DEVELOPMENT AND ANALYSIS OF A PARALLELIZED DIRECT POSITION ESTIMATION-BASED GPS RECEIVER IMPLEMENTATION

BY

MATTHEW PERETIC

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Assistant Professor Grace Xingxin Gao

ABSTRACT

Theoretical results conclude that one-step Direct Position Estimation (DPE)-based Global Navigation Satellite System (GNSS) receivers can achieve more accurate localization than their two-step counterparts. However, numerical solutions to DPE equations and approximations made for those equations introduce new effects that can reduce the accuracy improvement that such a one-step receiver may provide. This work examines effects that arise from those numerical solutions to DPE equations and from the approximations made for those equations. In light of the theoretical formulation of the DPE algorithm, resultant insights for design decisions of a DPE receiver implementation are presented, stemming from analysis of the localization and processing time results of a parallelized DPE receiver implementation developed specifically for this work. Additionally, a modular software architecture for the custom DPE receiver implementation and parallelization of portions of the DPE receiver algorithm for GPU operation are also proposed.

To my parents, Michael and Eileen Peretic, who have demonstrated steadfastly in their lives that "[love] does not seek its own... [love] bears all things, believes all things, hopes all things, endures all things."

ACKNOWLEDGMENTS

First, I wish to extend my sincere gratitude to my thesis adviser, Professor Grace Xingxin Gao. Her direction and patience have fostered both my technical and personal growth, and I am very thankful for this opportunity that she facilitated. Professor Gao, thank you for investing in me.

I am thankful to have shared my time at the University of Illinois with my friends and fellow research group members: Craig Babiarz, Sriramya Bhamidipati, Shubhendra Chauhan, Derek Chen, Arthur Chu, Shubh Gupta, Ashwin Kanhere, Cara Yang Kataria, Andy Lai, Enyu Luo, Tara Mina, Pulkit Rustagi, Akshay Shetty, Siddharth Tanwar, and Katherine Tsai. Working with you was always a team effort, and our collective friendship and humor were a source of joy.

I also wish to thank Dr. Juan Jose Jaramillo Jimenez, Dr. Jonathan Makela, Dr. Clark N. Taylor, and Dr. David Varodayan for their mentoring. Your advice and insight were timely and instrumental in my development, and I am appreciative of your support.

I wish to extend additional thanks for the use of datasets generated for academic use by the Grace Gao Research Group. In this regard, thank you to Arthur Chu, Shubhendra Chauhan, and the team of "Project GRIFFIN" of the United States Air Force Test Pilot School at Edwards Air Force Base. The use of these datasets enabled me to meet the analysis objectives of my thesis, and I am grateful for your meticulous work.

Finally, additional thanks to Enyu Luo for permission to utilize and expand upon the C++ object-oriented module paradigm that he developed for the Grace Gao Research Group.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Since the inception of the *Global Positioning System* (GPS), the two-step approach to computing navigation and timing solutions has been the popular choice for *Global Navigation Satellite System* (GNSS) receivers due to its speed and reliability in uncomplicated signal environments [1]. Such two-step methods compute ranges to satellites as intermediate measurements before computing navigation solutions. Typically, these intermediate measurements are found independently of each other using an approach known as scalar tracking, making them susceptible to errors when complicated effects are present in GNSS signals [2, 3]. Signal reflections or blockages, such as those from buildings, and interference on the electromagnetic spectrum, such as jamming or spoofing effects, can lead to loss of lock on satellite channels and significantly incorrect results from the receiver [4, 5, 6, 7, 8].

The *Direct Position Estimation* (DPE) algorithm, a one-step maximum likelihood formulation of the GNSS localization problem, has been proposed to specifically address these challenges [9]. The potential of the DPE approach has begun to be explored in the literature, as conceptual benefits, analytical advantages, demonstrated improvements, and processing techniques have been presented. However, two-step approaches remain dominant in the realm of GNSS receiver design, as the implementation practice of DPE is yet in its infancy. This work aims to support the growth of the DPE receiver body-of-knowledge and will begin with an overview of the DPE literature and the objectives of this work.

1.1 Related Works

The manner in which the one-step approach solves localization problems allows DPE to offer advantages over the two-step approach. While the DPE

approach itself will be introduced in detail in Chapter 2, this section will survey advantages offered by the DPE approach to motivate the work. Conceptually, the DPE approach can be more resilient to errors in received GNSS transmissions. Analytically, derivations prove that the DPE algorithm can theoretically achieve higher localization accuracy than two-step approaches. Demonstratively, DPE-based implementations have shown this localization improvement in practice, particularly in challenging environments. And, techniques for efficient processing have decreased the computational cost of software-defined DPE implementations.

1.1.1 Conceptual Advantages

Conceptually, DPE reduces vulnerabilities which are prevalent when satellite transmissions are separately tracked. Should the transmission from a satellite be reflected or a GNSS-like malicious transmission be present in the received signal, for a two-step approach, the satellite transmission delay found by the receiver to compute the ranging measurement may be corrupted by the presence of multiple GNSS-like signals in the received data [3, 10]. If the second step of a two-step approach does not account for this effect on the ranging measurement, the best that the receiver can do is discard that measurement [11]. Conversely, with the vector correlation used by DPE, as long as the line-of-sight-to-the-satellite transmission is present in the received signal, it will always contribute to the score of a given state [12].

1.1.2 Analytical Advantages

Analytical results have proven that the DPE algorithm is theoretically capable of achieving more accurate navigation solutions than two-step approaches, particularly in challenging signal environments. In [13], Cramér-Rao bound analysis is used to mathematically show that two-step approaches such as scalar tracking are not maximum-likelihood-optimal, while DPE, derived from the maximum likelihood estimate of receiver state $\boldsymbol{X}$, is. Closas *et al.* also show in [14] that DPE is theoretically guaranteed to compute the best possible navigation solution given a set of antenna voltage samples, while two-step approaches can, at best, only match the accuracy of DPE. Additionally,

in [15], Gusi-Amigó *et al.* show that DPE can maintain its performance at a higher level of noise power proportional to the number of available satellites as compared with two-step methods. The accuracy in lower signal-to-noise ratio environments is also analyzed in [15]. Bialer *et al.* show in [16] that DPE will outperform two-step approaches in dense multipath environments.

1.1.3 Demonstrated Advantages

The benefits of DPE begin to shine when the receiver's view of the sky is limited. Demonstratively, the proof-of-concept results presented in [4] show an improvement in localization accuracy using DPE over a two-step approach as the carrier-to-noise ratio decreases. Using only a 1-ms sample snippet in an environment with many signal reflections, Axelrad *et al.* generate position estimates in [12] within 50 m of the actual position using DPE. In an environment where half the sky is blocked, Ng and Gao add reflected transmissions to the replica signal constructed by DPE in [10], achieving position accuracy within 5 m of the actual position and 40 m of improvement in positioning accuracy over a two-step method. Chu and Gao provide an architecture for multi-receiver DPE-based localization in [17] and demonstrate that DPE-based receivers can generate position estimates as soon as line-of-sight to satellites is recovered by subjecting the receivers to high-dynamics aerial effects from a fixed-wing aircraft, in contrast to two-step methods which lose track in this scenario. Malicious GPS spoofers can even be localized when using a multi-receiver network of DPE receivers, as shown in [18].

1.1.4 DPE Computational Efficiency

Techniques to improve computation speed or localization accuracy that build on the basic DPE formulation have also been presented in the literature. In [19], a vector tracking loop is introduced and shown to provide successful tracking even when in a multipath environment. In [20], the Space-Altering Generalized Expectation Maximization (SAGE) algorithm is used to more efficiently search the navigation domain to find the maximum likelihood estimate. The search for the clock bias state is simplified in [21], providing another means of improving the efficiency. Axelrad *et al.* show in [22] how

to efficiently combine non-coherent correlations for different satellite channels for collective detection – a corresponding problem to DPE. Ng and Gao demonstrate in [23] that a DPE receiver can maintain track when duty-cycling sample set reads, increasing the time available to the receiver to process each iteration. And, [21, 22, 23] all utilize an efficient batch correlation method presented by [24].

1.2 Contributions

Yet, there remain barriers to broader usage of DPE in GNSS-based localization:

- While the benefits of DPE are evident in the analytical formulation of the approach, **DPE-based receiver implementations are solving numerical equations using sampled physical signals and discretized theoretical signal models**. This translation from analytical equations to the numerical implementation impacts the localization accuracy. Furthermore, approximations that are made to reduce the computational burden additionally impact the localization accuracy.

- **Naïve implementations of the DPE algorithm are more computationally expensive than those of two-step approaches**, limiting the practical use of the implementation.

- Implementations of DPE-based receivers have typically been developed to demonstrate a new theoretical result or improvements. However, **the impacts of implementation design choices on localization have not been the focus of prior works**, obscuring localization performance tradeoffs between different implementations.

This work aims to aid in reducing these barriers by the following contributions:

- **Identify effects present in the localization results which are caused by the nature of the one-step approach to localization, numerical representation of equations, and approximations made for efficiency.**

In the same vein of other DPE literature [25, 26], a signal-centred derivation of the DPE algorithm is presented. From this derivation, some benefits of DPE over two-step approaches are identified to highlight important aspects when solving the DPE algorithm. And, approximations for computational efficiency that are used in the DPE receiver implementation of this work are introduced.

In light of the mathematics behind the DPE receiver implementation, new conclusions about the DPE-based localization accuracy are drawn and identified in localization results from the DPE-based receiver implementation. A clock-aiding effect is identified in grid-based DPE, and this effect is found to be responsible for improved vertical accuracy in localization results. A position-domain discriminator-like step is shown to improve the accuracy of the receiver implementation, and a justification for this improvement is provided. Additionally, a limit on the accuracy of DPE-based receiver implementations that arises from a discretized cross-correlation step is identified and found to manifest in the localization results.

- **Leverage parallel programming principles to present a computationally faster implementation of a DPE-based single-constellation GNSS receiver.**

  A grid-based approach to a DPE-based receiver implementation determines the navigation solution by evaluating candidate navigation solutions and choosing the best one. Each candidate may be evaluated independently of the others. Furthermore, the candidate navigation solutions are evaluated based on a cross-correlation score between the received signal and a discretized theoretical replica, and the samples of the theoretical replica may also be constructed independently of each other. For these reasons, the DPE algorithm is well-suited for parallel processing. Use of a *graphics processing unit* (GPU) for more efficient DPE processing has been suggested in [27, 28], but an implementation has not yet been presented in the literature.

  A custom software-defined parallelized DPE-based GNSS receiver is developed for this work and evaluated against a comparable software-defined sequential DPE-based receiver. To accomplish this, the DPE al-

gorithm is parallelized. The implementation is developed in the CUDA C/C++ programming language [29] and evaluated on an NVIDIA Jetson TX2 portable GPU. Additionally, to support future research, the receiver implementation is decomposed into seven modular subtasks packaged with well-defined inputs and outputs, and the implementation is parameterized in a manner that allows its run speed to be tuned by a user for the host hardware.

- **Provide insights into implementation design decisions stemming from analysis of localization and processing time results from the custom DPE-based receiver implementation.**

  Implementation details, such as the choice of candidate navigation solutions and sampling frequency, play a role in the localization results. Different candidate navigation solution "grids" are used by the DPE-based receiver implementation, and insights for the benefits provided by these grids are presented by comparing experimental data with analytically-identified effects. The manifestation of these effects in the navigation solutions computed by the DPE-based receiver provides a foundation for future research into numerical implementations of the DPE algorithm. Speedup from the parallelized DPE-based receiver implementation is evaluated and remaining bottlenecks are identified to provide guidance on tuning and GPU hardware selection.

The thesis is organized into six chapters. This chapter introduces the objectives and contributions of the work. Chapter 2 derives the DPE algorithm from the signal structure of a GNSS transmission and identifies some approximations and techniques used by a numerical implementation of DPE. Chapter 3 presents a modular DPE-based receiver software architecture and details for the implementation used in this work. Chapter 4 presents new analysis of effects present in DPE and DPE-based receiver implementations. Chapter 5 presents results from the receiver implementation for a stationary receiver dataset, two mobile receiver datasets, and the computational efficiency of the implementation. Chapter 6 concludes the thesis.

CHAPTER 2

BACKGROUND

A GNSS consists of a set of satellites in accurately known orbits transmitting known messages to their users. In the context of navigation, the namesake purpose of GNSS, a user may operate a GNSS receiver to compute position estimates using knowledge of the satellites' positions and the signals observed by the receiver's antenna [1, 30]. Implicitly coupled with the question of position is one of timing. Not only is timing often desired by the user to provide context for the position estimates, but receiver algorithms also must typically determine the signal receive time with respect to some reference when estimating position [1]. This leads to the *position-time* state representation:

$$\mathbf{x} \triangleq [x \quad y \quad z \quad \delta t]^\top = [\mathbf{p} \quad \delta t]^\top \tag{2.1}$$

where

- $\mathbf{p} \triangleq [x \quad y \quad z]^\top$ describes the 3D position estimate, assumed without loss of generality in this work to be a geographic and Cartesian coordinate system, such as *Earth-centered, Earth-fixed* (ECEF).
- δt describes the time estimate, assumed without loss of generality in this work to be a *clock bias* with respect to a time standard, such as GPS time.

Many receiver algorithms and architectures have been developed for generating position-time estimates using GNSS [31, 32, 33, 34, 35]. However, for the physically implemented GNSS, the received signals observed are subject to real-world interferences that may be able to drive the receiver into a failure mode [3]. Due to the breadth of potential interferences, these algorithm-derailing effects have not been universally overcome, and GNSS receiver design remains an area of active research for further study of the failure modes and development of improved receiver countermeasures [7, 36, 37, 38, 39].

DPE, the subject of this work, is one such recently developed approach that mitigates the impact of some interferences to *conventional GNSS receivers* [9]. This chapter will introduce fundamental concepts of GNSS-based navigation and formally present the DPE approach. The introduction to GNSS-based navigation will also summarize the conventional receiver approach, known as *scalar tracking*, in order to highlight the primary improvements introduced by DPE.

For simplicity of discussion, the numerical solutions presented in this work are exclusive to a single-constellation GPS receiver implementation. Different GNSS constellations use different channel sharing schemes, which will result in differently-shaped probability density functions over the same range of navigation solutions. Thus, to enable effective discussion of a numerical implementation compared to its analytical formulation, this work will focus solely on GPS. However, the concepts presented are expected to be applicable without loss of generality to all GNSSs, as the concepts draw their assumptions from the problem of satellite-based localization.

2.1 GNSS-Based Navigation

As the presence of stars in the sky can provide references for position and time, the fundamental idea of GNSS-based navigation is that a user operating a *GNSS receiver* can estimate their own state $\mathbf{x}$ by receiving signals transmitted from a constellation of artificial satellites. By taking measurements of the voltage of a GNSS receiver's antenna, the receiver runs an algorithm to process these samples into measurements, then filters the measurements into a state estimate. And, it is the conceptual approach that the processing algorithm takes to the problem of localization that will paint with a broad brush the capabilities of the receiver.

This section will first introduce details of the GPS constellation relevant to understanding the conceptual approaches of scalar tracking and DPE. Then, the scalar tracking approach will be summarized for reference to DPE.

2.1.1 GPS Signal Structure

The GPS constellation consists of 31 satellites orbiting such that they pass over the same point on Earth approximately twice per day [1]. Following the physics of orbital mechanics, the trajectory of each GPS satellite is parameterized into an *ephemeris*, and the set of all GPS *ephemerides* are made publicly available, downloadable both online and from the GPS satellites themselves [1]. Each ephemeris parameterizes a satellite trajectory such that, given a time within the range of validity for the ephemeris, a series of equations may be solved to find the *position-velocity-time* (PVT) state of that satellite.

By finding a time-identifier pattern of bits in the *navigation data* of the received transmission [1], a receiver can determine the time at the satellite when the current set of samples were sent. Since the ephemerides are sets of orbital parameters that solve for the PVT states of the satellites at given times, knowing the transmission time for a signal from satellite i means satellite i can be used as a position-time reference.

The signal from satellite i can be modeled as follows [25]:

$$\begin{aligned} S_t^i\{\tau\} = a_t^i D_t^i\{(f_{C/A} + f^i_{code_d,t})\tau + \phi^i_{code,t}\} G^i\{(f_{C/A} + f^i_{code_d,t})\tau + \phi^i_{code,t}\} \\ \exp\{j2\pi((f^i_{L1} + f^i_{carr_d,t})\tau + \phi^i_{carr,t})\} \end{aligned} \tag{2.2}$$

where the functions defined are referenced to the time t at the beginning of a finite-length sampling window and the parameter τ is the index of a sample in the window of K samples $[0, \frac{1}{f_s}, \frac{2}{f_s}, ..., \Delta T - \frac{1}{f_s}]$. f_s and ΔT are the sampling frequency and time duration of the sampling window, respectively.

Dissecting Equation 2.2, the reader will observe four constituent functions:

- a_t^i is the amplitude of the received signal, assumed to be constant over the sampling window.
- $\exp\{\cdot\}$ describes the *carrier wave*, broadcasting at frequency f_{L1} and experiencing some Doppler shift $f^i_{carr_d,t}$ and phase shift $\phi^i_{carr,t}$.
- $G^i\{\cdot\}$ is an encoding function returning either $+1$ or -1 according to the *Pseudo-random Number* (PRN) *Code Division Multiple Access* (CDMA) channel sharing method that allows all GPS satellites to

transmit on the same carrier frequency f_{L1}. $G^i\{\cdot\}$ is a function of the PRN code frequency $f_{C/A}$ experiencing some Doppler shift $f^i_{code_d,t}$ and with the PRN code phase $\phi^i_{code,t}$.

- $D^i_t\{\cdot\}$ is a function describing the transmitted navigation data returning either $+1$ or -1 according to the navigation data being transmitted. Present in this data are the identifier bit patterns that allow for recovery of transmission time. One navigation data bit is timed to be exactly the length of 20 complete PRN codes, making $D^i_t\{\cdot\}$ dependent on exactly the same parameters as $G^i\{\cdot\}$.

Once the parameters of the functions in Equation 2.2 are known, the receiver can determine the transmitted navigation data by removing the other components of $S^i_t\{\tau\}$ and recovering the unscaled $D^i_t\{\cdot\}$. f_{L1} and $f_{C/A}$ are known constants of the GPS implementation, so it remains for the receiver to estimate the parameters $\phi^i_{code,t}$, $f^i_{carr,t}$, $\phi^i_{carr,t}$, and $f^i_{carr,t}$.

These four parameters are known as *channel parameters*:

- The *code phase*, $\phi^i_{code,t}$
- The *code frequency shift*, $f^i_{code,t}$
- The *carrier phase*, $\phi^i_{carr,t}$
- The *carrier frequency shift*, $f^i_{carr,t}$

List 2.1: Channel Parameters

When the receiver has estimates of these channel parameters for satellite i, the receiver is said *to have track* of satellite i. And, by tracking a subset of the satellites available in the GPS constellation, the receiver can utilize these satellites as landmarks and begin to determine information about itself through a localization algorithm.

2.1.2 Two-Step Localization

The two-step approach to GNSS-based navigation, typically implemented by a process called scalar tracking, uses the principle of *multilateration* [1].

In multilateration, an unknown position-time state can be found given the Euclidean distance to known position-time states. In the case of a GPS receiver, the receiver's antenna is the unknown state, the GPS satellites are known states, and the ranges are found by processing measurements from the antenna. Thus, the two steps arise from:

1. Processing the received signal to compute range measurements from the antenna voltage samples.

2. Computing the position-time estimate from the range measurements by the principle of multilateration.

The range measurements are computed by taking the difference of the satellite i's transmission time t^i_{tx} with the current time estimate at the receiver t and multiplying by the speed of light c, providing a *time-of-flight* (ToF) measurement to the satellite. However, it should be noted these ranges are estimates. Passive environmental effects, active broadcasts in the same or nearby bands, and reflections of the GPS signals themselves can interfere with the receiver's track of the satellite channel parameters. And, the time estimate of the receiver is subject to error.

Acknowledging this, the range estimate to the i^{th} satellite will not be exactly equal to the true range r^i and is appropriately called the *pseudorange* ρ^i:

$$\rho^i = c(t - t^i_{tx}) = \sqrt{(x^i - x)^2 + (y^i - y)^2 + (z^i - z)^2} + c(\delta t^i - \delta t) + \epsilon^i \quad (2.3)$$

where the difference between the true range and ρ^i is captured by the term ϵ^i. Due to this inaccuracy in the pseudorange estimates, the multilateration problem of locating the receiver will not have exact solutions but will instead be an optimization problem to find the *maximum likelihood* receiver state $\mathbf{x}$ given the pseudoranges.

If pseudorange measurements are significantly corrupted, the receiver position estimate can be drawn away and channel parameter tracking can be lost. These pseudorange corruptions can be compensated for using optimization techniques other than least-squares. Robust estimators [40, 41, 42] and weighting the pseudorange by estimates of the channel noise [43, 44] are among a variety of demonstrated alternative techniques. Additionally, algorithms have been introduced to identify pseudoranges that may be corrupted

to an extent that produces hazardously misleading position-time information [11, 45].

While there is flexibility in the choice of optimization technique used, the two-step architecture remains – scalar pseudorange measurements are first computed from separately tracked channels, then all concurrent measurements are used together in position-time estimation.

2.2 Direct Position Estimation Formulation

In contrast to the two-step approach of scalar tracking, the one-step approach of DPE arises from the observation that the channel parameters of satellite i can be completely determined by knowing the relative position, time, and *velocity* between the receiver's antenna and a given satellite.

$$\boldsymbol{X} \triangleq [x \quad y \quad z \quad \delta t \quad \dot{x} \quad \dot{y} \quad \dot{z} \quad \dot{\delta t}]^\top = [\mathbf{p} \quad \delta t \quad \dot{\mathbf{p}} \quad \dot{\delta t}]^\top \tag{2.4}$$

This section will mathematically derive the DPE method of estimating the receiver's PVT state from this observation. This section will also simplify the computation of DPE and examine the meaning of the operations performed on the physical signals by DPE.

2.2.1 Revisiting the Received Signal Model

Given the channel parameters, the transmission from every satellite can be completely reconstructed at a given PVT state (Equation 2.2). The receiver's PVT state can then be determined by choosing the state whose reconstructed transmissions most closely matches the received signal.

To show this observation mathematically, for the *position-velocity-time* (PVT) state $\boldsymbol{X}$ and referring to List 2.1 and Equation 2.2 for the channel parameters and their roles, the channel parameters may be written in terms of a PVT solution as follows [25]:

- $\phi^i_{code,t} = f_{C/A}(t - \text{TOW}^i - \frac{N^i_{code,t}}{f_{C/A}}) - \frac{f_{C/A}}{c}(\|\mathbf{p}^i - \mathbf{p}\| + (c\delta t - c\delta t^i))$
- $f^i_{code,t} = \frac{f_{C/A}}{c}(\frac{(\mathbf{p}^i - \mathbf{p})^\top}{\|\mathbf{p}^i - \mathbf{p}\|}(\dot{\mathbf{p}} - \dot{\mathbf{p}}^i) + (c\dot{\delta}t^i - c\dot{\delta}t))$
- $\phi^i_{carr,t} = f_{L1}(t - \text{TOW}^i - \frac{N^i_{code,t}}{f_{C/A}}) - \frac{f_{L1}}{c}(\|\mathbf{p}^i - \mathbf{p}\| + (c\delta t - c\delta t^i))$
- $f^i_{carr,t} = \frac{f_{L1}}{c}(\frac{(\mathbf{p}^i - \mathbf{p})^\top}{\|\mathbf{p}^i - \mathbf{p}\|}(\dot{\mathbf{p}} - \dot{\mathbf{p}}^i) + (c\dot{\delta}t^i - c\dot{\delta}t))$

List 2.2: Definitions of the Channel Parameters

where t is the signal receive time, TOW^i is the GPS *time-of-week* (TOW) associated with the identifier bits being used for global time reference, and $N^i_{code,t}$ is the integer number of CDMA codes elapsed since the identifier bits of the TOW^i.

The code phase and carrier phase parameters $\phi^i_{code,t}$ and $\phi^i_{carr,t}$ are found by computing the number of cycles elapsed since the time-alignment transmission and subtracting the time the signal spent travelling to the receiver, which leads to the dependence on $\boldsymbol{X}$. The code frequency and carrier frequency shifts $f^i_{code,t}$ and $f^i_{carr,t}$ are the result of the Doppler effect due to the relative velocity between satellite i and the receiver, which also leads to the dependence on $\boldsymbol{X}$.

The signal received $S_t\{\cdot\}$ is the superposition of the transmissions of all GPS satellites in view $S^i_t\{\cdot\}$ plus the effects of noise $N_t\{\cdot\}$ [25]:

$$S_t\{\tau\} = \sum_i S^i_t\{\tau\} + N_t\{\tau\} \tag{2.5}$$

For this work, noise is defined to include passive environmental effects, other active broadcasts in the same or nearby bands, modelling errors such as satellite position inaccuracies, and reflections of the GPS signals themselves.

Since the received signal $S_t\{\cdot\}$ is composed of the transmissions of the individual satellites, $S_t\{\cdot\}$ is also a function of the receiver's PVT state $\boldsymbol{X}$. However, since the individual transmissions are summed, the received signal is also a function of the *amplitude* of each transmission. While the amplitude of the received signal could be ignored in scalar tracking due to the independence of each channel, it *cannot be ignored when reconstructing the expected composite signal*. Thus, the received signal $S_t\{\cdot\}$ can be known for a given PVT state $\boldsymbol{X}$ and the transmission amplitude $\boldsymbol{a}$ from each satellite.

2.2.2 Derivation of the DPE Algorithm

Following the derivation presented in [26], finding the replica signal $\hat{S}_t\{\cdot\}$ of maximum likelihood to the received signal $S_t\{\cdot\}$ is equivalent to finding $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{a}}$, the maximum likelihood estimates of the receiver state and the amplitudes of each satellites' transmission, according to the invariance property of *maximum likelihood estimation* (MLE) [9]. Assuming $N_t\{\cdot\}$ is *additive white Gaussian noise* (AWGN), this yields the least-squares minimization problem:

$$\hat{\boldsymbol{a}}, \hat{\boldsymbol{X}} = \underset{\boldsymbol{a},\boldsymbol{X}}{\arg\min} \, ||\boldsymbol{y} - \boldsymbol{C}\boldsymbol{a}||^2 \tag{2.6}$$

where $\boldsymbol{y}$ is the vector of K antenna voltage samples in the sampling window $[0, \frac{1}{f_s}, \frac{2}{f_s}, ..., \Delta T - \frac{1}{f_s}]$, $\boldsymbol{C}$ is a $K \times M$ matrix of the samples of the reconstructed signal $S_t\{\cdot\}$ in the sampling window $[0, \frac{1}{f_s}, \frac{2}{f_s}, ..., \Delta T - \frac{1}{f_s}]$ for the M visible satellites, and $\boldsymbol{a}$ is the vector of received amplitudes of the M visible satellites [26].

Performing the maximum likelihood estimation on Equation 2.6 is possible, but would be an $(8 + M)$-dimensional problem, as the state PVT states $\boldsymbol{X}$ and the received signal amplitudes $\boldsymbol{a}$ are unknown. Additionally, for the objective of GPS navigation, estimates of the received satellites' amplitudes are typically not of importance to the user. Thus, it would be desirable to modify this formulation to eliminate the explicit simultaneous estimation of $\hat{\boldsymbol{a}}$ from the minimization function.

