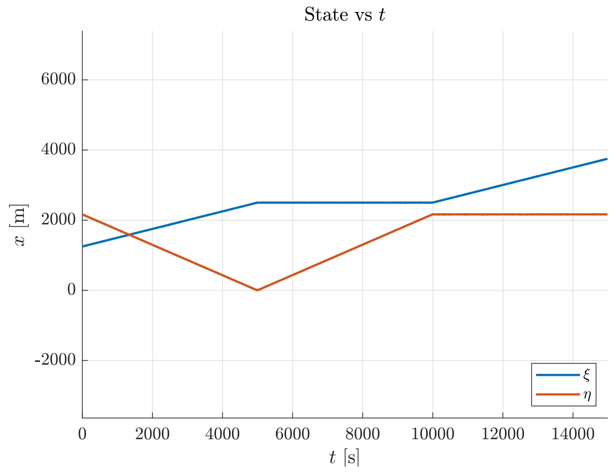


(a) No Near-Far Trajectory

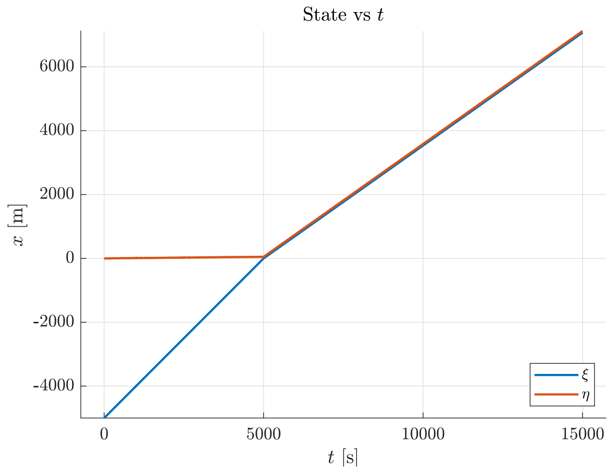


(b) Near-Far Trajectory

In each case, the rover starts from the left and ends up at the right (in the direction of positive $\xi$). The first test case stays roughly in between the pseudolites (good *dilution of* *precision*), whereas the second one passes within 50 meters of a pseudolite and strays far outside the pseudolite triangle, which combines near-far affects with poor geometry. Here are the states as a function of time for each case:



(a) No Near-Far States



(b) Near-Far States

We use (3) to generate the noisy measurements, where $v_{i,r}^k$ is distributed as function of the SNR of the received signal. We will make a few assumptions to simplify the generation of the noisy distribution. First, we assume that the strongest signal (the one coming from the pseudolite closest to us), has the maximum possible SNR after CDMA acquisition. We will let $v_{i,r}^k \sim \mathcal{N}(\mu = r_i, \sigma = \sigma_\rho)$ for the closest pseudolite, where $\sigma_\rho = \sigma_{\text{min}} = 10$ m, so the closest pseudolite always has a range measurement standard deviation of 10 meters. To obtain the standard noise deviations from the other pseudolites, we will leverage the fact that power drops with distance squared, or that $P_2/P_1 = r_2^2/r_1^2$. We assume the entire noise floor is due to contributions from the first signal (weaker signals won't contribute significantly to noise floor), and that the noise floor stays the same for both signals 1 and 2, so $\text{SNR}_2 = \text{SNR}_1(r_2^2/r_1^2)$. SNR is related to range standard deviation by the following equation [3]:

$$\sigma_r = \frac{c}{2\sqrt{2}B\sqrt{\text{SNR}}} \propto \frac{1}{\sqrt{SNR}} \tag{5}$$

The inverse square law for power and distance and the inverse root law for range standard deviation and SNR cancel out and the relationship simplifies to:

$$\sigma_i = \left(\frac{r_i}{r_{\text{min}}}\right)\sigma_{\text{min}} \tag{6}$$

All of this to say that each $v_{i,r}^k \sim \mathcal{N}(r, \sigma = (r_i/r_{\text{min}})\sigma_{\text{min}})$. We can add this noise to the simulated measurements, and pass that in along with the relative SNR of the signals (with added noise) as observations. The following are the plots for the simulated measurements:

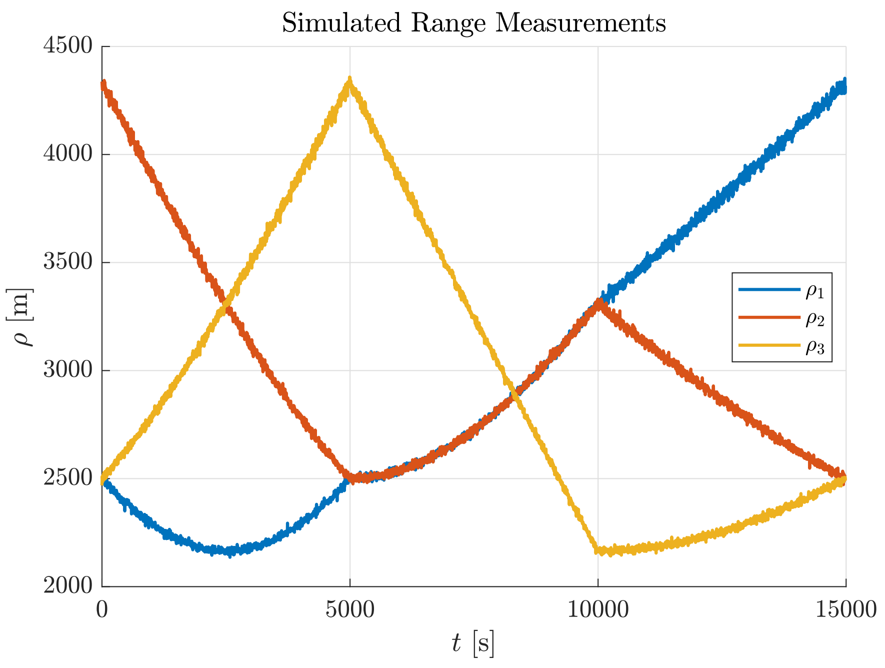


Figure 5: Simulated Measurements for No Near-Far Trajectory

We should expect to see roughly the same noise deviations since the distances only change by a factor of two (so the noise standard deviations should only change by a factor of 2, which shouldn't be very visible on the graph). In the near-far trajectory, however, we should expect to see a drastic increase in the noise variance when the rover approaches a pseudolite closely, and we should see that variance fade as the rover moves further away:

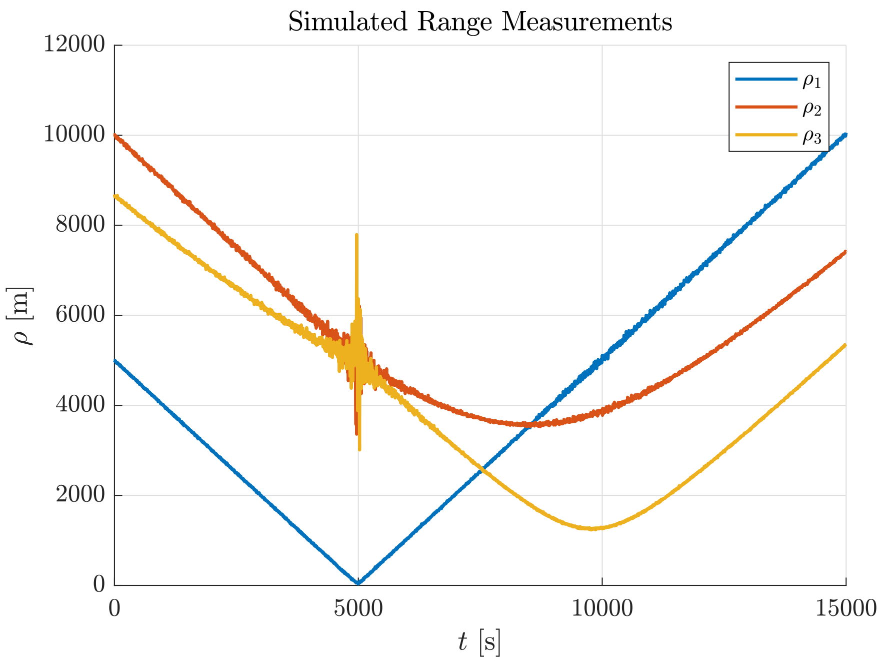


Figure 6: Simulated Measurements for Near-Far Trajectory

All that's left is to generate measurements with clock biases, which drift according to a polynomial $b^t = c_2 t^2 + c_1 t + c_0$. We assume the pseudolites are synchronized since there will likely be a base station surveying and correcting them periodically. We can assume the receiver has a chip-scale atomic clock (drifts  100 $\mu$s a day [4]), so over the duration of four hours we would expect the clock to drift 16 $\mu$s. We'll arbitrarily set the initial bias $c_0 = 10$ $\mu$s, and $c_2 = 3.55\text{e} - 14$ s/$s^2$ and $c_1 = 5.33\text{e} - 10$ s/s to achieve a 16 $\mu$s drift over 4 hours. Here are the new sets of measurements, after multiplying the clock biases by the speed of light:

