IMPROVING THE STATE ESTIMATES OF A MANEUVERING TARGET
USING A CONSTRAINED GRID-BASED FILTER: A COMPARATIVE STUDY

by Mark Eric Silbert

B.S. in Computer Science, May 1982, Drexel University
M.S. in Computer Science, October 1988, Rutgers State University

A Dissertation submitted to

the Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

August 31, 2012

Dissertation directed by

Shahram Sarkani
Professor of Engineering Management and Systems Engineering

Thomas A. Mazzuchi
Department Chair of Engineering Management and Systems Engineering and Professor
of Engineering Management and Systems Engineering and of Decision Sciences
The School of Engineering and Applied Science of The George Washington University certifies that Mark Eric Silbert has passed the Final Examination for the degree of Doctor of Philosophy as of April 30, 2012. This is the final and approved form of the dissertation.

IMPROVING THE STATE ESTIMATES OF A MANEUVERING TARGET USING A CONSTRAINED GRID-BASED FILTER: A COMPARATIVE STUDY

Mark Eric Silbert

Dissertation Research Committee:

Shahram Sarkani, Professor of Engineering Management and Systems Engineering, Dissertation Co-Director

Thomas A. Mazzuchi, Department Chair of Engineering Management and Systems Engineering and Professor of Engineering Management and Systems Engineering and of Decision Sciences, Dissertation Co-Director

Timothy Eveleigh, Adjunct Professor of Engineering Management and Systems Engineering, Committee Member

Edward Lile Murphree, Professor of Engineering Management, Committee Member

Michael Stankosky, Professor of Engineering Management and Systems Engineering, Committee Member
© Copyright 2012 by Mark Eric Silbert
All rights reserved
DEDICATION

To Laney,
Elyse,
Dean,
Janet,

and in loving memory of Abe.

iv
ACKNOWLEDGMENTS

I would like to thank Professors Thomas Mazzuchi and Shahram Sarkani for the
dedication and knowledge they brought to this effort.

I am especially grateful to NAVAIR and ONR for supporting me by funding this effort.

I would like to give a special thanks to Dr. Oliver Drummond for providing early
encouragement. Dr. Drummond helped put my ideas on a solid, mathematical foundation.
With his comments and insight, he helped make me believe that I might be on to
something new, novel and useful to the data fusion community.

I also give thanks to Dr. Craig Agate for his help and guidance. He helped me better
understand some of the finer points of track filtering, especially on particle filtering, that
proved invaluable for this research.

I would like to thank Mr. Charles Rea for our many discussions and being there as a
sounding board for many of the ideas presented in this research.

And finally, I would like to thank Dr. Tim Eveleigh for reading and re-reading this
dissertation to provide helpful comments to improve it.
ABSTRACT

IMPROVING THE STATE ESTIMATES OF A MANEUVERING TARGET USING A CONSTRAINED GRID-BASED FILTER: A COMPARATIVE STUDY

Tracking maneuvering targets is an important but difficult problem. This problem occurs when tracking maneuvering boats or even people wandering around. If the target motion is linear and the measurement errors are Gaussian, then a Kalman filter provides the optimal estimates. However, since maneuvering targets do not generally exhibit linear motion, a Kalman filter will often produce poor estimates for these targets. A better approach would be to exploit the kinematic constraints of the target motion to restrict the predicted states to those where the target could have transitioned. But exploiting these constraints will produce bounded, non-Gaussian prediction distributions. A Kalman filter must assume that this prediction distribution is Gaussian. Grid-based filters (GBFs) can work with any prediction distribution so they allow the kinematic constraints to be exploited. As a result, GBFs should be very effective at producing accurate state estimates. But current implementations of GBFs typically require large memories and exponential processing. As a result, although grid-based filters should be effective, they have been avoided due to their perceived exponential computational complexity. A novel approach for implementing a GBF, called a constrained grid-based filter (CGBF), has been developed that can track targets moving in two dimensions by using a well-confined, two-dimensional grid. This paper will expose the problems with current GBFs and then explain how they are avoided in a CGBF. As a result, this grid-based approach is enormously more computationally efficient and can effectively exploit the kinematic constraints of the target. The paper will discuss how the kinematic constraints of the
target are used to restrict the possible predicted states, including the formalism to show how the predicted state from the CGBF is better bounded than from a Kalman filter.

In systems engineering, when two or more alternative solutions exist, it is important to conduct an analysis of alternatives (AoA) to determine the best systems approach. An AoA will be conducted to compare the tracking performance of a CGBF to a Kalman filter for maneuvering target scenarios. The target state estimates (position and velocity) will be analyzed via Monte Carlo analysis. This study will employ two comparison methods including the comparison paradigm presented by Kirubarajan and Bar-Shalom. The intent of this study is to determine how maneuverable the target must be to gain the benefit from a CGBF over a Kalman filter. Keep in mind that when all Kalman filter assumptions hold, it is the optimal filter. This paper will discuss the target motion model, the CGBF implementation, and the Kalman filter used for the study. The results will show that by restricting the prediction distribution based on the presumed (realistic) kinematic constraints of the target, a more informed filter will result. Thus, by violating the one key Kalman filter assumption, it will be shown that the CGBF consistently outperforms a Kalman filter, even for the linear motion target case. Furthermore, the improvement from the CGBF increases relative to a Kalman filter as the target becomes more maneuverable.

Although the CGBF is much less computational than current GBF methods, it is still more computational than a Kalman filter. The paper will discuss the grid and sample sizing needed to obtain quality estimates from a CGBF. It will be shown that the sizes are much smaller than what may be expected and that the CGBF is quite stable even when using these small sizes.
# TABLE OF CONTENTS

Dedication ................................................................................................................................. iv  
Acknowledgments ...................................................................................................................... v  
Abstract ...................................................................................................................................... vi  
List of Figures ............................................................................................................................ x

Chapter 1 - Introduction ........................................................................................................ 1  
  1.1 Background ......................................................................................................................... 6  
  1.2 Motivation .......................................................................................................................... 12  
  1.3 The Kalman Filter .............................................................................................................. 13  
  1.3.1 The 1D Kalman Filter ...................................................................................................... 14  
  1.4 Quantifying a Maneuvering Target .................................................................................... 15  
  1.4.1 The Maximum Practical Limit on the Maneuver Index .............................................. 16

Chapter 2 - The Grid-Based Filter (GBF) ........................................................................... 18  
  2.1 The 2D GBF ....................................................................................................................... 18  
    2.1.1 Defining the 1D Motion Model ................................................................................... 19  
      2.1.1.1 Determining the Kinematically-Feasible Region ................................................. 21  
      2.1.1.2 Computing the Distribution of the Feasible Region .......................................... 21  
    2.1.2 Performing the Motion Update (Target Predict) ......................................................... 22  
    2.1.3 Determining the Updated State Estimate ..................................................................... 24  
    2.1.4 Limiting the Grid Extent ............................................................................................. 26  
    2.1.5 Determining Grid Size and Sampling Size .................................................................. 27  
  2.2 The Straightforward 4D GBF ............................................................................................. 28

Chapter 3 - The Specialized 2D Constrained GBF ............................................................... 30  
  3.1 Computing the 5-Tuple ....................................................................................................... 30  
  3.2 The 2D Target Motion Model ............................................................................................ 32  
    3.2.1 Analyzing the 2D Motion Model ................................................................................ 34  
      3.2.1.1 When the Target Cannot Accelerate ..................................................................... 34  
      3.2.1.2 When the Target Cannot Turn .............................................................................. 36  
    3.2.2 Determining the 2D Kinematically-Feasible Transition Region .................................. 38  
    3.2.3 Computing the Distribution of the Feasible Transition Region .................................. 40  
    3.2.4 The Mean and Variance of the Transition Density (Single Maneuver) ....................... 43  
      3.2.4.1 The Mean and Variance of the Speed and Course ............................................... 44
LIST OF FIGURES

Figure 1-1: Aircraft tracking various surface targets via radar........................................................ 1
Figure 1-2: Ideal operating range for the three prominent filters for tracking maritime targets. .... 4
Figure 1-3: Maritime operating region versus ideal operating region of key filters.......................... 5
Figure 1-4: Ideal operating region for GBFs.................................................................................. 6
Figure 2-1: All possible initial (2D) states of a target moving in one-dimension. ....................... 19
Figure 2-2: Minimal (left) and maximal (right) displacement over duration $\tau$............................ 20
Figure 2-3: Region of feasible transition states............................................................................. 21
Figure 2-4: Distribution within the transition region (50 million samples). ................................. 22
Figure 2-5: The transition of a single cell..................................................................................... 24
Figure 2-6: After all cells have been transitioned........................................................................ 24
Figure 2-7: After summing the mass from all transition densities. ........................................... 24
Figure 2-8: Updated state distribution after masking out mass not within measurement error.... 25
Figure 3-1: The algorithm for computing the destination grid for each update. ............................ 31
Figure 3-2: A target moving clockwise(right-hand turn) around a constant radius circle............ 35
Figure 3-3: Coarse-grained bounds of the predicted positional containment region............... 39
Figure 3-4: True predicted positional containment region.......................................................... 39
Figure 3-5: Position uncertainty distribution; up to 2 maneuvers within time interval............ 42
Figure 3-6: Position uncertainty distribution; up to 4 maneuvers within time interval............ 42
Figure 3-7: Position uncertainty distribution; up to 8 maneuvers within time interval............ 43
Figure 3-8: Example plot of $f(x) = \frac{1}{x}(1 - \cos \tau x)$ for $\tau = 2$. (from wolfram.com)........ 50
Figure 3-9: Example plot of $f(x) = \frac{1}{x}\mathrm{Si}(\tau x)$ for $\tau=2$. (from wolfram.com)............ 52
Figure 3-10: Each entire cell is moved per Monte Carlo sample; mass is apportioned. .......... 67
Figure 3-11: The mirrored state $x_k',y_k'$ from the predicted state $x_k,y_k$............................. 69
Figure 4-1: Block diagram of the comparison methodology....................................................... 73
Figure 4-2: Constant velocity target; first four updates of the two filters. ................................. 80
Figure 4-3: Constant velocity target; the 5th, 10th, 15th, and 20th updates.......................... 81
Figure 4-4: Position comparison for a constant velocity target.................................................. 82
Figure 4-5: Position and velocity error plots for constant velocity target.......................... 82
Figure 4-6: State error plot for a constant velocity target......................................................... 83
Figure 4-7: Speed and course error plots.......................................................... 84
Figure 4-8: Constant velocity target, maneuver index = 1/4.................................................. 86
Figure 4-9: Constant velocity target, maneuver index = 1/2.................................................. 87
Figure 4-10: Constant velocity target, maneuver index = 3/4.................................................. 88
Figure 4-11: Constant velocity target, maneuver index = 1.................................................... 89
Figure 4-12: Constant velocity target, maneuver index = 2.................................................... 90
Figure 4-13: Constant velocity target, maneuver index = 3.................................................... 91
Figure 4-14: Maneuvering target, maneuver index = 1/4.................................................... 94
Figure 4-15: Maneuvering target, maneuver index = 1/2.................................................... 95
Figure 4-16: Maneuvering target, maneuver index = 3/4.................................................... 96
Figure 4-17: Maneuvering target, maneuver index = 1.................................................... 97
Figure 4-18: Maneuvering target, maneuver index = 2.................................................... 98
Figure 4-19: Maneuvering target, maneuver index = 3.................................................... 99
Figure 4-20: Target maneuvering schedule. ............................................................................... 102  
Figure 4-21: Peak RMS errors for CGBF and Kalman filter. (a) Position errors; (b) Velocity errors. ........................................................................................................................................... 107  
Figure 4-22: Overall RMS errors for CGBF and Kalman filter. (a) Position errors; (b) Velocity errors. ........................................................................................................................................... 108  
Figure 4-23: RMS errors for the GBF and Kalman filter during the non-maneuvering intervals. (a) Position errors; (b) Velocity errors. ................................................................................................................................. 109  
Figure 4-24: RMS errors for GBF and Kalman filter during the non-maneuvering intervals. (a) Position errors; (b) Velocity errors. ................................................................................................................................. 109  
Figure 5-1: Percentage reduction in (a) position error, (b) speed error, and (c) course error that CGBF offers over Kalman filter as grid size increases. ....................................................................................... 111
Chapter 1 - Introduction

Tracking maneuvering targets is an important but difficult problem. This problem occurs when needing to track maneuvering boats, or even people wandering around. For example, consider the problem of tracking all the boats and ships in a coastal region, as shown in Fig. 1-1. For the purposes of tracking, all the entities to be tracked are called targets. Tracking a target entails maintaining an estimate of its position and velocity. Each position and velocity estimate is called the target’s state. Each of the targets move independently with different speeds and directions. Generally, the tracking of these targets is accomplished with the use of surveillance radars mounted along the shorelines or on aircraft flying overhead. Fig. 1-1 depicts a future Broad Area Maritime Surveillance (BAMS) aircraft maintaining maritime surveillance. The surveillance radars used to track maritime traffic typically have slow scan rates which result in each target being revisited once every five seconds to once every 30 seconds, or sometimes revisited even less often.

Figure 1-1: Aircraft tracking various surface targets via radar.
To maintain track from scan to scan, a radar tracker generally assumes each target moves with constant velocity to be able to assign the new radar measurement to the existing radar track. For many targets, particularly larger ships such as cargo ships, tankers, and cruise ships, using the constant velocity assumption is quite reasonable. However, maneuvering targets, such as pleasure boats, speed boats, go-fast boats, and jet skis, often invalidate this constant velocity assumption making it difficult to maintain a track on them. Confounding this problem is that often the state estimates are not that accurate, especially in terms of estimating the target's velocity, so the accuracy of the predicted state of the target deteriorates quickly. As the accuracy in the predicted state deteriorates, it becomes more difficult to assign the new measurement to the proper track. With the slow update rates of many surveillance radars, it becomes increasingly likely for the radar measurements to be incorrectly assigned, resulting in poor tracking of the targets.