The estimate of $\hat{\boldsymbol{a}}$ is found by taking the derivative of Equation 2.6 with respect to $\hat{\boldsymbol{a}}$ assessed at the maximum likelihood state $\hat{\boldsymbol{X}}$ and setting the resultant equation equal to zero [9] (intermediate steps omitted for brevity):

$$\hat{\boldsymbol{a}} = (\hat{\boldsymbol{C}}^*\hat{\boldsymbol{C}})^{-1}\hat{\boldsymbol{C}}^*\boldsymbol{y} \tag{2.7}$$

where $\hat{\boldsymbol{C}}$ is the matrix $\boldsymbol{C}$ assessed at $\hat{\boldsymbol{X}}$, and $\hat{\boldsymbol{C}}^*$ is the Hermitian transpose of $\hat{\boldsymbol{C}}$. Intuitively, this result means that the maximum likelihood estimate of $\boldsymbol{a}$ is the least-squares solution of the difference between the maximum likelihood reconstructed signal and the actual received signal. This also gives an expression for $\hat{\boldsymbol{a}}$ that depends on $\boldsymbol{X}$ through $\boldsymbol{C}$. Substituting this result into Equation 2.6 and expanding gives:

$$\begin{aligned}\hat{\boldsymbol{X}} &= \arg\min_{\boldsymbol{X}} ||\boldsymbol{y}^*\boldsymbol{y} - 2\boldsymbol{y}^*\boldsymbol{C}(\hat{\boldsymbol{C}}^*\hat{\boldsymbol{C}})^{-1}\boldsymbol{C}^*\boldsymbol{y} + \boldsymbol{y}^*\boldsymbol{C}(\hat{\boldsymbol{C}}^*\hat{\boldsymbol{C}})^{-1}\boldsymbol{C}^*\boldsymbol{y}|| \\ &= \arg\min_{\boldsymbol{X}} ||\boldsymbol{y}^*\boldsymbol{y} - \boldsymbol{y}^*\boldsymbol{C}(\hat{\boldsymbol{C}}^*\hat{\boldsymbol{C}})^{-1}\boldsymbol{C}^*\boldsymbol{y}|| \\ &= \arg\max_{\boldsymbol{X}} ||\boldsymbol{y}^*\boldsymbol{C}(\hat{\boldsymbol{C}}^*\hat{\boldsymbol{C}})^{-1}\boldsymbol{C}^*\boldsymbol{y}||\end{aligned} \tag{2.8}$$

Further reduction comes through a property of the CDMA codes used by GPS. Taking a look at the structure of $\boldsymbol{C}$:

$$\boldsymbol{C} = \begin{bmatrix} c^1_{[0]} & c^2_{[0]} & \cdots & c^M_{[0]} \\ c^1_{[\frac{1}{f_s}]} & c^2_{[\frac{1}{f_s}]} & \cdots & c^M_{[\frac{1}{f_s}]} \\ c^1_{[\frac{2}{f_s}]} & c^2_{[\frac{2}{f_s}]} & \cdots & c^M_{[\frac{1}{f_s}]} \\ \vdots & & & \vdots \\ c^1_{[\Delta T - \frac{1}{f_s}]} & c^2_{[\Delta T - \frac{1}{f_s}]} & \cdots & c^M_{[\Delta T - \frac{1}{f_s}]} \end{bmatrix} \tag{2.9}$$

A given column i consists of the K samples of the reconstructed transmission from satellite i. Thus, when multiplying $(\boldsymbol{C}^*\boldsymbol{C})$, the elements are the sum of the element-wise multiplication of the samples of two signals. By definition, the sum of element-wise multiplications of samples is the *cross-correlation* between two signals. Thus, the element at index (i, j) of the resultant matrix has the value of the cross-correlation between the signal from satellite i and the signal from satellite j [26].

The PRN codes used by GPS for CDMA are *Gold codes*, which have an auto-correlation score of K for a signal with K samples and a cross-correlation score between different codes significantly smaller in magnitude than K [30]. Thus:

$$[(\boldsymbol{C}^*\boldsymbol{C})]_{i,j} = \begin{cases} K, & i = j \\ [(\boldsymbol{C}^*\boldsymbol{C})]_{i,j} \ll K, & i \neq j \end{cases} \tag{2.10}$$

leading to the approximation $(\boldsymbol{C}^*\boldsymbol{C}) \approx NI_M$ [26], where I_M is the identity matrix of size $M \times M$. Substituting this result into Equation 2.8 gives:

$$\hat{\boldsymbol{X}} = \arg\max_{\boldsymbol{X}} \frac{1}{K} ||\boldsymbol{y}^*\boldsymbol{C}\boldsymbol{C}^*\boldsymbol{y}|| \tag{2.11}$$

Again following from the definition of C, multiplying $(\boldsymbol{C}^*\boldsymbol{y})$ produces a resultant vector of length M where the element at index i is the cross-correlation between the reconstructed signal from satellite i and the signal

actually received. The multiplication ($\boldsymbol{C}^*\boldsymbol{y}$) then performs the sum of the element-wise squares of each satellites' cross-correlation score with y. This result is the *vector cross-correlation* between the received signal and the expected transmission from each satellite without the explicit estimation of the received transmissions' amplitudes $\boldsymbol{a}$ [19].

So, for the cross-correlation function $\mathcal{R}_t^i$ between the received signal S_t and the reconstructed transmission of satellite i (as given in Equation 2.2),

$$\mathcal{R}_t^i(\boldsymbol{X}) = \sum_i S_t\{\tau\} D_t^i\{\cdot\} G^i\{\cdot\} \exp\{\cdot\} \tag{2.12}$$

the vector cross-correlation function $\mathcal{R}_t$ is the summation of the cross-correlations $\mathcal{R}_t^i$,

$$\mathcal{R}_t(\boldsymbol{X}) = \sum_i \mathcal{R}_t^i(\boldsymbol{X}) \tag{2.13}$$

allowing the objective function of DPE to be written as:

$$\hat{\boldsymbol{X}} = \arg\max_{\boldsymbol{X}} \frac{1}{K} ||\boldsymbol{y}^* \boldsymbol{C} \boldsymbol{C}^* \boldsymbol{y}|| = \arg\max_{\boldsymbol{X}} ||\mathcal{R}_t(\boldsymbol{X})|| \tag{2.14}$$

In summary, the DPE approach provides a one-step maximum likelihood estimation of the receiver's PVT state $\boldsymbol{X}$ by comparing the signal received to the expected transmission at the state $\boldsymbol{X}$ for each satellite through a *vector cross-correlation*.

2.2.3 DPE Terminology

For the remainder of this work, the action of evaluating the similarity between the expected signal and the received signal through the vector cross-correlation will be referred to as the *replica-received cross-correlation*. As described in Section 2.1.1, the received signal for a DPE-based receiver is a set of antenna voltage samples. A DPE-based receiver will solve this equation (or an approximation thereof) for sample set after sample set, providing localization results to the user. In this work, the process of acquiring a sample set and computing a DPE navigation solution from it will be referred to as one *timestep*.

2.3 Numerical Solutions to DPE

As the GPS navigation messages and PRN codes are highly non-linear and a function of eight dimensions, a receiver implementation cannot be expected to analytically solve an objective function which uses the replica-received cross-correlation, such as Equation 2.14. Thus, implementing this operation numerically and studying the effects that arise from the numerical implementation is the focus of this work.

This section will highlight numerical DPE techniques presented in other works which will be employed in this work's DPE receiver implementation.

2.3.1 Grid-based versus Iterative DPE Approaches

Two classes of algorithms exist to numerically solve the DPE objective function: iterative and grid-based. The grid-based approach to DPE computes a solution for the objective function by constructing a grid of candidate receiver states and evaluating the cross-correlation between the received signal and replica signal for each grid point's state [12]. The cross-correlations are the likelihood for each grid point, meaning the grid is the discrete representation of the *probabilistic manifold* of PVT estimates for DPE. Finding the maximum likelihood state on the manifold provides a numerical solution to the objective function of DPE. Each of the candidate receiver states may be evaluated independently of the others, which can be exploited for computational efficiency by parallelizing the evaluation of each state across the many threads of a GPU. This approach is visualized in Figure 2.1.

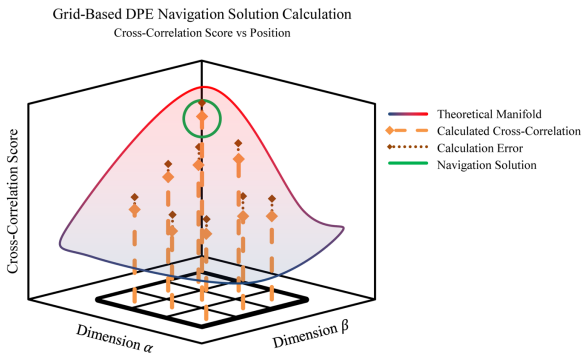

Figure 2.1: The grid-based approach to DPE evaluates multiple states simultaneously and chooses the best one.

In contrast, iterative algorithms for DPE are more suited for computation on a CPU, as they evaluate the cross-correlation between the replica and received signals at one state, quantify the replica-received error at that state, then move the state and try again until the replica-received error is below some threshold. This is visualized in Figure 2.2. Closas proposes the use of the Space-Alternating Generalized Expectation Maximization (SAGE) algorithm in [20], which reduces the search space for clock bias estimates to improve computation cost. However, as Cheong *et al.* observe in [21], the reduction in search space by the SAGE algorithm may fail to correct errors in the clock bias. And, as Chapter 4 will show, such errors would also propagate into the vertical estimates. Furthermore, iterative methods are subject to converging to local minima, while a grid-based approach can be easily configured to sample candidate points over a large domain and reject local minima with proper filtering.

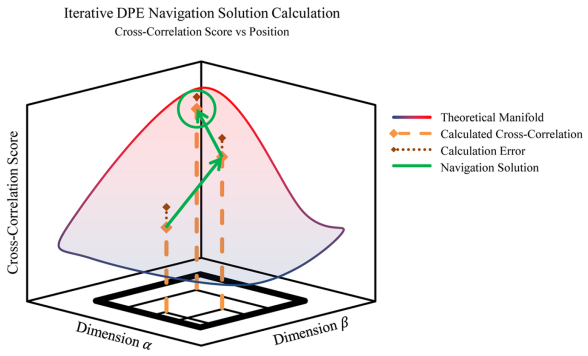

Figure 2.2: The iterative approach to DPE begins at one state and progressively refines that update until a convergence condition is met.

A grid-based approach will be studied in this work. Not only does the grid-based approach offer the advantages outlined above as compared to the iterative approach, but further study of a parallelized approach to DPE is desirable. The idea of a GPU-based implementation has been suggested in the literature [27, 28] and grid-based DPE has been demonstrated in multiple experiments (such as [18, 25, 26]), but a GPU-specific implementation has yet to be demonstrated and evaluated in the literature.

2.3.2 Improving the Computational Efficiency

Even with parallelization, individually computing the received signal replica and its replica-received cross-correlation score for each candidate receiver state on an 8-dimension grid would be computationally prohibitive except for the most powerful of computers or for grids which may be trivially small. However, in exchange for a slight degradation of the accuracy of the replica-received scores, the two concepts of *manifold decoupling* and *batch correlation* reduce the computational cost of the problem to one that is much more practically implementable. These techniques allow the scores for all candidate grid points to be generated together in an efficient manner and turn the step of scoring all points on the grid from one of intense computation into a much faster look-up step.

Manifold Decoupling

Since the received signal is dependent on the PVT state of the receiver, a naïve implementation of DPE measurement estimation is an 8-dimension optimization problem. However, as proposed in [46] and demonstrated in [25], GNSS-based position and velocity estimation can be performed independently of each other, given a reasonable starting estimate of the PVT state.

This is due to the scale of the GNSS with respect to the search space of an initialized receiver. For a fixed position state, the shape of the cross-correlation function with respect to velocity states will not be significantly affected by the exact position state chosen over a range of position states on the order of 100 meters [46]. Similarly, for a fixed velocity state, the shape of the cross-correlation function with respect to position states will not be significantly affected by the exact velocity state chosen over the range of reasonable receiver velocity changes [46].

Thus, finding the maximum likelihood PVT state may be approximated by estimating the maximum likelihood position-time state from a position-time manifold and the maximum likelihood velocity-drift state from a velocity-drift manifold. This approximation will hold as long as the previous iteration's estimate of the PVT state is close to the actual PVT state of the current sample set, as each manifold will estimate four states and hold four states fixed:

$$\hat{\boldsymbol{X}}_t = \begin{bmatrix} \mathbf{x}_t & \dot{\mathbf{x}}_t \end{bmatrix}^\top \approx \begin{bmatrix} \arg\max_{\mathbf{x}_t} ||\mathcal{R}_t(\mathbf{x}_t, \hat{\dot{\mathbf{x}}}_{t-1})|| & \arg\max_{\dot{\mathbf{x}}_t} ||\mathcal{R}_t(\hat{\mathbf{x}}_{t-1}, \dot{\mathbf{x}}_t)|| \end{bmatrix}^\top \tag{2.15}$$

This reduces the estimation problem from an 8-dimensional optimization problem to two 4-dimensional optimization problems – a total reduction in the number of points searched by $\frac{n^4}{2}$ for a hypercubic grid of size n.

Batch Correlation

With decoupled position-time and velocity-drift manifolds, another optimization is made possible. Both manifolds are capable of approximating the cross-correlation function with respect to a given channel parameter then solving for the value of that channel parameter value for a given state to find their cross-correlation score.

For the position-time manifold, referring to the channel parameters given in List 2.1, the velocity state $\dot{\mathbf{p}}$ is known and the unit vector to the satellite $\frac{(\mathbf{p}^i - \mathbf{p})^\top}{\|\mathbf{p}^i - \mathbf{p}\|}$ is effectively constant over the size of the grid. Thus, the frequency parameters $f^i_{code,t}$ and $f^i_{carr,t}$ are constant and the phase parameters $\phi^i_{code,t}$ and $\phi^i_{carr,t}$ are a function of the position-time state $\mathbf{x}$. If the PRN code chips $G^i_{rec}\{\cdot\}$ are retrieved from the received signal, the circular cross-correlation property of the PRN codes of GPS may be leveraged to compute the cross-correlation score for all code phases $\phi^i_{code,t}$ [24]. This is accomplished by a property of the *fast Fourier transform* (FFT) – the circular cross-correlation function of two discrete signals u and v of length ΔT can be found by:

$$\text{corr}(u, v)[k] = \sum_{m=0}^{m=\Delta T} u^*[m]v[m+k] = \mathcal{F}^{-1}(\mathcal{F}^*(u)\mathcal{F}(v))[k] \tag{2.16}$$

for the FFT function $\mathcal{F}$. Thus, the position-time correlation scores may be approximated by:

$$\mathcal{R}_t(\mathbf{x}_t, \hat{\dot{\mathbf{x}}}_{t-1}) \approx \mathcal{F}^{-1}(\mathcal{F}^*(G^i_{rec}\{\cdot\})\mathcal{F}(G^i\{\mathbf{x}\})) \tag{2.17}$$

For the velocity-drift manifold, referring to the channel parameters given in List 2.1, the position state $\mathbf{p}$ is known. Thus, the phase parameters $\phi^i_{code,t}$ and $\phi^i_{carr,t}$ are constant, and the frequency parameters $f^i_{code,t}$ and $f^i_{carr,t}$ are a function of the velocity-drift state $\dot{\mathbf{x}}$. Even with the phase parameters known, computing the cross-correlation scores for every frequency is still

computationally expensive. However, the power of a given frequency in the carrier wave can be used to score that frequency [24]. Thus, if the carrier wave $\exp_{rec}\{\cdot\}$ is retrieved from the received signal, the velocity-drift correlation scores may be approximated by:

$$\mathcal{R}_t(\hat{\mathbf{x}}_{t-1}, \dot{\mathbf{x}}_t) \approx \mathcal{F}(\exp_{rec}\{\cdot\}) \tag{2.18}$$

Manifold Scoring

After the batch correlations, each state on the position-time or velocity-drift grid will look up its score from the batch correlation results by back-calculating the expected $\phi^i_{code,t}$ or $f^i_{carr,t}$, respectively, for each tracked channel i and sum the results. Since the batch correlations are effectively sampling the continuous cross-correlation function or FFT-based frequency difference function at specific code delays or Doppler frequencies, a grid point will interpolate between two samples if its back-calculated value does not exactly match that of one of the sample indices.

Once each point is scored, the maximum likelihood state may be found. This state is the measurement of the software-defined DPE receiver. The maximum likelihood state may be chosen in different ways, such as simply choosing the grid state with the highest score or by applying a peak detector step to resolve the navigation solution estimate to a sub-grid-level precision. The means of choosing a maximum likelihood state is a signal tracking problem, which will be addressed in Section 2.3.3.

2.3.3 DPE Signal Tracking

The better the generation of the expected transmission $\boldsymbol{C}$ in Equation 2.14, the better the estimation of the receiver's PVT state. With a perfect model of the world, any state $\boldsymbol{X}$ may calculate the expected transmission exactly. However, the DPE objective function of Equation 2.14 is modeled as an estimation problem following the assumption that such a perfect model does not exist.

Open-Loop versus Closed-Loop

The simplest and typically least accurate method of generating the signal replica is to pick the state on the manifold with the highest replica-received cross-correlation score from the previous DPE receiver iteration and back-calculate the channel parameters for the next timestep from that state. This is considered an *open-loop* implementation, and its accuracy is dependent on both the choice of candidate receiver states and the signal transmission model used in the back-calculation.

In contrast, a *closed-loop* DPE can significantly improve the accuracy of the constructed signal through an additional processing step. A channel-domain tracker will analyze a recent set of samples and, using knowledge of the GNSS navigation message format and the navigation message received previously, the receiver can better determine what navigation data should be received next by working with the estimates of the channel parameters. A navigation-domain tracker, alternatively, will analyze the shape of the cross-correlation function $\mathcal{R}_t$ based the scores of the grid of candidate receiver states to refine the navigation solution to a sub-grid point-level precision.

However, tracking loops are an extra step of processing, and, if the tracking loop produces an erroneous conclusion about the channel parameters, the navigation solution accuracy can be degraded. Channel-domain tracking, as seen when used in two-step approaches, is susceptible to locking onto multipath reflections [36], and navigation-domain tracking may mischaracterize the shape of the manifold. Thus, designing tracking loops that reliably improve accuracy and reject failure modes is an important area of study on its own, as they have a significant impact on the host receiver.

Weighted Average-Based Navigation-Domain Tracking

A weighted average-based navigation-domain tracker will be introduced to demonstrate how such trackers can be used in DPE and their impact on the receiver's localization solutions. Utilized by Ng and Gao in [10] and [23], this signal tracker estimates the navigation solution at the peak of the manifold by weighting all the states on the grid by their replica-received score and computing the average. However, the mathematical effects of such tracking and its achievable accuracy were not the focus of [23] and were not studied

in the work.

This work will follow up on the weighted average-based navigation-domain tracker with a parallelized implementation in Chapter 3, mathematical analysis in Chapter 4, and demonstration in Chapter 5. For the remainder of this work, when referring to the closed-loop DPE receiver implementation, the weighted average-based navigation-domain tracker is being utilized. Alongside this, to analyze the impact of the tracker, as well as more clearly isolate effects arising from the numerical implementation of Equation 2.14, an open-loop implementation will also be considered.

2.4 Conclusion

The advantages of GNSS-based localization using DPE are the result of a single-step, maximum-likelihood approach to GNSS localization, providing an algorithm robust to certain vulnerabilities introduced by the intermediate measurements of two-step methods. Studying the two-step approach in Section 2.1.1 introduces fundamental ideas of GNSSs and leads to the understanding that facilitates the single-step approach of DPE. Derivation of DPE mathematically in Section 2.2 shows the resilience to the vulnerabilities of two-step approaches. And, in order to make a numerical implementation of DPE more computationally efficient and accurate, the ideas of grid-based DPE, batch correlation, and closed-loop operation were introduced.

CHAPTER 3

DPE RECEIVER IMPLEMENTATION

Under the grid-based numerical approach put forth in Section 2.3, principles of parallel computing can be used to improve the efficiency of the DPE algorithm. The signal replica can be constructed in a parallelized manner, the scores of each candidate point can be computed independently, and the manifold can be assessed with parallelized reduction techniques. This makes DPE well-suited for implementation on a GPU, as each candidate state can be assessed on separate processing threads – each thread executing the same instructions, but operating on different memory locations corresponding to different candidate points.

This chapter presents the first of the three areas of contributions of this work: a software-defined DPE receiver implementation developed and tuned for a portable GPU. The software segment of the receiver will be implemented using a proposed seven-task modular decomposition; two of the modules implement a parallelized DPE algorithm and five of the modules perform parallelized supportive tasks to advance the DPE algorithm to the next timestep. The hardware segment of the receiver responsible for generating the antenna voltage samples will be introduced. Lastly, implementation parameters and hardware-specific tuning will specified.

3.1 Software Segment

To provide the researcher with the benefits of abstraction when developing a receiver implementation, the software segment of a software-defined GPS receiver may be decomposed into the seven tasks of List 3.1.

The DPE algorithm itself is implemented in the *Batch Scoring* and *Assess Manifold* modules – a pair of tandem modules which provide the researcher with flexibility through abstraction to make changes to the way in which the

1. **Initialization:** place the grid and load channel parameters
2. **Acquire Samples:** load samples for the current processing iteration
3. **Batch Scoring:** compute the cross-correlation scores for the grid
4. **Assess Manifold:** score the grid points and generate the measurement
5. **Filter:** filter the measurements
6. **Channel Propagation:** update channels and satellite states
7. **Logging:** record results

List 3.1: DPE Task Decomposition

scores are batch-calculated and the manifolds are assessed. The seven tasks of List 3.1 are executed sequentially for a set of samples; the processing of that set constitutes one timestep. Then, a new set of samples is acquired, and the next timestep runs.

For operation on a GPU, the software of this work is developed in NVIDIA's CUDA C/C++ 9.0.252 parallel computing platform [47]. The Nsight IDE provided by NVIDIA is used for profiling the GPU usage [48].

The DPE algorithm is wrapped in a custom C++ object-oriented software architecture developed by the Grace Gao Research Group at the University of Illinois at Urbana-Champaign. Under this architecture, a software-defined receiver is implemented by a *Flow*, consisting of an ensemble of *Modules* which are capable of sharing data by *Ports*. These three software constructs of the *Flow*, the *Module*, and the *Port* are specified in base classes, from which the implementation classes constituting an algorithm are derived. The relation of these three constructs is diagrammatically represented in Figure 3.1.

Following the seven-task decomposition given in List 3.1, the DPEFlow software-defined receiver program was created by developing the seven modules given in Figure 3.2.

3.1.1 Initialization

Developed for this work, the *DPInit* module loads an initialization file. The initialization data consists of a starting PVT state, channel parameters to visible satellites, the GPS time of the internal reference time, and satellite

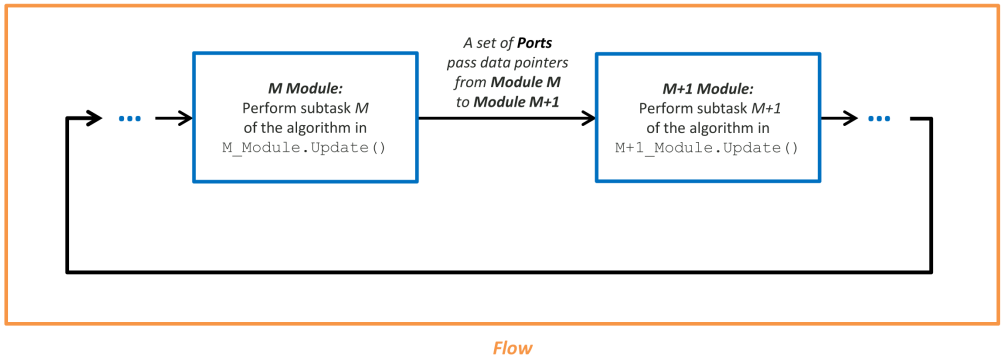

Figure 3.1: Block diagram representation of the software architecture used in this work.

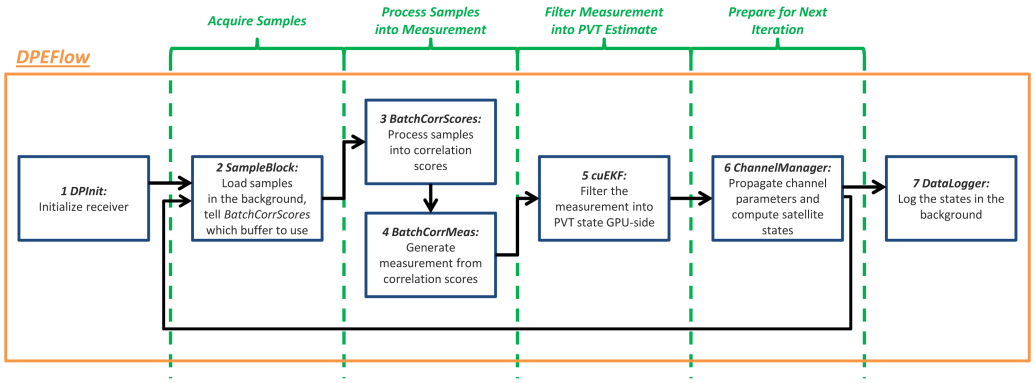

Figure 3.2: DPE algorithm design developed for this work.

ephemerides, and this information is made available to other modules in the flow. The functionality provided by the *DPInit* module is depicted in Figure 3.3.

Given the numerical-solution nature of this DPE algorithm implementation, an initial PVT state is required. This initial state will be the center point of the first timestep's manifold grid and will be the starting state of the measurement filter. For reasons explained in Section 3.1.6, the channel parameters for this initial state are also provided. To provide a global time reference for the clock states, a reference time is loaded as well.

Following the standard for distributing satellite orbit data, a *Receiver Independent Exchange Format* (RINEX) file parser is included in *DPInit* to load the satellite ephemerides and bypass the 12.5 minute ephemeris download-from-satellite time [1]. All ephemerides in the RINEX file with the same issue time are stored in the same instance of a custom class, and all of these instances are accumulated in a vector.

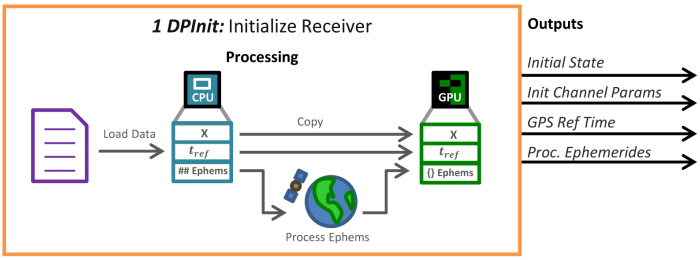

Figure 3.3: *DPInit* module created for this work.

3.1.2 Acquire Samples

SampleBlock is adapted for this work from an existing module developed by the Grace Gao Research Group and is responsible for loading the antenna voltage measurements from a file into GPU memory. As CPU $\rightarrow$ GPU memory transfers can be time-consuming, it is desirable to keep the algorithm waiting on these transfers as little as possible. Thus, *SampleBlock* uses a multi-buffer design to pre-load samples for the next iteration while other modules are processing the samples of the current iteration, parallelizing the memory transfers and minimizing the time each new iteration must wait for samples. This parallelization is diagrammatically represented in Figure 3.4.

According to settings specified in the flow, an array of buffers is created, each one sized to the number of antenna voltage samples to be processed in one iteration of the DPE algorithm. Samples are asynchronously read into the buffers from a source file. Once a buffer is full, it is copied to device memory. Once the device copy completes, that buffer is ready to be processed by the DPE algorithm.