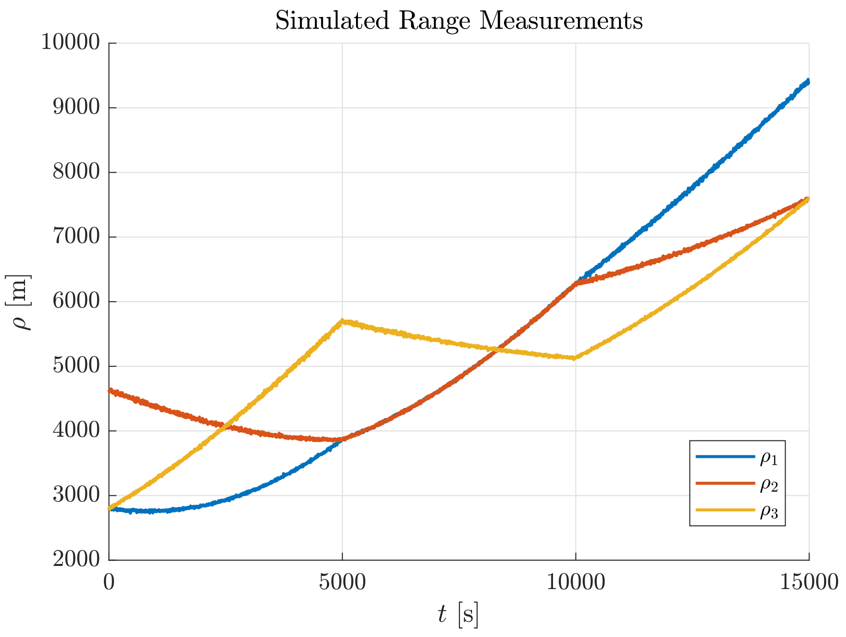


Figure 7: Simulated Measurements for No Near-Far Trajectory and Clock Bias

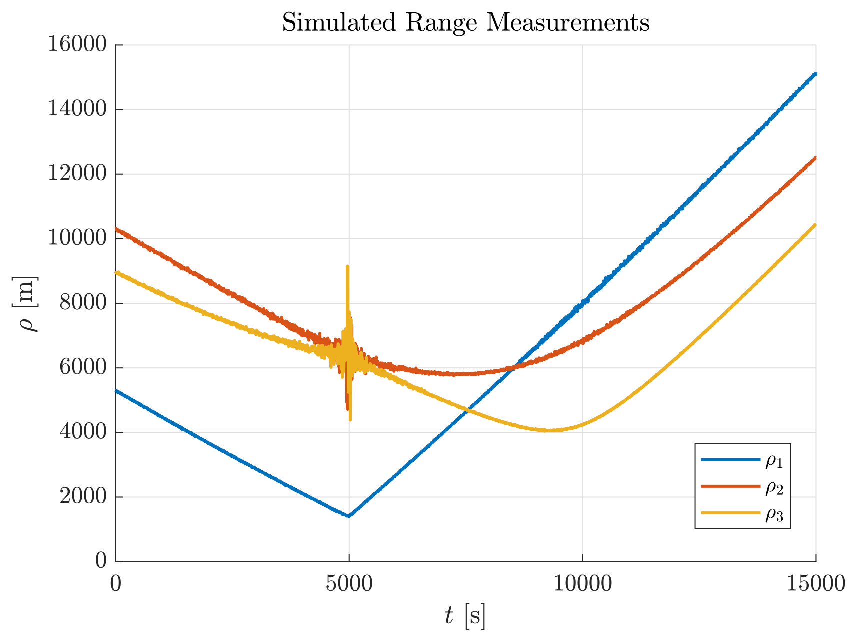


Figure 8: Simulated Measurements for Near-Far Trajectory and Clock Bias

## 3 Objectives

### 3.1 Level 1

Apply a nonlinear least-squares and a particle filter approach to the trilateration problem for both trajectories, no clock bias.

### 3.2 Level 2

Apply a nonlinear least-squares and a particle filter approach to the trilateration problem for both trajectories, with clock bias.

### 3.3 Level 3

Apply a joint nonlinear least-squares/particle filter approach to the trilateration problem for both trajectories, with clock bias. Use the robustness of the particle filter to keep initializing the NLS estimates for clock bias.

## 4 Methods

### 4.1 Level 1

#### 4.1.1 Nonlinear Least Squares

The nonlinear least squares cost function is simply the square of the difference between the actual measurement from the pseudolite vs the measurement generated from the position currently being evaluated multiplied by the inverse of the estimate of the measurement variance [5]:

$$J_{\text{NLS}}(x) = \sum_{i=1}^{3} \frac{\left(\sqrt{(\xi_i - \xi^k)^2 + (\eta_i - \eta^k)^2} - \rho_i\right)^2}{\sigma_{i,r}^2} \tag{7}$$

Since $\rho_i$ is unbiased, we can use it to calculate the ratio of distances to the minimum $\rho_i$ and compute $\sigma_{i,r}$ from that ratio of distances and $\sigma_{\text{min}}$. We need to pick a good initial condition, but after that, we use `fminsearch` from MATLAB® to compute the optimal $x$ to minimize $J_{\text{NLS}}$. We use the estimate of $x$ from the previous timestep to initialize the solver at the current timestep.

### 4.1.2 Particle Filter

We use a sequential-importance-resampling particle filter initialized about a uniform belief 200 meters in each direction centered on the starting location. The particle filter consists of two steps: the dynamics propagation and the measurement update. The dynamics propagation step is simply a random walk of all the particles; there isn't much dynamically happening with the rover moving around how it pleases in two dimensions. Even though the rover typically only moves a maximum of 5-10 meters between each timestep, adding Gaussian noise with standard deviation of 30 meters in each direction ended up yielding good results (this parameter can be varied, however).

The measurement update is the interesting step. We assume that the range error is Gaussian, so the distribution for each measurement is a two-dimensional radial Gaussian distribution centered around each pseudolite. Fortunately in the particle filter implementation, the distributions only need to be known to a normalizing constant. The following is the equation defining this radial Gaussian ceneterd at pseudolite $i$:

$$p(\xi^k, \eta^k | \rho_i) = \exp\left(-\frac{1}{2}\left(\frac{\sqrt{(\xi_i - \xi^k)^2 + (\eta_i - \eta^k)^2} - \rho_i}{\sigma_r}\right)^2\right) \tag{8}$$

Technically, we are missing part of the distribution that accounts for a range measurement in one direction having a distribution that extends in the opposite direction, but we assume this contribution is negligible. The following is a plot of a sample distribution generated by a range measurement of 250 meters, noise deviation of 20 meters, about a pseudolite centered at $(0, 0)$, shown by a yellow point:

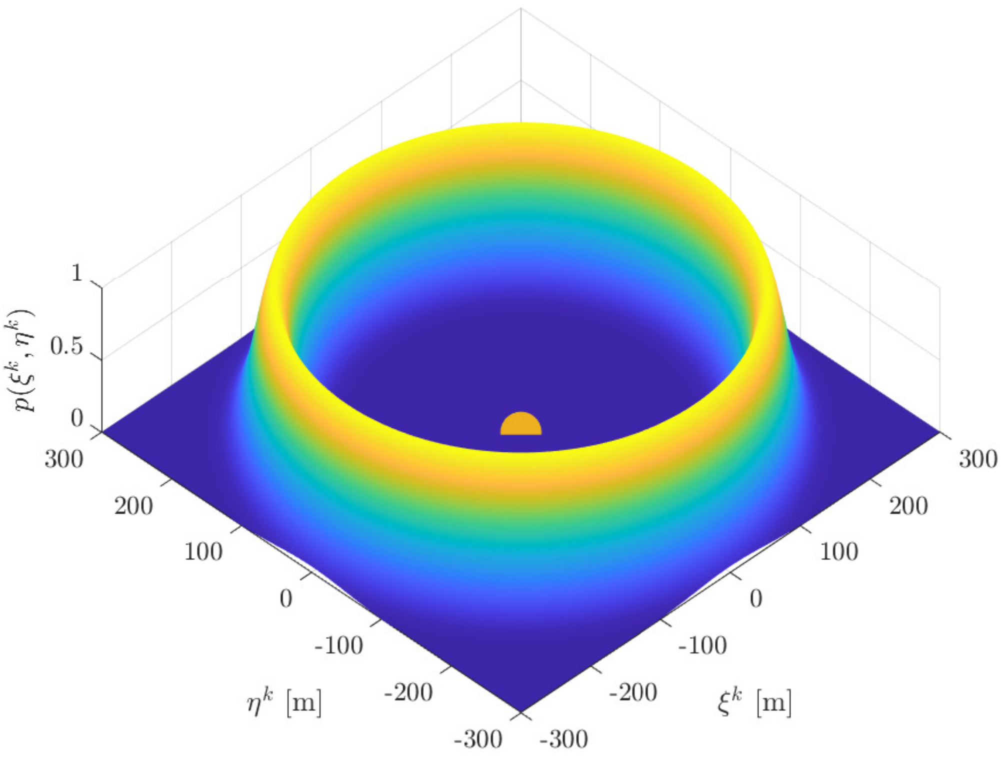


Figure 9: Radial Gaussian Distribution (not normalized)

The weights of each particle in the particle filter is determined by the point-wise multiplication of these distributions evaluated at each particle. The weights are then normalized to form a proper distribution of particles.

At this point the maximum a-posteriori (MAP) and minimum mean-squared error (MMSE) estimates can be computed by taking the particle with the maximum weight and the weighted sum of all particles respectively. An estimate of the error covariance can be made by taking the weighted sum of the difference between each particle and the MMSE and MAP estimates multiplied by its transpose to form a $2 \times 2$ matrix.

At this point we can resample the particles based on their weights, and repeat from the dynamics updating step.

This implementation uses 1000 particles for each step.

## 4.2 Level 2

### 4.2.1 NLS

The NLS state is augmented with the clock bias, with the modified cost function:

$$J_{\text{NLS}}(x) = \sum_{i=1}^{3} \frac{\left(\sqrt{(\xi_i - \xi^k)^2 + (\eta_i - \eta^k)^2} - \rho_i + cb\right)^2}{\sigma_{i,r}^2} \tag{9}$$

Where $b$ is the clock bias estimate and $c$ is the speed of light.

#### 4.2.2 PF

The particle filter is simply augmented with a clock bias state, which is updated via a random walk dynamics update step. This bias is simply subtracted out of the range measurement when it comes time to do the measurement update step.

## 4.3 Level 3

Level three involves using the NLS estimate of clock bias to discipline the particle filter.

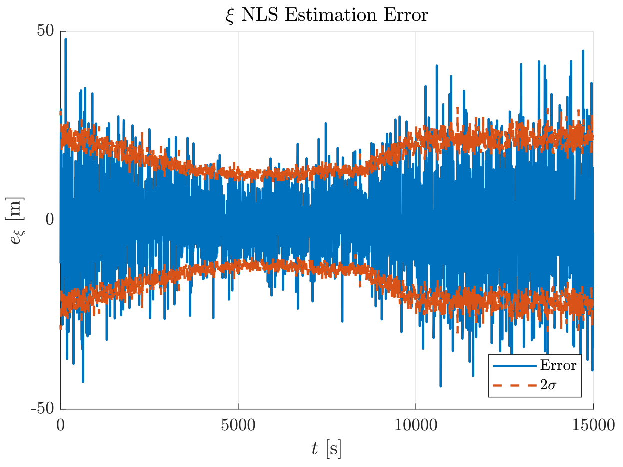
## 5 Results

The following plots show the estimation errors for $\xi$, $\eta$, and $b$ (where applicable). It is the estimation error and $2\sigma$ confidence bounds that characterize the performance of the estimator.

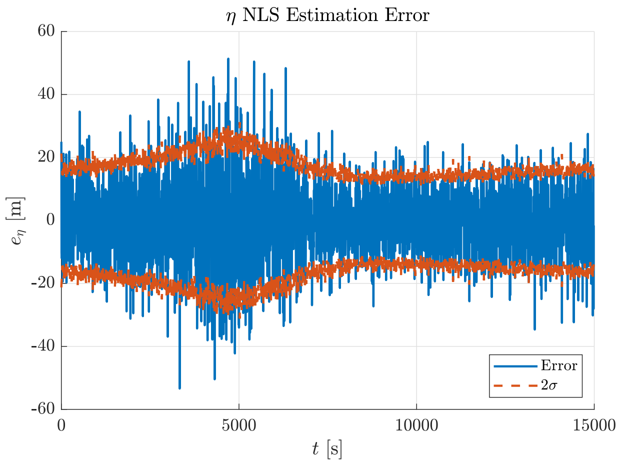
### 5.1 Level 1

The following plots detail the results of nonlinear least squares and the particle filter MMSE and MAP estimates of both trajectories without any clock bias:

#### 5.1.1 NLS

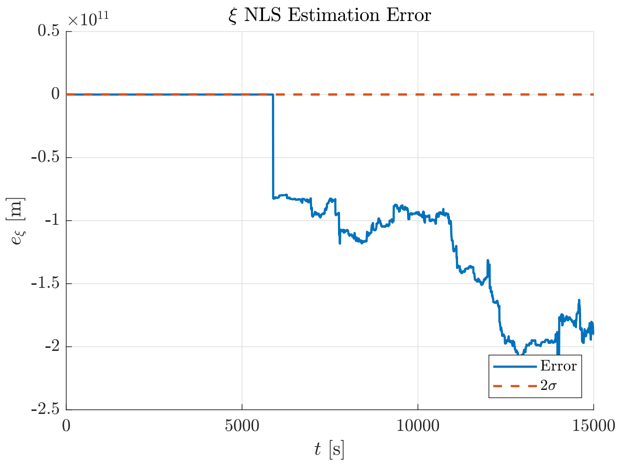


(a) East estimation error

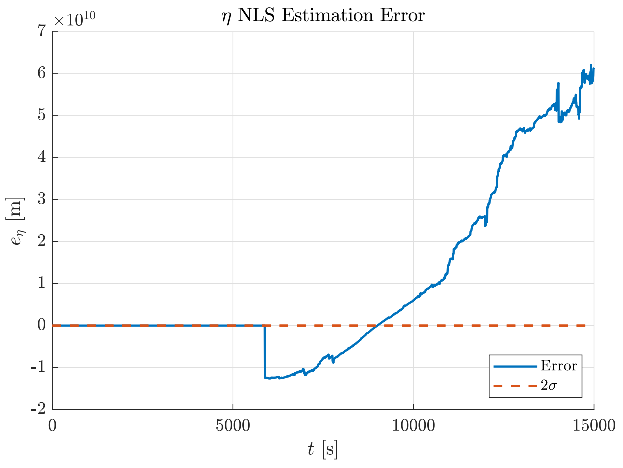


(b) North estimation error

Figure 10: NLS estimation errors for trajectory with no near-far and no clock bias