As a result, the difficulty of the tracking problem is dependent on the type and behavior of the targets to be tracked. For targets that exhibit constant or nearly constant velocity, a Kalman filter is sufficient to maintain track on them. If the targets occasionally maneuver and do not maneuver quickly, the Kalman filter may still be sufficient however, these types of targets are generally better tracked using an interacting multiple model filter (IMM). An IMM is basically a bank of Kalman filters executing simultaneously for each target track where the motion model is differenced across the filters. For example, a common implementation of an IMM is to form a bank of two filters where one filter uses a constant velocity target motion while the other assumes a (mildly) maneuvering target motion. Often, the maneuvering target motion model is implemented the same as the constant velocity motion model but employs a larger process noise. The state estimates from the multiple filters are then combined using a weighted average where with the covariance from each of the filters is used to compute the weights. Note that since multiple Kalman filters are executing for each target track, the IMM is more computational than a Kalman filter. In fact, if there are $k$ motion models in the IMM, then the IMM is larger than $k$ times more
computational than the Kalman filter. Kirubarajan and Bar-Shalom showed in [55] how maneuverable a target must be to gain benefit from using an IMM over a Kalman filter. Their results showed it doesn't take much of a maneuvering target to obtain benefit from an IMM.

If the targets maneuver often and/or generate (relatively) high-G maneuvers, the Kalman filter and the IMM will often be insufficient to track these targets accurately. For these targets, a more robust filter method must be employed. Particle filters are one of the key alternative approaches for dealing with these highly maneuvering targets. Particle filters can be thought of as thousands of simple Kalman filters running simultaneously in an attempt to approximate the state estimate and covariance numerically. While particle filters can handle a wide range of target dynamics, they are not without their shortcomings. First, they are much more computational than an IMM. Second, the particle filter is reliant on a having the particles approximate the underlying distribution. Maintaining this distribution turns out to be problematic because there is little theoretical treatment for how to ensure the particles will appear to be from the proper distribution. As a result, if the underlying distribution is not maintained, the particle filter will eventually fail causing a loss of the target track. This failure is called particle collapse (or particle degeneracy).

There are two aspects of the tracking problem that exacerbate the particle collapse problem: high dimensionality and accurate measurements. As the number of dimensions in the filter increases, the domain space increases exponentially requiring exponentially more particles to properly approximate the underlying distribution. This problem is often referred to as the curse of dimensionality. Thus, particle filters are really meant for low dimension tracking problems. Fortunately, maritime surveillance requires a relatively low dimension filter since the state estimates for the targets are usually only 4D (2D position plus 2D velocity). However, measurement accuracy is much more problematic for particle filters used for maritime surveillance tracking. In general, unlike Kalman filters, particle filters prefer noisy or large errors in the measurement. The reason is that as the accuracy in the measurement increases (i.e.,
becomes tighter due to less error), the more particles are needed to assure a sufficient number of particles will survive the measurement update. Requiring much more particles again leads to exponential processing which typically leads to the demise of the particle filter. Kalman filters and IMMs, on the other hand, provide better state estimates as the measurement accuracy improves. This difference between Kalman filters and particle filters is often referred to as the \textit{SNR (signal-to-noise) problem}. Kalman filters prefer measurements with high SNR while particle filters prefer low SNR. In the maritime surveillance domain, although the time between measurements updates tends to be large, the measurements tend to be highly accurate, especially in range. So the measurements are high SNR. Thus, particle filters often fall short for tracking highly maneuvering maritime targets due to the high SNR measurements. The ideal operating range for these three filter types is summarized in Fig. 1-2. The graph on the left shows the ideal operating range for the filters based on dimensionality of the filter and target maneuvering. The right-hand graph is the ideal operating range for the filters based on magnitude of the measurement error and target maneuvering. From the figure, it can be seen that particle filters are well-suited for dealing with maneuvering targets but Kalman filters and IMMs are well-suited for high-dimension and small measurement error. Fig. 1-3 overlays the operating range required for maritime target tracking. As the figure shows, none of these filters completely cover the operating range required for the tracking of all maritime target behaviors.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Ideal operating range for the three prominent filters for tracking maritime targets.}
\end{figure}
As a result, there is no perfect track filter solution to deal with tracking of all types of maritime targets. Each filter method offers some advantages but falls short in some aspect of the surveillance problem. Thus, a track filter is needed that can work with accurate measurements yet deal with highly maneuvering targets. Fortunately, this filter need only operate at a relatively low dimension (i.e., 4D).

This research focused on developing a filter that can better address the operating region needed for tracking all maritime targets. The research began by re-considering the grid-based filter (GBF). A GBF can be thought of as a method that bridges the Kalman filter-like methods to the particle filter methods. The idea behind a GBF is to discretize the region where the targets are presumed to reside into a set of cells. The next step is to model how the targets would transition to other cells over time. Finally, the measurements are used to increase the probability of the target being in a cell. Since the underlying distributions are calculated numerically like the particle filter, it can deal with maneuvering targets. Furthermore, since the measurement error distributions need to be discretized as well, the GBF prefers accurate measurements just like the Kalman filter. The downfall of a GBF is that, like a particle filter, it suffers from the curse of dimensionality. However, a GBF is even more quickly beaten by dimensionality. From a computational standpoint, these filters are practically limited to only two dimensions. Fig. 1-4

Figure 1-3: Maritime operating region versus ideal operating region of key filters.
compares the ideal operating region for GBFs to the operating region needed for tracking maritime targets. GBFs offer a good fit in terms of handling the measurement error requirements but unfortunately work at lower dimensions than is required. Notice, however, the GBF needs to deal with only two more dimensions than its ideal operating region. Perhaps there is a way to modify a GBF so it could deal with four dimensions. If so, then GBFs would completely cover the operating region required for maritime tracking of all targets. This research set out to extend the GBF method to allow it to work on the four dimensional filtering problem that occurs in maritime target tracking. The focus was to find a practical approach for employing a 4D GBF that can address maritime target tracking and remain computationally feasible. The research led to the development of a modified GBF, called a constrained grid-based filter (CGBF). This filter fully captures the required operating region for tracking all types of maritime targets. To show the benefit of the CGBF, its performance will be compared to a Kalman filter against a target exhibiting differing degrees of maneuvering.

![Figure 1-4: Ideal operating region for GBFs.](image)

### 1.1 Background

Over the years, many systems have been developed to address the target tracking problem. However, tracking multiple maneuvering targets remains a notable weakness of these systems. At the heart of a tracking system is the target track filter. The track filter computes the most likely
target state given all the measurements collected on that target. This most likely state is typically based on the one that minimizes the mean squared error (MSE). In the early 1960s, Rudolf Kalman published a paper offering a “new approach to linear filtering and prediction problems” [54]. In this paper, Kalman described a filter that yields minimum MSE state estimates for systems that exhibit linear dynamics. This filter became known as the Kalman filter. Since its debut, it has been extensively studied and has emerged as the premier method for target track filtering due to its computational efficiency and its filtering effectiveness. The Kalman filter is now used in many applications from GPS tracking to air traffic control [47]. If the motion is linear and the measurement errors are Gaussian, then a Kalman filter can provide accurate estimates of the target state efficiently through analytical formulization. Furthermore, under these conditions, the Kalman filter can be shown to be “optimal”, meaning no other method can produce a better estimate (on the average) than this filter [11]. However, although the Kalman filter is efficient and effective for computing state estimates of a moving target, it can produce poor estimates when tracking a maneuvering target. The key reason that these poor estimates occur is because the Kalman filter is a linear filter. It assumes the dynamics of the target to be tracked can be modeled as a linear system [49], but maneuvering targets do not generally exhibit such motion. The reason the target motion needs to be linear is due to a property of the normal distribution. A normal random variable remains normal through a linear transform [12]. If the transform is not linear, there is no such guarantee that the normal variable will remain normal. To remain the optimal solution, the Kalman filter requires the target motion to be linear. If the distributions are not normal, the Kalman filter still is the best linear filter [11].

The Kalman filter includes a process noise parameter which provides a means to capture the uncertainty in the target motion. Thus, a straightforward way to allow the Kalman filter to track maneuvering targets would be to simply employ a large process noise. The idea is to treat the non-linear target motion simply as larger uncertainty in its linear motion. However, increasing the
process noise parameter results in larger errors in the computed state estimates, as well as larger uncertainties in these state estimates [10]. These larger errors and uncertainties cause more ambiguity making the tracking process more difficult. It became clear that more research was needed to address maneuvering targets.

Since the Kalman filter was so effective, researchers began by looking for enhancements and/or extensions to the Kalman filter. One of the early extensions was the extended Kalman filter (EKF). Radar was (and still is) the dominant sensor used for target tracking. Radars were advancing as well. One of the early refinements to a radar system was the ability to measure the closing rate of the target. This closing rate is known as the range rate. Actually, the range rate is a closing rate only when the target is moving radially towards, or away from, the radar. If the target is moving along some other path, then the range rate is measuring only that component of the target's velocity that is moving radially towards (or away from) the radar. This radial component is proportional to the cosine of the difference of the angles between the target's course and the radial angle from the target to the radar. This radial angle to the target is often called the azimuth angle. Cosine is a non-linear function. Since this range rate is a function of cosine, the (linear) Kalman filter had to be extended to address this non-linear quantity. The resulting filter, called an EKF, includes this range rate measurement into the calculations by performing a first-order approximation of its additional information. Since the Kalman filter is inherently a linear filter, the intent is to linearize the range rate information. But the cosine cannot always be estimated well with only a first-order approximation, so the EKF is a suboptimal filter [11]. As a result, an EKF will at times perform no better than a Kalman filter and could even perform worse by failing to properly track the target.

Several other extensions to the Kalman filter have subsequently emerged including iterated extended Kalman filters (IEKFs), unscented Kalman filters (UKFs), and interacting motion model Kalman filters (IMMs) [23]. All these methods have the advantage that they are based on the
Kalman filter and have closed-form solutions. This makes them computationally efficient and well-grounded in a mathematical formulation. However, all these methods make assumptions about the maneuvering target that are often not true; this could lead to erroneous estimates.

In addition to radars becoming better, computers and computational power had also improved. These improvements let research pursue more numerical (and thus, more computational) methods. Two methods that emerged were grid-based filters (GBFs) and particle filters (PFs) [4]. These methods were broadly classified as sequential Monte Carlo (SMC) techniques. A key feature of these methods is that they allow many of the restricted assumptions placed on the closed-form methods to be relaxed. It was now possible to consider more general, non-linear target dynamics. Any presumed motion of the target could now be modeled without being limited to linear dynamics. As pointed out by Li and Jilkov [59], a good target model is vital to achieve quality state estimates. In addition, the measurement errors could be modeled more realistically. For example, all the Kalman filter-like methods assume Gaussian error on the measurements even though radar measurements are generally not Gaussian (at least not in rectilinear space).

The idea behind a GBF is to first discretize the region where the targets are presumed to reside into a set of cells. The next step is to model how the targets would transition to other cells over time. Finally, measurements are used to increase the probability of the target being in a cell. Unfortunately, it was quickly realized that even for the faster computers that were becoming available, this approach would be far too computational to execute in real-time. Although the grid-based filter approach is intriguing, due to its computational burden, it never really gained any widespread use. Indeed, there are very few papers in the open literature that focus on these types of filters. What was needed was a way to achieve the benefit of the grid-based filter without the exponential computations it required.
The solution seemed to emerge in the 1990s with the development of particle filters. The idea behind particle filters is to “smartly” sample the region, creating “particles” for those areas with relatively high likelihood of containing the target. Each particle represents a possible state for the target. The particles are weighted corresponding to how likely the target is deemed to be in that state. At each measurement update, the particles are propagated according to the presumed target motion to generate a proposal density [27]. The next step is to use the samples along with the measurement to re-weight the samples so they approximate the update distribution. This process of re-weighting the samples to make them look like ones from the update distribution is referred to as importance sampling [4]. After the importance sampling, re-sampling is performed where the weights are used to generate a new set of (equally weighted) samples. The purpose of the re-sampling is to generate new samples from the higher weighted (i.e., more likely) samples, thereby effectively discarding the lower weighted (i.e., less likely) ones. However, just because a particle has more weight than another particle does not mean it will always be a more correct state estimate. Therefore, care needs to be taken when deciding which particles to discard. It could turn out that only a few particles will have most of the weight, meaning the entire update distribution will be based on only these few samples. When this occurs, the particle filter fails. This failure is called particle collapse (or particle degeneracy) and results in a loss of the track for the target [34]. Unfortunately, little theory exists to determine how the importance sampling should be done to avoid particle collapse [69]. However, it is known that high dimensionality and/or accurate measurements exacerbate the particle collapse problem [34]. As the number of dimensions in the filter increases, the domain space increases exponentially requiring exponentially more particles. This problem is often referred to as the curse of dimensionality. While Kalman filter-like methods welcome accurate measurements, particle filters have trouble dealing with these. The reason is the more accurate the measurement, the more particles are needed to ensure there will be many located where the measurement falls. Only the particles that are within the measurement region will have significant weight; all the other particles will have very low weight. If only few
particles end up in the measurement region, particle collapse is imminent. As a result, importance sampling remains a key weakness of particle filters. Despite this problem, particle filters remain the key SMC method with research continuing to look for ways to improve the technique. For example, Daum and Huang have recently developed a particle filter which uses particle flow and log-homotopy to completely eliminate the need for importance sampling and re-sampling [32][25]. It is not clear, however, how this method compares to Kalman filter methods since there is currently no published work on such a comparison.

As previously mentioned, due to the computations required for a grid-based filter, this type of filter has been mostly ignored. This is unfortunate because GBFs offer all the same advantages as particle filters yet have no need for down-sampling. The focus of this research is to find a practical approach to employing a grid-based filter that maintains all its advantages but is computationally feasible. The research led to the development of a modified GBF called a constrained grid-based filter (CGBF). It will be shown that for targets moving in two dimensions, such as maritime targets, this specially crafted grid-based filter runs faster than real-time on a laptop PC.

Of course, the key test of a grid-based filter is to see how its tracking performance stands up to the more conventional methods such as a Kalman filter. Therefore, a trade study will be conducted. Conducting a trade study is a form of analysis of alternatives (AoA), which is a key endeavor of systems engineering [70]. Trade studies involve comparing alternative solutions to a particular problem to determine which solution contributes to the overall best system design. In order to determine the best solution, performance metrics must be determined at the subsystem and/or system level. Toward this end, this research studies the comparison of this novel grid-based track filter against the standard Kalman filter based on the tracking accuracy against a maneuvering target.
1.2 Motivation

Designers of surveillance and tracking systems must balance a number of factors when selecting solution technologies. These choices include the type of sensing and associated sensor processing that complement expected target phenomenology. System engineers must trade off the needs for target temporal, spatial, and identification accuracies against platform and sensor capabilities. The design space is further complicated by the expected target dynamics and behaviors, target density, environmental conditions (e.g., sea state), sensor standoff range, and computational loading. Taken together, all these constraints and opportunities yield a multi-dimensional trade space that more often than not, is mapped by rules of thumb and simplifying assumptions that may or may not hold true upon system deployment. What is needed are deterministic means to allow system engineers make better decisions in allocating surveillance and tracking functional performance requirements to sensing, platform, and processing technologies. Toward this end, this research is focused on the processing technologies, with particular emphasis on the target track filter.

