ASE 390P-7 Problem Set 4

Posting Date: October 8, 2024

You need not hand in anything. Instead, be prepared to answer any of these problems—or similar problems—on an upcoming take-home exam. You may discuss your solutions with classmates up until the time that the exam becomes available, but do not swap work (including code).

Readings

All background reading material is found on Canvas. You'll find valuable material about quadrature sampling in [1]. A simple introduction to bandpass sampling and the direct-to-baseband architecture is presented in [2]. The notes by Proakis titled "Sampling and reconstruction of signals" are a valuable reference. Drawing from Vaughan's original paper [3], these notes cover the bandpass sampling theorem, including the "spikey hair plot." Chris Hegarty offers a valuable model for GNSS quantization effects in [4]. My chapter titled "Interference" in the Springer Handbook on GNSS (found on Canvas as `interference_humphreys.pdf`) also has a useful discussion on quantization effects [5].

Problems

1. In this problem you will explore the difference between low- and high-side mixing on carrier phase tracking. Consider the process of converting a bandpass signal $2a(t)\cos 2\pi f_c$, centered at f_c, to an intermediate frequency via mixing with a signal $\cos 2\pi f_l t$:

   $$x_u(t) = 2a(t)\cos 2\pi f_c t \cos 2\pi f_l t = a(t)\cos\left[2\pi(f_c - f_l)t\right] + \text{HFT}$$

   Filtering at the intermediate frequency removes the high-frequency term (HFT), leaving only

   $$x(t) = a(t)\cos\left[2\pi(f_c - f_l)t\right]$$

   By definition, the frequency of a physical signal (the number of cycles per second) is a positive quantity. Therefore, if $f_c < f_l$, we express $x(t)$ as

   $$x(t) = a(t)\cos\left[2\pi(-f_c + f_l)t\right]$$

   by invoking the identity $\cos(y) = \cos(-y)$.

   The original bandpass signal's center frequency f_c can be decomposed as

   $$f_c = f_{c,\text{nom}} + f_{c,D}$$

   where $f_{c,\text{nom}}$ is the nominal value (e.g., 1575.42 MHz for GPS L1), and $f_{c,D}$ is a Doppler value due to satellite-to-receiver relative motion and satellite clock frequency offset. Likewise,

   $$f_l = f_{l,\text{nom}} + f_{l,D}$$

where $f_{l,\text{nom}}$ is the nominal local oscillator value and $f_{l,D}$ is a Doppler value due to receiver clock frequency offset. When $f_{c,\text{nom}} < f_{l,\text{nom}}$, the mixing operation is referred to as *high-side mixing* (HS mixing), whereas when $f_{l,\text{nom}} < f_{c,\text{nom}}$, it is *low-side mixing* (LS mixing). The intermediate frequency is defined as

$$f_{\text{IF}} = |f_{c,\text{nom}} - f_{l,\text{nom}}|$$

Note that f_{IF} is always chosen to be large enough that $f_c < f_l$ for HS mixing and $f_l < f_c$ for LS mixing over the range of expected $f_{c,D}$ and $f_{l,D}$. The downmixed signal $x(t)$ is modeled as

$$x(t) = a(t)\cos\left[2\pi f_{\text{IF}} t + \theta(t)\right]$$

Explain the different effect of HS and LS mixing on $\theta(t)$. What implications might this different effect have for signal acquisition and tracking?

2. Consider the block diagram of the venerable GP2015 front end in Figure 2 of the document `gp2015-datasheet-sept2007.pdf` on Canvas. This front end has been used in many software GPS receiver applications. Trace the GPS L1 C/A signal through this front end by diagramming in the frequency domain the effect of each mixing and filtering stage. To appreciate the effects of high- and low-side mixing, model the GPS L1 C/A signal as asymmetric about f_{L1}, even though it's actually symmetric. Identify each analog mixing stage as either high-side or low-side mixing. You may wish to use several diagrams to make the process clear. The narrowest filter in the chain is the 1.9-MHz SAW filter applied at 35.42 MHz. This is the filter that selects only the GPS L1 C/A signal. The final analog mixing stage centers the GPS L1 C/A signal at $f_{\text{IF}_3} = 4.309$ MHz. After this stage, the signal is digitized by two-bit quantization at a sampling rate of $40/7 \approx 5.714$ MHz (locked to the same 10-MHz external reference driving the rest of the downconversion). Draw another diagram that shows the frequency spectrum of the digitized signal. Does the digitization introduce a phase reversal (the digital equivalent of high-side mixing)? Identify the final intermediate frequency f_{IF_4} that applies for the discrete-time signal. Give this value in Hz to six decimal places.