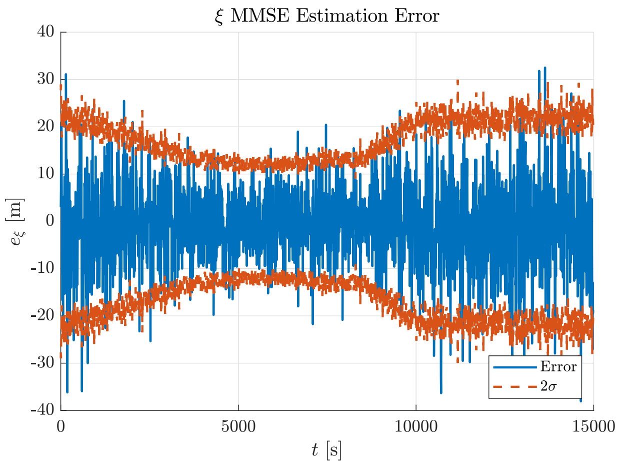
(a) East estimation error



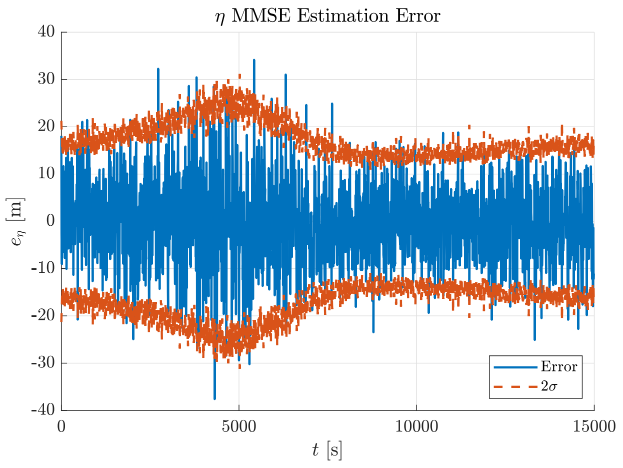
(b) North estimation error

Figure 11: NLS estimation errors for trajectory with near-far and no clock bias

### 5.1.2 PF

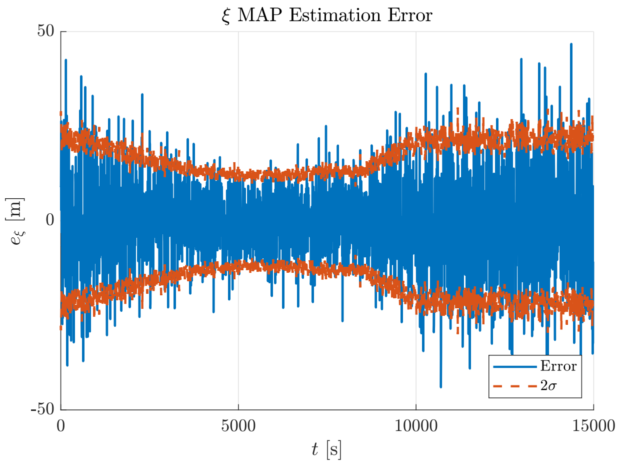


(a) East estimation error

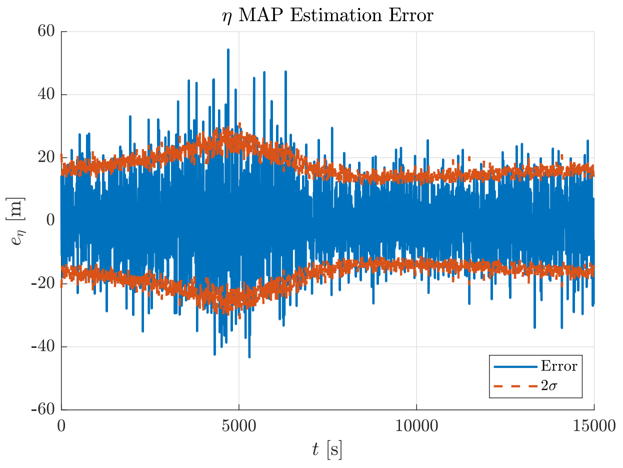


(b) North estimation error

Figure 12: PF MMSE estimation errors for trajectory with no near-far and no clock bias

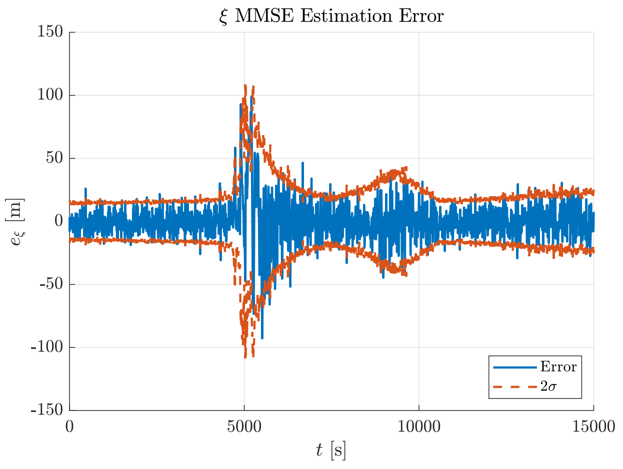


(a) East estimation error

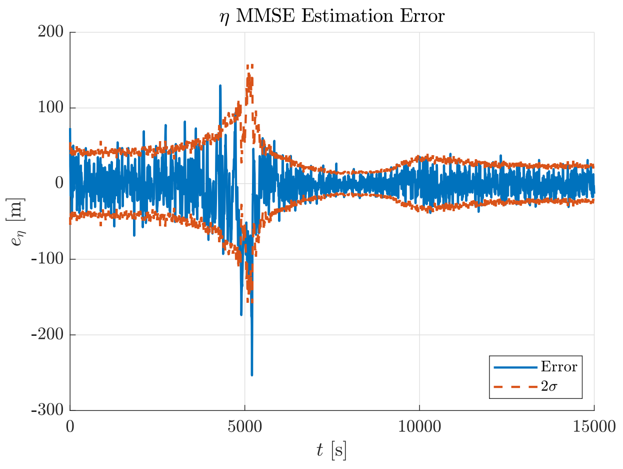


(b) North estimation error

Figure 13: PF MAP estimation errors for trajectory with no near-far and no clock bias

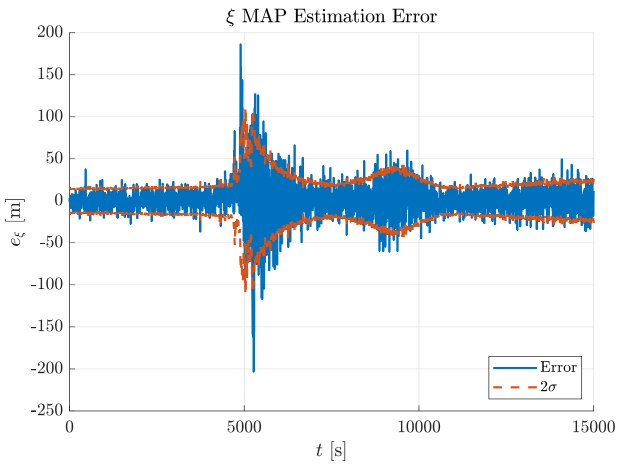


(a) East estimation error

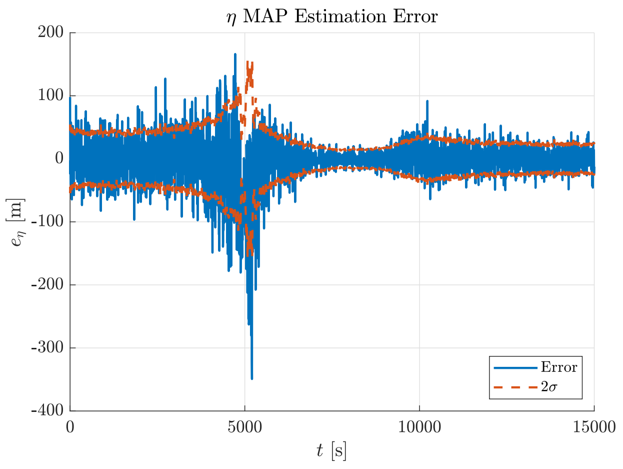


(b) North estimation error

Figure 14: PF MMSE estimation errors for trajectory with near-far and no clock bias



(a) East estimation error

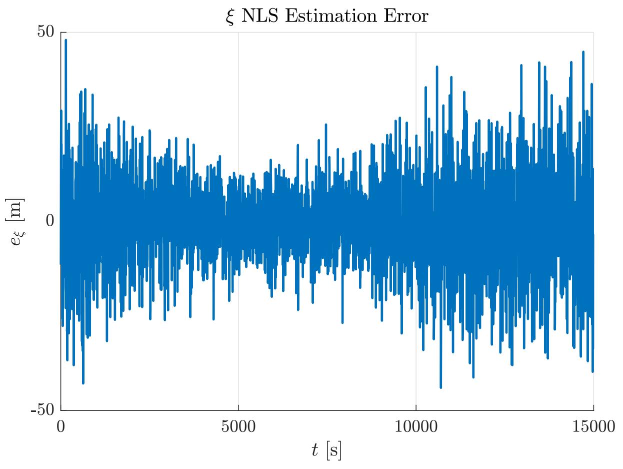


(b) North estimation error

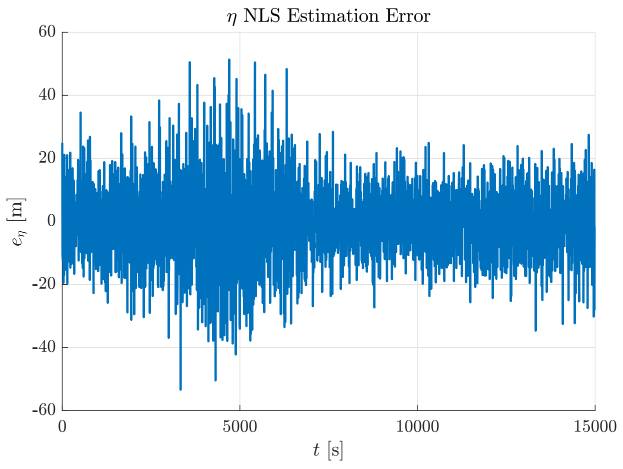
Figure 15: PF MAP estimation errors for trajectory with near-far and no clock bias