The target track filter is the key part of a target tracking system. Thus, the performance of a tracking system can be improved by improving the track filter. But how can we improve upon the approaches that we already have? It seems the key is to focus on utilizing the kinematic constraints of the target. Instead of modeling the target maneuvering as increased motion uncertainty, a better solution would be to exploit the kinematic constraints of the target. The idea is to limit the predicted states to only those that are possible for the target to achieve kinematically. But by imposing the kinematic constraints to limit the predicted state, the resulting prediction distribution will not be Gaussian. Therefore, a paradigm must be selected that will allow exploiting of the presumed target's kinematic constraints and non-Gaussian prediction distributions. The SMC methods provide this paradigm which includes grid-based filters and particle filters. Grid-based filters were selected for this study because they are more intuitive and
seem more of a bridge between closed-form methods, such as the Kalman filter, and particle filters. Furthermore, as shown in Fig. 1-4, if a grid-based filter can be extended to work in 4D, it will completely cover the required operating range for tracking maritime targets. In addition, grid-based filters avoid the importance sampling problem that plagues particle filters. These reasons provide ample justification to focus on the processing technology by looking for a means to extend grid-based filter methods.

### 1.3 The Kalman Filter

The Kalman filter equations are presented. There are many references to get a detailed discussion of this filter [54] [11] [17]. When all the Kalman filter assumptions hold, it is the standard approach for determining the optimal (i.e., minimum MSE) estimate of the target state based on the observed measurements. The filter has two steps: 1) the motion update step, commonly called the target predict, and 2) the estimation update. These two steps are repeated for each new measurement update.

Assume the $n^{th}$ measurement occurs at time $t_n$. Let the notation $X_{[n|k]}$ mean the state estimate at time $t_n$, using $k$ measurements. Suppose after $n-1$ measurements, the estimated target state is $X_{[n-1|n-1]}$ with covariance $P_{[n-1|n-1]}$, where the state is a $d$-dimensional vector and the covariance is a matrix of dimension $d \times d$. Let $\Phi$ be the (linear) dynamics model for the target motion and $Q$ be the process noise on that motion. Then the predicted target state and covariance at time $t_n$ is:

$$X_{[n|n-1]} = \Phi X_{[n-1|n-1]} \tag{1.1}$$

$$P_{[n|n-1]} = \Phi P_{[n-1|n-1]} \Phi^T + Q \tag{1.2}$$

where the 'T' in the superscript is for matrix transpose. The notation $X_{[n|n-1]}$ means the state estimate at time $t_n$, using $n-1$ measurements, i.e., before the $n^{th}$ measurement is included.
The updated target state estimate can then be determined. Let \( Z_n \) be the \( n \)th measurement and \( R_n \) be the covariance of the measurement errors on that measurement. The measurement is an \( m \)-dimensional vector and the measurement covariance is an \( m \times m \) matrix. Finally, let \( H \) be the observation matrix of size \( m \times d \). The observation matrix maps the target state space into the measurement space. Then the Kalman gain, \( K \) is given by:

\[
K = P_{[n|n-1]}H^T(HP_{[n|n-1]}H^T + R_n)^{-1} \tag{1.3}
\]

where the '-1' superscript means matrix inverse. The updated state estimate and covariance are:

\[
X_{[n|n]} = X_{[n|n-1]} + K(Z_n - HX_{[n|n-1]}) \tag{1.4}
\]

\[
P_{[n|n]} = (I - KH)P_{[n|n-1]} \tag{1.5}
\]

where \( I \) is the identity matrix.

1.3.1 The 1D Kalman Filter

It is instructive to see what the Kalman filter equations reduce to for the 1D case. In 1D, all the Kalman filter variables become scalars with the effect of performing estimation of a single parameter using a set of scalar measurements. In 1D, \( X = x \), \( P = \sigma^2 \), \( \Phi = 1 \), \( Q = 0 \), \( H = 1 \), \( Z = z \), and \( R = \sigma_R^2 \). Substituting these scalars into the Kalman filter predict equations, yields:

\[
x_{[n|n-1]} = x_{[n-1|n-1]} \tag{1.6}
\]

\[
\sigma_{[n|n-1]}^2 = \sigma_{[n-1|n-1]}^2 \tag{1.7}
\]

Thus, the predicted state is just equal to the previous state. Continuing with the Kalman filter update equations, yields:

\[
k = \sigma_{[n|n-1]}^2(\sigma_{[n|n-1]}^2 + \sigma_R^2)^{-1} = \frac{\sigma_{[n|n-1]}^2}{\sigma_{[n|n-1]}^2 + \sigma_R^2} \tag{1.8}
\]
The result shows that the Kalman gain is the ratio of the predicted uncertainty over the total uncertainty, where the total uncertainty is the sum of the predicted uncertainty plus the measurement uncertainty. Therefore, the gain is always $0 < k < 1$. The gain tends towards zero as the uncertainty in the measurement dominates over the predicted uncertainty. And conversely, the gain tends towards one as the predicted uncertainty dominates over the measurement uncertainty. The updated state estimate is the weighted average of the predicted state and the new measurement. And finally, the updated variance is the product of the uncertainties over the sum of the uncertainties. Therefore the updated variance follows the same rule used for resistors in parallel [72]:

$$\frac{1}{\sigma^2_{n|n}} = \frac{1}{\sigma^2_{n|n-1}} + \frac{1}{\sigma^2_R}$$ (1.11)

1.4 Quantifying a Maneuvering Target

Kirubarajan and Bar-Shalom quantified the maneuverability of the target by a maneuver index. The maneuver index is a unit-less quantity formed as the ratio of the uncertainty in the target position due to motion uncertainty over the uncertainty in the measurement. The index is defined as:

$$\lambda = \frac{\sigma^2_{\alpha} T^2}{\sigma_R}$$ (1.12)
where $\sigma_a$ is the process noise standard deviation (in each dimension), $T$ is the time between updates, and $\sigma_R$ is the measurement error standard deviation (in each dimension)[11][55].

Although there are no actual limits on the maximum value, it will be shown why the maneuver index should not exceed three in practice. As Kirubarajan and Bar-Shalom point out, index values less than a half are considered low maneuvering target problems. An index value above a half indicates tracking a high maneuvering target and thus, is more difficult for the Kalman filter to obtain accurate state estimates.

### 1.4.1 The Maximum Practical Limit on the Maneuver Index

To understand why the maximum value of the maneuver index value should be less than three, consider the predicted covariance equation (1.2). The uncertainty in the predicted covariance is driven by process noise, $Q$. The positional uncertainty in $Q$ is proportional to the positional uncertainty due to target motion over the duration of the update time interval.

Therefore:

$$\sigma_{[n|n-1]} \propto \sigma_a T^2$$  \hspace{1cm} (1.13)

where ‘$\propto$’ means “proportional to.” Assume the predicted position uncertainty is a constant, $\lambda$, times bigger than the error uncertainty in the measurement. In other words, the maneuver index can be thought of as the multiplier of how much bigger the predicted position uncertainty is than the measurement uncertainty. Therefore:

$$\sigma_{[n|n-1]} = \lambda \sigma_R$$ \hspace{1cm} (1.14)

Then from (1.9), the updated position is:

$$x_{[n|n]} = \frac{\lambda^2 \sigma_R^2 z_n + \sigma_R^2 x_{[n|n-1]}}{\lambda^2 \sigma_R^2 + \sigma_R^2} = \frac{1}{\lambda^2 + 1} \left( \lambda^2 z_n + x_{[n|n-1]} \right)$$ \hspace{1cm} (1.15)
Consider a few cases for $\lambda$. If $\lambda = 0$, meaning there is no uncertainty, \textit{(i.e., perfect confidence)} in the prediction, then (1.15) yields $x_{[n|n]} = x_{[n|n-1]}$, as expected. If $\lambda = 1$, meaning the uncertainty in the prediction is identical to the uncertainty in the measurement, then (1.15) yields $x_{[n|n]} = \frac{1}{2}(z_n + x_{[n|n-1]})$ which is simply the average of the prediction and the measurement. And finally, consider the case $\lambda \gg 1$ meaning large uncertainty in the prediction. For $\lambda \gg 1$, (1.15) yields $x_{[n|n]} \approx z_n$, which says that as the uncertainty in the prediction gets large, the estimate becomes increasingly more reliant on just using the measurement.

So the question is: how large can $\lambda$ get before the estimate is basically just the measurement? To answer this question, let $\gamma$ be the proportion of the updated estimate that is only from the measurement. For example, if $\gamma = 0.9$, then 90% of the estimate is from using the measurement alone. Since the updated position is a weighted average of the prediction and the measurement, it should be clear from (1.15) that this occurs when:

$$\frac{\lambda^2}{\lambda^2 + 1} = \gamma$$

(1.16)

Solving for $\lambda$, yields:

$$\lambda = \sqrt{\frac{\gamma}{1 - \gamma}}$$

(1.17)

Using $\gamma = 0.9$, $\lambda = \sqrt{\frac{0.9}{1 - 0.9}} = \sqrt{9} = 3$. So once the predicted position uncertainty is three times larger than the measurement uncertainty, the Kalman filter is doing little more than simply using the new measurement as the updated positional estimate. As a result, the maximum practical limit for the maneuver index is three when using a Kalman filter. Interestingly, Kirubarajan and Bar-Shalom must have realized this fact but didn’t mention why they limited $\lambda$ to 2.5 in their study.
Chapter 2 - The Grid-Based Filter (GBF)

A GBF is a sequential Monte Carlo (SMC) type method that can be better thought of as a discretized-space implementation of a Kalman filter. Like a particle filter, the GBF allows for dynamic, non-linear target motion and non-Gaussian measurement models but without the need for importance sampling. Furthermore, GBFs can offer more accurate target position uncertainty distributions since they are captured directly into the grid structure. To better explain how a GBF works, consider a simple example of a target moving in one-dimension (1D), say along the \( x \) axis. Since the target is moving in 1D, it has a 1D position and a 1D velocity. Thus, a 2D grid is needed to maintain all of the possible target states.

2.1 The 2D GBF

Continuing with the example, suppose we want to track this target. At each measurement time update, the position of the target is observed, with some measurement error. The goal is to provide the best estimate of target's position and velocity, at each moment in time based on the observed measurements. The position and velocity of the target is referred to as the target state. Assume that the position of the target is measured with error that is bounded by some known maximum error, \( E \). Thus the measurements are \( z_i = w_i + e_i \), where \( w_i \) is the true position of the target at time \( t_i \) and \( e_i \) is \(-E \leq e_i \leq +E\).

For simplicity, assume the measurement errors are uniformly distributed. Also assume that the maximum acceleration and maximum speed of the target are known. Let this known maximum acceleration be \( A \) m/s\(^2\) and the maximum speed be \( S \) m/s. Therefore, upon the arrival of the first measurement, \( z_1 \), at time \( t_1 \), the true target position must be somewhere between \( z_1-E \) and \( z_1+E \) and its velocity must be between \(-S\) and \(+S\). All of the possible target states can be represented as a two-dimensional plot where position is plotted along the horizontal axis and the velocity is along the vertical axis, as Fig. 2-1 illustrates.
Figure 2-1: All possible initial (2D) states of a target moving in one-dimension.

The region in Fig. 2-1 is divided up into cells, forming a grid. Each cell within the grid can be thought of as a particular possible state of the target at time $t_1$. Let each cell contain the probability of the target being in that state. Since the measurement errors are assumed to be uniformly distributed, if all possible initial speeds of the target are equally likely, then each cell in the grid has equal probability.

### 2.1.1 Defining the 1D Motion Model

Suppose at a later time, $t_2$, the target is measured at $z_2$. Like a Kalman filter, the predicted state of the target must first be determined. For the 1D case, the target motion can be modeled as:

\[
\begin{align*}
\dot{x}_2 &= \dot{x}_1 + aT \\
\Delta x &= \dot{x}_1 \tau + \frac{1}{2}aT^2
\end{align*}
\]

where $T = t_2 - t_1$, and the acceleration $a$, is bounded by the target's presumed maximum acceleration, $A$. As a result of this bound on acceleration, if the initial velocity of the target is truly $\dot{x}_1$, then its ending velocity, $\dot{x}_2$, must be between $\dot{x}_1 - AT$ and $\dot{x}_1 + AT$. Similarly, the displacement of the target must be between $\dot{x}_1 T - \frac{1}{2}AT^2$ and $\dot{x}_1 T + \frac{1}{2}AT^2$. Note that displacement and final velocity are correlated. For example, it's impossible for the target to achieve the maximal displacement without also achieving its maximal velocity. The relation between ending velocity and displacement can be calculated. This is found by computing the minimal and
maximal displacement as a function of the initial and ending velocities. Consider Fig. 2-2. It shows the initial and ending velocities over the time interval, $T$. The minimal displacement occurs when the target decelerates as fast as it can and then accelerates as fast as it can, so that it reaches a specified ending velocity at the end of the time interval (left-hand figure). Let $\tau$ be the time within the interval (i.e., $0 \leq \tau \leq T$) when that target should 'switch' from decelerating to accelerating. Conversely, the maximal displacement occurs when the target accelerates as fast as it can and then decelerates as fast as it can to achieve the specified ending velocity (right-hand figure).

**Figure 2-2: Minimal (left) and maximal (right) displacement over duration $\tau$.**

Since the graphs in Fig. 2-2 are velocity versus time, from calculus the displacement is simply the area under the curve. It should be clear from the figure that the area is the sum of two trapezoids. Therefore, the minimal displacement, $\Delta x_L$ is:

$$\Delta x_L = \frac{1}{2} [\dot{x}_1 + (\dot{x}_1 - A\tau)] \tau + \frac{1}{2} [\dot{x}_2 + (\dot{x}_2 - A(T - \tau))] (T - \tau) \quad (2.3)$$

The time $\tau$, when the target needs to switch from decelerating to accelerating, is found as follows:

$$\dot{x}_1 - A\tau + A(T - \tau) = \dot{x}_2 \quad (2.4)$$

$$\tau = \frac{1}{2} \left( T - \frac{\dot{x}_2 - \dot{x}_1}{A} \right) \quad (2.5)$$
The maximal displacement is found similarly.