3. In lecture we considered an analog signal $x_a(t)$ sampled by impulses:

$$x_\delta(t) = \sum_{n=-\infty}^{\infty} x_a(nT)\delta(t - nT)$$

We showed that the Fourier transform of the impulse-sampled signal $x_\delta(t)$ is related to $X_a(f)$, the Fourier transform of $x_a(t)$, by

$$X_\delta(f) = \frac{1}{T}\sum_{n=-\infty}^{\infty} X_a\left(f - \frac{n}{T}\right)$$

We can derive a similar relationship between $X_a(f)$ and the Fourier transform of the discrete-time signal

$$x(n) = x_a(nT), \quad -\infty < n < \infty$$

To make this easier, we'll define the frequency variable $\tilde{f} = fT = f/f_s$. This variable, which has units of cycles per sample and is often called the *normalized frequency*, is used as the frequency variable for discrete-time signals. For example, a discrete-time sinusoid can be represented as

$x(n) = \cos 2\pi \tilde{f} n$. For discrete-time signals, only frequencies in the range $-\frac{1}{2} \leq \tilde{f} \leq \frac{1}{2}$ are unique; all frequencies $|\tilde{f}| > \frac{1}{2}$ are aliases.

The Fourier transform of a discrete-time signal $x(n)$ is defined by

$$X(\tilde{f}) = \sum_{n=-\infty}^{\infty} x(n) e^{-j2\pi \tilde{f} n}$$

and the inverse transform is defined by

$$x(n) = \int_{-1/2}^{1/2} X(\tilde{f}) e^{j2\pi \tilde{f} n} d\tilde{f}$$

Derive the relationship between $X(\tilde{f})$ and $X_a(f)$.

Hint: This is a standard relationship whose derivation can be found in many texts that treat digital signal processing. You're free to use such a text as a guide or you may perform the derivation yourself following these steps:

(a) Express $x(n) = x_a(nT)$ in terms of $X_a(f)$.

(b) Equate this expression with the inverse transform definition given above.

(c) Express the integral that goes from $-\infty$ to ∞ as an infinite sum of integrals of width f_s.

(d) Make a change of variable $f = \tilde{f} \cdot f_s$ in this infinite sum of integrals expression to eliminate f in favor of $\tilde{f}$.

(e) Make some deductions to arrive at the desired relationship between $X(\tilde{f})$ and $X_a(f = \tilde{f} \cdot f_s)$.

4. In lecture we showed that uniform (first-order) sampling and digital down-conversion can be employed to obtain in-phase $I(m)$ and quadrature $Q(m)$ samples of the baseband complex representation of a bandpass signal $x_a(t)$. If quantization effects are ignored, then with sufficient processing $I(m)$ can be made to approach $x_c(mT_l)$ and $Q(m)$ can be made to approach $x_s(mT_l)$ with arbitrary accuracy, where $x_c(t)$ and $x_s(t)$ are the continuous-time in-phase and quadrature components of the baseband representation $x_l(t) = x_c(t) + jx_s(t)$ and T_l is the complex sampling interval.

   Knowing how to go back and forth between a discrete-time baseband representation of a signal in terms of $I(m)$ and $Q(m)$ and a discrete time representation of the signal at the original carrier frequency—or at some intermediate frequency—is useful for simulation and analysis of baseband signals.

   Write two Matlab functions for converting between these signal representations. Your function for converting from $I(m)$ and $Q(m)$ samples to a bandpass representation at some arbitrary f_{IF} should adhere to the following interface:

```
function [xVec] = iq2if(IVec,QVec,Tl,fIF)
% IQ2IF : Convert baseband I and Q samples to intermediate frequency samples.
%
% Let xl(m*Tl) = I(m*Tl) + j*Q(m*Tl) be a discrete-time baseband
% representation of a bandpass signal.  This function converts xl(n) to a
% discrete-time bandpass signal x(n) = I(n*T)*cos(2*pi*fIF*n*T) -
```

```
% Q(n*T)*sin(2*pi*fIF*n*T) centered at the user-specified intermediate
% frequency fIF, where T = Tl/2.
%
%
% INPUTS
%
% IVec -------- N-by-1 vector of in-phase baseband samples.
%
% QVec -------- N-by-1 vector of quadrature baseband samples.
%
% Tl ---------- Sampling interval of baseband samples (complex sampling
%               interval), in seconds.
%
% fIF --------- Intermediate frequency to which the baseband samples will
%               be up-converted, in Hz.
%
%
% OUTPUTS
%
% xVec -------- 2*N-by-1 vector of intermediate frequency samples with
%               sampling interval T = Tl/2.
%
%
%+------------------------------------------------------------------------+
% References:
%
%
%+========================================================================+
```