## 5.2 Level 2

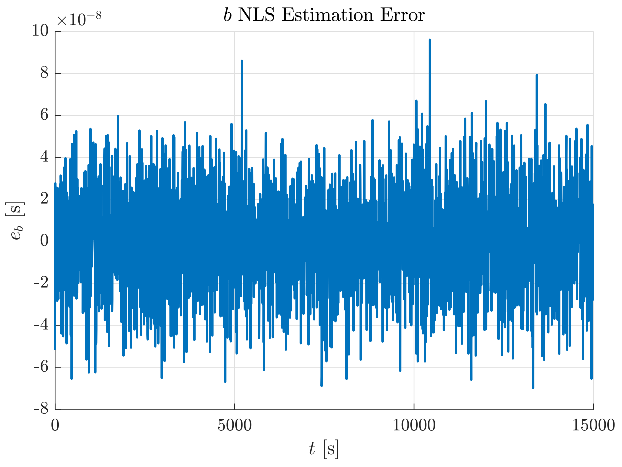
### 5.2.1 NLS



(a) East estimation errors

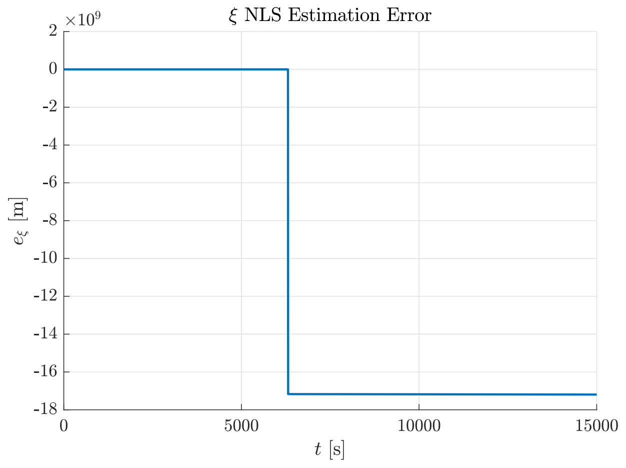


(b) North estimation errors

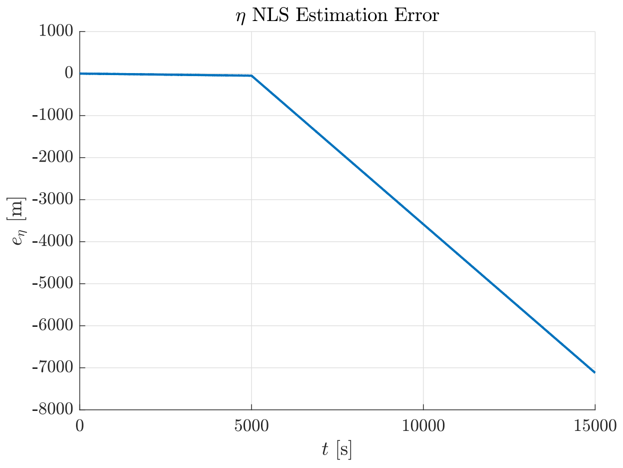


(c) Clock bias estimation errors

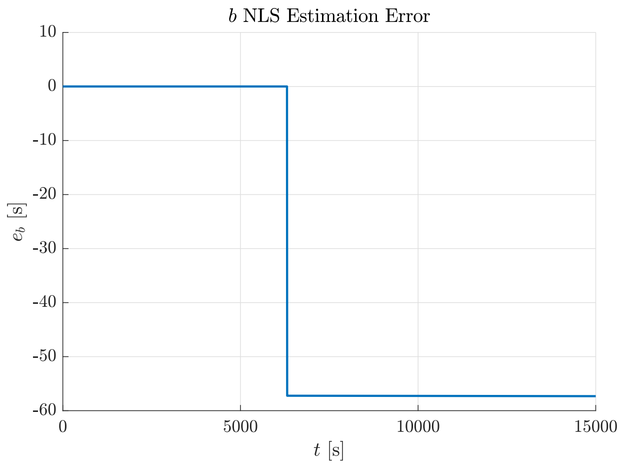
Figure 16: NLS estimation errors for trajectory with no near-far and clock bias