### 2.1.1.1 Determining the Kinematically-Feasible Region

By varying the ending velocity from $\hat{x}_1 - AT$ to $\hat{x}_1 + AT$, the region of all possible transition states can be calculated. The resulting region is shown in Fig. 2-3. The region forms a “surfboard” shape with pointy ends. In contrast, a Kalman filter assumes this transition region is elliptical with infinite extent due to its Gaussian distribution assumption. Clearly, the region is certainly not infinite; it is quite finite. Any states outside of the surfboard-shaped region are not feasible. The Kalman filter, on the other hand, assumes no such restriction. Since the region is tighter than what a Kalman filter assumes, exploiting this constraint should result in tighter (i.e., reduced uncertainty and likely more accurate) estimates of the target state.

![Figure 2-3: Region of feasible transition states.](image)

### 2.1.1.2 Computing the Distribution of the Feasible Region

Although the region boundaries can be easily calculated, the distribution of probability mass within the region is much more problematic. As pointed out, Kalman simply assumes the distribution is Gaussian. Since the region is finite and not elliptical, the probability density cannot be Gaussian. To estimate the probability density within the surfboard, a Monte Carlo analysis was performed. The target's acceleration was varied uniformly over the interval [-A, +A], per the
assumed distribution on target acceleration. However, in addition to varying the acceleration, it is also necessary to define the number of times the acceleration changes over the time interval. As shown in Fig. 2-2, at least two acceleration changes are required to obtain the minimal and maximal displacements. Thus, up to two acceleration changes were allowed: the first at the beginning of the time interval and the second some time later within the interval. Fig. 2-4 shows the results based on 50,000,000 random samples where each sample used two randomly selected acceleration changes. As expected, all samples remained within the surfboard region. Interestingly, the distribution appears elliptical toward the center of the surfboard shape but becomes elongated and pointy toward the ends. The fact that the distribution appears somewhat Gaussian in the middle is probably why the Kalman filter does reasonably well with its simplistic Gaussian assumption.

![Distribution within the transition region (50 million samples).](image)

**2.1.2 Performing the Motion Update (Target Predict)**

Now that each state transition (subject to the presumed target's kinematic constraints) has been established, this transition distribution is applied to every possible target prior state. A second grid is used to capture all the transitioned probability (*i.e.*, mass), as shown in Figs. 2-5
through 2-7. Figure 2-5 shows the transition for one grid cell. Figure 2-6 shows the second grid after all the cells from the first grid have been transitioned. As the cells are transitioned, the mass that falls into each cell of the second grid is accumulated resulting in the predicted distribution, as shown in Fig. 2-7.

The best estimate of the target’s state is found using the probabilities to compute the expected state. For example, to find the best estimate of the target’s position, the probabilities within each column are summed to form the probability of the target being at that particular location. Forming the expected values in position and velocity of the resulting prediction distribution (Fig. 2-7), would be the predicted state. The variance and covariance parameters of the distribution form the predicted covariance matrix. The calculation of these parameters is as follows. Let \( p_{ij} \) be the probability mass in cell \((i,j)\). Let \( x_j \) be the position at the center of column \( j \). And finally, let \( v_i \) be velocity and the center of row \( i \). Then,

\[
\bar{x} = \sum_{j=1}^{n} \left[ x_j \sum_{i=1}^{n} p_{ij} \right] \tag{2.6}
\]

\[
\bar{v} = \sum_{i=1}^{n} \left[ v_i \sum_{j=1}^{n} p_{ij} \right] \tag{2.7}
\]

\[
\sigma_{x}^2 = \sum_{j=1}^{n} \left( x_j - \bar{x} \right)^2 \sum_{i=1}^{n} p_{ij} \tag{2.8}
\]

\[
\sigma_{v}^2 = \sum_{i=1}^{n} \left( v_i - \bar{v} \right)^2 \sum_{j=1}^{n} p_{ij} \tag{2.9}
\]

\[
\sigma_{xv}^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} \left( x_j - \bar{x} \right) \left( v_i - \bar{v} \right) \tag{2.10}
\]
2.1.3 Determining the Updated State Estimate

This predicted distribution along with the next measurement is used to calculate the next estimated state. The measurement probability distribution is first discretized into cells sized to match the size of cells of the prediction distribution. Then, this discretized measurement
distribution is aligned and overlaid onto the prediction distribution. A cell-wise multiplication of the probabilities is performed for each cell. If the measurement errors are bounded as assumed, then all mass outside the error bounds of the measurement are simply discarded. Since it was assumed that the measurements errors do not exceed a known bound $E$, these outcomes cannot be possible (i.e., its cell-wise multiplication would be zero). This is illustrated in Fig. 2-8. As a result, this mass does not need to be accounted for. If the measurement errors are assumed to be uniformly distributed, there is no need to perform the cell-wise multiplication. Once the measurement update is complete, re-normalization is performed to assure the remaining mass sums to unity.

Figure 2-8: Updated state distribution after masking out mass not within measurement error.

Although it was assumed that the acceleration distribution was uniform, it does not need to be. If some target movements are deemed more likely than others, they can be weighted accordingly. However, even if in the initial state, the measurement errors, and the accelerations are all assumed to be uniformly distributed, the resulting distribution of the update state will not be uniformly distributed. It will reflect the uncertainty distribution of all next possible target states. Depending on the target motion assumed, the distribution will not likely be normally distributed either. In fact, it could even be multi-modal. More commonly, the distribution
becomes asymmetric (unlike the Gaussian assumption) due to the edges induced by the measurement bounds.

Using (2.6) - (2.10) on the resulting measurement update distribution (Fig. 2-8), yields the updated state estimate and covariance.

2.1.4 Limiting the Grid Extent

This grid-to-grid transition process is repeated for each measurement update. Therefore, only two grids are needed since the originating grid becomes reusable after the update is complete. The fact that the grids are reusable, suggests the first computational efficiency that was employed in this study. In the literature, the grids are placed over the entire region of interest [6]. In this study, the grids were placed only over the region of the measurement. But this is possible only if the measurements have finite extent. That is the reason it was assumed the measurements are bounded by some known maximum error. So, instead of requiring very large grids to fully capture the region of interest, much smaller grids can be used that need to only capture the measurement area. Namely, each new grid is centered on the new measurement. However, this refinement raises a different concern. In the discussion, it was assumed that it was known that the new measurement should be assigned to this track. If there were multiple measurements, such as in a multi-target environment, then there would be multiple tracks (i.e., grids). In this case, it would first be necessary to determine to which grid the new measurement should be assigned. This is the measurement-to-track assignment problem. In a Kalman filter, this decision is usually based on computing the Mahalanobis distance to each track and selecting the track with smallest distance [17][11]. For the grid-based filter, each grid would be temporarily updated with the new measurement, but only the grid with the highest probability (before normalizing) would be kept. All of the other temporary grids would be discarded (i.e., re-used). Although the grid-based filter can directly handle the measurement-to-track assignment issue, it will be assumed for the
remainder of this paper, that there is only one target and all measurements are from detecting that target.

Furthermore, placing the grids over each measurement alleviates another problem with current GBFs. If the grid is placed over the region instead, then the GBF cannot deal with a target that temporarily transitions outside the defined region of interest. If a target moves outside the defined region for a few updates and then returns to the region, the probability in the grid could go to zero causing a loss of track on the target. Therefore, GBFs that place the grid over a region need to deal with this boundary-edge problem but by placing the grids over the measurements, the boundary-edge problem is eliminated.

2.1.5 Determining Grid Size and Sampling Size

Note that the number of cells used in each dimension was not specified. Of course this is a parameter, but clearly, the more cells, the better the target state distribution information. However, increasing the target state distribution information comes at a high price of needing many more Monte Carlo samples. If the grids were placed over the entire region of interest (as the literature describes), 100 cells or more could be required for each dimension. But by placing the grid only over the new measurement, many fewer are needed. The size of these smaller grids will be addressed later but for now consider an $n \times n$ grid. The distribution of each cell within the grid must be determined. To get a good (numerical) estimate of the distribution of each cell, there must be many more samples than the number of cells used to represent the distribution. For example, suppose a 1D distribution is to be approximated using 50 cells. To get a good distribution over these 50 cells, many more than 50 samples must be used. If fewer than 50 samples are used, many of the cells won't get filled in and the distribution will not be correct, possibly even erratic. Obviously the more samples the better, but how many more? A simple answer would be to select the average number of samples that are desired to fall into each cell and then multiply that number by the number of cells. For the 50 cells example, suppose an average
of 100 samples per cell is desired. That would mean that at least $50 \times 100 = 5000$ samples should be generated. So returning to the 2D GBF, suppose an average of about $m$ Monte Carlo samples are desired per destination cell. The 2D GBF destination grid has $n^2$ cells. So, each originating cell would need to generate at least $n^2 m$ samples. And since there are $n^2$ cells in the originating grid, then $n^2 \times n^2 m = n^4 m$ samples must be generated for each update. As a numerical example, suppose there are 20 cells in each dimension and an average of 100 samples per bin is desired. Thus, $n = 20$ and $m = 100$, so there would need to be at least $20^4 \times 100 = 16$ million Monte Carlo samples per update. While these are huge numbers, this processing is doable with the computers available today.

2.2 The Straightforward 4D GBF

Producing a GBF for a target moving in one dimension is relatively easy to construct. It requires a 2D grid $(x, \dot{x})$. Therefore, a straightforward approach to extend this filter to a target moving in two-dimensions would be to require a four-dimensional (4D) grid $(x, \dot{x}, y, \dot{y})$. Unfortunately, this approach would be problematic since it would result in a sizable grid. In general terms, suppose the grid has $n$ cells in each dimension. Thus each grid would have $n^4$ cells. The distribution of each origination cell must be determined. As discussed previously, to get a good (numerical) estimate of the distribution of each cell, there must be many more samples than the number of cells used to represent the distribution. Again, let $m$ be the average number of Monte Carlo samples desired per cell. The 4D GBF has $n^4$ cells in the four-dimensional destination grid. So each cell in the originating grid would need to generate at least $n^4 m$ samples. And since there are $n^4$ cells in the originating grid, there would need to be at least $n^8 m$ Monte Carlo samples generated per update. Using the same numerical example as before, each grid would have $20^4 = 160,000$ cells of storage. However, the processing requirement would be much worse. Each cell in the originating grid would need to generate 16 million ($20^4 \times 100$) Monte Carlo samples, thereby requiring over 2.5 trillion ($20^8 \times 100$) samples per update. This is beyond
the processing capability of today's computers to do in real-time. This analysis seems to make a
good case for abandoning the grid-based approach. It is likely that this type of analysis is the
reason so little attention is given to grid-based methods. However, the next chapter will show that
a 4D grid can be avoided by recasting the problem using specialized 2D grids.
Chapter 3 - The Constrained GBF

The power of a GBF can be exploited as long as only a two-dimensional grid is required. Therefore, the trick is how to keep the grid two-dimensional for a target moving in 2D space. The solution is to assume the grid only discretizes the $x,y$ locations, and then extend what is stored in each cell. Instead of storing only the probability of the target being at this $x,y$ location, four other values are included as well:

1) the expected speed of the target given it is at this location,

2) the variance of those speeds,

3) the expected course angle of the target given it as this location, and

4) the variance of those course angles.

Thus, each cell location is actually maintaining a 5-tuple so the grid is really an $n \times n \times 5$. The next section will explain how the 5-tuple is computed.

Notice, however, that speed and course are being stored instead of the usual component velocities, $\dot{x}, \dot{y}$. There is a good reason for this. While it is easier to work with $\dot{x}, \dot{y}$, the kinematic constraints on a target are usually specified in terms of speeds and turn rates, not component velocities. By maintaining speed and course instead, those kinematic constraints can be directly exploited. Thus, for the purposes of ensuring all kinematic constraints are enforced, the method must resort to working with speed and course instead of component velocities.

3.1 Computing the 5-Tuple

The previous section revealed that each cell within the specialized 2D grid stores a 5-tuple. This section explains how the 5-tuple is computed. Recall that during each update, there are two grids, an originating grid and a destination grid. These grids can be thought of as the From and To
grids. Let the *From* grid be labeled *F* and the *To* grid be labeled *T*. Then *F*(x,y) is the cell at location *x*,*y* in the *From* grid. 'Dot notation' will be used to select each of the five values within a cell where *F*(x,y).prob, *F*(x,y).spd, *F*(x,y).svar, *F*(x,y).crs, and *F*(x,y).cvar are the probability, mean speed, speed variance, mean course, and course variance, respectively. The algorithm in Fig. 3-1 describes how these values are computed, except for the computation of the mean course and its variance. The mean and variance of the course are computed similarly to the mean and variance of the speed, but require additional work to deal with the circular nature of the course parameter. Appendix A provides the C code implementing the algorithm in Fig. 3-1.

```c
For Each (x_F, y_F) In F
    prb_F = F(x_F, y_F).prob/M;
    spd_F = Normal(F(x_F, y_F).spd, F(x_F, y_F).svar);
    crs_F = Normal(F(x_F, y_F).crs, F(x_F, y_F).cvar);
    \Delta x, \Delta y, spd_T, crs_T = RandomMove(spd_F, crs_F);
    x_TL = x_F - \Delta x/2;
    x_TH = x_F + \Delta x/2;
    y_TL = y_F - \Delta y/2;
    y_TH = y_F + \Delta y/2;
    For Each x_T In x_TL To x_TH
        For Each y_T In y_TL To y_TH
            p = CellOverlapFraction(x_T, y_T, x_TL, y_TL, x_TH, y_TH) * prb_F;
            T(x_T, y_T).prob = T(x_T, y_T).prob + p;
            T(x_T, y_T).spd = T(x_T, y_T).spd + p*spd_F;
            T(x_T, y_T).svar = T(x_T, y_T).svar + p*spd_F^2;
        End ForEach
    End ForEach
End ForEach

psum = 0;
For Each (x_T, y_T) In T
    If T(x_T, y_T).prob > 0
        psum = psum + T(x_T, y_T).prob;
        T(x_T, y_T).spd = T(x_T, y_T).spd/T(x_T, y_T).prob;
        T(x_T, y_T).svar = T(x_T, y_T).svar/T(x_T, y_T).prob - (T(x_T, y_T).spd)^2;
    End If
End ForEach

For Each (x_T, y_T) In T
    T(x_T, y_T).prob = T(x_T, y_T).prob/psum;
End ForEach
```

*Figure 3-1: The algorithm for computing the destination grid for each update.*
Notice there are three separate loops in Fig. 3-1. The first loop forms the (probability) weighted sums, \( \sum p_i, \sum p_i s_i, \) and \( \sum p_i s_i^2 \), based on the Monte Carlo samples. (It is also the loop that determines to which track a new measurement should be assigned in multi-target scenarios. The measurement would be assigned to the grid which yields the highest cumulative probability.)