The state of the set of buffers is managed using semaphores. Every time a full buffer is copied to device memory, a semaphore is incremented. Every iteration, this semaphore is checked to ensure there is a buffer available for processing, with an error being returned if not. Also every iteration, the buffer of samples that was processed during the previous iteration is marked for replacement by being added to a second semaphore tracking the number of buffers available to newly loaded samples. This second semaphore regulates the asynchronous sample loading by preventing more samples from being read when no buffers are available.

SampleBlock also provides the sampling frequency f_s and buffer size ΔT to the rest of the DPE algorithm.

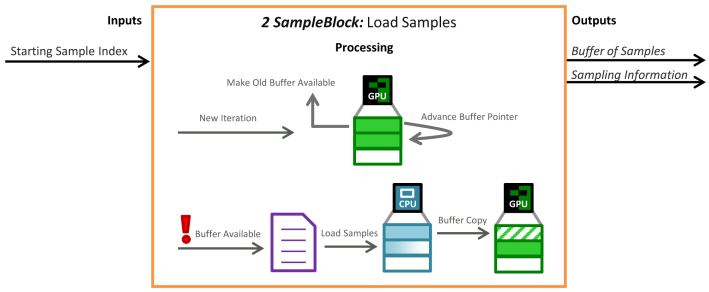

Figure 3.4: *SampleBlock* module adapted for this work.

3.1.3 Batch Scoring

The first of the two tandem modules implementing the DPE algorithm, the *BatchCorrScores* module was developed for this work to compute the correlation scores using the batch correlation technique described in Section 2.3.2. For the position-domain scores, this is accomplished by constructing the replica at the receiver's state estimate then cross-correlating with the received signal using FFTs. For the velocity-domain scores, this is accomplished by wiping off the PRN code and navigation bits from the received signal then computing the frequency components of the remaining carrier wave by FFT.

Components of the replica signals used for cross-correlation and wipe-off can be generated independently of the others, and each sample of each component can be generated independently of the others. Additionally, the FFTs which compute the scores can be performed independently of each other. Thus, the batch correlation step of the DPE algorithm was parallelized according to Figure 3.5. Each block in Figure 3.5 is a sub-task of the batch correlation, and each block is executed on one of three CUDA *streams* which are synchronized by CUDA *events*. This parallelization scheme ensures no computational work in the algorithm is redone.

Since the received signal is the superposition of all transmissions, the correlation score at a given state is also the superposition of all channels (Equation 2.5). Thus, the correlation scores for each channel being tracked may be computed separately. This way, the correlation function approximations are treated as independent look-up tables, and the correlation score for a state is computed as the sum of the score for each channel. For any step in the implementation that produces a different signal for each channel, the signals are stored contiguously in a 1D array – the last sample of one channel's signal

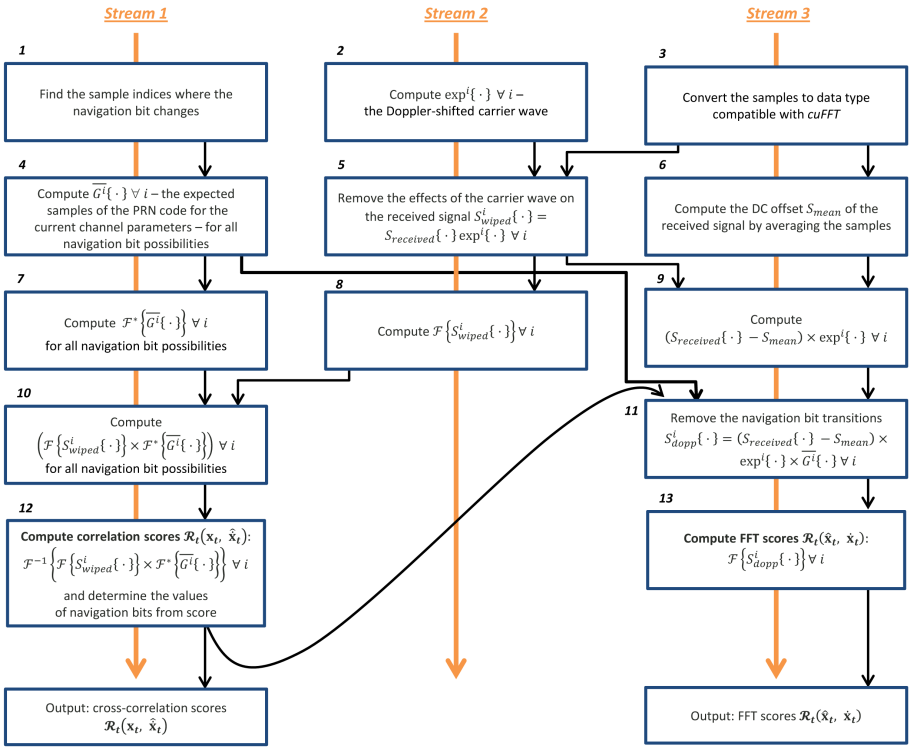

Figure 3.5: Parallelized implementation of Equations 2.17 and 2.18.

being followed by the first sample of the next channel's signal.

In this implementation, replicas of the carrier wave $\exp\{\cdot\}$, PRN code $G^i\{\cdot\}$, and navigation data $D^i_t\{\cdot\}$ are computed in blocks 2, 4, and 7, respectively, using the current estimates of the channel parameters. For the position-time scores, the carrier wave replica is used to wipe off the carrier effects from the received samples in block 5, and the result is Fourier transformed in block 8. This result is multiplied with the complex-conjugate of the FFT of the replica in blocks 7 and 10, and the inverse FFT is performed on the result in block 12. For the velocity-time scores, the DC-offset of the received samples is removed in block 9, and the navigation data replica and the C/A code replica are used to wipe off their effects from the received samples in block 11. The FFT of the result is then taken in block 13.

When constructing the navigation data replica $D^i_t\{\cdot\}$, the next navigation bit may not be known at the time of replica generation. In such a scenario, replicas are constructed for both navigation bit possibilities [25]. Correlation scores for both replicas are computed, and the navigation bit value that gives the higher correlation score for the current best code phase parameter

is selected. This is the cause of the dependence of block 12 on block 11. Additionally, this limits the sample length ΔT to the length of one navigation bit from a computational efficiency standpoint – if multiple navigation bits are included in the sample set, the number of possible values to check will scale exponentially with the number of navigation bits.

By leveraging the manifold decoupling and batch correlation techniques and by structuring the signal processing to not repeat computations, the *BatchCorrScores* module is capable of efficiently determining correlation scores as a function of channel parameters, as shown in Figure 3.6.

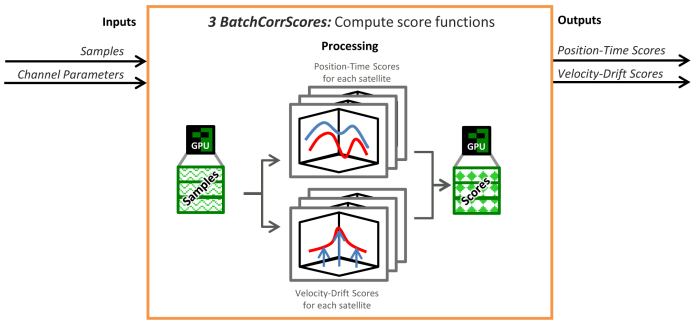

Figure 3.6: *BatchCorrScores* module developed for this work.

3.1.4 Assess Manifold

The second of the two tandem modules implementing the DPE algorithm, the *BatchCorrManifold* module was developed for this work to determine which PVT states will comprise the manifold, compute the cross-correlation score for each point, then generate a measurement from the manifolds.

The position-time and velocity-drift manifolds are computed in parallel on separate CUDA streams. Each thread will compute the score for a state one at a time until all states have been scored. The states are chosen according to a grid initialization function and are oriented along the local *East-North-Up* (ENU) coordinates of the PVT state corresponding to the channel parameter estimates used when generating the batch correlation scores.

Since both the position-time and velocity-drift manifolds are generated by looking up the batch correlation score for each state on the grid, the thread-level algorithm for this underlying concept is represented in Figure 3.7. The

differences between the way in which the two manifolds are generated lie simply in which states, channel parameters, and scores are used.

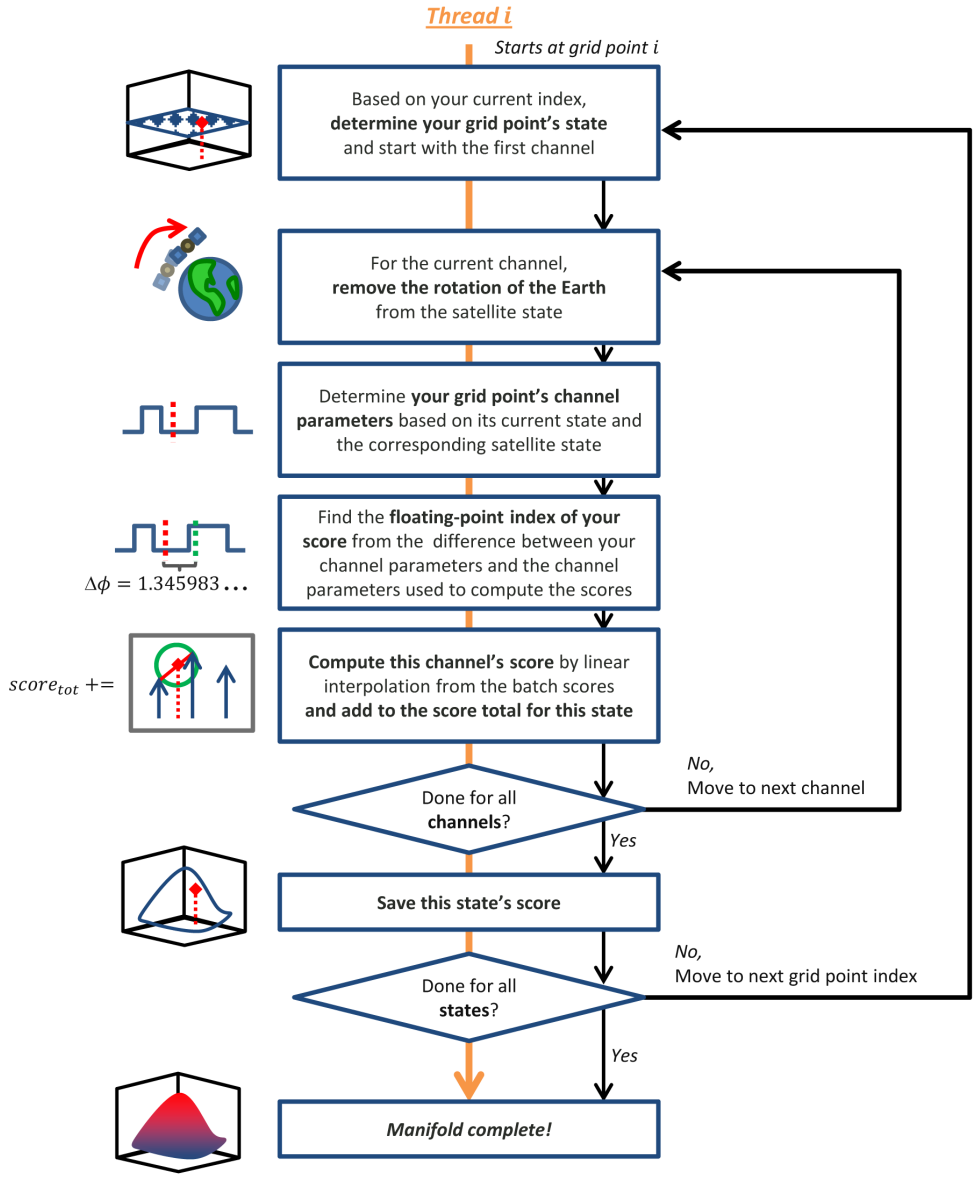

Figure 3.7: Computations performed by each thread when generating manifolds.

Position-Time Manifold

For a state on the position-time manifold $\mathbf{x}$, this is accomplished by finding the difference between the code phase of the point being evaluated $\phi^i_{code,t}(\mathbf{x})$

and the code phase used in the batch correlation $\hat{\phi}^i_{code,t-1}$. Following the definition given in List 2.2, $\phi^i_{code,t}(\mathbf{x})$ is found from the range between the satellite and the state $\mathbf{x}$. The code phase difference is used to linearly interpolate between the batch-computed code correlation scores.

Velocity-Drift Manifold

For a state on the velocity-drift manifold $\dot{\mathbf{x}}$, this is accomplished by finding the difference between the carrier frequency of the point being evaluated $f^i_{carr,t}(\dot{\mathbf{x}})$ and the carrier frequency used in the batch correlation $\hat{f}^i_{carr,t-1}$. Following the definition given in List 2.2, $f^i_{code,t}(\dot{\mathbf{x}})$ is found from the Doppler frequency between the satellite and the state $\dot{\mathbf{x}}$. The carrier frequency difference is used to linearly interpolate between the FFT scores.

Measurement Generation

Once each position-time state $\mathbf{x}_i$ or velocity-drift state $\dot{\mathbf{x}}_\mathbf{i}$ is scored with a value $w_{\mathbf{x}_i}$ or $w_{\dot{\mathbf{x}}_i}$, respectively, the maximum likelihood state may be found. This state is the measurement of the software-defined DPE receiver. In the open-loop implementation, the state with the highest score is chosen from the position-time and velocity-drift grids, as given in Equation 3.1. In the weighted-average closed-loop implementation, all states are weighted by their score and the result is averaged, as given in Equation 3.2.

$$\begin{bmatrix}\mathbf{x}_i & \dot{\mathbf{x}}_\mathbf{i}\end{bmatrix}^\top = \begin{bmatrix}\arg\max_{\mathbf{x}_t} w_{\mathbf{x}_i} & \arg\max_{\dot{\mathbf{x}}_t} w_{\dot{\mathbf{x}}_i}\end{bmatrix}^\top = \boldsymbol{Z}_t \tag{3.1}$$

$$\begin{bmatrix}\mathbf{x}_i & \dot{\mathbf{x}}_\mathbf{i}\end{bmatrix}^\top = \begin{bmatrix}\sum_i \frac{w_{\mathbf{x}_i}\mathbf{x}_i}{w_{\mathbf{x}_i}} & \sum_i \frac{w_{\dot{\mathbf{x}}_i}\dot{\mathbf{x}}_i}{w_{\dot{\mathbf{x}}_i}}\end{bmatrix}^\top = \boldsymbol{Z}_t \tag{3.2}$$

Whether performing a max operation in the open-loop implementation or a weighted addition with the closed-loop implementation, the operations are performed by a parallel reduce – the score for each grid point is added to a running total or compared to the maximum seen within the thread, then the threads exchange their results until one thread is left with the final state, the measurement $\boldsymbol{Z}_t$. $\boldsymbol{Z}_t$ is then passed to the filtering step of the DPE algorithm. The processing pipeline of the *BatchCorrManifold* module is shown in Figure 3.8.

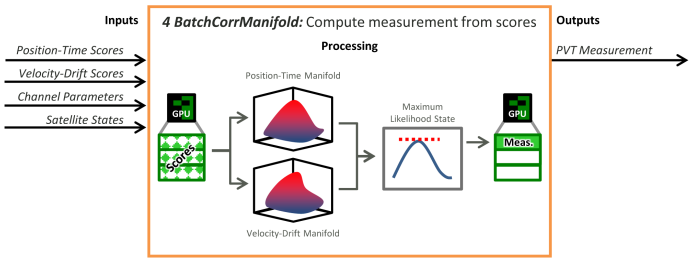

Figure 3.8: *BatchCorrManifold* module developed for this work.

3.1.5 Filtering

Developed for this work, the *cuEKF* module filters the measurements from *BatchCorrManifold*. While the state $\boldsymbol{Y}_t$ is the maximum likelihood state, error in the estimate is still expected due to interference effects on the received signal, discretization errors in the sampling and software-defined processing, and approximations in the DPE algorithm itself. Thus, an *extended Kalman filter* (EKF) is implemented to filter such errors and produce smoother, likely more accurate state estimates by modelling the uncertainty in measurements and integrating those in consideration of previous state estimates and their uncertainty [49, 50].

The EKF accomplishes this by modelling the state estimates at time t as a Gaussian distribution with mean $\boldsymbol{X}_{t|t}$ and covariance $\boldsymbol{P}_{t|t}$. Physical processes that are considered to have random effects often follow a Gaussian distribution [49], and Gaussian distributions may be combined using rather simple linear algebra equations. For a measurement with mean $\boldsymbol{Z}_t$ and noise covariance $\boldsymbol{R}_t$ at time index t, the *measurement update* of an EKF updates the previous best guess of the state $\boldsymbol{X}_{t|t-1}$ and its covariance $\boldsymbol{P}_{t|t-1}$ by multiplying the two distributions:

$$\begin{aligned} \boldsymbol{Y}_t &= \boldsymbol{Z}_t - h(\boldsymbol{X}_{t|t-1}) \\ \boldsymbol{K}_t &= \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t^\top(\boldsymbol{H}_t\boldsymbol{P}_{t|t-1}\boldsymbol{H}_t^\top + \boldsymbol{R}_t)^{-1} \\ \boldsymbol{X}_{t|t} &= \boldsymbol{X}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{Y}_t \\ \boldsymbol{P}_{t|t} &= (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H}_t)\boldsymbol{P}_{t|t-1} \end{aligned} \tag{3.3}$$

where $h(\cdot)$ is the function mapping the state to the measurement and $\boldsymbol{H}_t$ is the Jacobian of $h(\cdot)$ assessed at the state $\boldsymbol{X}_{t|t-1}$. Following the measurement

update, the EKF then predicts the state and covariance for the next iteration based on the state's expected motion between iterations:

$$\begin{aligned} \boldsymbol{X}_{t+1|t} &= f(\boldsymbol{X}_{t|t}) \\ \boldsymbol{P}_{t+1|t} &= \boldsymbol{F}_t \boldsymbol{P}_{t|t} \boldsymbol{F}_t^\top + \boldsymbol{Q}_t \end{aligned} \tag{3.4}$$

where $f(\cdot)$ is the function describing the expected motion of the state, $\boldsymbol{F}_t$ is the Jacobian of $f(\cdot)$ assessed at the state $\boldsymbol{X}_{t|t}$, and $\boldsymbol{Q}_t$ is the noise associated with the expected motion.

Each iteration, the EKF performs the measurement update to compute $\boldsymbol{X}_{t|t}$ followed by the prediction update to compute $\boldsymbol{X}_{t+1|t}$. These states are made available to the rest of the DPE algorithm.

To eliminate the need for CPU $\leftrightarrow$ GPU data transfers in this module, the EKF was implemented on the GPU using the *CUDA Basic Linear Algebra Subroutines* (cuBLAS) library [51], as shown in Figure 3.9.

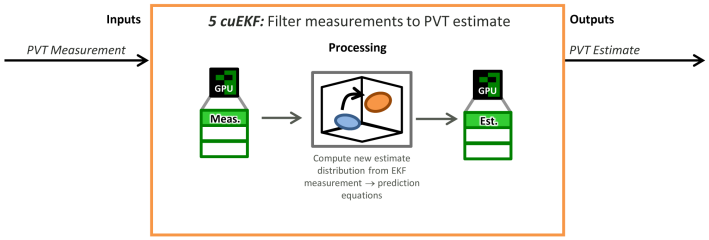

Figure 3.9: *cuEKF* module developed for this work.

3.1.6 Channel Propagation

Developed for this work, the *ChannelManager* module maintains current estimates of the channel parameters for all satellites being tracked. These parameters are used by both the *BatchCorrScores* and *BatchCorrManifold* modules and, as they describe the relationship between the receiver state and the state of each satellite, they are updated using $\boldsymbol{X}_{t|t}$. Further, an estimate of the transmission time of the next sample set is found, which is used to compute the corresponding satellite states. This functionality is shown in Figure 3.10.

The channel frequency parameters $f^i_{code,t}$ and $f^i_{carr,t}$ are computed according to their formulae as given in List 2.2. However, the channel phase parameters

$\phi^i_{code,t}$ and $\phi^i_{carr,t}$ are propagated forward from their previous estimates using the estimates of the corresponding frequency:

$$\begin{aligned}\phi^i_{code,t} &= (f^i_{code,t}\Delta T + \phi^i_{code,t-1}) \bmod L_{C/A} \\ \phi^i_{carr,t} &= (f^i_{carr,t}\Delta T + \phi^i_{carr,t-1}) \bmod 1.0\end{aligned} \tag{3.5}$$

Instead of the back-calculation and time-propagation equations for updating the channel parameter estimates, channel-domain tracking loops could be employed. Though beyond the scope of this work, the software architecture was designed to accommodate channel-domain tracking loops in this module.

Following the phase parameter update, the *ChannelManager* module then computes the satellite states for the current set of channel parameters. Since the code phase estimate is computed every iteration, a count is maintained of the code periods elapsed since the time-identifier bits were transmitted by each satellite. This allows the transmission time of the sample set to be accurately estimated. As the ephemeris for a satellite is the parameterization of its trajectory, a series of orbital mechanics equations is then solved to compute the PVT state for each satellite.

The channel parameters and the corresponding PVT for each satellite are made available to the next iteration of the DPE algorithm.

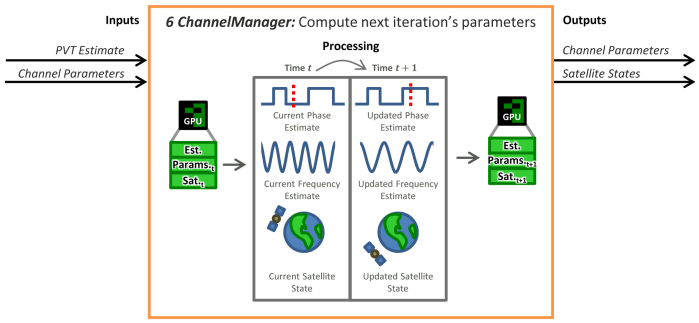

Figure 3.10: *ChannelManager* module developed for this work.

3.1.7 Logging

The *DataLogger* module is adapted for this work from a core functionality module in the group's software architecture. Data on the GPU from the current iteration is asynchronously copied to the CPU and recorded in a file.

Typically, the state estimate $\boldsymbol{X}_{t|t}$ is of greatest interest, though other data may be recorded using this module, as well. Values are stored in a comma-separated format for easy analysis. The parallelization of this functionality is shown in Figure 3.11.

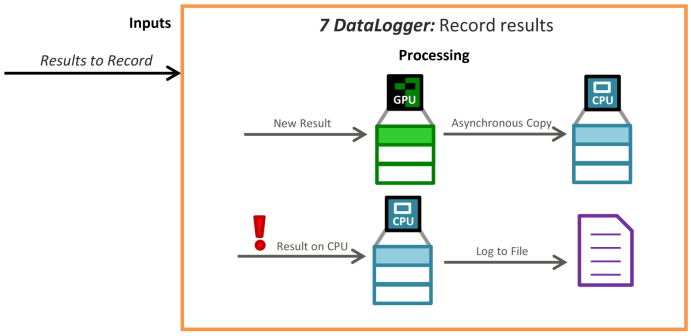

Figure 3.11: *DataLogger* module adapted for this work.

3.2 Hardware Segment

To evaluate the numerical DPE implementation, Chapter 5 processes simulated and real-world results using the software presented in Section 3.1. The host which generates and processes the datasets is presented in this section. The hardware segment consists of a GPS signal source, a signal sampler, and the GPU to process the samples.

3.2.1 Signal Source

For the simulated datasets, the GPS signals originate from the National Instruments GPS Simulation Toolkit [52]. This simulator is capable of generating the theoretical transmissions from up to 12 GPS satellites with adjustable amplitudes. By specifying the receiver position-time state in National Instruments LabVIEW, the simulator will generate the composite received signal for that state. The receiver state may also be programmed to follow a specified path over a period of time. By simulating the received signal using this hardware, the accuracy of the DPE implementation can be evaluated by comparing the results directly to the known receiver states and with only known effects on the received signal. After a simulated downconversion from

the GPS L1-band carrier frequency of $f_{L1} = 1575.42$ MHz [1] to 0 Hz, the samples are generated at a frequency of $f_s = 2.5$ MHz. This downconversion means that the frequency of the carrier wave in the received samples for satellite i will only consist of the Doppler frequency component $f^i_{carr,t}$.

3.2.2 Sampling

The receiver's voltage samples are acquired through the use of an Ettus Research *Universal Software Radio Peripheral* (USRP) [53]. The USRP is an radio front-end that will sample at a specified frequency; for this work, the sampling frequency $f_s = 2.5$ MHz. The samples are triggered by a Microsemi SA.45s *Chip-Scale Atomic Clock* (CSAC) that provides a clocking signal of 10 MHz $\pm$ 5×10^{-10} Hz to the USRP [54, 55]. The USRP also digitally down-converts the received signal from the GPS L1-band carrier frequency of $f_{L1} = 1575.42$ MHz [1] to 0 Hz, meaning the frequency of the carrier wave in the received samples for a satellite i will only consist of the Doppler frequency component $f^i_{carr,t}$. Additionally, during the digital down-conversion, the USRP converts the signal received into its *in-phase* (I) and *quadrature* (Q) form, resulting in one sample consisting of both the I and Q signal components.

The samples produced by the USRP are sent over Ethernet to a computer to be recorded in a data file using the USRP hardware drivers provided by Ettus Research. The I and Q components are 16 bits each, meaning one sample is 32 bits or 4 bytes.

3.2.3 GPU

The GPU used is an NVIDIA Jetson TX2 [56]. The TX2 consists of 2 streaming multiprocessors, each supporting up to a total of 2048 threads [57]. CUDA kernels – functions responsible for executing code on the GPU – launch a specified number of blocks, each block executing a specified number of threads. Each block can support up to 1024 threads, though that number may need to be tuned smaller depending on how many registers a thread uses, as each block has a maximum of 32768 registers available to it [57]. Thus, optimizing CUDA kernel launches for the TX2 entails choosing the

number of blocks and threads to launch for each kernel so that the GPU has as many threads as possible active at a time while staying under the register cap for each kernel block launched.

The TX2 has 8GB of RAM available to the CPU and GPU for memory allocation [57]. As sample sets, intermediate processing results, and the manifolds are all stored as arrays, this places a limit on the theoretical maximum allowable sizes of those steps of the implementation. However, 8GB is sufficient for the implementation considered in this work.

3.2.4 Hardware Integration

The hardware segment consists of the pipeline shown in Figure 3.12.

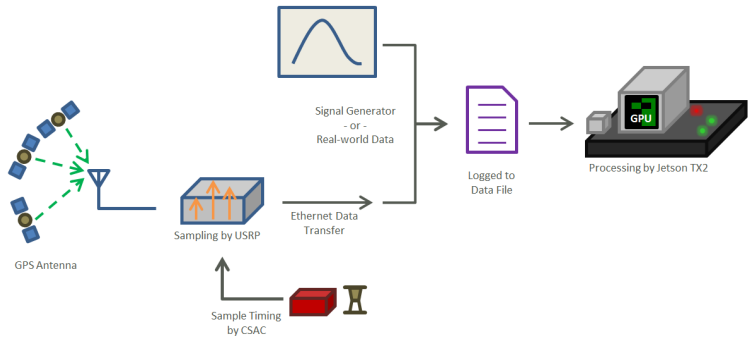

Figure 3.12: The hardware pipeline of the DPE receiver for this work.

3.3 Receiver Implementation

This section will specify the configuration of the software segment. The settings selected are an example tuning for the receiver implementation chosen to balance localization accuracy and computational efficiency.