(a) East estimation errors



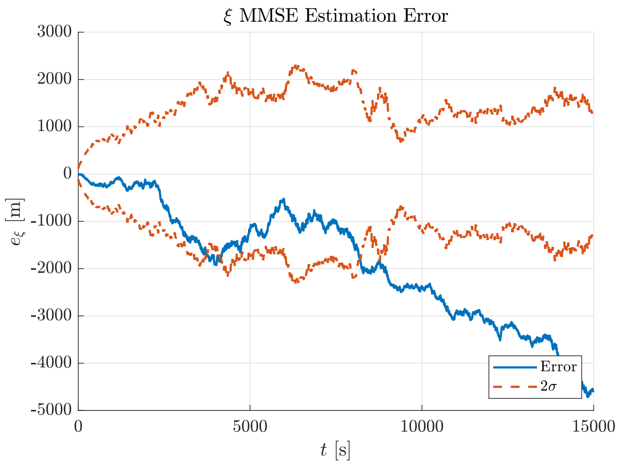
(b) North estimation errors



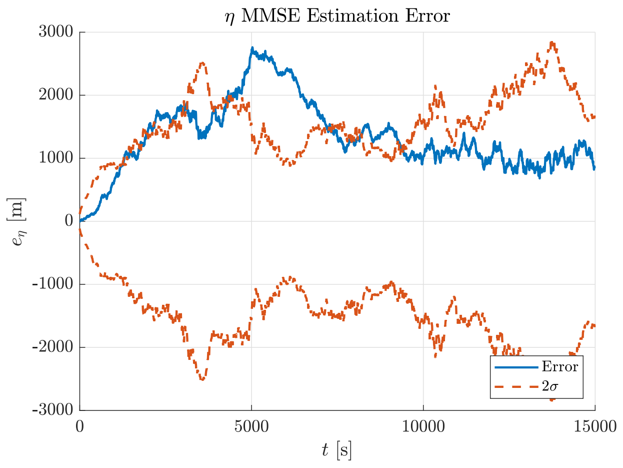
(c) Clock bias estimation errors

Figure 17: NLS estimation errors for trajectory with near-far and clock bias

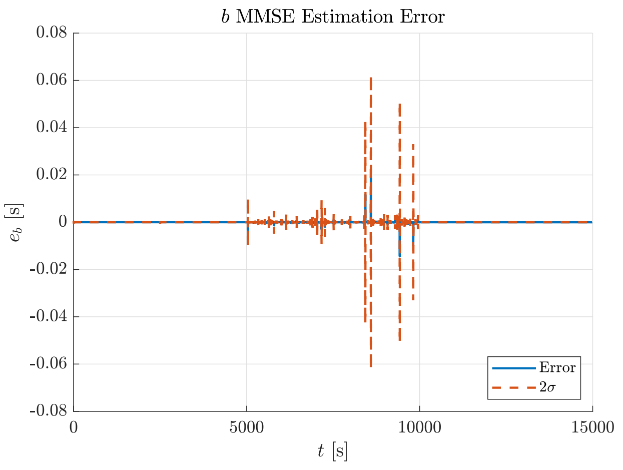
### 5.2.2 PF



(a) East estimation errors

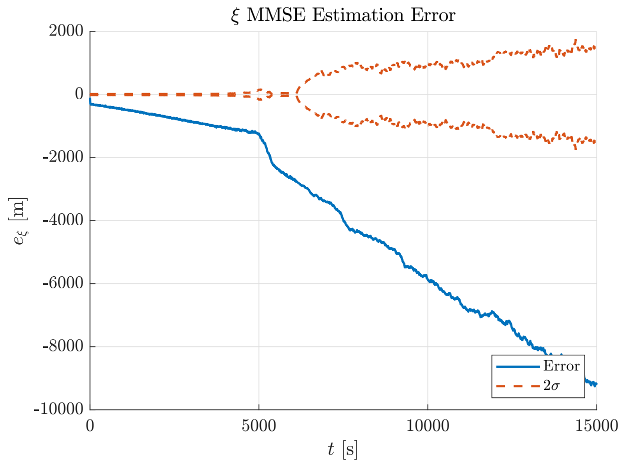


(b) North estimation errors

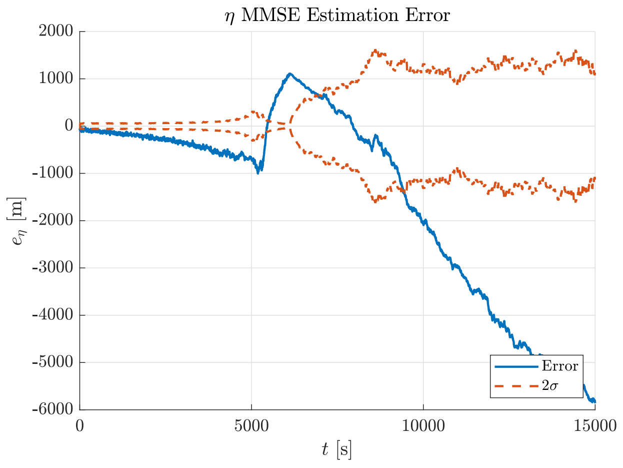


(c) Clock bias estimation errors

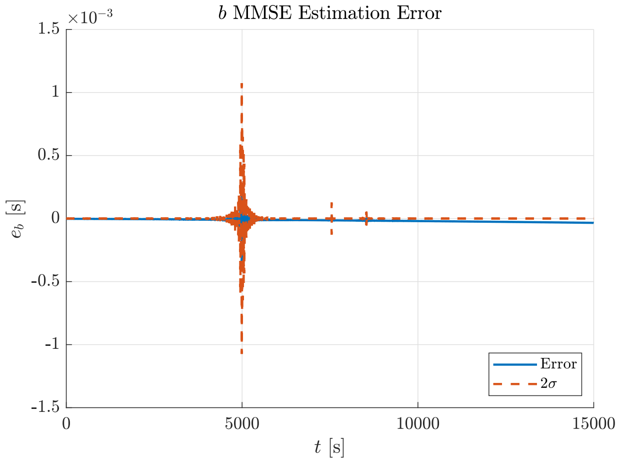
Figure 18: MMSE estimation errors for trajectory with no near-far and clock bias



(a) East estimation errors



(b) North estimation errors



(c) Clock bias estimation errors

Figure 19: MMSE estimation errors for trajectory with near-far and clock bias

# 6 Discussion

## 6.1 Level 1

We start with figure 10, which represents the baseline trilateration algorithm for this project. The swells in the $2\sigma$ bounds are representative of mainly geometry changes (in dilution of precision), but also smaller near-far effects as the rover gets closer to some pseudolites than others. The estimates appear unbiased.

For the same algorithm running on a trajectory with serious near-far interference, shown in figure 11, we see that right around the 5000 second mark when the serious near-far interference kicks in, the NLS algorithm diverges. While the NLS cost function should account for the differences in SNR from the three pseudolites, it is apparent that the noisy power estimates used to scale the terms in the NLS cost function are not enough to offset the errors brought about by those extremely noisy measurements.

The particle filter results for the no near-far trajectory without clock bias (figures 12 and 13) seem to have slightly lower variances than the NLS results. They exhibit the same behavior of swells from the data, further reinforcing the effect that pseudolite geometry has on the position estimates.

Unlike NLS, the particle filter results for the near-far trajectory (figures 14 and 15) show the filter getting kicked at the 5000 second mark but able to recover and continue an unbiased filtering.

## 6.2 Level 2

The NLS estimates of the clock drifting no near-far trajectory (figure 16) appear identical to the estimates without the clock drift, and this is confirmed by the magnitude of the clock bias errors (on the order of $10^{-}8$ seconds, which translates to 30 meters).

NLS running with the clock drift seemed to do well until the near-far kick at 5000 seconds in figure 17. This is similar to what happened in the NLS near-far case without the clock drift.

Only the MMSE estimates are shown in the particle filter case (figures 18 and 19), because the MAP estimates are the same (sad) story. Even though the filter is initialized perfectly, the filter simply isn't able to capture the clock drifting properly with the dynamic walk method. This can partially be explained by the effective sample size of the filter:

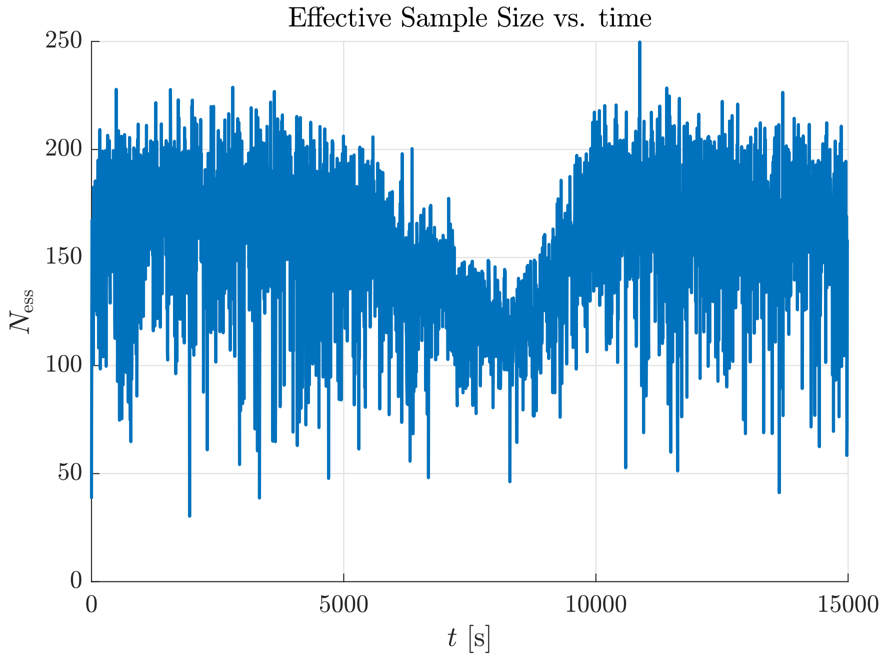


Figure 20: Effective Sample Size for PF without near far and clock drift

The filter was resampled with 1000 particles each iteration, but due to the nature of the random walk dynamics model, most of them ended up going to useless places. This is especially damaging when a paramater such as the clock bias is extremely sensitive and needs a very high resolution; the number of particles we would need to estimate the clock bias accurately would be inefficient compared to another method such as least squares.

It's interesting to note how there seems to be a smooth deviation away from the truth in the estimates in figure 19 before the near-far kick, after which the filter diverges randomly. This is reflected in the size of the $2\sigma$ bounds, which reflect some degree of confidence in the estimate before the near-far kick, and completely lose confidence after 5000 seconds. This behavior is not observed in figure 18, for which there was no near far problem and the filter diverged immediately. This inconsistency near the beginning of the filtering process suggests that the initialization of the particle filters is extremely important when trying to estimate the clock bias in PNT applications.

## 6.3 Level 3

The particle filter seems to handle the near-far kick quite well when there's no clock drift to worry about, and the NLS estimator seems to be able to compute the clock drift quite accurately when there's no near-far problem to worry about. It was this observation that lead to attempting a hybrid method by which the NLS would provide the time "measurement" to feed into the particle filter which could handle the regions of the trajectory with extreme near-far problems. The implementation had the same effect as the pure NLS estimator, where the time suddenly diverged and the particle filter broke because the time was too large.

Attempting to do positioning in a near-far heavy environment is extremely hard. It's probably easier to simply reinitialize the filter whenever near-far situations arise rather than trying to filter your way through it. The particle filter showed some promise in the simple case without any clock bias, but in the real world of clock bias there needs to be some sort of time correction that happens within the near-far environment before the particle filter can be viable.