After the sampling is complete, the second loop converts these weighted sums into mean and variance, using equations (3.1) - (3.3). The final loop re-normalizes the mass in the cells.

\[
P = \sum p_i \tag{3.1}
\]
\[
\bar{s} = \frac{1}{P} \sum p_i s_i \tag{3.2}
\]
\[
\sigma_s^2 = \frac{1}{P} \sum p_i s_i^2 - (\bar{s})^2 \tag{3.3}
\]

The algorithm in Fig 3-1 also references three external functions: Normal, RandomMove, and CellOverlapFraction. The 'Normal' function creates normally distributed random variables. Section 3.3.2 will describe a highly efficient means for generating normal-like variables that are good enough for this algorithm. 'RandomMove' will lead into a much broader discussion throughout all sections under 3.2 but will be mostly described in sections 3.2 and 3.2.3. 'CellOverlapFraction' will be explained in section 3.3. It will be shown how this is a refinement over current approaches for moving mass from one cell to a set of transition cells.

3.2 The 2D Target Motion Model

The algorithm in Fig. 3-1 refers to a function 'RandomMove'. This function creates a random, but kinematically feasible predicted state over the time interval. The function allows any target motion model to be used for a CGBF whereas a Kalman filter is restricted to a linear motion model (due to its Gaussian constraint). The motion model used for this CGBF was developed based on a generalization of the continuous-time nearly constant velocity (NCV) target model [59] [11]. The NCV target model assumes “small” perturbations to the target's velocity on
a recurring basis. The intent is to yield more realistic target motion. The perturbations are considered small if they are much less than the target’s actual velocity. These perturbations to the target’s velocity are really acceleration changes, so it is a small white acceleration model [59]. A (continuous-time) maneuvering target motion model can be derived from this NCV model by allowing any acceleration change that the target can achieve. Thus, the maneuvering target model was developed as follows. Assume the target maneuvers at times $t_k$, $k=1, 2, 3, ...$ These times are independent of the measurement update times. At each time $t_k$ the target undergoes a constant acceleration and/or a constant turn maneuver that persists until the next maneuver time. These accelerations and turn rates can be any that the target can achieve within its kinematic limits, including zero. Assume at time, $t_{k-1}$ the target is at location $(x_{k-1}, y_{k-1})$ with speed and course, $s_{k-1}, \theta_{k-1}$, respectively. Let $\dot{s}$ be a (randomly) selected acceleration (along the direction of travel) and $\dot{\theta}$ be a (randomly) selected turn rate that the target will undergo for the duration of the next maneuver. Let $\tau = t_k - t_{k-1}$ be the duration of this next maneuver. Then, the target state as a function of time over the interval 0 to $\tau$ would be:

\begin{align*}
    s(t) &= s_{k-1} + \dot{s}t \\
    \theta(t) &= \theta_{k-1} + \dot{\theta}t \\
    \dot{x}(t) &= s(t) \sin \theta(t) \\
    \dot{y}(t) &= s(t) \cos \theta(t)
\end{align*}

\begin{align}
    x(t) &= x_{k-1} + \int_0^t \dot{x}(t) dt = x_{k-1} + \frac{1}{\dot{\theta}^2} \left[ \dot{s} \sin \theta(t) - s(t) \dot{\theta} \cos \theta(t) \right]_0^t \\
    &= x_{k-1} + \frac{1}{\dot{\theta}^2} \left[ \dot{s} \sin \theta(t) - s(t) \dot{\theta} \cos \theta(t) - \dot{s} \sin \theta_{k-1} + s_{k-1} \dot{\theta} \cos \theta_{k-1} \right] \\
    y(t) &= y_{k-1} + \int_0^t \dot{y}(t) dt = y_{k-1} + \frac{1}{\dot{\theta}^2} \left[ \dot{s} \cos \theta(t) + s(t) \dot{\theta} \sin \theta(t) \right]_0^t \\
    &= y_{k-1} + \frac{1}{\dot{\theta}^2} \left[ \dot{s} \cos \theta(t) + s(t) \dot{\theta} \sin \theta(t) - \dot{s} \cos \theta_{k-1} + s_{k-1} \dot{\theta} \sin \theta_{k-1} \right]
\end{align}

for $0 \leq t \leq \tau$.  

33
Keep in mind that if small random accelerations and turn rates are selected at each time $t_k$, then this model reduces to the continuous-time NCV model (i.e., the continuous-time white noise acceleration model), except now the $x$ and $y$ components of the velocity are coupled [59].

### 3.2.1 Analyzing the 2D Motion Model

There are two interesting special cases to consider with these equations: 1) when the target cannot accelerate so $\dot{s} = 0$, and 2) when the target cannot turn so $\dot{\theta} = 0$. These cases will be investigated in the next two sections.

#### 3.2.1.1 When the Target Cannot Accelerate

Consider when $\dot{s} = 0$. Then $s(t) = s_{k-1}$. So (3.8) and (3.9) become:

\[
x(t) = x_{k-1} + \frac{1}{\theta^2} \left[ -s_{k-1} \dot{\theta} \cos \theta(t) + s_{k-1} \dot{\theta} \cos \theta_{k-1} \right] \\
= x_{k-1} + \frac{s_{k-1}}{\theta} \left[ -\cos \theta(t) + \cos \theta_{k-1} \right] \\
y(t) = y_{k-1} + \frac{s_{k-1}}{\theta} \left[ \sin \theta(t) - \sin \theta_{k-1} \right]
\]

(3.10) \hspace{1cm} (3.11)

Since there is no acceleration, the target must be simply moving along a constant radius circle. These equations may not seem to indicate this fact. However, consider Fig. 3-2. It shows the scenario where the initial course is zero, i.e., $\theta_{k-1} = 0$ and the target is moving clockwise (i.e., right-hand turn) around a circle of radius, $r$. From the figure, $(x, y)$ can be easily computed as:

\[
x(t) = x_{k-1} + r - r \cos \omega = x_{k-1} + r(1 - \cos \omega) \\
y(t) = y_{k-1} + r \sin \omega
\]

(3.12) \hspace{1cm} (3.13)
Equations (3.12) and (3.13) are in terms of $r$ and $\omega$, both of which can be determined. The length of the arc that the target moves through in $t$ time is $s_{k-1}t$. The angle $\omega$ that sweeps out this arc segment is $\dot{\theta}t$. Thus,

$$s_{k-1}t = r\dot{\theta}t$$  \hspace{1cm} (3.14)

Solving for $r$, yields:

$$r = \frac{s_{k-1}}{\dot{\theta}}$$  \hspace{1cm} (3.15)

Substituting these quantities into (3.12) and (3.13) yields:

$$x(t) = x_{k-1} + \frac{s_{k-1}}{\dot{\theta}}(1 - \cos \dot{\theta}t)$$  \hspace{1cm} (3.16)

$$y(t) = y_{k-1} + \frac{s_{k-1}}{\dot{\theta}} \sin \dot{\theta}t$$  \hspace{1cm} (3.17)

Figure 3-2: A target moving clockwise (right-hand turn) around a constant radius circle.

Equations (3.16) and (3.17) are identical to (3.10) and (3.11) when $\theta_{k-1} = 0$. Therefore, the motion equations do indeed reduce to a target moving around a constant radius circle when the target cannot accelerate.
### 3.2.1.2 When the Target Cannot Turn

For the second case, consider when $\dot{\theta} = 0$. The turn rate is in the denominator in (3.8) and (3.9). This seems to indicate an infinite displacement when the target cannot turn. Of course, if the target cannot turn, the motion should reduce to the case of a target moving along a straight line (constant course). So it is interesting to see what happens to (3.8) and (3.9) for the zero turn rate target. Consider (3.8) first. Taking the limit of $x(t)$ as $\dot{\theta}$ goes to zero yields:

$$
\lim_{\dot{\theta} \to 0} x(t) = \lim_{\dot{\theta} \to 0} \left\{ x_{k-1} + \frac{1}{\dot{\theta}^2} \left[ \dot{s} \sin \theta(t) - s(t)\dot{\theta} \cos \theta(t) - \dot{s} \sin \theta_{k-1} + s_{k-1} \dot{\theta} \cos \theta_{k-1} \right] \right\}
$$

(3.18)

$$
= x_{k-1} + \lim_{\dot{\theta} \to 0} \frac{\dot{\theta} [s_{k-1} \cos \theta_{k-1} - s(t) \cos \theta(t)] + [\dot{s} \sin \theta(t) - \dot{s} \sin \theta_{k-1}]}{\dot{\theta}^2}
$$

Since the limit is an indeterminate, l'Hôpital's rule can be used [83]. Therefore,

$$
\lim_{\dot{\theta} \to 0} x(t) = x_{k-1} + \lim_{\dot{\theta} \to 0} \left\{ \dot{\theta} \frac{[s(t) \sin \theta(t)] + [s_{k-1} \cos \theta_{k-1} - s(t) \cos \theta(t)]}{2\dot{\theta}} \right\}
$$

$$
= x_{k-1} + \lim_{\dot{\theta} \to 0} \left\{ \dot{\theta} \frac{s(t) \sin \theta(t) + s_{k-1} \cos \theta_{k-1} - s(t) \cos \theta(t)}{2} \right\}
$$

(3.19)

$$
= x_{k-1} + \frac{1}{2} \dot{s}(t) \sin \theta_{k-1} + \lim_{\dot{\theta} \to 0} \frac{s_{k-1} \cos \theta_{k-1} - s(t) \cos \theta(t)}{2\dot{\theta}}
$$

Equation (3.19) is still an indeterminate so applying l'Hôpital's rule again:

$$
= x_{k-1} + \frac{1}{2} \dot{s}(t) \sin \theta_{k-1} + \lim_{\dot{\theta} \to 0} \frac{s_{k-1} \cos \theta_{k-1} - s(t) \cos \theta(t)}{2\dot{\theta}}
$$
\[
\lim_{\theta \to 0} x(t) = x_{k-1} + \frac{1}{2} s(t) t \sin \theta_{k-1} + \lim_{\theta \to 0} \frac{s_{k-1}[t \sin \theta(t)]}{2} \\
= x_{k-1} + \frac{1}{2} s(t) t \sin \theta_{k-1} + \frac{1}{2} s_{k-1} t \sin \theta_{k-1} \\
= x_{k-1} + \frac{1}{2} t (s_{k-1} + \dot{s} t + s_{k-1}) \sin \theta_{k-1} \\
= x_{k-1} + t \left( s_{k-1} + \frac{1}{2} \dot{s} t \right) \sin \theta_{k-1} = x_{k-1} + \left( s_{k-1} t + \frac{1}{2} \dot{s} t^2 \right) \sin \theta_{k-1}
\]

(3.20)

So the new \( x \) position is just along the target's initial course, as expected. The same procedure can be done to find the new \( y \) position. Using (3.9),

\[
\lim_{\theta \to 0} y(t) \\
= \lim_{\theta \to 0} \left\{ y_{k-1} + \frac{1}{\theta^2} \left[ \dot{s} \cos \theta(t) + s(t) \dot{\theta} \sin \theta(t) - \dot{s} \cos \theta_{k-1} - s_{k-1} \dot{\theta} \sin \theta_{k-1} \right] \right\} \\
= y_{k-1} + \lim_{\theta \to 0} \frac{\dot{\theta} [s(t) \cos \theta(t) - s_{k-1} \sin \theta_{k-1} + [s(t) \cos \theta(t) - \dot{s} \cos \theta_{k-1}]]}{\theta^2} \\
= y_{k-1} + \lim_{\theta \to 0} \frac{\dot{\theta} [s(t) t \cos \theta(t)] + [s(t) \sin \theta(t) - s_{k-1} \sin \theta_{k-1}]}{2} + \frac{[-\dot{s} \sin \theta(t)]}{2} \\
= y_{k-1} + \lim_{\theta \to 0} \left\{ \frac{s(t) t \cos \theta(t)}{2} + \frac{s(t) \sin \theta(t) - s_{k-1} \sin \theta_{k-1} - \dot{s} t \sin \theta(t)}{2} \right\} \\
= y_{k-1} + \frac{1}{2} s(t) t \cos \theta_{k-1} + \lim_{\theta \to 0} \frac{-s_{k-1} \sin \theta_{k-1} + [s(t) - \dot{s} t] \sin \theta(t)}{2} \\
= y_{k-1} + \frac{1}{2} s(t) t \cos \theta_{k-1} + \lim_{\theta \to 0} \frac{-s_{k-1} \sin \theta_{k-1} + s_{k-1} \sin \theta(t)}{2} \\
= y_{k-1} + \frac{1}{2} s(t) t \cos \theta_{k-1} + \lim_{\theta \to 0} \frac{s_{k-1} \sin \theta(t)}{2} \\
= y_{k-1} + \frac{1}{2} s(t) t \cos \theta_{k-1} + \lim_{\theta \to 0} \frac{s_{k-1} \cos \theta(t)}{2} \\
= y_{k-1} + \frac{1}{2} s(t) t \cos \theta_{k-1} + \lim_{\theta \to 0} \frac{s_{k-1}}{2} \cos \theta_{k-1}
\]

(3.21)
Even though it has been shown that the motion equations are mathematically sound, there is no reason to depend on the computation of these equations to be of infinite precision when implemented. Instead, define $\epsilon_\theta > 0$ to be the smallest turn rate value at which smaller values are equivalent to the target simply moving along a constant straight-line course. Thus, the modified motion equations for position become:

$$x(t) = x_{k-1} + \begin{cases} \frac{1}{\dot{\theta}^2} \left[ \dot{s} \sin \theta(t) - s(t)\dot{\theta} \cos \theta(t) - \dot{s} \sin \theta_{k-1} + s_{k-1}\dot{\theta} \cos \theta_{k-1} \right] & \text{if } |\dot{\theta}| > \epsilon_\theta \\ \left( s_{k-1}t + \frac{1}{2}\dot{s}t^2 \right) \sin(\theta_{k-1} + \dot{\theta} \tau) & \text{otherwise} \end{cases}$$

$$y(t) = y_{k-1} + \begin{cases} \frac{1}{\dot{\theta}^2} \left[ \dot{s} \cos \theta(t) + s(t)\dot{\theta} \sin \theta(t) - \dot{s} \cos \theta_{k-1} - s_{k-1}\dot{\theta} \sin \theta_{k-1} \right] & \text{if } |\dot{\theta}| > \epsilon_\theta \\ \left( s_{k-1}t + \frac{1}{2}\dot{s}t^2 \right) \cos(\theta_{k-1} + \dot{\theta} \tau) & \text{otherwise} \end{cases}$$

for $0 \leq t \leq \tau$.