Your function for converting from a bandpass representation at some arbitrary f_{IF} to $I(m)$ and $Q(m)$ samples should be called `if2iq` and should adhere to the following interface:

```
function [IVec,QVec] = if2iq(xVec,T,fIF)
% IF2IQ : Convert intermediate frequency samples to baseband I and Q samples.
%
% Let x(n) = I(n*T)*cos(2*pi*fIF*n*T) - Q(n*T)*sin(2*pi*fIF*n*T) be a
% discrete-time bandpass signal centered at the user-specified intermediate
% frequency fIF, where T is the bandpass sampling interval.  Then this
% function converts the bandpass samples to quadrature samples from a complex
% discrete-time baseband representation of the form xl(m*Tl) = I(m*Tl) +
% j*Q(m*Tl), where Tl = 2*T.
%
%
% INPUTS
%
% xVec -------- N-by-1 vector of intermediate frequency samples with
%               sampling interval T.
%
% T ----------- Sampling interval of intermediate frequency samples, in
%               seconds.
%
```

```
% fIF --------- Intermediate frequency of the bandpass signal, in Hz.
%
%
% OUTPUTS
%
% IVec -------- N/2-by-1 vector of in-phase baseband samples.
%
% QVec -------- N/2-by-1 vector of quadrature baseband samples.
%
%
%+------------------------------------------------------------------------+
% References:
%
%
%+========================================================================+
```

For the function `iq2if`, you will need to resample the I and Q data at twice their original sampling rate. You could do this by generating a continuous signal from the digital samples using the ideal reconstruction formula, and then resampling this continuous signal at the new sampling rate (assuming the new sampling rate satisfies the Nyquist criterion). But the ideal reconstruction formula requires you to work with everlasting sinc functions, which isn't convenient. And for an arbitrary new sampling rate, the conversion function is somewhat complicated. But if we only wish to resample at an integer multiple R of the original sampling rate, and if we're willing to accept a slight distortion of the resampled signal, then there is an efficient way to resample: we insert R zeros between each pair of samples in the original sequence and low-pass filter the result. You'll find that the Matlab function `interp` from the signal processing toolbox handles all of this for you. For `iq2if`, $R = 2$.

For `if2iq` you could use one of the methods described in the papers [1] and [2] found on Canvas, but you can just as well employ the straightforward discrete-time implementation of the continuous-time "quadrature approach" to bandpass sampling discussed in lecture. You'll find the Matlab function `decimate` from the signal processing toolbox helpful for implementing the low-pass filtering and decimation operations.

The functions `if2iq` and `iq2if` should act as inverse operations; that is, aside from some high-frequency components lost in filtering, by calling `iq2if` and then `if2iq` you should recover your original data. You'll find that to satisfy this requirement you'll need to scale the output data in one of the functions by 2, or scale the output data in both by $\sqrt{2}$. Explain why this scaling is necessary.

5. Check to see that your functions `iq2if` and `if2iq` preserve the spectral shape of GPS data. Download the binary file

   `http://radionavlab.ae.utexas.edu/datastore/gnssSigProcCourse/niData01head_5MHz.bin`

   This file was recorded using the Radionavigation Lab's National Instruments data capture equipment. The file contains baseband complex (I and Q) samples of the GPS L1 C/A signal originally centered at 1575.42 MHz with a bandpass signal bandwidth of 4 MHz. Samples were taken at a complex sampling rate of $f_s = \frac{4}{0.8} = 5$ MHz. You can use the following snippet of Matlab code to read in 0.5 seconds of data and store it in a complex vector `Y`:

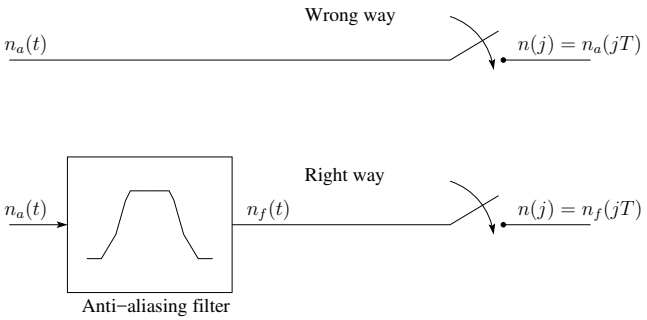

Figure 1: Two models of wideband noise sampling.

```
clear; clc;
%----- Setup
Tfull = 0.5;         % Time interval of data to load
fsampIQ = 5.0e6;     % IQ sampling frequency (Hz)
N = floor(fsampIQ*Tfull);
nfft = 2^9;          % Size of FFT used in power spectrum estimation

%----- Load data
fid = fopen('C:\yourDataPath\niData01head_5MHz.bin','r','l');
Y = fread(fid, [2,N], 'int16')';
Y = Y(:,1) + j*Y(:,2);
fclose(fid);
```

Use the `pwelch` function to estimate the power spectrum of the complex data just as you did in problem set 2 for the data from the Stanford "big dish." Next, use your `iq2if` function to convert the data to a bandpass signal at $f_{\rm IF} = 2.5$ MHz. Again use the `pwelch` function to estimate the power spectrum. This time the process will be similar to doing the power spectral estimate of the bandpass data from the GRID receiver (`dfDataHead.bin`) in problem set 2. Describe how this spectrum differs from that of the original baseband data. Finally, convert the bandpass data *back* to baseband with your function `if2iq` and compare the power spectrum of this data with the spectrum of the original data. What difference do you note?

6. Suppose we have a bandpass signal with $B = 4$ MHz centered at the GPS L1 frequency. Assuming $f_{\rm H} = f_{\rm L1} + B/2$ and $f_{\rm L} = f_{\rm L1} - B/2$, calculate the maximum wedge index $k_{\max}$ and the theoretical minimum sampling frequency $f_{s,\min}$ that avoids aliasing. Suppose that, just to be safe, we sample instead at $f_s = f_{s,\min}/0.8$. Explain why this is a bad idea. What if we set $W = B/0.8$ and $f_{\rm H} = mW$ for $m = 316$ and then sample at $2W$ Hz? Would this avoid aliasing? If not, devise a strategy (with specific numbers) for padding B with guard bands and sampling in such a way as to (1) avoid aliasing and (2) offer robustness to sample rate errors or carrier frequency errors. You may wish to consult the Proakis reading, "`sampling and reconstruction of signals_proakis.PDF`," found on Canvas.

7. It is important to understand how to model noise in the conversion from analog to digital signals. Consider the alternative models presented in Fig. 1. In both models, $n_a(t)$ is a Gaussian zero-mean white noise process with (two-sided) power spectral density $N_0/2$ W/Hz. This means that

the total power in $n_a(t)$, which is given by

$$P_n = \int_{-\infty}^{\infty} S_n(f)df = \int_{-\infty}^{\infty} \frac{N_0}{2}df$$

is infinite. Of course, this can't really be the case, but we model the noise as spectrally flat (white) nonetheless. If we model the sampling process according to the upper model of Fig. 1, then the variance of the discrete time samples is

$$\sigma_n^2 = E[n_a^2(jT)] = R_n(0) = P_n$$

In other words, our model predicts an infinite variance of the discrete-time noise samples, which isn't realistic or convenient. In the lower model in Fig. 1 the sampler is preceded by an anti-aliasing filter with unity gain and a (two-sided) noise-equivalent bandwidth of B Hz. In other words, if $H(f)$ is the anti-aliasing filter's frequency response, then $H_{\max} = H(0) = 1$ and

$$B = \int_{-\infty}^{\infty} |H(f)|^2 df$$

For this model, calculate the variance of the noise samples $n(j)$ in terms of N_0 and B. Note that if $S(f)$ is the power spectrum of the filtered signal $n_f(t)$, and $S_n(f) = N_0/2$ is the power spectrum of $n_a(f)$, these are related by $S(f) = |H(f)|^2 S_n(f)$.