## 7 Conclusion

The particle filter for positioning in near-far situations is quite robust if it doesn't have to deal with the time bias. Even outside near-far environments, due to the nature of the random walk dynamics update, the effective sample size after the measurement update is such that there isn't enough information to estimate the clock bias accurately and so different methods must be used. Increasing the number of particles is an option, but the efficiency of a least-squares approach for an accurate solution would be ideal. A potential solution would be to discipline the particle filter with least-squares until some near-far flag is triggered, and then to hold the clock bias in the particle filter fixed until the receiver is no longer in the near-far environment.

Within the near-far environment, a viable option is to use an Angle of Arrival positioning method. This way, if the pseudolite had a phased array antenna, it could receive a message from the rover and determine what angle the message came from, and combine it with range information from that one pseudolite to compute a position.

## References

[1] Fernández-Prades, Carles. "Tracking." GNSS SDR, Centre Tecnològic De Telecomunicacions De Catalunya, 16 Mar. 2020, gnss-sdr.org/docs/sp-blocks/tracking/.

[2] "Allan Variance." Wikipedia, Wikimedia Foundation, 26 Jan. 2021, en.wikipedia.org/wiki/Allan_variance#: :text=Allan%20variance%20is%20used%20as, typically%20expressed%20as%20phase%20noise.

[3] Wolff, Christian. "Radar Basics." Radartutorial, Christian Wolff, www.radartutorial.eu/01.basics/Radars%20Accuracy.en.html.

[4] Microsemi. "Chip-Scale Atomic Clock (CSAC) Performance During Rapid Temperature Change." Microsemi.com, Microchip Technology, Inc., www.microsemi.com/documentportal/doc_download/1243735-csac-performance-during-rapidtemp-change.

[5] Zhou, Yu. "An Efficient Least-Squares Trilateration Algorithm for Mobile Robot Localization." 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, doi:10.1109/iros.2009.5354370.

## Appendix

### 7.1 MATLAB® Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ian Thomas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;
clear;
clc;

set(groot, 'defaulttextinterpreter', 'latex');
set(groot, 'defaultAxesTickLabelInterpreter', 'latex');
set(groot, 'defaultLegendInterpreter', 'latex');

set(0, 'DefaultAxesLooseInset', [0,0,0,0])
set(0,'defaultAxesFontSize',12)

colors = get(gca, 'colororder');
close all;

rng(101)

%% Declaration of Parameters
```

```
% Pseudolite Locations
pseudolites = [0, 0; 5000, 0; 2500, 2500*sqrt(3)];
[npseudolites, dims] =  size(pseudolites);

% cpoly = [0];
cpoly = [3.5556e-14, 5.3333e-10, 1e-6];
c = 3e8;

n = 3;

sig_r0 = 10;
dfac = 0.05;

sig_particles = 30; % m
sig_time = 1e-7;

velsmoothing = false;
sig_theta = pi/6;
sig_vel = 20;

nls_cost = @(x, p, r, s) sum((1./s.^2).*...
    (sqrt((x(1)-p(:, 1)).^2+(x(2)-p(:, 2)).^2)-r+c*x(3)).^2);


%% Define Geometry

load traj1

figure(2)
hold on;
grid on
axis equal
xlabel('$\xi$ [m]')
ylabel('$\eta$ [m]')
title('Trajectory of Rover')
plot(xitrue, etatrue , 'LineWidth', 1.5);
scatter(pseudolites(1, 1), pseudolites(1, 2), 50, colors(1, :), 'filled')
scatter(pseudolites(2, 1), pseudolites(2, 2), 50, colors(2, :), 'filled')
scatter(pseudolites(3, 1), pseudolites(3, 2), 50, colors(3, :), 'filled')
legend('Path', 'Pseudolite 1', 'Pseudolite 2', 'Pseudolite 3',...
    'location', 'nw')

figure(20)
hold on
```

```
grid on
axis equal
xlabel('$t$ [s]')
ylabel('$x$ [m]')
title('State vs $t$')
plot(t, xtrue , 'LineWidth', 1.5);
legend('$\xi$', '$\eta$', 'location', 'se')


%% Generate noisy measurements independent of distance from pseudolites

[bias, r, rel_sig] = ...
    get_noisy_measurements(t, xtrue, pseudolites, cpoly, sig_r0, dfac);

figure(3)
hold on;
grid on;
xlabel('$t$ [s]')
ylabel('$\rho$ [m]')
title('Simulated Range Measurements')
plot(t, r(1, :), 'LineWidth', 1.5)
plot(t, r(2, :), 'LineWidth', 1.5)
plot(t, r(3, :), 'LineWidth', 1.5)
legend('$\rho_1$', '$\rho_2$', '$\rho_3$')


%% Particle Filter

getrpdf = @(particles, pseudolite, r, s) rpdf(particles(:, 1),...
    particles(:, 2), pseudolite(1), pseudolite(2), r, s);
getrpdfs = @(particles, pseudolites, rs,  ss) ...
    getrpdf(particles, pseudolites(1, :), rs(1, :), ss(1)).*...
    getrpdf(particles, pseudolites(2, :), rs(2, :), ss(2)).*...
    getrpdf(particles, pseudolites(3, :), rs(3, :), ss(3));

nparticles = 1000;

% Prior is a uniform 200 x 200 m distribution centered around xtrue(1)
sample_size = [nparticles, 1];

xi0 = xitrue(1) + 200*rand(sample_size) - 100;
eta0 = etatrue(1) + 200*rand(sample_size) - 100;

ss = rel_sig(:, 1) * sig_r0;
dists = rel_sig(:, 1);
```

```
b1 = (r(2, 1)*dists(1)-r(1, 1)*dists(2))/(dists(2)-dists(1))/c;
b2 = (r(3, 1)*dists(1)-r(1, 1)*dists(3))/(dists(3)-dists(1))/c;
b3 = (r(3, 1)*dists(2)-r(2, 1)*dists(3))/(dists(3)-dists(2))/c;

b0 = mean([b1, b2, b3]) + ...
    mean(rel_sig(:, 1))*std([b1, b2, b3])*randn(sample_size);

particles = [xi0, eta0, b0];

xhat_NLS = zeros(n, T);
xhat_mmse = zeros(n, T);
xhat_map = zeros(n, T);
Phat = zeros(n, n, T);
% Measurement update and resample step for t = 0

% w = getrpdfs(particles, pseudolites, r(:, 1)+b0'*c, ss);
w = getrpdfs(particles, pseudolites, r(:, 1), ss);
if sum(w) ~= 0
    w = w / sum(w);
else
    w = ones(sample_size)/nparticles;
end

mu = sum(w.*particles, 1)';
P = wcov(particles, w);

% xhat_NLS(:, 1) =
xhat_mmse(:, 1) = mu;
Phat(:, :, 1) = P;

[~, max_ind] = max(w);
xhat_map(:, 1) = particles(max_ind, :);

xhat_NLS(:, 1) = fminsearch(@(x) nls_cost(x, pseudolites, r(:, 1), ss), [1250;0;0]);

% Resample particle set
particles = sample_pmf(particles, w, nparticles);

%%

for i = 2:T

    disp(i)
    if i > 3 && velsmoothing
```

```
        % Velocity smoothing dynamics update
        vel = xhat_mmse(1:2, i-1) - xhat_mmse(1:2, i-2);
        theta_prev = atan2(vel(2), vel(1));
        thetas = theta_prev + sig_theta*randn(sample_size);
        mags = norm(vel) + sig_vel*randn(sample_size);

        particles(:, 1) = particles(:, 1) + mags.*cos(thetas);
        particles(:, 2) = particles(:, 2) + mags.*sin(thetas);
        particles(:, 3) = particles(:, 3) + sig_time*randn(nparticles, 1);
    else
        % random walk dynamics update
        particles(:, 1:2) = particles(:, 1:2) +...
            sig_particles*randn(nparticles, 2);
        particles(:, 3) = particles(:, 3) + sig_time*randn(nparticles, 1);
    end

    ss = (rel_sig(:, i)) * sig_r0;
    dists = rel_sig(:, i);