3.3.1 PVT State Convention

PVT states are defined with position-velocity states $\mathbf{p}$ and $\dot{\mathbf{p}}$ in the ECEF coordinate frame and time states δt and $\dot{\delta t}$ as the negative offset with respect to an internal reference time kept by the DPE algorithm based on the

number of samples received. Additionally, δt and $\dot{\delta t}$ are stored computationally as the distances $c\delta t$ and $c\dot{\delta t}$ to be of comparable magnitude to the ECEF $\mathbf{p}$ and $\dot{\mathbf{p}}$ states. Though not a requirement, this distance form of the time states is often more convenient for computations, particularly in linear algebra operations for optimization and filtering.

3.3.2 DPInit

The DPE algorithm is initialized by the results of a 60-second scalar tracking algorithm. The scalar tracking algorithm acquires lock of the available satellites to provide the DPE algorithm with channel parameters and achieved a reasonably accurate PVT state estimate such that the true receiver state is within the DPE receiver's manifold when initialized from this estimate.

3.3.3 SampleBlock

Samples are loaded from a datafile in sets of $\Delta T = 20\,\text{ms}$ long. This sample set length was chosen to match the length of one navigation bit so that the value of only one navigation bit needs to be determined by the *BatchCorrScores* module. While more than one navigation bit can occur in the sample set if the Doppler effect increases the received code frequency ($f^i_{code,t} > 0$), the Doppler effect is small ($f^i_{code,t} \ \pm 10\,\text{kHz}$) compared to the nominal code frequency ($f_{C/A} = 1.023\,\text{MHz}$). Thus, only a few samples could contain the second navigation bit, and having the value of the second navigation bit wrong would contribute minimally to the overall batch correlation score.

Each sample set consists of $f_s \times \Delta T = 2.5\,\text{MHz} \times 20\,\text{ms} = 50 \times 10^3$ samples. With each sample being 4 bytes, each buffer requires 200 kb. A total of 32 buffers are allocated for the rolling buffer implementation, taking a total of 6.4 Mb of device memory for this module.

3.3.4 BatchCorrScores

The majority of the steps in the *BatchCorrScores* module follow the number of samples in the sample set currently being processed. The replica $G^i\{\cdot\}$

used in Equation 2.17 to compute position-time cross-correlation is 50×10^3 samples to match the sample set, since cross-correlation is a sample-by-sample comparison of two signals. Thus, the FFTs used to transform the sample set and the replica $G^i\{\cdot\}$ as well as the following *inverse fast Fourier transform* (IFFT) are both 50×10^3-point transforms. A $G^i\{\cdot\}$ replica must be constructed and Fourier transformed for each tracked satellite i, then the IFFT must be taken for each tracked satellite to generate that channel's cross-correlation scores.

For the velocity-drift scores, however, since the cross-correlation scores are approximated by the FFT coefficients, there is flexibility in the choice of number of FFT points. By extending the number of points P in $\exp_{rec}\{\cdot\}$ of Equation 2.18 to a power of 2, the execution time for the FFT is reduced to $\mathcal{O}(P\log(P))$ [58]. If the new elements are added by zero-padding the end of the sample set, the precision of the FFT will also be increased by sinc interpolation. This precision increase will continue as more zeros are added to the end of the sample set. Empirically, a 524288-point FFT was chosen, as this is 50×10^3 rounded to the nearest power of two and multiplied by 8. The multiplication by 8 was empirically chosen to minimize the linear interpolation distance performed by the velocity-drift manifold by having the FFT perform a more accurate sinc interpolation, instead.

3.3.5 BatchCorrManifold

Three position-time and velocity-drift grids were used by *BatchCorrManifold* when evaluating the DPE receiver algorithm: *Spread Grid 7m*, *RNGrid 7m*, and *Spread Grid 6m*.

Spread Grid 7m

The candidate points of the *Spread Grid 7m* manifold were chosen with a higher density towards the center of the grid and lower density towards the extremities of the grid. This provides higher resolution for good starting estimates and a wide coverage to increase the range of poor estimates tolerable by the receiver. The candidate spacing density is shown in Figure 3.13 with a 1D slice of the position dimension grid.

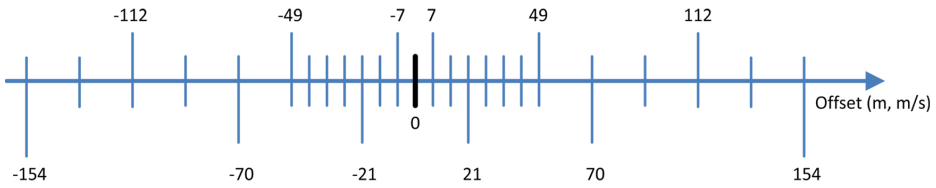

Figure 3.13: A 1D slice of the *Spread Grid 7m*.

As shown in Figure 3.13, the spacing increases from 7 m between points to 21 m between points after the seventh point from the center of the grid. Table 3.1 provides the specifics for the spacing of the candidate points in each dimension.

Table 3.1: *Spread Grid 7m* manifold dimensions

Dim.	**Axis**	**Span**	**Inner Spacing**	**Outer Spacing**	**Points per Dim.**
Position	East, North, Up	$\pm 154\,\mathrm{m}$	$7\,\mathrm{m}$	$21\,\mathrm{m}$	25
Clock Bias	Bias Range $c\delta t$	$\pm 154\,\mathrm{m}$	$7\,\mathrm{m}$	$21\,\mathrm{m}$	25
Velocity	East, North, Up	$\pm 15.4\frac{\mathrm{m}}{\mathrm{s}}$	$0.7\frac{\mathrm{m}}{\mathrm{s}}$	$2.1\frac{\mathrm{m}}{\mathrm{s}}$	25
Clock Drift	Drift Range $c\dot{\delta t}$	$\pm 5.5\frac{\mathrm{m}}{\mathrm{s}}$	$0.25\frac{\mathrm{m}}{\mathrm{s}}$	$0.75\frac{\mathrm{m}}{\mathrm{s}}$	25

As previous open-loop DPE implementations have shown position-domain localization accuracy on the order of 50 m [12, 26], this spacing ensures that 49 m in every direction around the estimate chosen by the receiver is covered by the higher-resolution spacing of points. Additionally, with the total coverage extending to 154 m when including the lower-resolution spacing of points, initialization perturbations up to 80 m – the subject of the experiments in Section 5.1 – will still have maximum likelihood candidate points contained within the grid even when compounded with the effects of the 50 m-order open-loop DPE accuracy.

RNGrid 7m

In comparison with the structured *Spread Grid 7m*, the *RNGrid 7m* was generated to cover the same position-time domain space with the same number of points as the *Spread Grid 7m*, but using largely randomly-selected candidate points.

A total of $(25^4 - 17)$ points of the 25^4 points were chosen as 4D uniform

random variables, *i.i.d.* with respect to each dimension. All candidate grid points were chosen *i.i.d.* with respect to each other.

The other 17 points in the position-time domain grid were chosen explicitly. Sixteen of these were the 16 vertices of a 4D hypercube spanning ± 154 m in each dimension to ensure *RNGrid 7m* covers the same position-time space as *Spread Grid 7m*. The one other explicitly chosen grid point was a candidate at the center of the grid so that the algorithm could remain at the same estimate over multiple timesteps.

A 3D slice of the *RNGrid 7m* used in this work is shown in Figure 3.14.

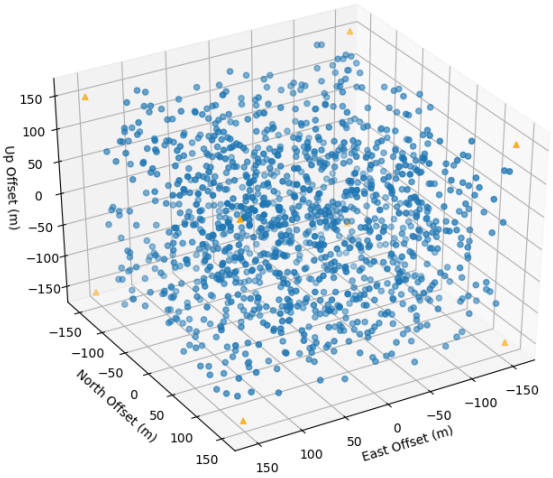

Figure 3.14: A 3D slice of the *RNGrid 7m* for grid points with clock bias between 153 and 154 m. The randomly chosen grid points are represented as blue circles, and 8 of the 16 vertex points that are visible in this plot are represented as orange triangles.

As Section 5.1 uses this grid to study position-time domain effects, the velocity-drift grid of the *RNGrid 7m* configuration was kept identical to the *Spread Grid 7m*. The specifications of the grid are provided in Table 3.2.

The random selection of grid points allows the effects introduced by the structure in the choice of states in the *Spread Grid 7m* to be easily identified. Thus, the justification for the domain covered by the *RNGrid 7m* follows from that of *Spread Grid 7m*. While Table 3.2 shows that the average spacing is now 12 m between points, the *7m* moniker in *RNGrid 7m* was used to signify the comparative relationship with *Spread Grid 7m*.

Table 3.2: *RNGrid 7m* manifold dimensions

Dim.	**Axis**	**Span**	**Mean Spacing**		**Points per Dim.**
Position	East, North, Up	$\pm 154\,\mathrm{m}$	$12.3\,\mathrm{m}$		25
Clock Bias	Bias Range $c\delta t$	$\pm 154\,\mathrm{m}$	$12.3\,\mathrm{m}$		25
Dim.	**Axis**	**Span**	**Inner Spacing**	**Outer Spacing**	**Points per Dim.**
Velocity	East, North, Up	$\pm 15.4\frac{\mathrm{m}}{\mathrm{s}}$	$0.7\frac{\mathrm{m}}{\mathrm{s}}$	$2.1\frac{\mathrm{m}}{\mathrm{s}}$	25
Clock Drift	Drift Range $c\dot{\delta}t$	$\pm 5.5\frac{\mathrm{m}}{\mathrm{s}}$	$0.25\frac{\mathrm{m}}{\mathrm{s}}$	$0.75\frac{\mathrm{m}}{\mathrm{s}}$	25

Spread Grid 6m

The candidate points of the *Spread Grid 6m* manifold were chosen in the same manner as *Spread Grid 7m*, but with a 6-m spacing for the high-density points and a corresponding adjustment to the lower-density points. This size reduction was chosen to better suit the mobile dataset analyzed in Section 5.2. The candidate spacing density is shown in Figure 3.15 with a 1D slice of the position dimension grid:

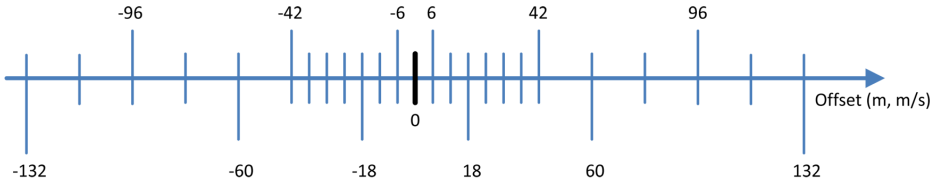

Figure 3.15: A 1D slice of the *Spread Grid 6m*.

As shown in Figure 3.15, the spacing increases from 6 m between points to 18 m between points after the seventh point from the center of the grid. Table 3.3 provides the specifics for the spacing of the candidate points in each dimension.

Table 3.3: Spread Grid 6m manifold dimensions

Dim.	**Axis**	**Span**	**Inner Spacing**	**Outer Spacing**	**Points per Dim.**
Position	East, North, Up	$\pm 132\,\mathrm{m}$	$6\,\mathrm{m}$	$18\,\mathrm{m}$	25
Clock Bias	Bias Range $c\delta t$	$\pm 132\,\mathrm{m}$	$6\,\mathrm{m}$	$18\,\mathrm{m}$	25
Velocity	East, North, Up	$\pm 13.2\frac{\mathrm{m}}{\mathrm{s}}$	$0.6\frac{\mathrm{m}}{\mathrm{s}}$	$1.8\frac{\mathrm{m}}{\mathrm{s}}$	25
Clock Drift	Drift Range $c\dot{\delta}t$	$\pm 5.5\frac{\mathrm{m}}{\mathrm{s}}$	$0.25\frac{\mathrm{m}}{\mathrm{s}}$	$0.75\frac{\mathrm{m}}{\mathrm{s}}$	25

With the receiver in the mobile dataset of Section 5.2 moving approximately 3 m between iterations but with no initialization perturbations, this grid size offers a slightly higher resolution in exchange for slightly less range.

3.3.6 cuEKF

While the *cuEKF* module was developed and verified for this work, it was not utilized in the experimental localization results in order to better isolate and study the effects of the numerical implementation of the DPE receiver.

3.4 Conclusion

This chapter introduced the DPE receiver implementation developed specifically for this work. A seven-task decomposition of the DPE algorithm was presented in List 3.1 and implemented as the seven *Modules* described in Section 3.1. Considerations for efficient GPU processing were also incorporated in the modules. The receiver implementation processes pre-recorded GPS datasets which were acquired through a simulated or real-world radio front-end which downconverts the received signal and samples at a rate of $f_s = 2.5\,\text{MHz}$. An NVIDIA Jetson TX2 is used as the host hardware for the software implementation, and the parallelized *Modules* are tuned to this hardware. Three DPE manifold grids were also specified.

CHAPTER 4

ANALYSIS AND DESIGN INSIGHTS FOR A NUMERICAL DPE IMPLEMENTATION

While the benefits of DPE can be proven analytically, the numerical implementation of DPE will require approximations that may degrade these benefits. Following the DPE algorithm and techniques presented in Chapter 2, and given the DPE-based receiver implementation specified in Chapter 3, this chapter presents the second of the three areas of contributions of this work: new analysis of certain effects present in the localization results of a DPE-based receiver implementation.

Three specific effects will be studied in this chapter. First, a state coupling effect arising from the nature of satellite-based navigation will be derived for DPE. Second, a position-domain signal tracker using weighted averaging will be justified. Third, a limit on the open-loop DPE accuracy based on the sampling frequency will be determined.

4.1 Satellite Geometry and DPE

Grid-based DPE also offers the potential to provide improved resilience to errors as compared to iterative methods. This improvement arises from a coupling between the accuracy of the estimates of clock bias and the accuracy of the estimates of the receiver's vertical state. While [59] shows this effect for two-step receivers, this work will show that the same effect occurs in one-step receivers.

4.1.1 Linear Model for DPE Maximum Likelihood

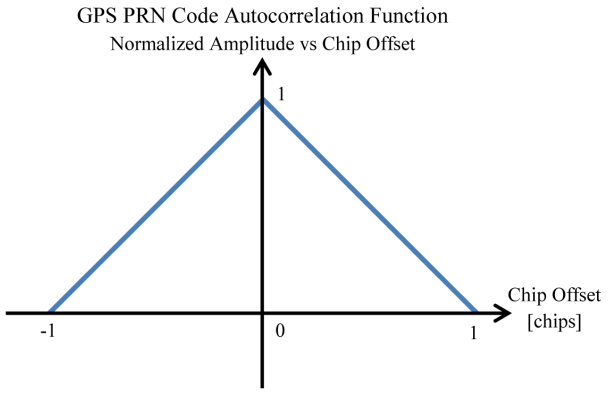

Figure 4.1: The autocorrelation score as a function of code chips delayed for GPS PRN codes.

As shown in Figure 4.1, assuming the replica and received signals have matched frequencies, GPS PRN codes have a cross-correlation function proportional to the difference in code phase signals over a specific range of code phases. This relation can be modeled as shown in Equation 4.1, with units expressed in number of chips of the PRN code sequence:

$$score(\Delta\phi^i_{code,t}) = \begin{cases} 1 - |\Delta\phi^i_{code,t}|, & 0 \leq |\Delta\phi^i_{code,t}| \leq 1 \\ 0, & \Delta\phi^i_{code,t} > 1 \end{cases} \tag{4.1}$$

where $\Delta\phi^i_{code,t} = \phi^i_{code,t_{received}} - \phi^i_{code,t_{replica}}(\mathbf{x}) + \epsilon^i_{\phi^i}$ and:

- $\phi^i_{code,t_{replica}}(\mathbf{x}) = ||\mathbf{x}^i - \mathbf{x}||$ is the code phase estimate computed by the receiver for state $\mathbf{x}$.
- $\phi^i_{code,t_{received}} = ||\mathbf{x}^i - \mathbf{x}_{actual}||$ is the code phase of the received signal at the receiver's true state $\mathbf{x}_{actual}$.
- $\epsilon^i_{\phi^i}$ is a collection of the errors in the replica and received signals, and is responsible for any shift in the maximum score away from $\phi^i_{code,t_{replica}}(\mathbf{x}) - \phi^i_{code,t_{received}}$.
- $\mathbf{x} = [\mathbf{p} \quad \delta t]^\top$ can be described in terms of the receiver's true state by $\mathbf{p} = \mathbf{p}_{actual} - \Delta\mathbf{p}$ and $\delta t = \delta t_{actual} - \Delta\delta t$.

Substituting the definitions above into Equation 4.1 and using a Taylor series approximation of the vector norm gives Equation 4.2:

$$\begin{aligned}\Delta\phi^i_{code,t} &= ||\mathbf{p}^i - \mathbf{p} - \Delta\mathbf{p}|| - ||\mathbf{p}^i - \mathbf{p}|| - (\delta t_{actual} - \delta t) + \epsilon^i_{\phi^i} \\ &\approx \frac{(\mathbf{p}^i - \mathbf{p})}{||\mathbf{p}^i - \mathbf{p}||}\Delta\mathbf{p} + \Delta\delta t + \epsilon^i_{\phi^i} \\ &= [(-\mathbf{1}^i)]^\top \Delta\mathbf{p} + \Delta\delta t + \epsilon^i_{\phi^i}\end{aligned} \tag{4.2}$$

According to the substitution in Equation 4.2, $[-\mathbf{1}^i]^\top = \frac{(\mathbf{p}^i - \mathbf{p})}{||\mathbf{p}^i - \mathbf{p}||}$. Then, the solution to the objective function of Equation 2.14 is found by maximizing the sum of the cross-correlation scores for all the tracked channels, as given in Equation 4.3:

$$\begin{aligned}\hat{\boldsymbol{X}} &= \arg\max_{\boldsymbol{X}} \sum_i (1 - |\Delta\phi^i_{code,t}|)^2 \\ &= \arg\min_{\boldsymbol{X}} \sum_i |\Delta\phi^i_{code,t}|^2 \\ &= \arg\min_{\boldsymbol{X}} \left| \boldsymbol{\Gamma} \begin{bmatrix} \Delta\boldsymbol{p} \\ \Delta\delta t \end{bmatrix} \right|^2\end{aligned} \tag{4.3}$$

with the satellite geometry matrix $\boldsymbol{\Gamma} = \begin{bmatrix} (-\mathbf{1}^1) & 1 \\ (-\mathbf{1}^2) & 1 \\ \vdots & \vdots \\ (-\mathbf{1}^M) & 1 \end{bmatrix}$.

4.1.2 Geometric Impact

A comparison between the satellite-geometric formulation of DPE given in Equation 4.3 and the formulation of two-step approaches given on page 204 of [1] shows that satellite geometry plays the same role in both DPE and two-step-based receivers. Thus, effects that occur in the receiver due to the satellite geometry in two-step-based receivers will occur in DPE-based receivers as well.

The aforementioned resilience to noise in grid-based DPE arises from a difference in satellite geometry between the horizontal plane and the vertical axis. In the horizontal plane, a receiver will be able to receive signals from satellites from any direction, provided there is no interference from structures

or terrain. This means that, for a state slightly shifted from the actual receiver position in the horizontal plane, the code phase errors for all channels will be either positive or negative. However, as the surface of the Earth blocks signals from satellites that are below the horizon for a near-Earth receiver, the receiver will only receive satellites which are above it. A state slightly shifted from the actual receiver position in the vertical dimension will have code phase errors for all channels with the same sign. Such a shift in all the channels is very similar to a clock bias shift, which leads to a strong coupling between the vertical estimates and the clock bias estimates, as Misra shows in [59]. And, to confirm this, the same demonstration done by Misra in [59] for two-step receivers will be conducted for DPE in Chapter 5.

This coupling manifests itself numerically in the *dilution of precision* (DOP) matrix $DOP = (\mathbf{\Gamma}^\top \mathbf{\Gamma})^{-1}$. This matrix originates from the covariance of the error of the position estimate, as shown in Equation 4.4 [1]:

$$Cov \begin{bmatrix} \Delta \boldsymbol{p} \\ \Delta \delta t \end{bmatrix} = \sigma^2 (\mathbf{\Gamma}^\top \mathbf{\Gamma})^{-1} = \sigma^2 DOP \tag{4.4}$$

As the DOP matrix is a covariance matrix, the diagonal terms are proportional to the variance in a given direction while the off-diagonal terms are proportional to the covariance between two directions. If the position states of the DOP matrix are expressed in terms of receiver-referenced ENU coordinates, GPS receivers will typically have much higher variance in the vertical direction than the other directions, due to the effects described above. Furthermore, also due to the effects described above, the covariance terms between the vertical dimension and the clock bias dimension will typically be significant, while all other off-diagonal terms are not significant, indicating a strong coupling between the dimensions. This means that the error in the vertical estimate will typically be directly proportional to the error in the clock bias estimate; for GNSSs, the vertical error will be directly proportional to the clock bias error by factor between 1 and 2. And, as Misra demonstrates in [59], if the receiver has an accurate model of the error in its clock, the coupling can be leveraged to significantly reduce the error in the receiver's vertical estimates.

4.1.3 DPE Receiver Design Insights

Position-time manifold grids can be designed to leverage the coupling between vertical and horizontal estimates. If the DPE algorithm begins in a low-confidence state, such as one acquired through coarse acquisition, a grid with large, independent variations in the vertical and clock bias states should be used. The variation in such a grid would increase the likelihood of finding a state with offsetting errors in the vertical and clock bias dimensions. If the DPE algorithm begins in a state of reasonable confidence, such as one acquired through scalar tracking or previous DPE timesteps, a grid with vertical and clock bias states that vary together should be used. This would allow a grid of the same size to refine the estimate by searching a smaller space at a higher resolution.

An uncertainty metric could be used to adjust which grids are used during operation, as well. When there is greater variance in the time and clock bias states, the variation between the vertical and clock bias states could increase. The state covariance from a Kalman filter or the sharpness of the manifold in the previous timestep could be used as such metrics.

4.2 Grid-based Signal Tracking

In GNSSs, signal tracking is a part of the receiver algorithm responsible for maintaining highly accurate estimates of the channel parameters [1]. This idea of tracking can also be utilized by DPE in the position domain to improve the receiver state estimate. The grid points themselves are samples of the replica-received correlation function at different code phases, and, with the points being the sum of the scores of each channel at a certain code phase, a step that better resolves the peak of the manifold will perform vector tracking. Weighted average-based position-domain vector tracking was utilized by Ng and Gao in [10] and [23]; however, its mathematical justification was not explored in those works. The following justification of weighted average-based position-domain vector tracking was developed for this work.

4.2.1 Analytical Model of Manifold Scoring

The auto-correlation function of GPS PRN codes has the form shown in Figure 4.1. Assuming perfect signal transmission and reconstruction, if the DPE receiver's estimate $\mathbf{x}$ is exactly at the ground truth location $\mathbf{x}_{actual}$, all tracked channels should have this form and lie on top of each other, as shown in Figure 4.2 (staggered for visual clarity).

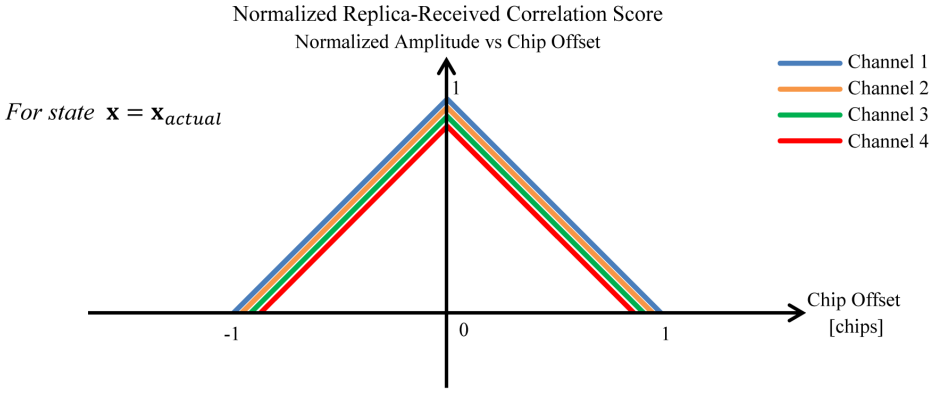

Figure 4.2: Theoretical shape of correlation functions of multiple tracked channels at the ground truth location of a receiver.

However, if the DPE receiver's estimate is slightly off the ground truth location by some perturbation $\Delta\mathbf{x}$, the correlation scores of the channels will begin to move away from the center of the plot based on the estimate-receiver and satellite-receiver relative geometry, as shown in Figure 4.3.

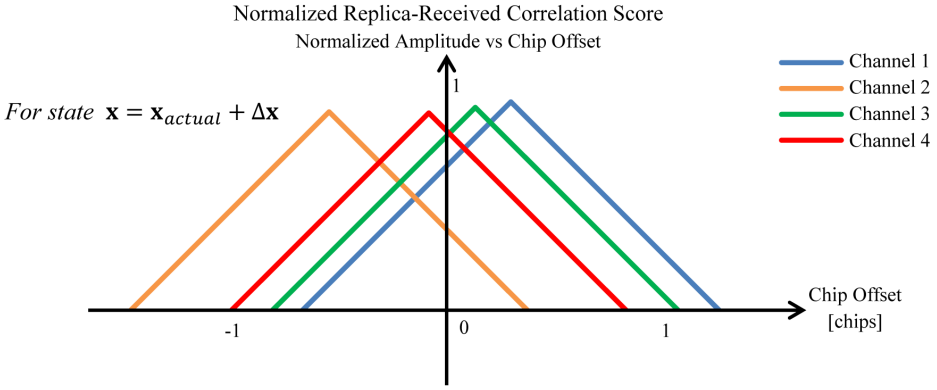

Figure 4.3: The replica-received correlation plots for a receiver state estimate nearby – but not equal to – the ground truth location of the receiver.

The chip offsets on the x-axis of Figure 4.3 are referenced with respect to the grid point at state $\mathbf{x}$. For clearer visualization of how the cross-correlation functions shift based on the state perturbation applied, a range of chip offsets are shown in Figure 4.3. However, the grid points cannot compute the true shape of the function; only the cross-correlation scores at zero chip offset from itself – the value of the cross-correlation function on the y-axis – can be computed for a given grid point.

Consider now two states co-linear with the ground truth and perturbed at $\Delta\mathbf{x}$ from the ground truth in opposite directions. Assuming the three points have the same satellite geometry due to their proximity, the state perturbed by $-\Delta\mathbf{x}$ will have a mirror image replica-received correlation plot with respect to the state perturbed by $\Delta\mathbf{x}$. In other words, the correlation plots will shift the same amounts in opposite directions, as shown in Figure 4.4.

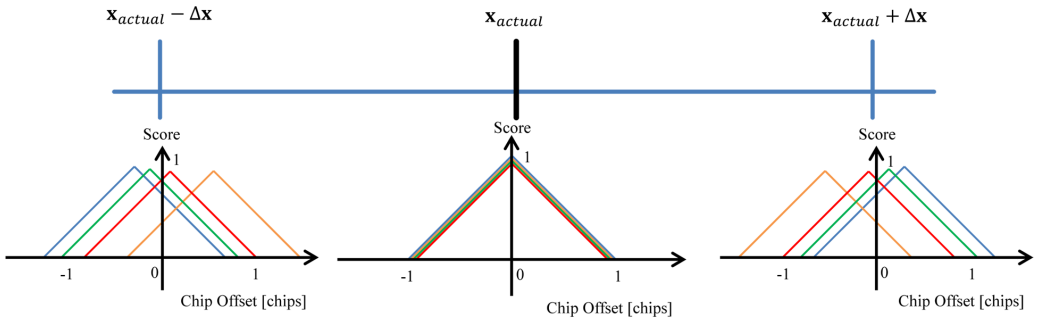

Figure 4.4: The replica-received correlation plots for two receiver state estimates the same distance away in opposite directions from the ground truth location of the receiver.