8. Perform GPS L1 C/A signal acquisition on the complex baseband data in the binary file

   `http://radionavlab.ae.utexas.edu/datastore/gnssSigProcCourse/niData01head_5MHz.bin`

   described earlier.

   Determine the following for each signal present:

   (a) PRN identifier

   (b) Approximate apparent Doppler frequency (in Hz)

   (c) Approximate start time of the first full C/A code expressed in μs assuming that the first sample corresponds to $\tau_0 = 0$.

   (d) Approximate carrier-to-noise ratio C/N_0.

   Hand in a paper copy of your code, including all signficant subroutines.

   Hints:

   (a) Note that because the data in `niData01head_5MHz.bin` are complex baseband data, you'll need to either (1) shift them up to some intermediate frequency (e.g., 2.5 MHz) using your `iq2if.m` function, or (2) operate on the data with a complex baseband (zero-intermediate-frequency) model in mind. The GNSS signal model introduced in lecture was for a real bandpass signal with a non-zero f_{IF}:

$$x(j) = A(\tau_j)D[\tau_j - t_d(\tau_j)]C[\tau_j - t_s(\tau_j)]\cos[2\pi f_{\text{IF}}\tau_j + \theta(\tau_j)] + n(j) \quad (1)$$

By contrast, a complex baseband (zero-intermediate-frequency) model of a GNSS signal is as follows:

$$x(j) = A(\tau_j)D[\tau_j - t_d(\tau_j)]C[\tau_j - t_s(\tau_j)]\exp\left[i\theta(\tau_j)\right] + n(j) \tag{2}$$

Note that in (2), $x(j)$ and $n(j)$ are complex numbers whereas in (1) they are real.

If you choose to operate directly on the complex baseband data, then you'll also need to alter the "recipe" by which complex accumulations are calculated as follows:

$$S_k = \sum_{j=j_k}^{j_k+N_k-1} x(j)\exp\left[-i\hat{\theta}(\tau_j)\right] C[\tau_j - \hat{t}_{s,k}]$$

(b) You might find it helpful to look over the script `exampleAccumulation.m`, which gives a minimalist realization of a single correlation and accumulation operation performed on one GPS L1 C/A code. Note that the script is written for real-valued data with a non-zero $f_{\rm IF}$ rather than complex-valued baseband data, but you could adapt it for the complex case as described above.

(c) You'll have to try all PRNs. The 1-ms GPS L1 C/A PRN codes are Gold codes built up by combining two maximum-length LFSR sequences. For GPS L1 C/A, the length of the LFSR is $n = 10$ and the length of the m-sequences is $N = 2^{10} - 1 = 1023$.

In lecture, we identified the characteristic polynomials for the GPS L1 C/A codes as

$$\begin{aligned} f_1(D) =& 1 + D^3 + D^{10} \\ f_2(D) =& 1 + D^2 + D^3 + D^6 + D^8 + D^9 + D^{10} \end{aligned}$$

Start off with both the f_1 and f_2 LFSRs loaded with the all-ones sequence `1111111111`. Obtain the different GPS PRN Gold codes by circularly shifting the output of the f_2 LFSR before modulo-2 adding it to the output of the f_1 LFSR.

The GPS Interface Specification (IS), posted on Canvas as `IS-GPS-200L.pdf`, has details on generating the GPS L1 C/A codes. See Section 3.3.2.3 and the accompanying figures. Also see Table 3-Ia. The easiest way to interpret this table is to take the code delay chips (column 5) and shift the f_2 sequence by that amount before modulo-2 adding it to the f_1 sequence.

(d) Under the constant Doppler model, the phase estimate $\hat{\theta}(\tau_j)$ over the accumulation interval is related to the constant Doppler estimate $\hat{f}_{{\rm D},k}$ by

$$\hat{\theta}(\tau_j) = 2\pi(\tau_j - \tau_{j_k})\hat{f}_{{\rm D},k} + \hat{\theta}(\tau_{j_k}), \quad j = j_k, j_k+1, \ldots, j_k+N_k-1$$

It doesn't matter to what value you set $\hat{\theta}(\tau_{j_k})$ since your complex accumulation S_k probes both the in-phase and the quadrature parts of the incoming data, so you might as well set $\hat{\theta}(\tau_{j_k}) = 0$.