**3.2.2 Determining the 2D Kinematically-Feasible Transition Region**

Using the described 2D target motion model, it is interesting to see what the resulting prediction region looks like. Recall in the 1D case, the bounds of the containment region was straightforward to calculate and it looked like a surfboard. For the 2D case, the region becomes four-dimensional making it much more difficult to visualize. However, the positional containment region, which is two-dimensional, can be found. Assume the target is initially at the origin moving due north with an initial speed, $s_1$. As before, let $T$ be the duration of the prediction time interval. Let $\tau_1, \tau_2, \tau_3, \ldots, \tau_k$ be the durations of the maneuvers within a prediction time interval and sums to the interval, $T$. Thus, $\tau_1 + \tau_2 + \tau_3 + \cdots + \tau_k = T$. Using the maximum accelerations and turn rates, the coarse-grained bounds of the positional prediction can be found, as shown in Fig. 3-3. This positional region is coarse because it assumes the turns happen instantaneously even though a turn rate is specified. The true positional containment region would be somewhat smaller when the turn rate is confined to the one specified. Using (3.8) and (3.9) the
true positional containment region can be computed, as shown in Fig. 3-4. In either the coarse or true case, the containment region is clearly not elliptical as a Kalman filter would infer by its Gaussian assumption.

Figure 3-3: Coarse-grained bounds of the predicted positional containment region.

\[ u_{2L} = s_i T - \frac{1}{2} AT^2 \]
\[ u_{2H} = s_i T + \frac{1}{2} AT^2 \]

\[ (-u_{2H} \sin(\hat{\theta}T), u_{2H} \cos(\hat{\theta}T)) \quad (u_{2H} \sin(\hat{\theta}T), u_{2H} \cos(\hat{\theta}T)) \]
\[ (-u_{2L} \sin(\hat{\theta}T), u_{2L} \cos(\hat{\theta}T)) \quad (u_{2L} \sin(\hat{\theta}T), u_{2L} \cos(\hat{\theta}T)) \]

Figure 3-4: True predicted positional containment region.

\[ w_L = -\frac{A}{\dot{\theta}^2} \sin(\hat{\theta}T) - \frac{s_i - AT}{\dot{\theta}} \cos(\hat{\theta}T) + \frac{s_i}{\dot{\theta}} \]
\[ w_H = \frac{A}{\dot{\theta}^2} \sin(\hat{\theta}T) - \frac{s_i + AT}{\dot{\theta}} \cos(\hat{\theta}T) + \frac{s_i}{\dot{\theta}} \]
\[ h_L = -\frac{A}{\dot{\theta}^2} \cos(\hat{\theta}T) + \frac{s_i - AT}{\dot{\theta}} \sin(\hat{\theta}T) - \frac{s_i}{\dot{\theta}} \]
\[ h_H = \frac{A}{\dot{\theta}^2} \cos(\hat{\theta}T) + \frac{s_i + AT}{\dot{\theta}} \sin(\hat{\theta}T) - \frac{s_i}{\dot{\theta}} \]
3.2.3 Computing the Distribution of the Feasible Transition Region

As in the 1D case, the probability mass distribution within the kinematically-feasible region is required. To estimate this probability density, the same Monte Carlo technique was used as for the 1D case. However, in addition to varying the target's acceleration, the target's turn rate must also be varied. So now a maneuver is a newly executed acceleration and/or turn rate. In a given unit of time, a target can execute any number of maneuvers and have each maneuver persist for any duration (provided, of course, that the sum of the durations does not exceed the unit of time). Thus, as before in the 1D case, the number of maneuvers and the duration of those maneuvers must also be varied. For the 1D case, it was necessary to have at least two maneuvers to fill in the entire feasible region. How many maneuvers are required for the 2D case? Since the turn rate is a non-linear quantity, it seems infinitely many maneuvers are required. But how does the number of maneuvers affect the probability distribution? To address these questions, the Monte Carlo analysis varied the number of maneuvers over the time interval. Each Monte Carlo run executes a (different) random movement of the target over a specified duration. This random movement is what the function 'RandomMove' performs in the algorithm in Fig. 3-1. Consider Figs. 3-5 - 3-7. Fig. 3-5 allowed up to two maneuvers, Fig. 3-6 allowed up to four maneuvers, and Fig. 3-7 allowed up to eight maneuvers. The colors denote probability from red (most probable) to blue (least probable). In all cases, the target started at the origin with an initial course of zero degrees (measured clockwise from North). The figures show that as the maximum number of maneuvers increases, the prediction distribution becomes tighter, i.e., there is less uncertainty of where the target could have transitioned to over the time interval. In other words, the more often the target maneuvers, the less uncertainty in its final position. At first this may seem counter-intuitive but with some thought, it does make sense. As the number of maneuvers increases, there is more chance for the velocity changes to cancel each other out. Thus, the final position of the target tends towards the location it would have if it had simply moved with constant velocity. This is not a new idea. Li and Jilkov documented this property in [59]. As they explained, as the number of
maneuvers decreases to one, the motion can be represented by a constant acceleration model. As the number of maneuvers increases towards infinity, the motion tends toward a constant velocity model. Notice, however, as the number of maneuvers increases, the uncertainty region becomes Gaussian. This should be no surprise either. The final position is a sum of these random maneuvers. Each random maneuver is based on a set of random variables that are independent and selected from identical distributions. Thus, the final position is based on the sum of random variables that are independent and identically distributed. By virtue of the Central Limit Theorem, the resulting distribution must become Gaussian [12]. Therefore, the Kalman filter effectively treats the target motion as a large (i.e., infinite) sum of maneuvers over the time interval.

As a result of this analysis, it seems best to keep the number of maneuvers to at most two to avoid overly optimistic estimates about the position uncertainty from the uncertainty in the target motion. So in answer to the question of how many maneuvers are needed, two are sufficient; one at the beginning of the time and a second somewhere within the time interval. This is identical to the 1D case considered earlier. The function 'RandomMove' in Fig 3-1 uses this method for computing a random trajectory.

Regardless of the number of maneuvers, the predicted position distribution is well bounded, unlike the infinite extent assumption from a normal distribution. In addition, when the number of maneuvers is kept small, the position distribution is clearly not elliptical and thus again, cannot be normally distributed. Furthermore, it is more realistic to assume the target will make only a few maneuvers over the interval rather than many. Thus, between the finite extent of the distribution and the more tailored computed distribution, it is natural to expect the grid-based filter to outperform the Kalman filter since the grid-based filter will exploit this information.
Figure 3-5: Position uncertainty distribution; up to 2 maneuvers within time interval.

Figure 3-6: Position uncertainty distribution; up to 4 maneuvers within time interval.
3.2.4 The Mean and Variance of the Transition Density (Single Maneuver)

Although formulating the distribution for the 2D transition uncertainty density analytically is unknown, the mean and variance of this distribution can be determined. As shown in the previous section, the spread of the distribution depends on the number of maneuvers over the update time interval. To help simplify the discussion, it will be initially assumed that there is only one maneuver over the update time interval, i.e., \( T = \tau \). The case where up to two maneuvers are permitted is considered in section 3.2.5. Since it is assumed for now that there is only one maneuver, this is the equivalent of determining the mean and variance for a single maneuver. Thus, the maneuver duration, \( \tau \), will be used to emphasize this fact. Refer to (3.4) - (3.9). The state at the end of the maneuver is found using \( t = \tau \). Recall that the subscript \( k \) refers to the (true) state of the target after the \( k^{th} \) maneuver within an update time interval. A subscript of zero is the true state of the target at the beginning of the time interval. So:
\[ s_1 = s(t) = s_0 + \dot{s} \tau \] (3.24)
\[ \theta_1 = \theta(t) = \theta_0 + \dot{\theta} \tau \] (3.25)
\[ \dot{x}_1 = \dot{x}(t) = s(t) \sin \theta(t) = s_1 \sin \theta_1 \] (3.26)
\[ \dot{y}_1 = \dot{y}(t) = s(t) \cos \theta(t) = s_1 \cos \theta_1 \] (3.27)
\[ x_1 = x(t) = x_0 + \frac{1}{\dot{\theta}^2} \left[ \dot{s} \sin \theta_1 - s_1 \dot{\theta} \cos \theta_1 - \dot{s} \sin \theta_0 + s_0 \dot{\theta} \cos \theta_0 \right] \] (3.28)
\[ y_1 = y(t) = y_0 + \frac{1}{\dot{\theta}^2} \left[ \dot{s} \cos \theta_1 + s_1 \dot{\theta} \sin \theta_1 - \dot{s} \cos \theta_0 - s_0 \dot{\theta} \sin \theta_0 \right] \] (3.29)

### 3.2.4.1 The Mean and Variance of the Speed and Course

Determining the mean and variance of the speed and course is fairly straightforward. Using (3.24),

\[ E[s_1] = E[s_0 + \dot{s} \tau] = E[s_0] + E[\dot{s} \tau] = E[s_0] + \tau E[\dot{s}] = s_0 \] (3.30)

And similarly from (3.25),

\[ E[\theta_1] = E[\theta_0 + \dot{\theta} \tau] = E[\theta_0] + \tau E[\dot{\theta}] = \theta_0 \] (3.31)

Therefore the expected speed and course after the target maneuvers is identical to what they were prior to the maneuver. Calculation of the variances for the speed and course are also straightforward.

\[ \text{Var}[s_1] = \text{Var}[s_0 + \dot{s} \tau] = \text{Var}[s_0] + \text{Var}[\dot{s} \tau] = \text{Var}[s_0] + \tau^2 \text{Var}[\dot{s}] = \tau^2 \sigma_s^2 \] (3.32)
\[ \text{Var}[\theta_1] = \text{Var}[\theta_0 + \dot{\theta} \tau] = \text{Var}[\theta_0] + \tau^2 \text{Var}[\dot{\theta}] = \tau^2 \sigma_\theta^2 \] (3.33)

The selection of accelerations and turn rates were assumed to be uniformly distributed over the range of possible rates. As a result, \( \sigma_s^2 = \frac{(2A)^2}{12} = \frac{A^2}{3} \) and \( \sigma_\theta^2 = \frac{(2\dot{\theta})^2}{12} = \frac{\dot{\theta}^2}{3} \), where \( A \) is the maximum acceleration and \( \dot{\theta} \) is the maximum turn rate. The variance of the velocity is quadratic with respect to time, or equivalently, the standard deviation of the velocity is linear in time.
3.2.4.2. Mean and Variance of the Component Velocities

With a little more work, the mean and variance of the component velocities can also be determined. It should be pointed out that the means are not found by just plugging in the expected values into (3.26) and (3.27) to get $E[\dot{x}_1] = s_0 \sin \theta_0$ and $E[\dot{y}_1] = s_0 \cos \theta_0$. This result is what the Kalman filter uses because it assumes the distributions are normal in rectilinear space, not in polar space, which is what is assumed for the CGBF. Using (3.26) and (3.27),

$$E[\dot{x}_1] = E[s_1 \sin \theta_1]$$
$$E[\dot{y}_1] = E[s_1 \cos \theta_1]$$

Since it is assumed that the acceleration change is independent of the course change,

$$E[\dot{x}_1] = E[s_1 \sin \theta_1] = E[s_1]E[\sin \theta_1]$$
$$E[\dot{y}_1] = E[s_1 \cos \theta_1] = E[s_1]E[\cos \theta_1]$$

The $E[s_1]$ was already found in (3.30). Using the properties of sine and cosine,

$$E[\sin \theta_1] = E[\sin(\theta_0 + \dot{\theta}_\tau)] = E[\sin \theta_0 \cos \dot{\theta}_\tau + \sin \dot{\theta}_\tau \cos \theta_0]$$
$$= \sin \theta_0 E[\cos \dot{\theta}_\tau] + \cos \theta_0 E[\sin \dot{\theta}_\tau]$$

$$E[\cos \theta_1] = E[\cos(\theta_0 + \dot{\theta}_\tau)] = E[\cos \theta_0 \cos \dot{\theta}_\tau - \sin \theta_0 \sin \dot{\theta}_\tau]$$
$$= \cos \theta_0 E[\cos \dot{\theta}_\tau] - \sin \theta_0 E[\sin \dot{\theta}_\tau]$$

The remaining expected values can be found using the definition of expected value. Assume, as before, that $\dot{\theta}$ is a uniform random variable on the interval $[-\Theta, +\Theta]$. Therefore,

$$E[\sin \dot{\theta}_\tau] = \int_{-\Theta}^{+\Theta} \left(\frac{1}{2\dot{\Theta}}\right) \sin \dot{\theta}_\tau d\dot{\theta} = -\left(\frac{1}{2\dot{\Theta}}\right) \cos \dot{\theta}_\tau \bigg|_{-\Theta}^{+\Theta} = -\left(\frac{1}{2\dot{\Theta}}\right) (\cos \dot{\theta}_\tau - \cos \dot{\theta}_0)$$
$$= 0$$

$$E[\cos \dot{\theta}_\tau] = \int_{-\Theta}^{+\Theta} \left(\frac{1}{2\dot{\Theta}}\right) \cos \dot{\theta}_\tau d\dot{\theta} = \left(\frac{1}{2\dot{\Theta}}\right) \sin \dot{\theta}_\tau \bigg|_{-\Theta}^{+\Theta} = \left(\frac{1}{2\dot{\Theta}}\right) (\sin \dot{\theta}_\tau + \sin \dot{\theta}_0)$$
$$= \frac{\sin \dot{\theta}_\tau}{\dot{\Theta}}$$
Thus,

\[ E[\sin \theta_1] = \sin \theta_0 \frac{\sin \dot{\theta}}{\dot{\theta}} \]  
(3.42)

\[ E[\cos \theta_1] = \cos \theta_0 \frac{\sin \dot{\theta}}{\dot{\theta}} \]  
(3.43)