So, for all possible shifts $\Delta\mathbf{x}$ in a given direction, until the shifts are greater than one chip for any channel, and assuming all points have the same satellite geometry across the grid, the replica-received correlation plots for each channel will shift at a constant rate. Due to the piecewise linearity of the cross-correlation function, a constant-rate shift results in a linear change in the score. Thus, the composite replica-received correlation plot for M channels has the same form as the auto-correlation function for one channel, as shown Figure 4.5.

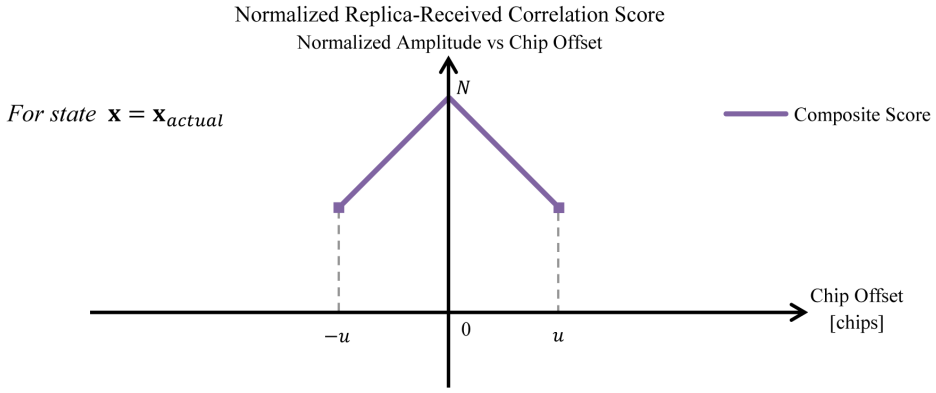

Figure 4.5: The sum of the individual replica-received correlation plots at the ground truth location of the receiver over a select range where all channels are above their noise floors.

The rate at which the individual cross-correlation plots is a function of the relative locations of that channel's satellite and the receiver. Channels with satellite elevation closer to the horizon will move into the correlation function's noise floor faster than those closer to zenith. For simplicity, in this analysis, it is assumed that the DPE grid has no candidate points with channels in the noise floor. Thus, Figure 4.5 visualizes only a range of offsets bounded by u in which each channel has a non-zero score.

4.2.2 Geometric Solution of Position

Due to this known and piecewise-linear structure, the ground truth position may be geometrically computed. Figure 4.6 visualizes geometric relations used in the computation of the ground truth $\mathbf{x}_{actual}$ from two states $\mathbf{x}_1 \in [-u, 0)$ and $\mathbf{x}_2 \in (0, u]$ with $|\mathbf{x}_1| > |\mathbf{x}_2|$.

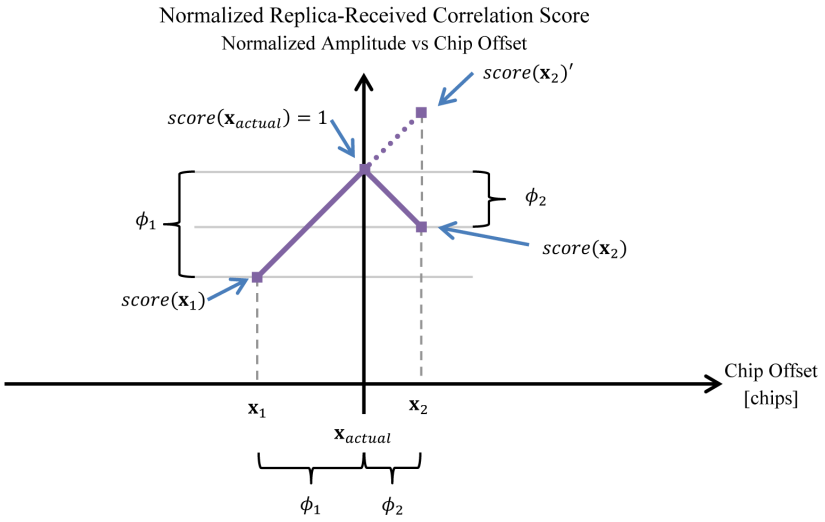

Figure 4.6: The geometry of the replica-received correlation scores for two states nearby the ground truth location of the receiver.

The $score(\cdot)$ values and the locations of $\mathbf{x}_1$ and $\mathbf{x}_2$ are the only values available to the receiver. However, other terms present in Figure 4.6 will be used to compare the weighted average of $\mathbf{x}_1$ and $\mathbf{x}_2$ to an exact calculation of the receiver's true location $\mathbf{x}_{actual}$. Equation 4.5 geometrically solves for the value of $\mathbf{x}_{actual}$ through linear interpolation on the line from $score(\mathbf{x}_1)$ to $score(\mathbf{x}_2)'$:

$$\mathbf{x}_{actual} = \frac{(score(\mathbf{x}_{actual}) - score(\mathbf{x}_1))\mathbf{x}_2 + (score(\mathbf{x}_2)' - score(\mathbf{x}_{actual}))\mathbf{x}_1}{score(\mathbf{x}_2)' - score(\mathbf{x}_1)} \tag{4.5}$$

For simplicity, assuming the composite correlation function is normalized, following the formulation of Equation 4.1, the width and height of the triangles drawn in Figure 4.6 are equivalent:

- $score(\mathbf{x}_{actual}) - score(\mathbf{x}_1) = \phi_1$
- $score(\mathbf{x}_{actual}) - score(\mathbf{x}_2) = \phi_2$
- $score(\mathbf{x}_2)' = score(\mathbf{x}_2) + 2\phi_2$
- $score(\mathbf{x}_2) - score(\mathbf{x}_1) = \phi_1 - \phi_2$

If the correlation plot is not normalized, a scale factor s may be used to relate the values instead. Using this simplification, though, the following substitutions may be made into Equation 4.5:

$$\begin{aligned}\mathbf{x}_{actual} &= \frac{\phi_1\mathbf{x}_2 + (score(\mathbf{x}_2) + 2\phi_2 - score(\mathbf{x}_{actual}))\mathbf{x}_1}{score(\mathbf{x}_2) + 2\phi_2 - score(\mathbf{x}_1)} \\ &= \frac{\phi_1\mathbf{x}_2 + (2\phi_2 - \phi_2)\mathbf{x}_1}{2\phi_2 + \phi_1 - \phi_2} \\ &= \frac{\phi_1\mathbf{x}_2 + \phi_2\mathbf{x}_1}{\phi_1 + \phi_2}\end{aligned} \tag{4.6}$$

4.2.3 The Weighted Average Position-Domain Tracker

In comparison, the weighted average between $\mathbf{x}_1$ and $\mathbf{x}_2$ is given by Equation 4.7.

$$\begin{aligned}\mathbf{x} &= \frac{score(\mathbf{x}_1)\mathbf{x}_1 + score(\mathbf{x}_2)\mathbf{x}_2}{score(\mathbf{x}_1) + score(\mathbf{x}_2)} \\ &= \frac{(1 - |\phi_1|)\mathbf{x}_1 + (1 - |\phi_2|)\mathbf{x}_2}{2 - |\phi_1| - |\phi_2|}\end{aligned} \tag{4.7}$$

When comparing Equation 4.6 and Equation 4.7, it is clear that the weighted average does not compute the true position of the manifold peak. However, the weighted average is capable of converging to the true position over multiple iterations due to the trends of the numerator coefficients. Let $\mathbf{x}_1$ be further away from $\mathbf{x}_{actual}$ than $\mathbf{x}_2$. Then,

- The factors of the $\mathbf{x}_1$ term, ϕ_2 (Equation 4.6) and $1 - |\phi_1|$ (Equation 4.7), will be smaller numbers.
- The factors of the $\mathbf{x}_2$ term, ϕ_2 (Equation 4.6) and $1 - |\phi_2|$ (Equation 4.7), will be larger numbers.

These factors will trend in the same direction as $\mathbf{x}_2$ approaches $\mathbf{x}_{actual}$. When $\mathbf{x}_2 = \mathbf{x}_{actual}$, $\phi_2 = 0$, and Equation 4.6 will keep $\mathbf{x}_2 = \mathbf{x}_{actual}$. When using the weighted average-based position-domain vector tracker, if the DPE grid is evenly spaced, $\mathbf{x}_2 = \mathbf{x}_{actual}$, and there is a third point $\mathbf{x}_3$ such that $\mathbf{x}_2 - \mathbf{x}_1 = \mathbf{x}_3 - \mathbf{x}_2$, as shown in Figure 4.7:.

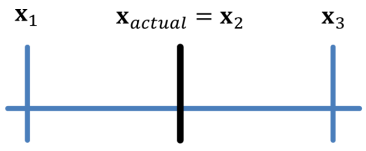

Figure 4.7: Visualization of the relations of $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$.

For these three points, the weighted average equation changes to Equation 4.8.

$$\mathbf{x} = \frac{(1 - |\phi_1|)\mathbf{x}_1 + (1 - |\phi_2|)\mathbf{x}_2 + (1 - |\phi_3|)\mathbf{x}_3}{3 - |\phi_1| - |\phi_2| - |\phi_3|} \tag{4.8}$$

where $|\phi_1| = |\phi_3|$, $\mathbf{x}$ will also equal $\mathbf{x}_2$, and the grid will remain at $\mathbf{x}_2$.

4.2.4 DPE Receiver Design Insights

By taking the weighted average of the states on the grid, the known structure of the manifold can be leveraged to numerically find the state at the maximum of the theoretical DPE manifold, regardless of whether that state is actually sampled by the grid. The weights used in this operation are simply the scores computed by each grid point, making the computation efficient when parallelized using reduction.

The weighted average does assume that the manifold is unimodal – the weighted average for a manifold with two peaks, for example, will converge to a state between the peaks. Also, this tracker does require successive timesteps to converge, but will remain at the maximum of the theoretical manifold once it is found. Finally, this approach does require a manifold grid that is symmetric about the center state of the grid, however, which forces an inherent structure in the states that are searched. This last requirement may be innocuous or even desired if the receiver is leveraging satellite geometry effects, but, nonetheless, must be considered when the receiver designer is choosing grid configurations. Such considerations are the price for the high accuracy and computational efficiency of the tracker, and good choices of such tradeoffs are integral to a well-designed receiver implementation.

4.3 Accuracy Limitations of DPE

The batch correlation technique introduced in Section 2.3.2 and utilized as described in Section 3.1.3 is crucial for making the computational cost of a numerical DPE implementation practically implementable in portable receivers. However, it is an approximation of the true scores of the grid states, and this approximation does introduce an accuracy-degrading effect, which will be studied in this section.

4.3.1 Numerical Cross-Correlation

The batch correlation technique operates on the principle that, for the DPE algorithm, all states on the position-time grid ultimately reduce to an estimate of code phases to each satellite and that all states on the position-time grid have the same relative velocity with respect to each satellite. And, since circular cross-correlation functions can be efficiently calculated by FFT, the circular cross-correlation function between the received signal and the receiver's best PVT estimate will efficiently give an approximation of the scores at every code delay and, thus, every state on the grid.

However, the numerical implementation of the cross-correlation function does not compute an analytical expression of the cross-correlation function. Instead, it computes cross-correlation scores at specific code delays, leaving the receiver to interpolate the score between what are effectively samples of the cross-correlation function. Over the majority of code delays, interpolating the score is an accurate approximation – the cross-correlation function is piece-wise linear when the replica is within one code chip of the received signal and is in a noise floor around zero when the replica has more than one code chip of error with respect to the received signal. However, the interpolation can be erroneous for a critical range of code delays.

Consider the difference between the analytical cross-correlation function and the numerical cross-correlation function, assuming the receiver has a perfect model for the received signal. The numerical implementation, consisting of the cross-correlations of two sampled signals, will be a discrete set of cross-correlation score values. If the replica signal is constructed in the numerical implementation with no code delay with respect to the received signal, linear interpolation between the cross-correlation samples will have no error compared to the analytical cross-correlation function. However, if the code delay is non-zero, linear interpolation between the cross-correlation samples will be erroneous near the peak of the cross-correlation function.

For clarification without loss of generality and for comparison with the localization results in Chapter 5, this effect is visualized in Figure 4.8 for the sampling frequency $f_s = 2.5\,\text{MHz}$ used in the receiver implementation. For the GPS L1 signal tracked by the receiver implementation of this work, the PRN code repeats 1023 code chips every 1 ms. Thus, the samples of the replica signal are spaced according to Equation 4.9:

$$\frac{(1 \times 10^{-3})\,\text{sec}}{1023\,\text{chips}} \times (2.5 \times 10^{6} \frac{\text{samples}}{\text{sec}}) \approx 2.5 \frac{\text{samples}}{\text{chip}} = 0.4 \frac{\text{chips}}{\text{sample}} \tag{4.9}$$

With the replica samples spaced at $0.4 \frac{\text{chips}}{\text{sample}}$, the worst-case numerical approximation of the analytical cross-correlation function by linear interpolation will occur when the replica is constructed at a code phase error of 0.2 chips, as shown in Figure 4.8.

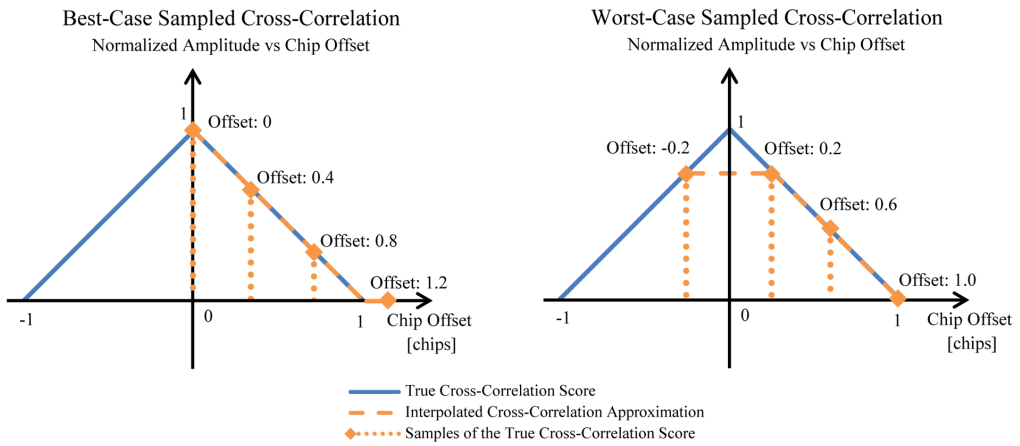

Figure 4.8: The computed cross-correlation scores compared to the theoretical cross-correlation scores under channel parameter error.

4.3.2 Resultant Localization Errors

Open-loop DPE, in particular, suffers detrimental accuracy effects from erroneous batch correlation approximations, as the following events occur:

1. When the DPE receiver implementation begins a new timestep, it constructs a replica signal at the receiver's previous PVT state, which, for the position-time component, was the state on the grid with the highest batch correlation score. Thus, the replica at the new timestep will have nearly the same code phase error as the highest-score position-time state of the previous timestep, with the only change in code phase error coming from movement of the receiver.

2. If the position-time grid includes the same state as the receiver's previous PVT estimate, there will be a sample of the cross-correlation function exactly at the code phase error of that state on the grid. All

other states on the position-time grid will be interpolations of the samples of the cross-correlation function.

3. If the code phase error of the previous timestep's PVT estimate has not changed sufficiently, the sample of the cross-correlation function corresponding to that PVT state will have the highest score, since it was the state with the highest score at the previous timestep. And, the grid point state at the previous PVT state will have a score exactly equal to this highest score, while all other grid points will, at best, be interpolations between the best score and some other lower score cross-correlation sample.

In essence, for any given timestep, the score for the previous-best PVT state will be the one most accurately computed in the current timestep. And, the PVT states that should score better than the previous-best PVT state will be underestimated – scoring worse than the previous best PVT state – within some range of code phase errors. This arises because of the batch correlation and the grid-based approach generating the replica for the previous best PVT state. This effect is visualized in Figure 4.9.

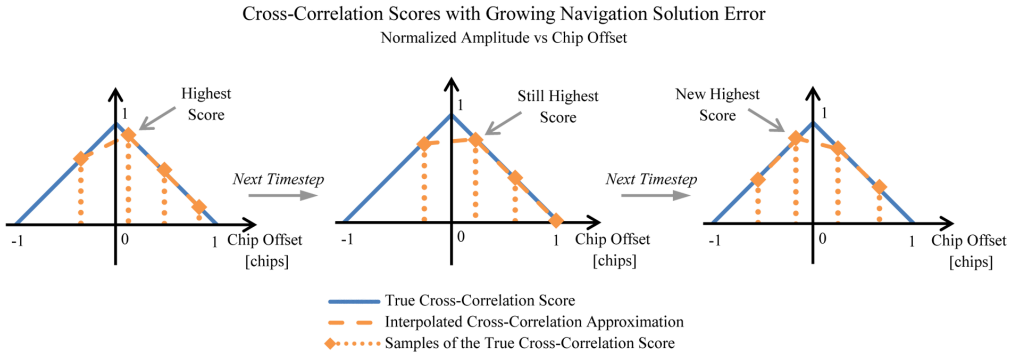

Figure 4.9: The progression of the error in the batch-correlated cross-correlation scores over multiple timesteps for a DPE receiver moving away from the estimated state.

As shown in Figure 4.9, for the batch correlation implementation of this work, there will be a range of 0.4 code chips with respect to a given satellite where the previous best PVT state will again have the highest cross-correlation score. In the worst-case scenario for such error, the receiver would still have to move a distance of 0.2 chips along the line-of-sight direction to

a given satellite before the receiver's PVT estimate would no longer be the highest score. This still translates to a 60-m range in the line-of-sight direction where the previous PVT state has the highest cross-correlation score for a specific satellite.

When transforming this error region to ENU coordinates, satellite geometry will play a significant role. For purely horizontal movement, the error region will still be 60 m. However, satellites can only be tracked with some elevation above the horizon, extending the error region further as a function of the geometry matrix $\mathbf{\Gamma}$. Nonetheless, this approximation of a 60-m open-loop DPE "deadband" provides a reasonable accuracy expectation and explanation for presented in Chapter 5 of this work.

The impact of satellite geometry also leads to a concerning effect: an open-loop grid-based DPE receiver implementation using batch correlation will have worse horizontal accuracy in urban environments, as the only satellites visible by line-of-sight in an urban canyon have high elevation. This is due to satellite geometry – horizontal error will minimally change the code phase error for high elevation satellites. This effect runs counter to the motivation of using DPE in urban environments for its multipath resilience.

4.3.3 DPE Receiver Design Insights

Batch correlation and grid-based DPE are beneficial numerical approximations for computational efficiency, particularly when implemented on a GPU. However, replica fidelity is biased towards the previous best guess on the grid and reduced for grid states closer to the true peak. Thus, estimates from such an implementation will not move from a maximum likelihood state unless the code phase error of that state exceeds a certain deadband range. This deadband range is a function of the sampling frequency and the line-of-sight vectors to all satellites. However, these effects can be mitigated when properly addressed in the DPE receiver design. Thus, this impact on accuracy should be considered for compensation by receiver design choices:

1. Increase the sampling frequency. This will reduce the time between samples and, thus, the size of the accuracy deadband. Increased frequency will come at a higher processing cost and more expensive hardware, but is guaranteed to decrease the size of the deadband propor-

tionately to the increase in the sampling frequency.

2. Add channel parameter-domain signal trackers. If the code phase is more accurately resolved by tracking chip transitions in the received signal instead of simply being back-calculated from the previous PVT estimate, the replica is more likely to have a sample at the true peak. This comes at a slight increase in computational cost and will provide improvement proportionately to how well the bit transitions can be tracked. However, it may be susceptible to multipath errors if the tracking loops are scalar.

3. Add position-domain signal trackers. Grid-based DPE samples the theoretical manifold at specific states. A position-domain tracker will filter these samples to estimate the scores of the theoretical manifold between these samples and estimate the true peak at a higher precision than that of the grid. If the filtering is more accurate, the added precision will lead to a more accurate replica at the next timestep. However, such trackers will only provide improvement as accurate as their filtering.

4.4 Conclusion

This chapter examined three effects that arise in numerical solutions to the DPE algorithm. Section 4.1 introduced a coupling between vertical and clock bias estimates that arises from the satellite geometry that can be leveraged depending on the choice of manifold grid. Section 4.2 justified the use of a weighted average-based signal tracker and the requirements such a signal tracker places on the choice of manifold grid. Section 4.3 identified an approximation error that will cause an open-loop DPE receiver to remain at a PVT estimate within a certain error range, but can be countered with compensation in other parts of the receiver design. These effects will be present in the localization results of Chapter 5, including experiments where the first and second will be leveraged and the third will be mitigated by receiver design.

CHAPTER 5

EXPERIMENTAL RESULTS

To evaluate the DPE receiver implementation detailed in Chapter 3, this chapter presents the third of the three areas of contributions of this work: localization accuracy and computational efficiency analysis of the DPE-based receiver on simulated and real-world data. The software-defined DPE-based receiver implementation developed specifically for this work generated all the results presented in this chapter from execution on an NVIDIA Jetson TX2 following the implementation parameters specified in Chapter 3. Furthermore, the three effects of numerical DPE studied in Chapter 4 are all present and analyzed in the results.

First, idealized GPS receiver data generated by simulation will study the best-case localization accuracy and to identity the causes of effects present in the localization results. Second, real-world GPS receiver data will evaluate the receiver implementation's performance when subjected to motion and unmodeled errors. Third, the speedup and GPU occupancy of the DPE receiver implementation presented in this work as compared to a sequential DPE implementation will be analyzed.

5.1 Localization Accuracy Analysis – Simulated Data

This section will study the DPE receiver implementation's ability to choose an accurate state from the manifold. In order to evaluate the best-case capabilities of the receiver, a simulated dataset of a stationary receiver is used. A stationary-receiver simulated dataset is desirable for this objective, as it will have the following properties:

- Nearly line-of-sight satellite transmissions, as only Klobuchar-modeled atmospheric deflections will be introduced.

- No fluctuation of received satellite power, which would be possible in real-world data due to atmospheric and terrain effects. Received satellite power will be modeled as a function of distance, which will vary minimally over the duration of the dataset.

- No movement of the receiver, meaning the cross-correlation manifold will have only minimal change of shape over the dataset, and the change that does occur will only be from the motion of satellites.

Given these three properties, it may be assumed that there is one maximum likelihood state in the dataset for a theoretically perfect solution to the DPE objective function of Equation 2.11 – in other words, there should exist one state where the replica-received cross-correlation scores for each satellite are at their maximum value. Also, with the clean satellite signal transmissions, the model used by the receiver implementation to construct the replica signals will be quite accurate to the actual received transmission. This, then, makes the dataset ideal to analyze how well the DPE receiver implementation solves this objective function in an ideal case. In particular, steady state localization error, convergence, and convergence time will be of interest in this analysis.

5.1.1 Localization Experiment Design

In order to test the DPE receiver implementation's localization capabilities, the receiver was initialized at 100 different states surrounding the true position of the receiver per experiment. The initialization states were chosen by randomly drawing *i.i.d.* from a uniform distribution:

- Magnitude values in the range $[50, 80]$ m and angle values in the range $[0, 2\pi)$ for the horizontal plane. These values were converted into rectangular offsets in the East and North directions and added to the initial state.

- Values in the range $[-80, -50] \cup [50, 80]$ m as an offset in the Up direction and added to the initial state.

- For certain experiments, values in the range $[-80, -50] \cup [50, 80]$ m as an offset in the clock bias direction and added to the initial state.

The lower bound of 50 m was selected according to the DPE accuracy limitations imposed by batch correlation as explained in Section 4.3 – the open-loop navigation solution should not be expected to move if errors are less than approximately 60 m. The upper bound was chosen to ensure that the manifold peak shape and the maximum likelihood state would be contained within the 154 m reach of the grids defined in Section 3.3.5.

The DPE receiver is also given accurate states of the satellites at initialization which were obtained through closed-loop scalar tracking. This ensured that error present in the open-loop DPE solution could only be caused by the DPE algorithm or the updates between timesteps. With the idealized simulated data and channel-domain tracking loops, the navigation solution computed by this scalar tracking had centimeter-level error with respect to the simulated state, confirming the satellite states used in initialization were accurate.

5.1.2 Satellite Geometry

The dataset used in this section consists of simulated transmissions from GPS satellites with the geometric configuration shown in Figure 5.1. Satellite with PRN 22 was not tracked by the DPE receiver implementation due to its low elevation.

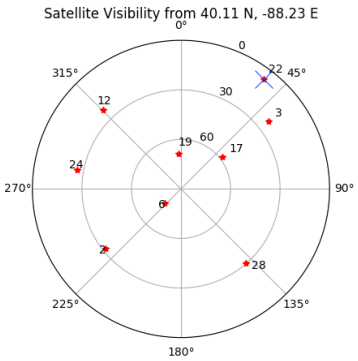

Figure 5.1: A skyplot of the satellites visible to the simulated receiver for the dataset used in Section 5.1.

The DOP matrix for the simulated dataset, expressed in the dimensions of East, North, Up, and clock bias, respectively, is given in Equation 5.1:

$$DOP_{sim} = \begin{bmatrix} 0.390438 & -0.013094 & -0.166666 & 0.139578 \\ -0.013094 & 0.607397 & 0.0867343 & -0.110702 \\ -0.166666 & 0.0867343 & 2.38308 & -1.51608 \\ 0.139578 & -0.110702 & -1.51608 & 1.09746 \end{bmatrix} \tag{5.1}$$

The horizontal DOP is the square root of the sum of the East and North variance values from DOP_{sim}: $HDOP = \sqrt{0.390438 + 0.607397} = 0.998917$. The vertical DOP is the square root of the Up-direction variance value from DOP_{sim}: $VDOP = \sqrt{2.38308} = 1.54372$. The clock bias DOP is the square root of the clock bias variance value from DOP_{sim}: $TDOP = \sqrt{1.09746} = 1.047597$. As expected from the discussion on satellite geometry in Section 4.1, the vertical accuracy will be more sensitive to measurement noise due to its higher DOP than the horizontal or clock bias dimensions. Furthermore, the covariance between the clock bias and the vertical dimensions is 1.51608 – a strong coupling, especially when compared to the other off-diagonal terms.

The satellite geometry and resulting DOP_{sim} matrix was computed for the beginning of the dataset and experienced negligible change over the dataset.

5.1.3 Localization – Open-Loop *Spread Grid 7m*

For the following experiments, the *Spread Grid 7m* manifold configuration was used as specified in Section 3.3.5. The simulated stationary dataset was processed with open-loop DPE. In order to study the effects of coupling between the vertical and clock bias dimensions, two sets of 100 randomly initialized runs each processed the first two seconds of the dataset: one set of 100 runs with no perturbation in the initial clock bias estimate, and one set of 100 runs with perturbation in the initial clock bias estimate. The offsets to the scalar tracking initialization state for both sets of 100 runs are shown in Figure 5.2.