    % Estimate clock bias roughly from relative power
    b1 = (r(2, i)*dists(1)-r(1, i)*dists(2))/(dists(2)-dists(1))/c;
    b2 = (r(3, i)*dists(1)-r(1, i)*dists(3))/(dists(3)-dists(1))/c;
    b3 = (r(3, i)*dists(2)-r(2, i)*dists(3))/(dists(3)-dists(2))/c;

    mb = mean([b1 b2 b3]);
    stdb = std([b1 b2 b3]);
    particles(:, 3) = mb + stdb*randn(sample_size);
    % Measurement Update
%     w = getrpdfs(particles, pseudolites, r(:, i)+particles(:, 3)'*c, ss);
    w = getrpdfs(particles, pseudolites, r(:, i), ss);
    if sum(w) ~= 0
        w = w / sum(w);
    else
        w = ones(sample_size)/nparticles;
    end
%     w = w .* normpdf(particles(:, 3), mb,...
%         mean(rel_sig(:, i))*stdb);
%     if sum(w) ~= 0
%         w = w / sum(w);
%     else
%         w = ones(sample_size)/nparticles;
%     end

    [~, max_ind] = max(w);
    xhat_map(:, i) = particles(max_ind, :);
    mu = sum(w.*particles, 1)';
    P = wcov(particles, w);

    xhat_mmse(:, i) = mu;
    Phat(:, :, i) = P;

    xhat_NLS(:, i) = fminsearch(@(x)...
        nls_cost(x, pseudolites, r(:, i), ss), xhat_NLS(:, i-1));

    % Resampling (draw from gaussian covariance)
    particles = sample_pmf(particles, w, nparticles);

end

%%

figure(4)
hold on
grid on
xlabel('$t$ [s]')
ylabel('$e_\xi$ [m]')
title('$\xi$ MMSE Estimation Error')
plot(t, xhat_mmse(1, :)-xtrue(1, :), 'LineWidth', 1.5);
plot(t, 2*sqrt(reshape(Phat(1, 1, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :))
plot(t, -2*sqrt(reshape(Phat(1, 1, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :), 'HandleVisibility', 'off')
legend('Error', '2$\sigma$', 'location', 'se')

figure(5)
hold on
grid on
xlabel('$t$ [s]')
ylabel('$e_\eta$ [m]')
title('$\eta$ MMSE Estimation Error')
plot(t, xhat_mmse(2, :)-xtrue(2, :), 'LineWidth', 1.5);
plot(t, 2*sqrt(reshape(Phat(2, 2, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :))
plot(t, -2*sqrt(reshape(Phat(2, 2, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :), 'HandleVisibility', 'off')
legend('Error', '2$\sigma$', 'location', 'se')

figure(6)
hold on
```

```
grid on
xlabel('$t$ [s]')
ylabel('$e_b$ [s]')
title('$b$ MMSE Estimation Error')
plot(t, xhat_mmse(3, :)-bias, 'LineWidth', 1.5);
plot(t, 2*sqrt(reshape(Phat(3, 3, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :))
plot(t, -2*sqrt(reshape(Phat(3, 3, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :), 'HandleVisibility', 'off')
legend('Error', '2$\sigma$', 'location', 'se')


figure(7)
hold on
grid on
xlabel('$t$ [s]')
ylabel('$e_\xi$ [m]')
title('$\xi$ MAP Estimation Error')
plot(t, xhat_map(1, :)-xtrue(1, :), 'LineWidth', 1.5);
plot(t, 2*sqrt(reshape(Phat(1, 1, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :))
plot(t, -2*sqrt(reshape(Phat(1, 1, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :), 'HandleVisibility', 'off')
legend('Error', '2$\sigma$', 'location', 'se')

figure(8)
hold on
grid on
xlabel('$t$ [s]')
ylabel('$e_\eta$ [m]')
title('$\eta$ MAP Estimation Error')
plot(t, xhat_map(2, :)-xtrue(2, :), 'LineWidth', 1.5);
plot(t, 2*sqrt(reshape(Phat(2, 2, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :))
plot(t, -2*sqrt(reshape(Phat(2, 2, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :), 'HandleVisibility', 'off')
legend('Error', '2$\sigma$', 'location', 'se')

figure(9)
hold on
grid on
xlabel('$t$ [s]')
ylabel('$e_b$ [s]')
title('$b$ MAP Estimation Error')
plot(t, xhat_map(3, :)-bias, 'LineWidth', 1.5);
plot(t, 2*sqrt(reshape(Phat(3, 3, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :))
plot(t, -2*sqrt(reshape(Phat(3, 3, :), 1, [])), '--', 'LineWidth', 1.5,...
    'Color', colors(2, :), 'HandleVisibility', 'off')
legend('Error', '2$\sigma$', 'location', 'se')

figure(10)
hold on
grid on
xlabel('$t$ [s]')
ylabel('$e_\xi$ [m]')
title('$\xi$ NLS Estimation Error')
plot(t, xhat_NLS(1, :)-xtrue(1, :), 'LineWidth', 1.5);
% plot(t, 2*sqrt(reshape(Phat(1, 1, :), 1, [])), '--', 'LineWidth', 1.5,...
%     'Color', colors(2, :))
% plot(t, -2*sqrt(reshape(Phat(1, 1, :), 1, [])), '--', 'LineWidth', 1.5,...
%     'Color', colors(2, :), 'HandleVisibility', 'off')
% legend('Error', '2$\sigma$', 'location', 'se')

figure(11)
hold on
grid on
xlabel('$t$ [s]')
ylabel('$e_\eta$ [m]')
title('$\eta$ NLS Estimation Error')
plot(t, xhat_NLS(2, :)-xtrue(2, :), 'LineWidth', 1.5);
% plot(t, 2*sqrt(reshape(Phat(2, 2, :), 1, [])), '--', 'LineWidth', 1.5,...
%     'Color', colors(2, :))
% plot(t, -2*sqrt(reshape(Phat(2, 2, :), 1, [])), '--', 'LineWidth', 1.5,...
%     'Color', colors(2, :), 'HandleVisibility', 'off')
% legend('Error', '2$\sigma$', 'location', 'se')

figure(12)
hold on
grid on
xlabel('$t$ [s]')
ylabel('$e_b$ [s]')
title('$b$ NLS Estimation Error')
plot(t, xhat_NLS(3, :)-bias, 'LineWidth', 1.5);
% plot(t, 2*sqrt(reshape(Phat(3, 3, :), 1, [])), '--', 'LineWidth', 1.5,...
%     'Color', colors(2, :))
% plot(t, -2*sqrt(reshape(Phat(3, 3, :), 1, [])), '--', 'LineWidth', 1.5,...
%     'Color', colors(2, :), 'HandleVisibility', 'off')
% legend('Error', '2$\sigma$', 'location', 'se')
```

## 7.2 Helper Functions

```
function P = wcov(x, w)
    mu = sum(w.*x);
    C = x - mu;
    P = zeros(3, 3);
    for i = 1:length(w)
        P = P + w(i) * C(i, :)' * C(i, :);
    end
end

function [bias, r, rel_sig] = get_noisy_measurements(...
    t, trajectory, pseudolites, cpoly, sig_r0, dfac)
%% Function get_noisy_measurements
%
% Purpose:
% from a trajectory and pseudolite locations, generate noisy measurements
%
% Inputs:
% t - time vector (row vector)
% trajectory - traj (n rows, T columns)
% pseudolites - list of positions of pseudolites
% cpoly - clock polynomial coefficients ()
% sig_r0 - standard range deviation of highest power signal (20 m)
% dfac - coefficient of noise for rel_power (0.05)
%
%

    [npseudolites, dims] =  size(pseudolites);
    T = length(t);
    r = reshape(sqrt(sum((trajectory - ...
        reshape(pseudolites', [dims, 1, npseudolites])).^2, 1)),...
        [T, npseudolites])';

    rel_sig = (r ./ min(r)); % figure out the SNR ratios
    r = r + sig_r0*rel_sig.*randn(size(r)); % apply adjusted noise
%     figure(10)
%     plot(t', rel_power')
    rel_sig = rel_sig + dfac*rel_sig.*randn(size(rel_sig));

    bias = polyval(cpoly, t);
    r = r + bias*3e8;

end

function v = rpdf(X, Y, x, y, r, s)
    v = exp(-0.5*((sqrt((X-x).^2+(Y-y).^2)-r)./s).^2);
%     if sum(v) < numel(v)*1e-10
%         v = ones(size(v));
%     end
end

function s = sample_pmf(v, w, n)
% Draw from pmf
    edges = cumsum([0; w(:)]);
    edges = edges/edges(end);
%     if isnan(sum(edges))
%         hai
%     end
    [~, ~, i] = histcounts(rand(1, n), edges);
    s = v(i, :);
end
```