From (3.36), (3.37), (3.42), and (3.43),

\[ E[\dot{x}_1] = s_0 \sin \theta_0 \frac{\sin \dot{\theta}}{\dot{\theta}} \]  
(3.44)

\[ E[\dot{y}_1] = s_0 \cos \theta_0 \frac{\sin \dot{\theta}}{\dot{\theta}} \]  
(3.45)

When \( \dot{\theta} \) is small, \( \frac{\sin \dot{\theta}}{\dot{\theta}} \approx 1 \), which yields the same expected velocity components as from a Kalman filter. The variances of the component velocities can now be found. Recall that, in general,

\[ \text{Var}[X] = E[X^2] - (E[X])^2 \]  
(3.46)

Thus, from (3.36), (3.37), (3.44), and (3.45),

\[ \text{Var}[\dot{x}_1] = E[\dot{x}_1^2] - (E[\dot{x}_1])^2 = E[s_1^2]E[\sin^2 \theta_1] - \left( s_0 \sin \theta_0 \frac{\sin \dot{\theta}}{\dot{\theta}} \right)^2 \]  
(3.47)

\[ \text{Var}[\dot{y}_1] = E[\dot{y}_1^2] - (E[\dot{y}_1])^2 = E[s_1^2]E[\cos^2 \theta_1] - \left( s_0 \cos \theta_0 \frac{\sin \dot{\theta}}{\dot{\theta}} \right)^2 \]  
(3.48)

Using (3.46), \( E[s_1^2] = \text{Var}[s_1] + (E[s_1])^2 \), so from (3.30) and (3.32),

\[ E[s_1^2] = \tau^2 \sigma_s^2 + s_0^2 \]  
(3.49)

\( E[\sin^2 \theta_1] \), and \( E[\cos^2 \theta_1] \) still need to be determined. These expected values can be found using the definition of expected value and Wolfram Mathematica Online Integrator [89].
\[ E[\sin^2 \theta_1] = \int_{-\theta}^{\theta} \left( \frac{1}{2\theta} \right) \sin^2(\theta_0 + \dot{\theta} t) \, d\theta = \frac{2(\theta_0 + \dot{\theta} t) - \sin 2(\theta_0 + \dot{\theta} t)}{8\theta t} \]

\[ = \frac{2(\theta_0 + \dot{\theta} t) - \sin 2(\theta_0 + \dot{\theta} t) - 2(\theta_0 - \dot{\theta} t) + \sin 2(\theta_0 - \dot{\theta} t)}{8\theta t} \]

\[ = \frac{1}{8\theta t} \left[ 4\dot{\theta} t - \sin 2(\theta_0 + \dot{\theta} t) + \sin 2(\theta_0 - \dot{\theta} t) \right] \]

\[ = \frac{1}{2} - \frac{\sin 2\theta_0 \cos 2\dot{\theta} t}{4\dot{\theta} t} \]

\[ = \frac{1}{2} \left( 1 - \frac{\sin 2\theta_0}{2\dot{\theta} t} \cos 2\theta_0 \right) \quad (3.50) \]

\[ E[\cos^2 \theta_1] = \int_{-\theta}^{\theta} \left( \frac{1}{2\theta} \right) \cos^2(\theta_0 + \dot{\theta} t) \, d\theta = \frac{2(\theta_0 + \dot{\theta} t) + \sin 2(\theta_0 + \dot{\theta} t)}{8\theta t} \]

\[ = \frac{[2(\theta_0 + \dot{\theta} t) + \sin 2(\theta_0 + \dot{\theta} t) - 2(\theta_0 - \dot{\theta} t) - \sin 2(\theta_0 - \dot{\theta} t)]}{8\theta t} \]

\[ = \frac{1}{8\theta t} \left[ 4\dot{\theta} t + \sin 2(\theta_0 + \dot{\theta} t) - \sin 2(\theta_0 - \dot{\theta} t) \right] \]

\[ = \frac{1}{2} + \frac{\sin 2\theta_0 \cos 2\dot{\theta} t}{4\dot{\theta} t} \]

\[ = \frac{1}{2} \left( 1 + \frac{\sin 2\theta_0}{2\dot{\theta} t} \cos 2\theta_0 \right) \quad (3.51) \]

Therefore, using (3.47) - (3.51),

\[ \text{Var}[\dot{x}_1] = (\tau^2 \sigma_x^2 + s_0^2) \left( \frac{1}{2} - \frac{\cos 2\theta_0 \sin 2\dot{\theta} t}{4\dot{\theta} t} \right) - \left( s_0 \sin \theta_0 \frac{\sin \dot{\theta} t}{\dot{\theta} t} \right)^2 \quad (3.52) \]

\[ \text{Var}[y_1] = (\tau^2 \sigma_y^2 + s_0^2) \left( \frac{1}{2} + \frac{\cos 2\theta_0 \sin 2\dot{\theta} t}{4\dot{\theta} t} \right) - \left( s_0 \cos \theta_0 \frac{\sin \dot{\theta} t}{\dot{\theta} t} \right)^2 \quad (3.53) \]
Not very insightful equations but they emphasize how different the CGBF prediction model is from the one in the Kalman filter. To gain some appreciation for these equations, consider the case when \( \theta_0 = 0 \), i.e., the target's course is due north. Then, (3.52) and (3.53) become:

\[
\text{Var}[\dot{x}_1] = (\tau^2 \sigma_s^2 + s_0^2) \left( \frac{1}{2} \frac{\sin 2\Theta \tau}{4\Theta \tau} \right)
\]

\[
\text{Var}[\dot{y}_1] = (\tau^2 \sigma_s^2 + s_0^2) \left( \frac{1}{2} + \frac{\sin \Theta \tau}{4\Theta \tau} \right) - \left( s_0 \frac{\sin \Theta \tau}{\Theta \tau} \right)^2
\]

Since \( \Theta \tau \) is small, as it goes to zero, i.e., \( \Theta \tau \to 0 \), the ratio \( \sin \Theta \tau / \Theta \tau \to 1 \), so:

\[
\text{Var}[\dot{x}_1] \to (\tau^2 \sigma_s^2 + s_0^2) \left( \frac{1}{2} - \frac{1}{2} \right) = 0
\]

\[
\text{Var}[\dot{y}_1] \to (\tau^2 \sigma_s^2 + s_0^2) \left( \frac{1}{2} + \frac{1}{2} \right) - (s_0)^2 = \tau^2 \sigma_s^2
\]

The variance in the \( x \) direction should go to zero because when \( \Theta \tau = 0 \), the target is (only) moving exactly along its initial course, so there is no uncertainty in course. The resulting variance in the \( y \) direction (i.e., the variance in position uncertainty along the target's initial course) agrees with the process model for the Kalman filter. Thus, the CGBF uses a directional process model.

**3.2.4.3 The Mean and Variance of the Position**

Determining the mean and variance of the position is a little more complicated. The mean of the position is considered first. Without any loss of generality, assume the target is initially moving due north. This assumption does not limit the generality because the distribution is being computed along the initial course of the target. Thus, any other initial course is just a rotation of the mean position to align with the target's initial course. With this assumption, \( \theta_0 = 0 \).

The distribution must be symmetric about the initial course since the target can turn left just as easily as turning right. Since the target is assumed to have an initial course of zero, i.e., due
north, the distribution must be symmetric about the $y$-axis. Therefore, $E[x_1] = x_0$. However, this result will be proven. The $E[x_1]$ is computed from (3.28).

$$E[x_1] = E\left[x_0 + \frac{1}{\dot{\theta}^2} \left[ \dot{s} \sin \theta_1 - s_1 \dot{\theta} \cos \theta_1 - \dot{s} \sin \theta_0 + s_0 \dot{\theta} \cos \theta_0 \right] \right]$$  \hspace{1cm} (3.58)

Looking at the equation should raise a concern. The random variable $\dot{\theta}$ occurs in the denominator. Since this variable could be zero, the expected value may not be defined. Recall this issue was addressed by defining the modified motion equations (3.22) and (3.23). But let's continue on and see what happens. Since $\theta_0 = 0$, (3.58) can be simplified to

$$E[x_1] = E[x_0] + E\left[\dot{s} \sin \theta_1 + s_1 \dot{\theta} \cos \theta_1 + s_0 \dot{\theta} \right]$$  \hspace{1cm} (3.59)

And since $\dot{s}$ and $\dot{\theta}$ are assumed to be independent random variables,

$$E[x_1] = E[x_0] + E[\dot{s}]E\left[\frac{\sin \theta_1}{\dot{\theta}^2} \right] - E[s_1 \dot{\theta}]E\left[\frac{\cos \theta_1}{\dot{\theta}} \right] + E[s_0]E\left[\frac{1}{\dot{\theta}} \right]$$  \hspace{1cm} (3.60)

But $E[\dot{s}] = E[\dot{\theta}] = 0$, so the $E\left[\frac{\sin \theta_1}{\dot{\theta}^2} \right]$ does not need to be evaluated. Whatever value it returns will be multiplied by zero, so it will be ignored. After simplifying:

$$E[x_1] = x_0 - s_0 E\left[\frac{\cos \theta_1}{\dot{\theta}} \right] + s_0 E\left[\frac{1}{\dot{\theta}} \right]$$

$$= x_0 + s_0 E\left[\frac{1 - \cos \theta_1}{\dot{\theta}} \right]$$  \hspace{1cm} (3.61)

The remaining expected value is somewhat troublesome. There still is a random variable in the denominator. The expected value depends on the range of $\dot{\theta}$ and the value of $\tau$. However, if
the target cannot turn (i.e., $\dot{\Theta} = 0$) then the expected value should simply be $x_0$. The expected value for this case can be solved using l'Hôpital's rule:

$$\lim_{\dot{\theta} \to 0} E \left[ \frac{1 - \cos \dot{\theta} \tau}{\dot{\theta}} \right] = \lim_{\dot{\theta} \to 0} E \left[ \frac{\tau \sin \dot{\theta} \tau}{1} \right] = 0 \quad (3.62)$$

So for the case when the target cannot turn, $E[x_1] = x_0$, as expected. When the target can turn (i.e., $\dot{\Theta} > 0$) the integral must be solved. Using WolframAlpha[88], the solution can be obtained.

$$E \left[ \frac{1 - \cos \dot{\theta} \tau}{\dot{\theta}} \right] = \int_{-\Theta}^{\Theta} \left( \frac{1 - \cos \dot{\theta} \tau}{\dot{\theta}} \right) \left( \frac{1}{2\dot{\theta}} \right) d\dot{\theta} = \frac{\log(\frac{1}{2} \dot{\theta} \tau) - Ci(\dot{\theta} \tau)}{2\dot{\theta}} \bigg|_{-\Theta}^{\Theta} = 0 \quad (3.63)$$

where $Ci(x)$ is the cosine integral function. To better visualize that this integral is zero, consider the integrand function $f(x) = \left( 1 - \frac{\cos x \tau}{x} \right)$. Since $\cos(-x) = \cos x$, the integrand is an odd function, (i.e., $f(-x) = -f(x)$). An example plot of $f(x)$ is shown in Fig. 3-8. Integrating area that is symmetric to the $y$-axis (i.e., from $-\Theta$ to $\Theta$) will always yield zero.

![Figure 3-8: Example plot of $f(x)$](from wolfram.com)

The $E[y_1]$ is computed from (3.29).

$$E[y_1] = E \left[ y_0 + \frac{1}{\dot{\theta}^2} [\dot{s} \cos \theta_1 + s_1 \dot{\theta} \sin \theta_1 - \dot{s} \cos \theta_0 - s_0 \dot{\theta} \sin \theta_0] \right] \quad (3.64)$$
Again, the random variable is in the denominator. This will be ignored for now to see if it drops out like it did for the previous case. Since \( \theta_0 = 0 \), (3.64) can be simplified to:

\[
E[y_1] = E\left[y_0 + \frac{1}{\theta^2} \left( \hat{s} \cos \hat{\theta} \tau + s_1 \hat{\theta} \sin \hat{\theta} \tau - \hat{s} \right) \right]
= E[y_0] + E\left[\frac{\hat{s} \cos \hat{\theta} \tau}{\theta^2} \right] + E\left[\frac{s_1 \hat{\theta} \sin \hat{\theta} \tau}{\theta^2} \right] - E\left[\frac{\hat{s}}{\theta^2} \right] \tag{3.65}
\]

Since \( \hat{s} \) and \( \hat{\theta} \) are assumed to be independent random variables,

\[
E[y_1] = E[y_0] + E[\hat{s}]E\left[\frac{\cos \hat{\theta} \tau}{\theta^2} \right] + E[s_1]E\left[\frac{\sin \hat{\theta} \tau}{\theta} \right] - E[\hat{s}]E\left[\frac{1}{\theta^2} \right] \tag{3.66}
\]

But \( E[\hat{s}] = E[\hat{\theta}] = 0 \), so fortunately \( E\left[\frac{\cos \hat{\theta} \tau}{\theta^2} \right] \) and \( E\left[\frac{1}{\theta^2} \right] \) can be ignored. Therefore:

\[
E[y_1] = y_0 + s_0 \theta \left[\frac{\sin \hat{\theta} \tau}{\theta} \right] \tag{3.67}
\]

The remaining expected value appears troublesome. The expected value depends on the range of \( \hat{\theta} \) and the value of \( \tau \). However, if the target cannot turn (i.e., \( \hat{\theta} = 0 \)) then the expected value should simply be \( y_0 + s_0 \tau \). The expected for this case can be obtained using l'Hôpital's rule.

\[
\lim_{\theta \to 0} E\left[\frac{\sin \hat{\theta} \tau}{\theta} \right] = \lim_{\theta \to 0} E\left[\frac{\tau \cos \hat{\theta} \tau}{1} \right] = \tau \tag{3.68}
\]

So for the case when the target cannot turn, \( E[y_1] = y_0 + s_0 \tau \), as expected. When the target can turn (i.e., \( \hat{\theta} > 0 \)) the integral must be solved. Using WolframAlpha, the solution can be found.

\[
E\left[\frac{\sin \hat{\theta} \tau}{\theta} \right] = \int_{-\hat{\theta}}^{\hat{\theta}} \left( \frac{\sin \hat{\theta} \tau}{\hat{\theta}} \right) \left( \frac{1}{2\hat{\theta}} \right) d\theta = \frac{\text{Si}(\hat{\theta} \tau)}{2\hat{\theta}} \bigg|_{-\hat{\theta}}^{\hat{\theta}} = \frac{1}{\hat{\theta}} \text{Si}(\hat{\theta} \tau) \tag{3.69}
\]
Si(x) is the sine integral function. The resulting expected value is bounded by τ, i.e., \(0 < \frac{1}{\Theta} Si(\Theta \tau) < \tau\). To visualize the result, an example plot is shown in Fig. 3-9. A key point of the result is that the faster the target can turn, the less it is expected to travel along its current heading. This is in contrast to a Kalman filter which always assumes simple linear motion.