(e) If you set $\Delta t_{\rm step} = T$, as suggested in the lecture notes, then you don't need to generate $C[\tau_j - \hat{t}_{s,k}]$ for each test value of $\hat{t}_{s,k}$. You can simply generate $C[\tau_j - \hat{t}_{s,k}]$ only once for $\hat{t}_{s,k} = 0$ and $j = 0, 1, \ldots, N_k$. Then use this code replica in ever iteration when you correlate agains the data samples isolated for acquisition, $\{x(j)\}_{j_k}^{j_k+N_k-1}$. This simple approach to code replica correlation effectively tests the hypothesis that the code start time is τ_{j_k} and that $x(j_k)$ is the first sample after this start time.

Note that if your accumulation interval is longer than one code period, which is 1 ms for the GPS L1 C/A code, you still only need to search through one code period's worth of data since the code repeats.

(f) You can check your results by comparing them with the output of the GRID receiver shown below. The GRID results apply at 33.8 seconds into the longer data set `niData01_5MHz.bin`, of which only the first second is available in `niData01head_5MHz.bin`. But because the Doppler and pseudorange values change relatively slowly, they should serve as a good approximate check for your values. You can check your Doppler results directly against the values below, but you'll only be able to check your code offset values in a differential sense. Let ρ^i be the pseudorange for PRN i, and let $\Delta\rho^{ij} \equiv \rho^i - \rho^j$. Then $\mod(\Delta\rho^{ij}/c, N_c T_c)$ should be close to the time difference between your estimated code start times for PRNs i and j, where $N_c T_c$ is the nominal code period. The modulo operation is necessary because the time of flight of two signals can differ by more than one code period.

```
============ GRID: General Radionavigation Interfusion Device =============
 Receiver time:    0 weeks     33.8 seconds        Build ID:         1652
 GPS time:      1577 weeks 134496.1 seconds
------------------------------------------------------------------------------
CH  TXID    Doppler       BCP           PR      C/N0     Az     El   Status
             (Hz)       (cycles)     (meters)  (dB-Hz)  (deg)  (deg)
-------------------------GPS_L1_CA Channels-------------------------------
 1    1u    1121.60     -38257.91  20985472.06  44.5    299.3   29.2    6
 2    2    -1551.17      52357.06  19972502.03  47.7     53.8   38.9    6
 3    5    -1222.22      41011.94  18551086.07  45.8     43.8   69.7    6
 4   10    -2711.30      91429.41  19765301.28  47.4     43.1   42.3    6
 5   12    -2652.97      89477.40  20277962.34  45.2    209.0   39.1    6
 6   15     3103.75    -104943.52  21700953.46  42.8    158.6   21.1    6
 7   21     2049.57     -69279.79  23269270.58  40.6    273.5    9.7    6
 8   24     2186.67     -73970.56  22529297.11  41.1    272.6   10.9    6
 9   29     2459.39     -83244.16  19915944.89  47.0    322.6   43.6    6
10   30    -1424.31      47820.58  19728164.68  45.8    251.4   43.5    6
11   --   ---------  ------------  -----------  ----   ------  -----    -
12   --   ---------  ------------  -----------  ----   ------  -----    -
------------------------------Navigation Data-------------------------------
 X:  -742008.16   Y: -5462244.69   Z:  3198013.59   deltRx:  -1902021.04
 Xvel:     0.07   Yvel:     0.03   Zvel:    -0.00   deltRxDot:     -0.03
==============================================================================
```

9. **Extra Credit:** Recreate the simulation component of the plots of loss due to quantization in Fig. 5 in [4], posted on Canvas. Hand in your plots and your code.

References

[1] D. Rice and K. Wu, "Quadrature sampling with high dynamic range," *IEEE transactions on Aerospace and Electronic Systems*, no. 6, pp. 736–739, 1982.

[2] W. Waters and B. Jarrett, "Bandpass signal sampling and coherent detection," *IEEE Transactions on Aerospace and Electronic Systems*, no. 6, pp. 731–736, 1982.

[3] R. G. Vaughan, N. L. Scott, and D. R. White, "The theory of bandpass sampling," *IEEE Transactions on signal processing*, vol. 39, no. 9, pp. 1973–1984, 1991.

[4] C. Hegarty, "Analytical model for GNSS receiver implementation losses," *NAVIGATION*, vol. 58, no. 1, p. 29, 2011.

[5] T. E. Humphreys, "Interference," in *Springer Handbook of Global Navigation Satellite Systems*. Springer International Publishing, 2017, pp. 469–503.