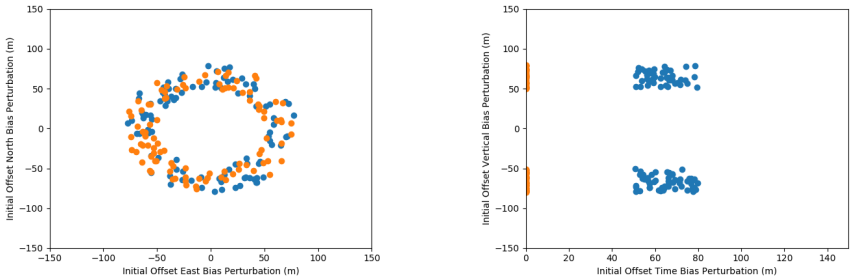

Figure 5.2: The offsets to the scalar initialization state for the results presented in Section 5.1.3. The initializations with no clock bias perturbation are shown in blue, and the initializations with clock bias perturbation are shown in orange.

The final *root mean square* (RMS) errors for all 100 runs in these two sets are shown in Figure 5.3.

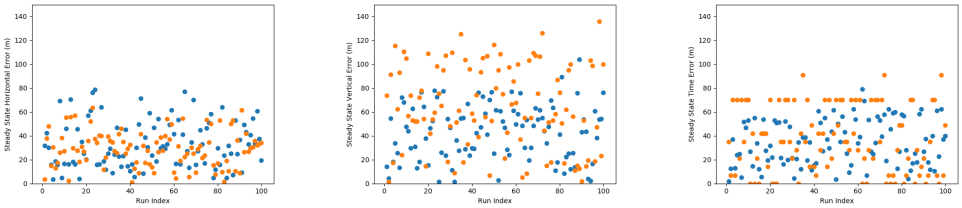

Figure 5.3: The RMS error after two seconds of processing for the randomly initialized simulated dataset using the DPE receiver implementation with the *Spread Grid 7m* manifold grid. The initializations with no clock bias perturbation are shown in blue, and the initializations with clock bias perturbation are shown in orange.

Furthermore, by plotting the error in one dimension with respect to another, the expected coupling between only the vertical and clock bias dimensions may be observed, as shown in Figure 5.4.

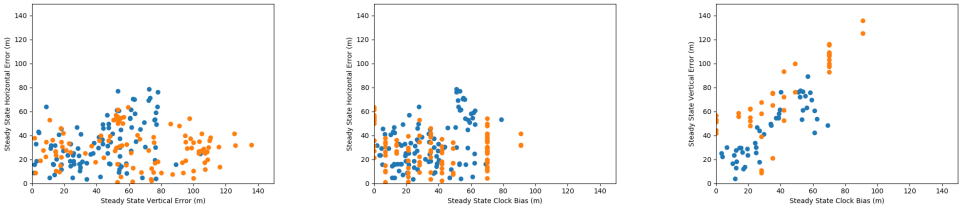

Figure 5.4: The RMS error after two seconds of processing for the randomly initialized simulated dataset using the DPE receiver implementation with the *Spread Grid 7m* manifold grid. The initializations with no clock bias perturbation are shown in blue, and the initializations with clock bias perturbation are shown in orange.

As expected from DOP_{sim} and comparable to the scalar tracking results presented in [59], Figure 5.4 shows a linear correlation between vertical error and clock bias error, and no correlation between other states. This coupling allows for relatively low vertical RMS errors in the final states, as statistically characterized in Table 5.1.

Table 5.1: RMS and standard deviation of the error in the randomly initialized simulated dataset using the DPE receiver implementation with the *Spread Grid 7m* manifold grid. The initializations with no clock bias perturbation are provided in the top row, and the initializations with clock bias perturbation are provided in the bottom.

	Horizontal	**Vertical**	**Clock Bias**	**Geometric**
RMS (m) ***No δt Perturbation***	32.78455	72.96034	45.58069	92.06323
RMS (m) ***δt Perturbed***	38.24430	48.88085	39.13977	73.37497
Standard Deviation (m) ***No δt Perturbation***	14.90965	34.90779	27.43428	36.88651
Standard Deviation (m) ***δt Perturbed***	19.40486	23.03195	18.72932	28.82295

Interestingly, perturbing the clock bias state noticeably improves the vertical RMS error, as well as the clock bias error. This is likely due to the numerical sampling of the manifold grid. When the DPE receiver's state estimate moves, the state on which the grid is centered will also move at the next timestep. This will cause a different set of points to be sampled; with

the increased sampling density at the center of the grid, the states around the new estimate will be sampled at a higher resolution, as well.

Initial state estimates with more error typically resulted in the receiver moving to a greater number of intermediate state estimates. For the 100 runs without the clock bias perturbation, the DPE receiver implementation moved once at most and, on average, 0.89 times before settling on the final state. For the 100 runs with the clock bias perturbation, the DPE receiver implementation moved five times at most and, on average, 1.68 times before settling on the final state.

Additionally, with the way the random perturbations were drawn, runs in the clock bias-perturbed set would be initialized with states with less than 30 m difference in their vertical and clock bias error half the time. With the linear structure of the candidate points in the *Spread Grid 7m*, the coupling between the vertical and clock bias states may be exploited, as the manifold would immediately be in a region with many states having comparable vertical and clock bias errors, beneficially utilizing the coupling. Conversely, runs that did not have clock bias perturbations would have states with around 65 m difference in their vertical and clock bias error. States with comparable vertical and clock bias error would be in the lower-resolution region of the grid, and they may have significant error in the horizontal dimension, giving a low score and preventing the grid from moving that direction.

Thus, perturbing the clock bias forced more states – typically including states with less error – on the theoretical replica-received correlation manifold to be sampled, increasing the receiver implementation's likelihood of finding a state near the true peak. These better estimates are reflected by the improvement in the covariance for the vertical and clock bias states.

With the improvement in the vertical and clock bias accuracies from perturbing the initial clock bias comes a slight decrease in horizontal accuracy. However, as Table 5.1 has a lower geometric error, and Equation 2.11 shows that DPE optimizes with respect to low geometric error, this increase in horizontal error is reasonable, given the definition of the objective function.

5.1.4 Localization – Open-Loop *RNGrid 7m*

For the following experiments, the *RNGrid 7m* manifold configuration was used as specified in Section 3.3.5. The simulated stationary dataset was processed with open-loop DPE – weighted average tracking was not used. In order to study the effects of coupling between the vertical and clock bias dimensions, two sets of 100 randomly initialized runs each processed the first two seconds of the dataset: one set with no perturbation in the initial clock bias estimate, and one set with perturbation in the initial clock bias estimate. The offsets to the scalar tracking initialization state for both sets of 100 runs are shown in Figure 5.5.

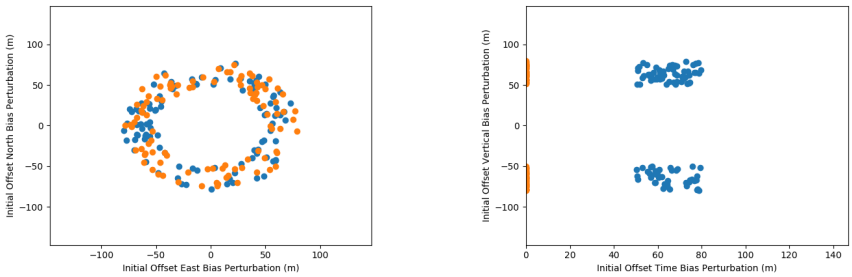

Figure 5.5: The offsets to the scalar initialization state for the results presented in Section 5.1.4. The initializations with no clock bias perturbation are shown in blue, and the initializations with clock bias perturbation are shown in orange.

The final RMS errors for all 100 runs in these two sets are provided in Figure 5.6.

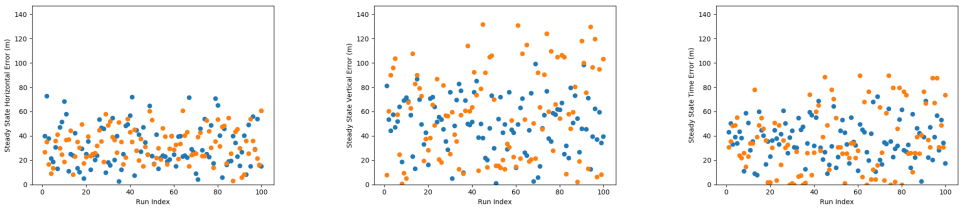

Figure 5.6: The RMS error after two seconds of processing for the randomly initialized simulated dataset using the DPE receiver implementation with the *RNGrid 7m* manifold grid. The initializations with no clock bias perturbation are shown in blue, and the initializations with clock bias perturbation are shown in orange.

Furthermore, by plotting the error in one dimension with respect to an-

other, the expected coupling between only the vertical and clock bias dimensions may be observed, as shown in Figure 5.7.

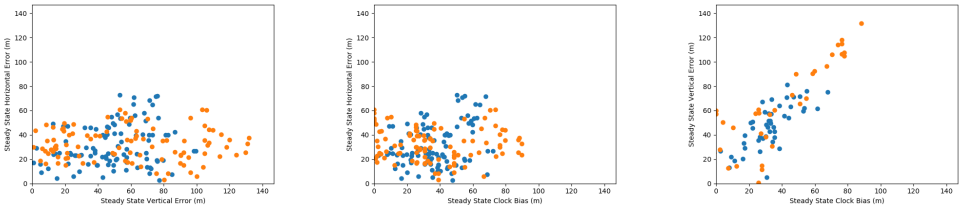

Figure 5.7: The RMS error after two seconds of processing for the randomly initialized simulated dataset using the DPE receiver implementation with the *RNGrid 7m* manifold grid. The initializations with no clock bias perturbation are shown in blue, and the initializations with clock bias perturbation are shown in orange.

Again, Figure 5.7 shows a linear correlation between vertical error and clock bias error, and no correlation between other states. Table 5.2 shows the final RMS errors for these experiments.

Table 5.2: RMS and standard deviation of the error in the randomly initialized simulated dataset using the DPE receiver implementation with the *RNGrid 7m* manifold grid. The initializations with no clock bias perturbation are provided in the top row, and the initializations with clock bias perturbation are provided in the bottom.

	Horizontal	**Vertical**	**Clock Bias**	**Geometric**
RMS (m) ***No*** δt ***Perturbation***	35.10287	69.31776	44.66751	89.62338
RMS (m) δt ***Perturbed***	35.99064	54.11594	40.16401	76.40033
Standard Deviation (m) ***No*** δt ***Perturbation***	13.39996	37.10285	25.989167	37.57627
Standard Deviation (m) δt ***Perturbed***	17.44745	22.07944	15.95802	26.03454

Again, perturbing the clock bias state noticeably improves the vertical RMS error, as well as the clock bias error. As with *Spread Grid 7m*, *RNGrid 7m* will sample a different set of points as the state estimate moves.

Initial state estimates with more error again typically result in the receiver moving to more intermediate state estimates. For the 100 runs without the

clock bias perturbation, the DPE receiver implementation moved once at most and, on average, 0.94 times before settling on the final state. For the 100 runs with the clock bias perturbation, the DPE receiver implementation moved five times at most and, on average, 1.85 times before settling on the final state. Again, sampling more states on the theoretical replica-received manifold improves the receiver implementation's likelihood of finding a state near the true peak.

However, in *RNGrid 7m*, there is no structure in the choice of candidate points, so any exploitation of the coupling between vertical and clock bias error will stem only from the density of the states in the grid.

The horizontal error also decreases slightly for the *RNGrid 7m* when the initial clock bias is perturbed. Again, since Table 3.2 shows the geometric error decreased with the clock bias perturbations, this worsening of the horizontal accuracy is an effect of Equation 2.11 optimizing for lower geometric error.

5.1.5 Localization – Closed-Loop *Spread Grid 7m*

In the following experiment, the *Spread Grid 7m* manifold configuration was used as specified in Section 3.3.5. The simulated stationary dataset was processed with closed-loop DPE using the weighted-average position-domain tracker. Due to the improvement observed in Sections 5.1.3 and 5.1.4 when perturbing the initial clock bias estimate, the clock bias was perturbed for this experiment, as well.

Since the weighted average tracker position-domain tracker requires a symmetric manifold to converge and remain on the true peak of the manifold, the *RNGrid 7m* manifold was not evaluated with the tracker, as the randomness does not guarantee symmetry. Also, 60 seconds of the dataset was processed to observe convergence of the weighted average position-domain tracker. The RMS error over all 100 randomly initialized runs for every sample set is shown in Figure 5.8.

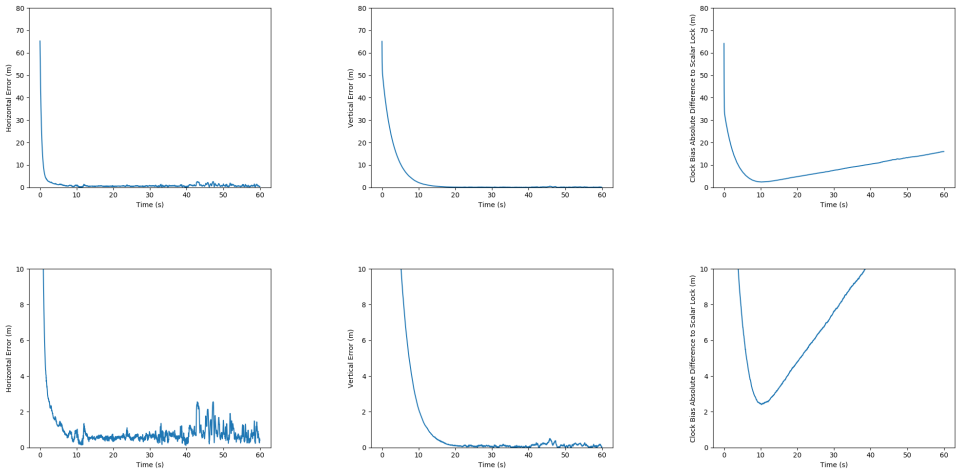

Figure 5.8: The RMS error after two seconds of processing for the randomly initialized simulated dataset using the DPE receiver implementation with the *Spread Grid 7m* manifold grid. The initial clock bias estimate was perturbed in this dataset.

Figure 5.8 shows that localization error in the closed-loop implementation continuously decreases over the first 20 seconds – a difference from the open-loop implementation, which jumped at most five times in the first one second of data before remaining steady for the rest of the dataset. The convergence in the closed-loop case is due to two reasons. First, as discussed in Section 4.2, the weights used in the weighted average are close but not exactly equal to the geometrically ideal values of linear interpolation. Second, Section 4.3 shows that the batch correlation method of manifold state score calculation will often compute cross-correlation scores that undervalue the states on the peak. In such a scenario, the manifold itself will have a form where the true peak is "sliced off" and the computed maximum is at the navigation solution of the previous timestep. As shown in Figure 4.9, the "sliced off" region of the manifold will still have higher scores than the side walls of the manifold, so the weighted average of the manifold will be closer to the peak than the computed maximum point. This effect will pull the navigation solution towards the true peak over multiple timesteps until it has converged on the true peak, which manifests in the decreasing error of Figure 5.8.

After the first 20 seconds of the dataset, the exponential error decay stops and experiences steady-state trends. To study this steady-state region, analysis will focus first on the horizontal and vertical error, then focus second on the clock bias error.

Horizontal and Vertical Error

As seen in Figure 5.8, the weighted average-based signal tracker converges to significantly more accurate horizontal and vertical estimates of the true receiver position than the open-loop method. Additionally, the errors in the open-loop and weighted-average position domain signal tracker exhibit some different effects compared to each other:

1. In the open-loop implementation, the navigation solution will vary over time for approximately one second before reaching and remaining at a final state for the rest of the dataset. In the closed-loop implementation, the navigation solution will converge to a region of less than 3 m RMS position error within the first 20 seconds of the dataset, but the navigation solution will continue to move during the rest of the dataset.

2. In the open-loop implementation, different random initializations will arrive at different navigation solutions with standard deviation on the order of tens of meters. In the closed-loop implementation, different random initializations will arrive nearly the same navigation solution with standard deviation on the order of micrometers.

By picking the state with the highest score, the open-loop implementation does not utilize any information about the shape of the manifold and is only capable of evaluating a finite set of states around the initialization state. In contrast, the closed-loop implementation is steered to the peak of the cross-correlation manifold by moving the grid until the scores on either side of the center of the manifold are balanced. The weights used in the weighted average may also resolve to values between grid points, allowing the receiver to sample any possible point. As a result, the closed-loop receiver is capable of getting much closer to the true peak of the cross-correlation manifold.

The jitter that remains in the horizontal and vertical error of the closed-loop implementation is likely due to the atmospheric effects modeled by the simulator but not the receiver implementation. These errors will prevent the replica-received correlation score functions of the individual channels from perfectly aligning at the ground-truth state, and thereby from matching Figure 4.2. Thus, there will be some error in the estimation of the ground truth state that cannot be overcome without better replica modelling. As this error is on the order of 3 m RMS, it is likely smaller than the 7-m spacing of grid

points in the open-loop implementation, and is "filtered" by the grid only being able to reach a finite subset of states.

To characterize the steady-state jitter in the closed-loop implementation, Table 5.3 shows the mean value from 30 s to 60 s of the RMS error at each timestep for all random initializations, and the mean value from 30 s to 60 s of the standard deviation in the error at each timestep for all random initializations in the horizontal and vertical dimensions.

Table 5.3: RMS and standard deviation of the error in the randomly initialized simulated dataset using the DPE receiver implementation with the *Spread Grid 7m* manifold grid for both weighted average closed-loop (CL) DPE and open-loop (OL) DPE. The initial clock bias estimate was perturbed in this dataset.

	Horizontal RMS Error Over All Timesteps	**Vertical RMS Error Over All Timesteps**
CL RMS at Each SS Timestep Over All Runs (m)	0.950763	0.143969
CL Standard Deviation at Each SS Timestep Over All Runs (m)	0.000303	0.000850
OL RMS at Each SS Timestep Over All Runs (m)	38.24430	48.88085
OL Standard Deviation at Each SS Timestep Over All Runs (m)	19.40486	23.03195

In the statistical characterization of the horizontal and vertical error of Table 5.3, it can be seen that the vertical error is an order of magnitude better than the horizontal error. This is due to the coupling to a very accurate lock of the clock bias, the justification for which will be discussed next.

Clock Bias Error

In Figure 5.8, while the horizontal and vertical position plots converge near zero in steady-state, the clock bias plot grows linearly for the remainder of the dataset. At first glance, it may appear that this is a loss of lock and increasing error in the clock bias term of the DPE receiver's state estimates.

However, with further analysis, it may be concluded that the receiver is maintaining lock on a drifting clock.

First, as indicated on the y-axis of Figure 5.8, the clock bias plot is showing the absolute difference with respect to the scalar tracking used to initialize the dataset. This scalar tracking solution is the state to which all perturbations are added. So, a difference with respect this state is not an error unless that state is known to be at the true location with respect to the received signal. And, the linearity in the growing clock bias of Figure 5.8 is indicative of receiver clock drift.

All clock bias estimates from 30 s to 60 s were fit to a first-order equation using the `linregress` function of the SciPy library [60]. With a coefficient of determination of $R^2 = 0.99888$, the linear growth in the clock bias state depicted in Figure 5.8 can be modeled for sample set index τ as given in Equation 5.2:

$$\delta t_{growth}(\tau) = 0.00556 \frac{\text{m}}{\text{sampleset}} \tau - 0.70281\,\text{m} \tag{5.2}$$

With the strong linearity over this region as indicated by the coefficient of linearity, it would appear that this effect is caused by a clock drift in the simulated data itself. Converting Equation 5.2 into units of clock drift by using the speed of light as $3 \times 10^8 \frac{m}{s}$ and the length of one sample set as $20 \frac{ms}{sampleset}$ (specified in Section 3.1.2) gives Equation 5.3:

$$\delta t_{growth}(\tau) = 0.92665 \frac{\text{ns}}{\text{s}} \tau - 2.34272\,\text{ns} \tag{5.3}$$

If the DPE receiver implementation is accurately locked to the true clock bias, this linear regression shows that the drift is less than one nanosecond per second or less than one part per billion (ppb), which is within the rating of 50 ppb of the simulator module used [61]. Furthermore, if the trend is latent in the convergence phase of the clock bias plot, the scalar tracking solution used for initialization has 2.35 ns of error, indicating that this solution is accurate, as expected.

As the fit and parameters of the regression in Equation 5.3 are reasonable, it may be concluded that the DPE receiver implementation not only correctly tracks a drift in the clock bias, but that this drift is correctly tracked regardless of initialization, according to Table 5.4. The RMS for the steady-state interval from 30 s to 60 s of the standard deviations between each of the 100

runs for the clock bias is in the micrometer range, showing that all random initializations converged to nearly the same estimate.

Table 5.4: RMS and standard deviation of the error in the randomly initialized simulated dataset using the DPE receiver implementation with the *Spread Grid 7m* manifold grid for both weighted average closed-loop (CL) DPE and open-loop (OL) DPE. The clock bias is referenced to the scalar initialization clock bias. The initial clock bias estimate was perturbed in this dataset.

	Clock Bias RMS Difference Over All Timesteps
CL RMS at Each SS Timestep Over All Runs (m)	15.98248
CL Standard Deviation at Each SS Timestep Over All Runs (m)	0.000610
OL RMS at Each SS Timestep Over All Runs (m)	39.13977
OL Standard Deviation at Each SS Timestep Over All Runs (m)	18.72932

Since it may be concluded that the DPE receiver implementation very accurately tracks the clock bias in the navigation solution, the vertical accuracy will also benefit through the coupling of these two states.

The clock drift is not noticeable in Sections 5.1.3 and 5.1.4, as the open-loop results are only processed for two seconds. By Equation 5.2, the clock bias error at two seconds is $-0.147\,\text{m}$, significantly smaller than the grid point spacing.

5.1.6 Conclusion – Stationary Data

Using the idealized simulation data, theoretical effects of DPE and the grid-based implementation could be easily isolated and studied. By analyzing the results of many randomly initialized runs, the predicted coupling between the vertical and clock bias states was identified. The performances of different grids were shown to vary due to this coupling, and both grids had strengths

depending on the initial error in the data. Also, the effect of the batch correlation biasing scores towards the navigation solution of the previous timestep was identified. The navigation solutions only moved a few times and typically never moved again after the navigation solution had less than 0.2 to 0.3 code chips of error.

Additionally, a discriminator-like weighted averaging step demonstrated improved accuracy by leveraging a prediction about the shape of the manifold. Given time to converge, the weighted averaging step was also able to mitigate the effects of inaccurate score calculation from the batch correlation approximation.

Comparing Grids in Open-Loop

Comparing Table 5.1 and Table 5.2, the standard deviations for all cases were very similar, all differing with their counterpart on the other grid by less than 3 m. Thus, the uncertainties in the estimates for both grids were similar. However, a clear difference in the RMS error emerged. When the clock bias was not perturbed, *RNGrid 7m* reached better solutions, with lower vertical, clock bias, and geometric error. When the clock bias was perturbed, *Spread Grid 7m* reached better solutions, with lower vertical, clock bias, and geometric error.

As discussed in Section 5.1.3 and Section 5.1.4, perturbing the clock bias estimates produced better estimates of the clock bias due to the grids sampling more points. Thus, these experiments show that structure in selection of grid points in the manifold does impact the accuracy of the solution. When fewer candidate states are sampled and the coupling between vertical and clock bias states cannot be well-used, it is better to have no structure in the choice of candidate states being evaluated. However, when more candidate states are sampled and the coupling between vertical and clock bias states can be exploited, a structure in the points being sampled can lead the receiver implementation to better estimates.

Comparing Open and Closed-Loop Operation

With the unimodal manifold shape of the simulated data, the weighted average step aided the receiver in converging to sub-meter-level accuracies over

the first 20 seconds of the dataset. The different initialization perturbations converged to effectively the same state with only sub-millimeter-level differences, demonstrating the reliability of the discriminator-like step. The accuracy of this approach is significantly greater than decameter-level accuracy in the open-loop implementation.

The weighted average position-domain tracker also demonstrates how correct characterization of the true manifold can reject errors. The true manifold is unimodal, but inaccuracies in the batch correlation cause the computed manifold to have the true peak "sliced off". In this case, fitting the manifold to a unimodal shape correctly shifts the navigation solution towards the true peak, and the batch correlation error is effectively rejected over multiple timesteps. This motivates the advantages of position-domain discriminators and how much they can improve the accuracy of DPE if they accurately model the theoretical manifold shape.

5.2 Localization Accuracy Analysis – Real-World Data

This section will analyze the DPE receiver implementation's ability to localize itself in real-world environments. In order to evaluate the practical performance of the receiver, two datasets recorded during the flight of a C-12C Huron are used. These datasets were generated under "Project GRIFFIN", a Test Management Project by the United States Air Force Test Pilot School at Edwards Air Force Base for academic use by the Grace Gao Research Group [62]. These real-world datasets will have the following properties:

1. Real-world effects on the satellite transmissions, such as atmospheric and terrain effects on signal power, will be present in the received signal.

2. A mobile receiver will produce very different manifolds over the dataset, and the receiver will need to follow the true position of the receiver throughout the dataset for successful operation.

3. Satellite transmissions may reflect off of terrain, and atmospheric effects are not guaranteed to follow the Klobuchar model.

Given these three properties, the maximum likelihood state for a theoretically perfect solution to the DPE objective function of Equation 2.11 will

vary over time. Furthermore, some real-world effects on the satellite transmissions will not be reconstructed by the receiver implementation, meaning the DPE manifold computed by the receiver implementation will not match the theoretical DPE manifold for a receiver that could perfectly replicate the received signal. This dataset, then, will serve to evaluate how well a DPE receiver implementation can perform its objective of localization with real-world errors. In particular, localization error as a function of time will be of interest in this analysis.

5.2.1 Localization Experiment Design

In order to test the DPE receiver implementation's localization capabilities, the receiver was initialized at a state during the flight found by scalar tracking. The satellite states obtained during the scalar tracking initialization were also provided to the DPE receiver implementation. For both datasets, the *Spread Grid 6m* specified in Section 3.3.5 was used in an open-loop implementation.

Sixty seconds of each dataset was processed to evaluate the localization accuracy of the receiver over time. Ground truth position was obtained from a *time-space positioning information* (TSPI) system, which recorded the flight path of the aircraft with a position accuracy of $\pm 0.46\,\mathrm{m}$ and velocity accuracy of $\pm 6.1 \frac{\mathrm{mm}}{\mathrm{s}}$ [26].

5.2.2 Satellite Geometry – Mobile 1 Dataset

The first dataset, referred to hereafter as the *Mobile 1 Dataset*, was collected by a flight in the Mojave Desert northeast of Edwards Air Force Base in California. The flight profile took the aircraft nearly due east in a straight and level path. The horizontal component of the flight path is drawn in blue along with the aircraft's horizontal and vertical velocities in Figure 5.9.

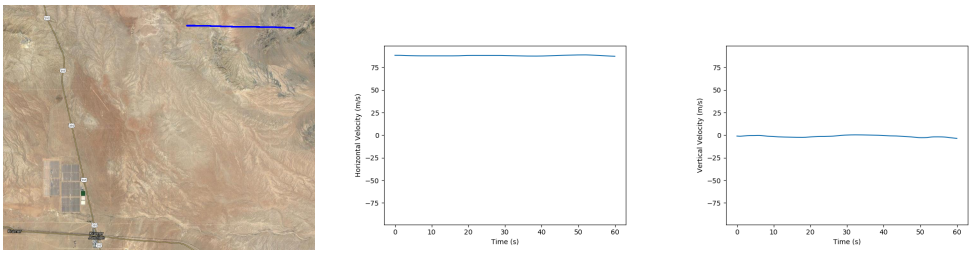

Figure 5.9: The flight path (left), horizontal velocity (center), and vertical velocity (right) of the *Mobile 1* dataset. The flight path is represented by the blue line, travelling from left to right in the image. Plotted using Google Maps [63].

The received signal consists of transmission from GPS satellites with the geometric configuration shown in Figure 5.10. The satellite with PRN 30 was not tracked by the DPE receiver implementation in this experiment.

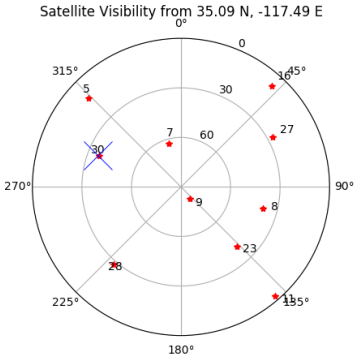

Figure 5.10: A skyplot of the satellites visible to the simulated receiver for the *Mobile 1* dataset. Satellites and their PRN are denoted with the red star and black text.

The DOP matrix for the *Mobile 1* real-world dataset, expressed in the dimensions of East, North, Up, and clock bias, respectively, is given in Equation 5.4:

$$DOP_{m1} = \begin{bmatrix} 0.402148 & -0.0293464 & -0.0509775 & 0.0396547 \\ -0.0293464 & 0.580979 & 0.461984 & 0.303643 \\ -0.0509775 & 0.461984 & 1.97873 & 1.12828 \\ 0.0396547 & 0.303643 & 1.12828 & 0.784079 \end{bmatrix} \tag{5.4}$$

The horizontal DOP is the square root of the sum of the East and North variance values from DOP_{m1}: $HDOP = \sqrt{0.402148 + 0.580979} = 0.991527$.

The vertical DOP is the square root of the Up-direction variance value from DOP_{m1}: $VDOP = \sqrt{1.97873} = 1.406674$. The clock bias DOP is the square root of the clock bias variance value from DOP_{m1}: $TDOP = \sqrt{0.784079} = 0.885482$. As expected from the discussion on satellite geometry in Section 4.1, the vertical accuracy will be more sensitive to measurement noise due to its higher DOP than the horizontal or clock bias dimensions. Furthermore, the covariance between the clock bias and the vertical dimensions is 1.12828 – a strong coupling, especially when compared to the other off-diagonal terms.

The satellite geometry and resulting DOP_{m1} matrix was computed for the beginning of the dataset and experienced negligible change over the dataset.

5.2.3 Localization – Mobile 1 Dataset

Over the 60 seconds of processed data, the DPE receiver's horizontal and vertical error with respect to the TSPI system and clock bias error with respect to the scalar initialization propagated forward are shown in Figure 5.11. In Figure 5.11, a gray vertical line is placed every time the DPE solution moves, as it does not move every iteration. This can be seen in Figure 5.12.

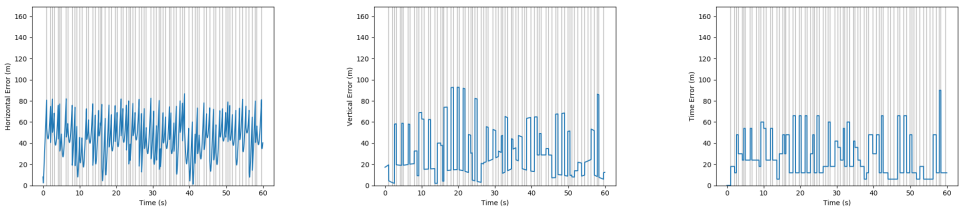

Figure 5.11: The RMS error over 60 seconds of data for the *Mobile 1* dataset using the DPE receiver implementation with the *Spread Grid 6m* manifold grid.

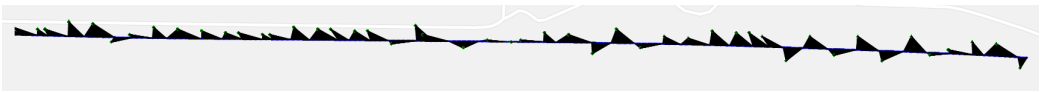

Figure 5.12: The solutions of the DPE receiver implementation over the path of the *Mobile 1* dataset. The black lines connect a DPE receiver implementation solution (green) to the corresponding ground truth as measured by the TSPI system (blue). Plotted using Google Maps [63].

With no exceptions, every piecewise step present in each of the plots of Figure 5.11 is caused by the DPE solution jumping to a new state. Then, the effects between the gray lines may be studied, as shown in Figure 5.13.

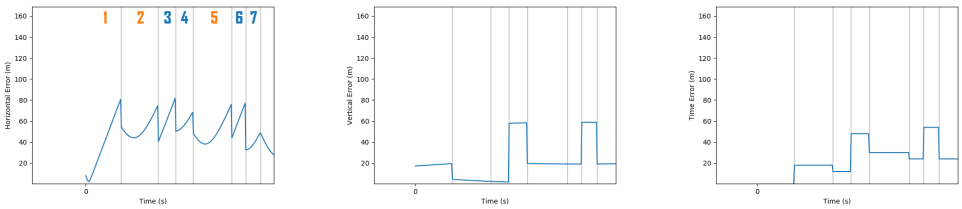

Figure 5.13: A look at the RMS error in the first seven different solutions of the DPE receiver implementation for the *Mobile 1* dataset.

For the horizontal error in Figure 5.11, many of the intervals consist of the error steadily growing until the DPE estimate moves again to a new lower-error state. This is consistent with the aircraft flying away from the point and eventually the receiver's antenna will have moved enough that the correlation score at another point on the grid becomes higher than the previous estimate.

However, some estimates on the horizontal plot show the error decreasing briefly after the DPE solution moves. This occurs when the DPE algorithm chooses a point ahead of the aircraft as its current state estimate. This can be seen by matching the intervals in Figure 5.13 with the different DPE solution estimates in Figure 5.14.

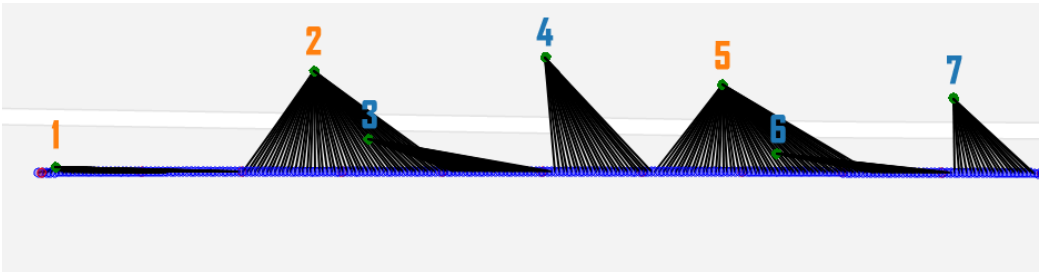

Figure 5.14: A look at the RMS error in the first seven different solutions of the DPE receiver implementation (green) as compared to the receiver's true position (blue) for the *Mobile 1* dataset. The black lines connect a DPE receiver implementation solution to the ground truth at that time. Plotted using Google Maps [63].

For the vertical error in Figure 5.11, the error between intervals is nearly linear as the altitude is relatively constant throughout the flight. This leads

to the plateau-like form of the vertical error plot, which is in contrast to the peaks of the horizontal error plot. If the DPE receiver implementation's navigation solution moves once every several iterations, the flight path velocities shown in Figure 5.9 should be reflected in significant horizontal error change between intervals and minimal vertical error change between intervals.

The slope that does exist between navigation solution movement intervals for the vertical error can be explained by the change – albeit minimal – in altitude over the flight path, as shown in Figure 5.15. Figure 5.15 also shows that the coupling between the vertical error and clock bias remains present in the real-world mobile data.

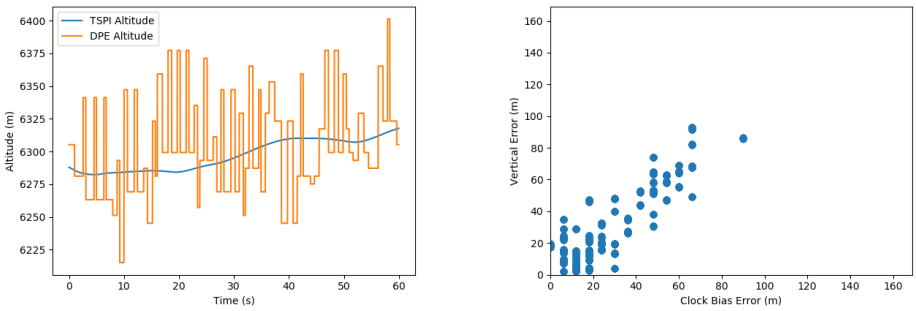

Figure 5.15: An overlay of the DPE-estimated altitude and the TSPI-measured altitude (left) and the vertical error plotted as a function of clock bias error (right).

Due to the highly-accurate timing of the CSAC used to trigger sampling intervals, there is no noticeable clock drift in the clock bias error results of Figure 5.13, and the clock bias error remains level between navigation solution movement intervals. This is the cause of the banding of points in the vertical-clock bias coupling plot of Figure 5.15: the clock bias error only ever resolves to a finite set of values since the clock does not noticeably drift.

The statistical distributions of the error after the first 10 seconds of data in these results are provided in Table 5.5. The first 10 seconds were intentionally omitted from statistical analysis to reduce the likelihood of bias from the scalar tracking aiding used for initialization.

Table 5.5 shows that the horizontal error is larger than the vertical error or clock bias error, a departure from the prediction of DOP_{m1} in Equation 5.4. However, the vertical position of the aircraft only varies over a range of 35.39 m while the horizontal position moves 5271.36 m from the

Table 5.5: RMS and standard deviation of the error over 60 seconds of data for the *Mobile 1* dataset using the DPE receiver implementation with the *Spread Grid 6m* manifold grid.

	Horizontal	**Vertical**	**Clock Bias**	**Geometric**
RMS (m)	44.60444	31.10273	26.67471	58.35988
Standard Deviation (m)	16.43598	23.94378	20.11102	19.86220

initial position.

Furthermore, by looking at the error in the DPE receiver implementation's navigation solution when the navigation solution moved as shown in Figure 5.16, it can be seen that the state estimate moves, on average, when the error is around 82 m. Considering that the average vertical velocity is $1.603\frac{\text{m}}{\text{s}}$ and the average horizontal velocity is $88.080\frac{\text{m}}{\text{s}}$, the true position of the aircraft changes significantly more in the horizontal direction than in the vertical direction between different navigation solution estimates. This effect manifests itself in the horizontal RMS error, meaning that even horizontally-accurate DPE receiver implementation estimates quickly become inaccurate until the solution moves again. Meanwhile, a vertically-accurate navigation solution will remain vertically accurate until the navigation solution moves to an inaccurate guess. Additionally, the vertical estimates benefit from the clock-bias coupling, and the clock bias estimates are free from noticeable drift.

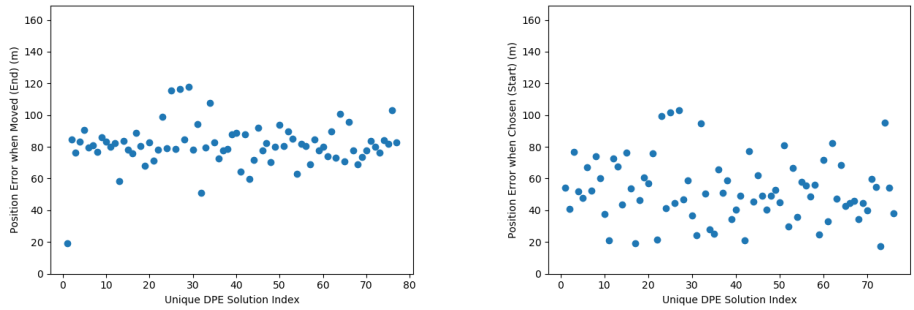

Figure 5.16: The error in the position estimates when the DPE receiver implementation estimate moved (left) and the error of the new position estimates when chosen by the DPE receiver (right).

While the left plot of Figure 5.16 shows that the DPE receiver implementation fairly consistently updates its position estimate when the error

is between 70 and 90 m, the number of timesteps that the receiver remains at a navigation solution has a larger variance, as shown in Figure 5.11 with the gray lines. This additional variance is reflected in the right plot of Figure 5.16, which shows the accuracy of newly-chosen estimates ranges from 20 m of error to 60 m of error.

These effects can be explained by the analysis in Section 4.3. An error of 80 m is on the order of 0.2 PRN code chips for a satellite with elevation a 40° elevation angle. Given $f_s = 2.5\,\text{MHz}$, it should be expected that the navigation solution moves once the receiver has accumulated around 0.2 code chips of phase error with respect to the average satellite. As for the error in a newly-chosen navigation solution, since scores nearest to the peak are the ones that are underestimated when interpolating, a state that scores higher than the previous navigation solution would be an optimization of distance closer to the true peak than the previous navigation solution (for a higher score) and distance away from the true peak (for better score fidelity). This expectation is reflected in the error of the new navigation solution having variance between 20 m and 80 m – grid states with less than 20 m of error are too close to the peak to have good replica fidelity and those greater than 80 m are far enough away from the peak that the previous state still has a higher score.

5.2.4 Satellite Geometry – Mobile 2 Dataset

The second dataset, referred to hereafter as the *Mobile 2 Dataset*, subjected the receiver to a more dynamic flight profile and terrain effects near the Black Mountain Wilderness in California. The flight profile included an S-bend curve and decreasing altitude. The horizontal component of the flight path is drawn in blue along with the aircraft's horizontal and vertical velocities in Figure 5.17.

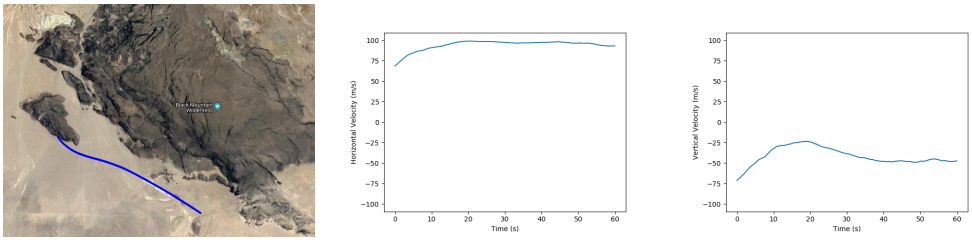

Figure 5.17: The flight path (left), horizontal velocity (center), and vertical velocity (right) of the *Mobile 2* dataset. The flight path is represented by the blue line, travelling from left to right in the image. Plotted using Google Maps [63].

The received signal consists of transmission from GPS satellites with the geometric configuration shown in Figure 5.18. The blue points on the top of the skyplot are estimations of the Black Mountain Wilderness terrain peaks and are included to indicate the possibility of terrain interference effects on low-elevation satellites, such as PRNs 5, 11, and 27. The Black Mountain Wilderness terrain has an altitude on the order of 1000 to 1150 m, as determined for the points selected through the Google Maps Elevation API [63]. The aircraft's altitude decreased from 1000 m to 850 m over the 60 s of the dataset, allowing the Black Mountain Wilderness terrain to rise above the horizon during the flight.

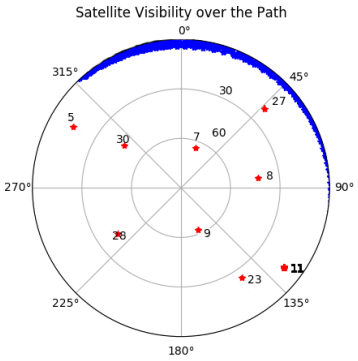

Figure 5.18: A skyplot of the satellites visible to the simulated receiver for the *Mobile 2* dataset. Satellites and their PRN are denoted with the red star and black text, and terrain elements are denoted with blue stars.

The DOP matrix for the *Mobile 2* real-world dataset, expressed in the dimensions of East, North, Up, and clock bias, respectively, is given in Equation 5.5:

$$DOP_{m2} = \begin{bmatrix} 0.354055 & 0.10108 & 0.14323 & 0.120878 \\ 0.10108 & 0.470432 & -0.0347885 & -0.0174103 \\ 0.14323 & -0.0347885 & 1.88099 & 1.08842 \\ 0.120878 & -0.0174103 & 1.08842 & 0.745301 \end{bmatrix} \tag{5.5}$$

The horizontal DOP is the square root of the sum of the East and North variance values from DOP_{m2}: $HDOP = \sqrt{0.354055 + 0.470432} = 0.908013$. The vertical DOP is the square root of the Up-direction variance value from DOP_{m2}: $VDOP = \sqrt{1.88099} = 1.371492$. The clock bias DOP is the square root of the clock bias variance value from DOP_{m2}: $TDOP = \sqrt{0.745301} = 0.863308$. As expected from the discussion on satellite geometry in Section 4.1, the vertical accuracy will be more sensitive to measurement noise due to its higher DOP than the horizontal or clock bias dimensions. Furthermore, the covariance between the clock bias and the vertical dimensions is 1.08842 – a strong coupling, especially when compared to the other off-diagonal terms.

The satellite geometry and resulting DOP_{m2} matrix was computed for the beginning of the dataset and experienced negligible change over the dataset.

5.2.5 Localization – Mobile 2 Dataset

Over the 60 seconds of processed data, the DPE receiver's horizontal and vertical error with respect to the TSPI system and clock bias error with respect to the scalar initialization propagated forward are shown in Figure 5.19. In Figure 5.19, a gray vertical line is placed every time the DPE solution moves, as it does not move every iteration. This can be seen in Figure 5.20.

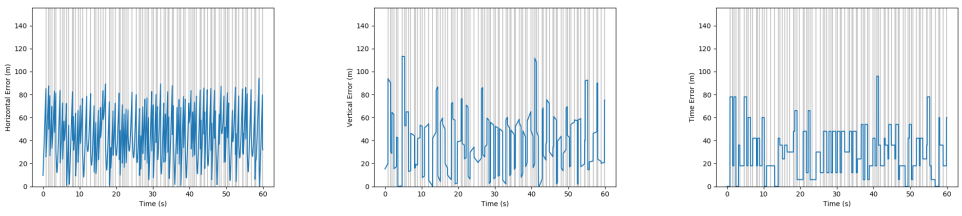

Figure 5.19: The RMS error over 60 seconds of data for the *Mobile 2* dataset using the DPE receiver implementation with the *Spread Grid 6m* manifold grid.

Again, with no exceptions, every piecewise step present in each of the plots

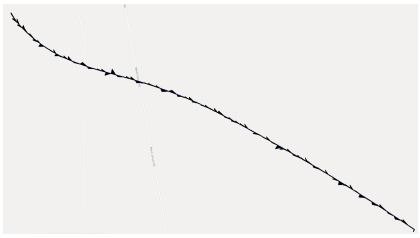

Figure 5.20: The solutions of the DPE receiver implementation over the path of the *Mobile 2* dataset. The black lines connect a DPE receiver implementation solution (green) to the corresponding ground truth as measured by the TSPI system (blue). Plotted using Google Maps [63].

of Figure 5.19 is caused by the DPE solution jumping to a new state. Then, the effects between the gray lines may be studied, as shown in Figure 5.21.

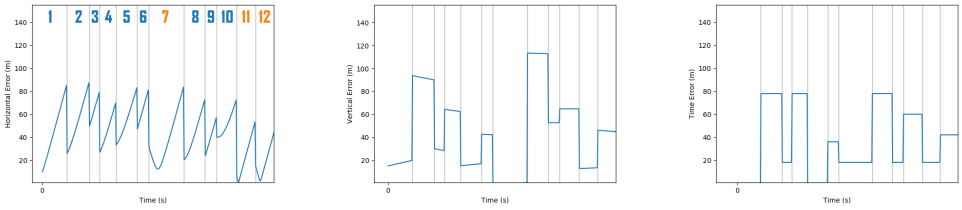

Figure 5.21: A look at the RMS error in the first 12 different solutions of the DPE receiver implementation for the *Mobile 2* dataset.

As with the *Mobile 1* dataset, horizontal error in the *Mobile 2* dataset either strictly increases or decreases then increases over the interval of a navigation solution. These effects are the result of newly-selected navigation solutions being chosen ahead or behind the flight path of the aircraft, as shown in Figure 5.22.

Compared to the horizontal error, the vertical error between navigation solution intervals in Figure 5.19 is much more level, yet it is steeper than the vertical error in the *Mobile 1* dataset. This is the result of the vertical velocity being smaller than the horizontal velocity, as shown in Figure 5.17, but larger than the vertical velocity in the *Mobile 1* dataset.

Figure 5.23 shows the altitude estimates of DPE overlaid on the TSPI-measured altitude and the coupling between the vertical and clock-bias errors.

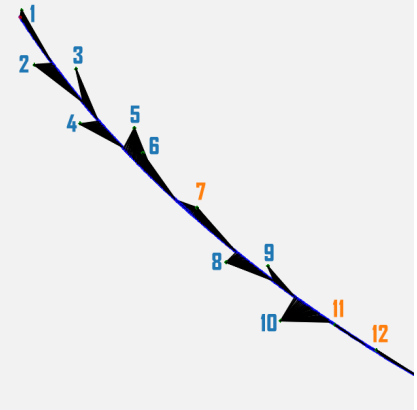

Figure 5.22: A look at the RMS error in the first 12 different solutions of the DPE receiver implementation (green) as compared to the receiver's true position (blue) for the *Mobile 2* dataset. The black lines connect a DPE receiver implementation solution to the ground truth at that time. Plotted using Google Maps [63].

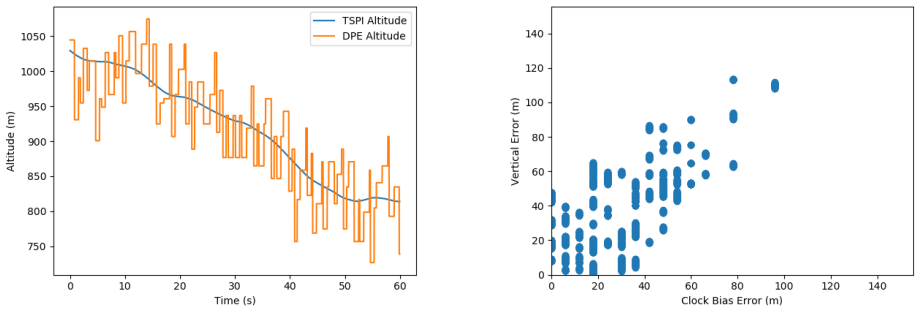

Figure 5.23: An overlay of the DPE-estimated altitude and the TSPI-measured altitude (left) and the vertical error plotted as a function of clock bias error (right).

Again, due to the highly-accurate timing of the CSAC used to trigger sampling intervals, there is no noticeable clock drift in the clock bias error results of Figure 5.21, and the clock bias error remains level between navigation solution movement intervals. This is the cause of the banding of points in the vertical-clock bias coupling plot of Figure 5.23: the clock bias error only ever resolves to a finite set of values since the clock does not noticeably drift.

The statistical distributions of the error after the first 10 seconds of data in these results are provided in Table 5.6. The first 10 seconds were intentionally omitted from statistical analysis to reduce the likelihood of bias from the scalar tracking aiding used for initialization.

Table 5.6: RMS and standard deviation of the error over 60 seconds of data for the *Mobile 2* dataset using the DPE receiver implementation with the *Spread Grid 6m* manifold grid.

	Horizontal	**Vertical**	**Clock Bias**	**Geometric**
RMS (m)	43.57418	37.65380	29.04197	61.85950
Standard Deviation (m)	19.69433	23.39240	19.60951	20.61531

Table 5.6 again shows that the horizontal error is larger than the vertical error or clock bias error. Though more than the *Mobile 1* dataset, the vertical position of the aircraft varies over a range of 215.5 m, which is still significantly less than the 6129.8 m of horizontal movement.

Furthermore, by looking at the error in the DPE receiver implementation's navigation solution when the navigation solution moved as shown in Figure 5.24, it can be seen that the state estimate moves, on average, when the error is around 88 m. Considering that the average vertical velocity is $42.56\frac{\text{m}}{\text{s}}$ and the average horizontal velocity is $94.45\frac{\text{m}}{\text{s}}$, the true position of the aircraft changes significantly more in the horizontal direction than in the vertical direction between different navigation solution estimates. This effect manifests itself in the horizontal RMS error, meaning that even horizontally-accurate DPE receiver implementation estimates quickly become inaccurate until the solution moves again. Meanwhile, a vertically-accurate navigation solution will remain vertically accurate until the navigation solution moves to an inaccurate guess. Additionally, the vertical estimates benefit from the clock-bias coupling, and the clock bias estimates are free from noticeable drift.

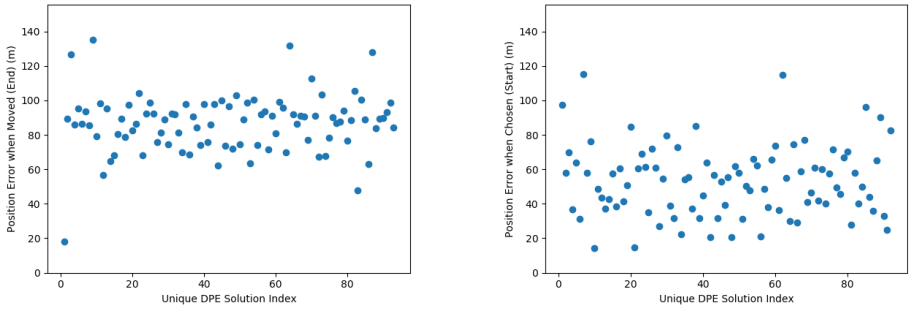

Figure 5.24: The error in the position estimates when the DPE receiver implementation estimate moved (left) and the error of the new position estimates when chosen by the DPE receiver (right).

Compared to Figure 5.16, Figure 5.24 shows more variation in when the DPE receiver updates its position estimate. The *Mobile 2* dataset typically moves its position estimate when the error is between 60 and 100 m. This can be attributed to the more dynamic flight profile, which experienced more altitude change and a horizontal S-bend maneuver. Again, the variance in the right plot of Figure 5.24 – the error of the new position estimate when chosen – is greater than that of the left plot. However, the error in the newly chosen estimates still typically ranges from 20 to 70 m.

The error plots of Figure 5.24 are still consistent with Section 4.3. The receiver state tends to move when the error to the average satellite is around or a little greater than 0.2 PRN code chips, as a horizontal error of 90 m is 0.2 PRN code chips for a satellite with elevation a 48° elevation angle. And, the bounds on the range of errors when new navigation solutions are chosen are consistent with the sampling effects – grid states with less than 20 m of error are too close to the peak to have good replica fidelity and those greater than 70 m are far enough away from the peak that others have a higher score.

5.2.6 Conclusion – Mobile Data

In this section, the DPE receiver implementation was demonstrated in two datasets experiencing receiver motion and real-world signal inaccuracies. The RMS error of the DPE receiver was comparable to the simulated data, with a degradation of the horizontal error on the order of 10 m due to the receiver moving and the batch correlation effects. Using the open-loop implementation, the effects of the numerical implementation of DPE could be easily studied.

The *Mobile 1* dataset was rather idealized for real-world data, as the receiver was flown in a straight-and-level flight path in an open-sky environment. The DPE-based receiver was seen to move navigation states when the error exceeded 80 m on average, which is attributed to the errors from the batch calculation. However, the receiver implementation still followed the navigation solution over the course of the flight, as the grid size was sufficiently large to include the true navigation solution when it moved out of the range of the batch correlation error.

The *Mobile 2* dataset subjected the receiver to a more dynamic flight

profile and nearby terrain. The horizontal RMS error was about 1 m less than that of the *Mobile 1* dataset, though the vertical and clock bias RMS errors were a few meters worse. There was more variation for when the navigation solution of the *Mobile 2* dataset moved, but it was still around the value to be expected from the batch correlation error.

5.3 GPU Usage Performance Analysis

This section will examine the computational performance of the DPE receiver implementation. The results presented are for the case of eight satellites being tracked in an open-sky stationary receiver simulated dataset using the *Spread Grid 7m* manifold grid. The DPE receiver implementation is initialized from state and channel parameter estimates acquired through scalar tracking. Additionally, the minimal satellite interference ensures that values are well-behaved – the maximum likelihood state is within the domain of the manifolds and the matrices in the EKF do not reach singular values. This is done to ensure the results are an accurate profiling of the implementation during intended operation.

Table 5.7 provides the information used in this analysis. First, GPU occupancy for the parallelized implementation is used to identify bottlenecks. Second, the speedup of the parallelized implementation is evaluated through comparison to a sequential implementation.

5.3.1 GPU Occupancy

The parallelized DPE receiver implementation processes one set of 20 ms of voltage samples in under one second. The *DPInit*, *SampleBlock*, and *DataLogger* modules are marked "Negligible", as their implementations are parallelized to perform reads and writes in the background of the main processing stream and their time cost is below the precision of the study of this work. The *cuEKF* and *ChannelManager* modules must execute on the main processing stream and do measurably contribute to the execution time, though their parallelizations reduce their time cost to two orders of magnitude smaller than that of the *BatchCorrScores* and *BatchCorrManifold* modules.

Table 5.7: Execution time of each module for one timestep.

Operation	**Parallelized DPE**		**Sequential DPE**	
	OL Time	**CL Time**	**OL Time**	**CL Time**
DPInit	Negligible	Negligible	–	–
SampleBlock	Negligible	Negligible	–	–
BatchCorrScores	0.113 s	0.113 s	0.460 s	0.452 s
Replica Gen.	*0.053 s*	*0.053 s*	*0.050 s*	*0.050 s*
FFTs	*0.060 s*	*0.060 s*	*0.410 s*	*0.402 s*
BatchCorrManifold	0.842 s	0.731 s	2.28 s	2.32 s
Pos. Manifold	*0.366 s*	*0.363 s*	*1.056 s*	*1.050 s*
Vel. Manifold	*0.373 s*	*0.367 s*	*0.940 s*	*0.936 s*
Measurement Gen.	*0.103 s*	*<0.001 s*	*<0.001 s*	*0.044 s*
cuEKF	<0.001 s	<0.001 s	–	–
ChannelManager	0.001 s	0.001 s	–	–
DataLogger	Negligible	Negligible	–	–
Total	**0.959 s**	**0.846 s**	**2.91 s**	**2.93 s**

The largest contributor to the overall time is *BatchCorrManifold*, requiring on the order of 800 ms. This comes from two kernels running in parallel to generate the position-time and velocity-drift manifolds. To score a state on the grid, a thread must compute the value of that state, remove the effect of the Earth rotating during the transmission time from each satellite state, compute the difference between the state's channel parameters and the scores' reference state, and interpolate to find the score for that state. This requires a significant number of registers, which limits the GPU occupancy to 25%.

The open-loop and closed-loop implementations differ in how the manifolds are scored. The open-loop implementation was developed with further analysis in mind, storing the scores for all grid points in memory before performing a reduction comparison to find the grid state with the highest score. This was done to give the researcher the opportunity to study the manifold shape after processing the dataset. The weighted average-based closed-loop implementation was developed with computational efficiency in mind, demonstrating much faster operation through the use of a reduction sum operation integrated into the manifold scoring step. As the integrated reduction operation does not store the manifold scores, it sacrifices modularity and ease of analysis for a nearly negligible execution cost.

The *BatchCorrScores* module also contributes noticeably to the time cost. The Fourier transforms which generate the position-time scores take 12 ms

and the high-resolution FFT which generates the velocity-drift scores takes 47 ms. As expected from Section 3.3.4, the velocity-drift scores are nearly four times the computation time, as the FFT of the velocity-drift scores is performed on an array $2^{3.38}$ times the length of the the array used for the position-time scores. The remaining 60 ms of the 115 ms is spent constructing the 50×10^3-sample signal replicas for each channel. The dependence of the velocity-drift scores on the position-time scores to determine the navigation bit also bottlenecks the velocity-drift operations, including the high-resolution velocity-drift FFT.

Figure 5.25 shows the CUDA kernel launches for one timestep using the Nsight IDE GPU profiling tool. The large number of registers required to score the position-time and velocity-drift manifolds prevents their respective CUDA kernels from executing concurrently, despite being assigned to separate streams. A different tuning in the number of threads dispatched would allow these kernels to execute concurrently, but with a negligible difference in total time.

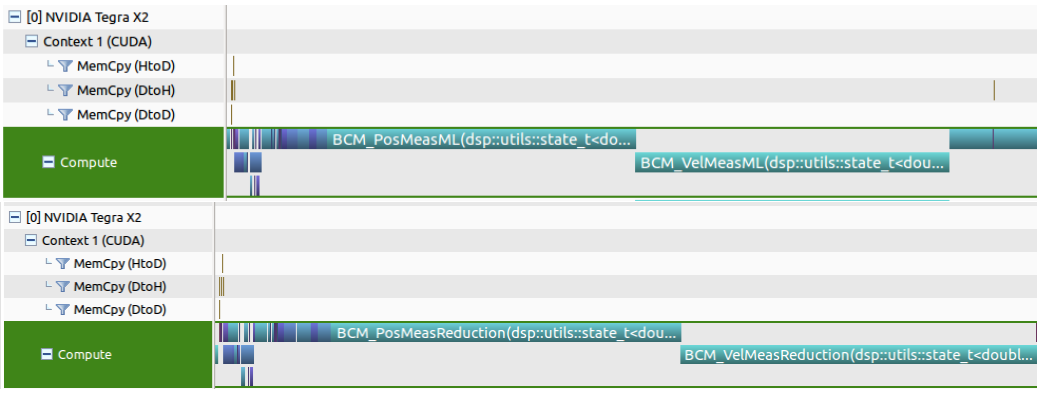

Figure 5.25: Time spent on GPU operations in one timestep of the DPE receiver implementation for the open-loop implementation (top) and the closed-loop implementation (bottom).

5.3.2 Parallelized GPU vs Sequential CPU

The sequential CPU implementation compared in Table 5.7 is acquired using `PyGNSS` – a Python-based software-defined GNSS receiver [64] configured to match the DPE implementation details of Chapter 3. `PyGNSS` performs the same computations in the DPE algorithm as the GPU implementation devel-

oped for this work, but with vector operations rather than CUDA kernels and without parallelization present in *SampleBlock*, *BatchCorrScores*, or *BatchCorrManifold*. This makes `PyGNSS` a suitable candidate for comparison to the GPU implementation. `PyGNSS` is run using Python 2.7 on a consumer-grade Asus G75VX commercial laptop with an Intel i7-3630QM 2.4GHz CPU and 12GB of RAM.

Overall, the parallelized implementation demonstrates a speedup on the order of three times as compared to the sequential implementation. As `PyGNSS` does not follow the software architecture presented in this work, timing for operations besides those of the DPE algorithm are not comparable. However, comparisons may be drawn within the DPE algorithm modules of the receiver implementation: *BatchCorrScores* and *BatchCorrManifold*.

For the *BatchCorrScores* and its equivalent operations in `PyGNSS`, both the sequential and parallelized implementation have similar time costs to construct the replica signal. However, the parallelized implementation performs the FFTs nearly six times faster than the sequential implementation, leading to an overall speedup for the scoring operations of over four times.

For the *BatchCorrManifold* and its equivalent operations, even with only 25% occupancy, the parallelized implementation computes the manifold scores almost three times faster than the sequential implementation. As the manifold grids consist of 25^4 points, this result shows that parallelization is crucial for larger DPE grid sizes to be practically implemented. In the open-loop case, Python's data management allows the largest score to be quickly found, while a more significant computation cost is required to perform the weighted average step in the closed-loop case. As seen by the closed-loop case of the parallelized implementation, an integrated reduction step can generate the measurement with time cost similar to that of Python's largest value look-up.

For the operations between timesteps, `PyGNSS` takes approximately 150 ms to run the EKF, update channel parameters and samples for the next timestep, and log values. The parallelized implementation reduces this time cost to approximately 2 ms. However, this comparison should be considered qualitatively and not quantitatively, as `PyGNSS` does have slight implementation differences in these steps. Nonetheless, this comparison does show that, for a DPE-based receiver implementation, asynchronous memory copies and efficient parallelization can minimize the computational cost of these supportive functions.

5.3.3 Conclusion – GPU Usage Analysis

The time cost analysis highlights aspects of hardware that are important to DPE receiver design. Back-calculating the code phases to the satellites was computationally expensive when assessing manifold grid points, causing register usage to be the limiting factor for GPU occupancy. Also, resolving the navigation bit when computing the velocity scores places a bottleneck on the parallelization of the position and velocity score batch calculation.

However, by comparing the execution times of a CPU DPE receiver implementation with a GPU DPE receiver implementation, the performance benefits of DPE implementations when leveraging parallel computing could be easily identified. The FFTs and manifold scoring of the DPE algorithm scale efficiently when parallelized as compared to their CPU counterparts. And, asynchronous data transfers keep the parallelized implementation executing the DPE algorithm during the majority of the execution.

Furthermore, the speedup of the parallelized DPE receiver achieves a target execution time set by Ng and Gao in [23], which demonstrates a DPE-based receiver implementation that tracks a moving vehicle using a 2% duty cycle. In [23], the receiver processes one 20-ms sample set out of every second of data using position-time and velocity-drift grids of 25^4 states each. While the `PyGNSS` sequential configuration compared in this work takes longer than one second to process a 20-ms sample set, the parallelized implementation presented in this work could achieve real-time operation using the 2% duty-cycling technique of [23], as one 20-ms timestep can be processed in less than one second.

5.4 Conclusion

This chapter evaluated the localization results of a numerical implementation of the DPE receiver algorithm. The approximations required for such an implementation result in errors that can be identified by the effect they have on the localization results. Once the causes of these effects are known, the receiver can be designed in a way to mitigate the impact of these effects on the localization results. And, the benefits of these approximations are reflected in the computational improvement of the GPU implementation.

In the idealized datasets of Section 5.1, the coupling between clock bias

and vertical states was shown to aid the accuracy of the navigation solutions, and, because of this, different grids may perform better or worse than each other depending on the current state estimate. Also, the batch correlation approximation makes a signature impact on the localization accuracy, and a position-domain signal tracker was shown to mitigate the impact with a correct characterization of the theoretical manifold. In the real-world data of Section 5.2, the DPE-based implementation tracked the flight path of a mobile receiver and validated an estimation of the "deadband" caused by batch correlation. Lastly, Section 5.3 demonstrated that a portable GPU can outperform a consumer-grade sequential CPU implementation, and the hardware usage was studied for insights into further optimization.

CHAPTER 6

CONCLUSION

To date, the DPE algorithm has been recognized for having the potential to mitigate errors that characteristically cause faults in classical two-step receivers, but the algorithm has thus far seen only limited application in practice due to the complexity of its implementation. Furthermore, approximations made for numerical representation of the theoretical equations and for computational efficiency can reduce the benefits gained by the one-step approach. With the objective of lowering these barriers to broader DPE usage, this work provided an implementation-oriented introduction to DPE, developed a custom parallelized DPE-based receiver implementation, studied effects of the numerical implementation, and evaluated the implementation by processing analyzed GPS datasets.

6.1 Contributions

Chapter 1 motivated the use of the DPE algorithm by a **survey of the conceptual, analytical, and demonstrated advantages** presented in the literature. Chapter 2 summarized from the literature the derivation of the DPE algorithm as a **signal-focused objective function**. Chapter 2 also presented prior **techniques for computational efficiency** that were employed in the receiver implementation developed for this work.

Chapter 3 provided a **parallelization of the DPE algorithm** designed for the CUDA parallel programming paradigm. This parallelization was **implemented as a software-defined GPS receiver** on an NVIDIA Jetson TX2 with supporting hardware to perform necessary RF functionality. Chapter 3 also detailed the implementation including sampling frequency and manifold grids.

Chapter 4 **identified a coupling between the error in the vertical**

and clock bias dimensions present in DPE. A position-domain tracker was also justified. And, a **degradation of cross-correlation scores close to the true peak** of DPE manifold caused by the batch correlation approximation was identified.

In simulated and real-world data, Chapter 5 **identified the three effects** studied in Chapter 4. The simulated data demonstrated the DPE receiver's ability to find the navigation solution. The real-world data subjected the receiver to unmodeled atmospheric and terrain effects. And, the **computational efficiency of the implementation** was analyzed.

6.2 Design Insights and Future Work

The results presented in this work provide insight into DPE-based receiver design. These insights also spur future work, as leveraging these trade-offs will contribute to a better DPE-based receiver. In particular:

- Operations necessary for GNSS-based localization, such as cross-correlation, are executed quickly with parallel processing. Then, by designing the other steps of the DPE algorithm to also be parallelizable, such as the evaluation of the manifold using a grid and reduction sum, a DPE-based receiver can execute more quickly on a GPU than a CPU. Future work should consider more ways to reduce register cost when assessing the manifold and tuning to the number of points on the grid.

- The coupling between clock bias and vertical errors can be leveraged by grid-based DPE. Both conceptually and from experimental results, a broad search over vertical estimates and clock biases should be conducted first to find the proportional relationship between the two. Then, a structured grid can reduce the search space considered when refining the estimate. Thus, grids that maximize the benefit from the coupling and heuristics that trigger grids to change during operation should be considered for future work.

- Particularly for grid-based DPE, batch correlation is a valuable technique for reducing the computational cost, but it comes at the price of accuracy. However, due to the structure of PRN cross-correlation, techniques other than point-to-point linear interpolation or the use of

signal trackers can overcome this accuracy cost. This effect should be considered in future work when implementing manifold scoring algorithms, quantifying the accuracy of a DPE solution, or implementing a signal tracker.

- Position-domain signal trackers are promising for improving the DPE accuracy. Correct characterization of the manifold shape can reject errors. While the true shapes of the theoretical manifolds in this work were unimodal or with minimal reflections, urban environments can introduce multipath sidelobes on the manifold. Future work should consider other peak detectors and methods for evaluating the structure of the grid. Additionally, signal trackers that do not require convergence time may be desirable for future work.

Proper implementation of the results and concepts presented in this work along with further refinements highlighted above will enable the DPE algorithm to serve as the basis for an effective GNSS receiver that can provide more robust performance than the classical scalar tracking-based architectures in compromised signal environments.

REFERENCES

[1] P. N. Misra and P. Enge, *Global Positioning System: Signals, Measurements, and Performance*, 2nd ed. Lincoln, Massachusetts: Ganga-Jamuna Press, 2011.

[2] K. V. Dyke, K. Kovach, and J. Lavrakas, "Status update on GPS integrity failure modes and effects analysis," in *Proceedings of the 2004 National Technical Meeting of The Institute of Navigation*, San Diego, CA, 1 2004, pp. 92–102.

[3] L. Montloin, L. Azoulai, A. Martineau, C. Milner, and C. Macabiau, "GNSS multipath failures modes analysis for airport surface operations," in *26th International Technical Meeting of The Satellite Division of the Institute of Navigation*, Nashville, TN, 9 2013. [Online]. Available: https://hal-enac.archives-ouvertes.fr/hal-00936846 p. 316.

[4] P. Closas and A. Gusi-Amigó, "Direct position estimation of GNSS receivers: Analyzing main results, architectures, enhancements, and challenges," *IEEE Signal Processing Magazine*, vol. 34, no. 5, pp. 72–84, 9 2017.

[5] A. Grant, S. Basker, P. Williams, and N. Ward, "GPS jamming and the impact on maritime navigation," *The Journal of Navigation*, vol. 62, no. 2, pp. 173–187, 4 2009. [Online]. Available: https://www.researchgate.net/publication/228897052

[6] J. Coffed, "The threat of GPS jamming: The risk to an information utility," Harris Corporation, Tech. Rep. [Online]. Available: www.nasa.gov

[7] G. X. Gao, M. Sgammini, M. Lu, and N. Kubo, "Protecting GNSS receivers from jamming and interference," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1327–1338, June 2016.

[8] T. Humphreys, "Statement on the vulnerability of civil unmanned aerial vehicles and other systems to civil GPS spoofing," Subcommittee on Oversight, Investigations, and Management of the House Committee on Homeland Security, Tech. Rep., 2012. [Online]. Available: http://rnl.ae.utexas.edu/images/stories/files/papers/Testimony-Humphreys.pdf

[9] P. Closas, C. Fernández-Prades, and J. A. Fernández-Rubio, "Maximum likelihood estimation of position in GNSS," *IEEE Signal Processing Letters*, vol. 14, no. 5, pp. 359–362, 4 2007.

[10] Y. Ng and G. X. Gao, "Direct position estimation utilizing non-line-of-sight (NLOS) GPS signals," in *Proceedings of the 29th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2016)*, Portland, Oregon, 9 2016, pp. 1279–284.

[11] R. G. Brown, "Receiver autonomous integrity monitoring," in *Global Positioning System: Theory and Applications, Volume II*, B. W. Parkinson, J. J. J. Spilker, P. Axelrad, and P. Enge, Eds. Washington, D.C.: American Institute of Aeronautics and Astronautics, Inc., 2012, pp. 143–165.

[12] P. Axelrad, B. K. Bradley, J. Donna, M. Mitchell, and S. Mohiuddin, "Collective detection and direct positioning using multiple GNSS satellites," *Navigation, Journal of the Institute of Navigation*, vol. 58, no. 4, pp. 305–321, 2011.

[13] P. Closas, C. Fernández-Prades, and J. A. Fernández-Rubio, "Cramér - Rao bound analysis of positioning approaches in GNSS receivers," *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 3775–3786, 2009.

[14] P. Closas, C. Fernández-Prades, and J. A. Fernández-Rubio, "Direct position estimation approach outperforms conventional two-steps positioning," in *2009 17th European Signal Processing Conference*, Glasgow, 8 2009, pp. 1958–1962.

[15] A. Gusi-Amigó, P. Closas, A. Mallat, and L. Vandendorpe, "Ziv-Zakai bound for direct position estimation," *Navigation, Journal of the Institute of Navigation*, vol. 65, no. 3, pp. 463–475, 10 2018.

[16] O. Bialer, D. Raphaeli, and A. J. Weiss, "Maximum-likelihood direct position estimation in dense multipath," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 5, pp. 2069–2079, 2013.

[17] A. H.-P. Chu and G. X. Gao, "Multi-receiver direct position estimation tested on a full-scale fixed-wing aircraft," in *Proceedings of the 30th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017)*, Portland, OR, 9 2015, pp. 3761–3766.

[18] S. Bhamidipati and G. X. Gao, "GPS spoofer localization for PMUs using multi-receiver direct time estimation," in *Proceedings of the 30th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2017)*, 2018, pp. 2780–2784.

[19] A. R. Kumar, "Direct position tracking in GPS using vector correlator," M.S. thesis, University of Illinois at Urbana-Champaign, 2015.

[20] P. Closas, C. Fernández-Prades, and J. A. Fernández-Rubio, "ML estimation of position in a GNSS receiver using the SAGE algorithm," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing*, Honolulu, HI, 4 2007, pp. 1045–1048.

[21] J. W. Cheong, J. Wu, A. G. Dempster, and C. Rizos, "Efficient implementation of collective detection," in *International Global Navigation Satellite Systems Society*, Sydney, New South Wales, Australia, 11 2011, pp. 15–17.

[22] P. Axelrad, J. Donna, and M. Mitchell, "Enhancing GNSS acquisition by combining signals from multiple channels and satellites," *22nd International Technical Meeting of the Satellite Division of the Institute of Navigation 2009, ION GNSS 2009*, vol. 5, pp. 3117–3128, 2009.

[23] Y. Ng and G. X. Gao, "Computationally efficient direct position estimation via low duty-cycling," in *Proceedings of the 29th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2016)*, Portland, Oregon, 9 2016, pp. 86–91.

[24] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen, *A Software-defined GPS and Galileo Receiver: A Single-Frequency Approach*, 1st ed. Basel, Switzerland: Birkhäuser, 2007.

[25] Y. Ng, "Improving the robustness of GPS direct position estimation," M.S. thesis, University of Illinois at Urbana-Champaign, 2016.

[26] A. H.-P. Chu, "GPS Multi-receiver direct position estimation for aerial applications," M.S. thesis, University of Illinois at Urbana-Champaign, 2018.

[27] T. Pany, J. Dampf, W. Bär, J. Winkel, C. Stöber, K. Fürlinger, P. Closas Gómez, and J. A. García Molina, "Benchmarking CPUs and GPUs on embedded platforms for software receiver usage," in *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*, Tampa, FL, 9 2015. [Online]. Available: https://upcommons.upc.edu/handle/2117/101077 pp. 3188–3197.

[28] Inside GNSS, "More than we ever dreamed possible," *Inside GNSS*, p. 352, 2015. [Online]. Available: https://insidegnss.com/more-than-we-ever-dreamed-possible/

[29] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with CUDA," *Queue*, vol. 6, no. 2, pp. 40–53, 4 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1365490.1365500

[30] Global Position Systems Directorate, "Systems engineering and integration interface specification IS-GPS-200," 2013.

[31] T. Hobiger, T. Gotoh, J. Amagai, Y. Koyama, and T. Kondo, "A GPU based real-time GPS software receiver," *GPS Solutions*, vol. 14, no. 2, pp. 207–216, 3 2010. [Online]. Available: http://link.springer.com/10.1007/s10291-009-0135-2

[32] M. S. Braasch and A. J. Van Dierendonck, "GPS receiver architectures and measurements," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 48–64, 1999.

[33] N. F. Krasner, "GPS receiver and method for processing GPS signals," *EP Patent*, vol. 1, no. 260, p. 830, 2010. [Online]. Available: https://patents.google.com/patent/US5663734A/en

[34] F. Engel, G. Heiser, P. Mumford, K. Parkinson, and C. Rizos, "An open GNSS receiver platform architecture," *Journal of Global Positioning Systems*, vol. 3, no. 1&2, pp. 63–69, 2010.

[35] C. Ma, G. Lachapelle, and M. E. Cannon, "Implementation of a software GPS receiver," in *Proceedings of the 17th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2004)*, Long Beach, CA, 9 2004, pp. 956–970.

[36] M. Braasch, "Performance comparison of multipath mitigating receiver architectures," in *2001 IEEE Aerospace Conference Proceedings*, Big Sky, MT, 3 2001, pp. 1309–3.

[37] J. S. Warner and R. G. Johnston, "GPS spoofing countermeasures," Los Alamos National Laboratory, Tech. Rep. [Online]. Available: http://mirror.thelifeofkenneth.com/lib/electronics_archive/GPS-Spoofing-Countermeasures.pdf

[38] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned aircraft capture and control via GPS spoofing," *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, 2014. [Online]. Available: http://rnl.ae.utexas.edu/images/stories/files/papers/unmannedCapture.pdf

[39] H. Hu and N. Wei, "A study of GPS jamming and anti-jamming," in *2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS)*. IEEE, 12 2009. [Online]. Available: http://ieeexplore.ieee.org/document/5406988/ pp. 388–391.

[40] L. Chang, K. Li, and B. Hu, "Huber's M-estimation-based process uncertainty robust filter for integrated INS/GPS," *IEEE Sensors Journal*, vol. 15, no. 6, pp. 3367–3374, 6 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7004783/

[41] N. L. Knight and J. Wang, "A comparison of outlier detection procedures and robust estimation methods in GPS positioning," *Journal of Navigation*, vol. 62, no. 04, p. 699, 10 2009. [Online]. Available: http://www.journals.cambridge.org/abstract_S0373463309990142

[42] A. Wieser and F. K. Brunner, "Short static GPS sessions: Robust estimation results," *GPS Solutions*, vol. 5, no. 3, pp. 70–79, 1 2002. [Online]. Available: http://link.springer.com/10.1007/PL00012901

[43] J. P. Collins and R. B. Langley, "Possible weighting schemes for GPS carrier phase observations in the presence of multipath," University of New Brunswick, Tech. Rep., 1999. [Online]. Available: http://gauss.gge.unb.ca/papers.pdf/acereport99.pdf

[44] S. Tay and J. Marais, "Weighting models for GPS pseudorange observations for land transportation in urban canyons," in *6th European Workshop on GNSS Signals and Signal Processing*, 2013. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00942180 p. 4.

[45] S. Hewitson and J. Wang, "GNSS receiver autonomous integrity monitoring (RAIM) performance analysis," *GPS Solutions*, vol. 10, no. 3, pp. 155–170, 7 2006. [Online]. Available: http://link.springer.com/10.1007/s10291-005-0016-2

[46] L. R. Weill, "A high performance code and carrier tracking architecture for ground-based mobile GNSS receivers," in *Proceedings of the 23rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2010)*, Portland, OR, 9 2010, pp. 3054–3068.

[47] Nvidia Corporation, "CUDA toolkit documentation 9.0." [Online]. Available: https://docs.nvidia.com/cuda/archive/9.0/

[48] Nvidia Corporation, "NVIDIA developer tools overview." [Online]. Available: https://developer.nvidia.com/tools-overview

[49] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME – Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[50] G. Welch and G. Bishop, "An introduction to the Kalman filter," University of North Carolina at Chapel Hill, Tech. Rep., 1995. [Online]. Available: http://www.cs.unc.edu/ gb

[51] Nvidia Corporation, "cuBLAS library documentation." [Online]. Available: https://docs.nvidia.com/cuda/cublas/index.html#using-the-cublasLt-api

[52] National Instruments, "NI global navigation satellite system toolkits." [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/en/nid/204980

[53] Ettus Research, "USRP N200/N210 networked series." [Online]. Available: https://www.ettus.com/content/files/07495_Ettus_N200-210_DS_Flyer_HR_1.pdf

[54] Microsemi Corportation, "SA.45s chip scale atomic clock." [Online]. Available: https://www.microsemi.com/product-directory/clocks-frequency-references/3824-chip-scale-atomic-clock-csac

[55] Microsemi Corportation, "SA.45s chip-scale atomic clock options 001 and 003," Aliso Viejo, CA, Tech. Rep.

[56] Nvidia Corporation, "Jetson TX2 module." [Online]. Available: https://developer.nvidia.com/embedded/buy/jetson-tx2

[57] Nvidia Corporation, "Jetson TX2 series data sheet." [Online]. Available: http://developer.nvidia.com/embedded/dlc/jetson-tx2-series-modules-data-sheet

[58] J. Wu and J. JaJa, "Optimized strategies for mapping three-dimensional FFTs onto CUDA GPUs," in *2012 Innovative Parallel Computing (InPar)*, San Jose, CA, 5 2012, pp. 1–12.

[59] P. N. Misra, "The role of the clock in a GPS receiver," *GPS World*, vol. 7, no. 4, pp. 60–66, 1996.

[60] E. Jones, T. Oliphant, P. Peterson et al., "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/

[61] National Instruments, "NI PXIe-5672 specifications: RF vector signal generator," Tech. Rep. [Online]. Available: http://www.ni.com/pdf/manuals/372342d.pdf

[62] A. G. Blackstock, J. DeMonte IV, P. J. Highland, M. S. Brodie, and J. P. Wilder, "Demonstration of aircraft state determination with blended solution of multiple GPS receivers: Project GRIFFIN," Edwards Air Force Base, CA, Tech. Rep., 2017.

[63] Google, "Google Maps Elevation API."

[64] E. Wycoff, Y. Ng, and G. X. Gao, "Python GNSS receiver," *GPS World*, no. 26 (2), pp. 52–57, 2 2015.