![Example plot of \(f(x) = \frac{1}{x} Si(tx)\) for \(\tau=2\). (from wolfram.com)](image)

**Figure 3-9: Example plot of \(f(x) = \frac{1}{x} Si(tx)\) for \(\tau=2\). (from wolfram.com)**

Summarizing, the expected values for the position of the target with an initial zero course, moving over duration \(\tau\) is:

\[
E[x_1] = x_0 \\
E[y_1] = y_0 + s_0 \frac{1}{\Theta} Si(\Theta \tau) 
\]

(3.70)  
(3.71)

Although the random variable occurred in the denominator, the expected values for the position exist and have been found. As a further check, the expected values will be found for the modified position equations (3.22) and (3.23).

\[
E[x_1] = E \left[ x_0 + \left( \frac{1}{\Theta^2} \left[ \dot{s} \sin \theta_1 - s_1 \dot{\theta} \cos \theta_1 - \dot{s} \sin \theta_0 + s_0 \dot{\theta} \cos \theta_0 \right] \right) \right] \quad \text{if } |\dot{\theta}| > \epsilon_\theta \\
\left( s_0 \tau + \frac{1}{2} \dot{s}^2 \right) \sin \left( \theta_0 + \dot{\theta} \tau \right) \quad \text{otherwise} 
\]

(3.72)
\[
E[y_1] = E \left[ y_0 + \left( \frac{1}{\hat{\theta}^2} \left[ \dot{s} \cos \theta_1 + s_1 \dot{\theta} \sin \theta_1 - \dot{s} \cos \theta_0 - s_0 \dot{\theta} \sin \theta_0 \right] \right) \right] \quad \text{if } |\hat{\theta}| > \epsilon_{\hat{\theta}}
\]

\[
\left( s_0 \tau + \frac{1}{2} \dot{s} \tau^2 \right) \cos(\theta_0 + \hat{\theta} \tau) \quad \text{otherwise}
\]

(3.73)

As before, assume the target is initially moving due north, so \( \theta_0 = 0 \). With this simplification:

\[
E[x_1] = E \left[ x_0 + \left( \frac{1}{\hat{\theta}^2} \left[ \dot{s} \sin \tau - s_1 \dot{\theta} \cos \tau + s_0 \right] \right) \right] \quad \text{if } |\hat{\theta}| > \epsilon_{\hat{\theta}}
\]

\[
\left( s_0 \tau + \frac{1}{2} \dot{s} \tau^2 \right) \sin \tau \quad \text{otherwise}
\]

(3.74)

\[
E[y_1] = E \left[ y_0 + \left( \frac{1}{\hat{\theta}^2} \left[ \dot{s} \cos \tau + s_1 \dot{\theta} \sin \tau - \dot{s} \right] \right) \right] \quad \text{if } |\hat{\theta}| > \epsilon_{\hat{\theta}}
\]

\[
\left( s_0 \tau + \frac{1}{2} \dot{s} \tau^2 \right) \cos \tau \quad \text{otherwise}
\]

(3.75)

And again assuming \( \dot{s} \) and \( \dot{\theta} \) are independent random variables,

\[
E[x_1] = x_0 + \left\{ \begin{array}{ll}
E[\dot{s}]E \left[ \frac{\sin \tau}{\hat{\theta}} \right] - E[s_1]E \left[ \frac{\cos \tau}{\hat{\theta}} \right] + s_0 E \left[ \frac{1}{\hat{\theta}} \right] & \text{if } |\hat{\theta}| > \epsilon_{\hat{\theta}} \\
E \left[ s_0 \tau + \frac{1}{2} \dot{s} \tau^2 \right] E[\sin \tau] & \text{otherwise}
\end{array} \right.
\]

(3.76)

\[
E[y_1] = y_0 + \left\{ \begin{array}{ll}
E[\dot{s}]E \left[ \frac{\cos \tau}{\hat{\theta}} \right] + E[s_1]E \left[ \frac{\sin \tau}{\hat{\theta}} \right] - E[\dot{s}]E \left[ \frac{1}{\hat{\theta}} \right] & \text{if } |\hat{\theta}| > \epsilon_{\hat{\theta}} \\
E \left[ s_0 \tau + \frac{1}{2} \dot{s} \tau^2 \right] E[\cos \tau] & \text{otherwise}
\end{array} \right.
\]

(3.77)

Since \( E[\dot{s}] = 0 \),

\[
E[x_1] = x_0 + \left\{ \begin{array}{ll}
s_0 E \left[ \frac{\cos \tau}{\hat{\theta}} \right] + s_0 E \left[ \frac{1}{\hat{\theta}} \right] & \text{if } |\hat{\theta}| > \epsilon_{\hat{\theta}} \\
s_0 \tau E[\sin \tau] & \text{otherwise}
\end{array} \right.
\]

(3.78)

\[
= x_0 + \left\{ \begin{array}{ll}
s_0 E \left[ \frac{1 - \cos \tau}{\hat{\theta}} \right] & \text{if } |\hat{\theta}| > \epsilon_{\hat{\theta}} \\
s_0 \tau E[\sin \tau] & \text{otherwise}
\end{array} \right.
\]

(3.79)

\[
E[y_1] = y_0 + \left\{ \begin{array}{ll}
s_0 E \left[ \frac{\sin \tau}{\hat{\theta}} \right] & \text{if } |\hat{\theta}| > \epsilon_{\hat{\theta}} \\
s_0 \tau E[\cos \tau] & \text{otherwise}
\end{array} \right.
\]

The expected values in (3.78) and (3.79) can be written as:
\[ E[x_1] = x_0 + s_0 \left[ \int_{-\theta}^{-\varepsilon_\theta} \left( \frac{1}{2\theta} \right) \left( \frac{1 - \cos \varepsilon_\theta \tau}{\theta} \right) d\theta + \tau \int_{-\varepsilon_\theta}^{+\varepsilon_\theta} \left( \frac{1}{2\theta} \right) \sin \varepsilon_\theta \tau d\theta \right] + \int_{+\varepsilon_\theta}^{+\theta} \left( \frac{1}{2\theta} \right) \left( 1 - \cos \varepsilon_\theta \tau \right) d\theta \] 

\[ E[y_1] = y_0 + s_0 \left[ \int_{-\theta}^{-\varepsilon_\theta} \left( \frac{\sin \varepsilon_\theta \tau}{\theta} \right) \left( \frac{1}{2\theta} \right) d\theta + \tau \int_{-\varepsilon_\theta}^{+\varepsilon_\theta} \left( \frac{1}{2\theta} \right) \cos \varepsilon_\theta \tau d\theta \right] + \int_{+\varepsilon_\theta}^{+\theta} \left( \frac{\sin \varepsilon_\theta \tau}{\theta} \right) \left( \frac{1}{2\theta} \right) d\theta \] 

These integrals are identical to the expected values for \( E[\sin \varepsilon_\theta \tau], E[\cos \varepsilon_\theta \tau], E\left[ \frac{\sin \varepsilon_\theta \tau}{\theta} \right] \), and \( E\left[ \frac{1 - \cos \varepsilon_\theta \tau}{\theta} \right] \) that were found earlier, except now with different limits of integration. Within the limits now defined, each function is well-behaved with no infinities in the range. First consider \( E[x_1] \). All three integrals in (3.80) do not need to be evaluated. As discussed earlier, since the integrands are odd functions and symmetrical area is being integrated, the area must sum to zero. Therefore, all the integrals for \( E[x_1] \) are zero. Now consider \( E[y_1] \). The second integral in (3.81) is identical to \( E[\cos \varepsilon_\theta \tau] \) that was already evaluated in (3.41) except with different limits:

\[ \int_{-\varepsilon_\theta}^{+\varepsilon_\theta} \left( \frac{1}{2\theta} \right) \cos \varepsilon_\theta \tau d\theta = \left( \frac{1}{2\theta} \right) \sin \varepsilon_\theta \tau \bigg|_{-\varepsilon_\theta}^{+\varepsilon_\theta} = \left( \frac{1}{2\theta} \right) \left( \sin \varepsilon_\theta \tau + \sin \varepsilon_\theta \tau \right) = \frac{\sin \varepsilon_\theta \tau}{\theta \tau} \] 

The first and last integrals in (3.81) are similar to evaluating \( E\left[ \frac{\sin \varepsilon_\theta \tau}{\theta} \right] \) in (3.69).
\[
\int_{-\theta}^{\epsilon_\theta} \left( \frac{\sin \theta \tau}{\theta} \right) \left( \frac{1}{2\theta} \right) d\theta + \int_{+\epsilon_\theta}^{+\epsilon_\theta} \left( \frac{\sin \theta \tau}{\theta} \right) \left( \frac{1}{2\theta} \right) d\theta = \frac{\sin(\theta \tau)}{2\theta} \left[ \frac{\epsilon_\theta^2}{\theta} \right] + \frac{\sin(\theta \tau)}{2\theta} \left[ +\epsilon_\theta \right]
\]

\[
= \frac{1}{2\theta} \left[ \sin(-\epsilon_\theta \tau) - \sin(-\epsilon_\theta \tau) + \sin(\epsilon_\theta \tau) - \sin(\epsilon_\theta \tau) \right] = \frac{1}{2\theta} \left[ -\sin(\epsilon_\theta \tau) + \sin(\epsilon_\theta \tau) + \sin(\epsilon_\theta \tau) - \sin(\epsilon_\theta \tau) \right] = \frac{1}{\theta} \left[ \sin(\epsilon_\theta \tau) - \sin(\epsilon_\theta \tau) \right] = \frac{1}{\theta} \sin(\epsilon_\theta \tau)
\]

(3.83)

Using these results in (3.80) and (3.81), the expected position using the modified motion equations is:

\[
E[x_1] = x_0
\]

(3.84)

\[
E[y_1] = y_0 + \frac{s_0}{\eta} \frac{k-1}{\epsilon_\theta} \left[ \sin(\epsilon_\theta \tau) - \sin(\epsilon_\theta \tau) + \sin(\epsilon_\theta \tau) \right]
\]

(3.85)

Since \( \epsilon_\theta \) is small, both \( \sin(\epsilon_\theta \tau) \approx 0 \) and \( \sin(\epsilon_\theta \tau) \approx 0 \), so \( E[y_1] \approx y_0 + \frac{s_0}{\eta} \frac{k-1}{\epsilon_\theta} \), which agrees with the previous result.

Now consider the variance of the position. To avoid the issue of possible division by zero, the modification equations (3.22) and (3.23) (or (3.72) and (3.73) as well) will be used to find the variance.

\[
E[x_1^2] = E\left[ \left( \begin{array}{c} x_0 \\ \frac{1}{\theta^2} \left[ s_0^2 \theta \sin^2 \theta - s_0 \theta \cos \theta \sin \theta + s_0 \theta \cos \theta \sin \theta \right] if |\theta| > \epsilon_\theta \\
\left( s_0 \theta^2 + \frac{1}{2} \tilde{s} \tau^2 \right) \sin(\theta_0 + \theta \tau) \end{array} \right) \right] \]

(3.86)

\[
E[y_1^2] = E\left[ \left( \begin{array}{c} y_0 \\ \frac{1}{\theta^2} \left[ s_0^2 \theta \cos^2 \theta - s_0 \theta \sin \theta \cos \theta - \sin \theta_0 \cos \theta \right] if |\theta| > \epsilon_\theta \\
\left( s_0 \theta^2 + \frac{1}{2} \tilde{s} \tau^2 \right) \cos(\theta_0 + \theta \tau) \end{array} \right) \right] \]

(3.87)
To help simplify the discussion, assume as before that the target is initially moving due north, so $\theta_0 = 0$. In addition, without any loss of generality, assume the target starts at the origin, so $(x_0, y_0) = (0,0)$. No generality is lost since the starting position only translates the distribution, but does not change it. Thus, the expected values are effectively finding $E[(x_1 - x_0)^2] = E[(\Delta x_1)^2]$ and $E[(y_1 - y_0)^2] = E[(\Delta y_1)^2]$. Thus,

$$E[(\Delta x_1)^2] = E \left[ \left( \frac{1}{\theta^2} \left[ \dot{s} \sin \theta \tau - s_1 \theta \cos \theta \tau + s_0 \theta \right] \sin \theta \tau + \frac{1}{2} \dot{s}^2 \theta^3 \right) \sin \theta \tau \right]$$

$$E[(\Delta y_1)^2] = E \left[ \left( \frac{1}{\theta^2} \left[ \dot{s} \cos \theta \tau + s_1 \theta \sin \theta \tau - \dot{s} \right] \cos \theta \tau + \frac{1}{2} \dot{s}^2 \theta^3 \right) \cos \theta \tau \right]$$

Squaring yields

$$E[(\Delta x_1)^2] = E \left[ \left( \frac{s^2 \sin^2 \theta \tau}{\theta^4} + s_1^2 \frac{\cos^2 \theta \tau}{\theta^2} + s_0^2 \frac{1}{\theta^2} - 2 s \sin \theta \tau \cos \theta \tau \frac{\sin \theta \tau}{\theta^3} + 2 s_0 \frac{\sin \theta \tau \cos \theta \tau}{\theta^3} - 2 s_0 s_1 \frac{\cos^2 \theta \tau}{\theta^2} \right) \sin^2 \theta \tau \right]$$

$$E[(\Delta y_1)^2] = E \left[ \left( \frac{s^2 \sin^2 \theta \tau}{\theta^4} + s_1^2 \frac{(1 - \cos \theta \tau)^2}{\theta^4} - 2 s \sin \theta \tau \cos \theta \tau \frac{1 - \cos \theta \tau}{\theta^3} \right) \cos^2 \theta \tau \right]$$
Since $E[\dot{s}] = 0$ and $E[s_1] = s_0$,

\begin{equation}
E[(\Delta x_1)^2] = E\left[ \left( \frac{s^2 \sin^2 \theta \tau}{\theta^4} + s_1^2 \cos^2 \theta \tau \cdot \frac{\sin \theta \cos \theta \tau}{\theta^2} + s_0^2 \frac{1}{\theta^2} - 2s_1 \frac{\sin \theta \cos \theta \tau}{\theta^3} - 2s_0 \frac{\cos \theta \tau}{\theta^2} \right) \right] \text{ if } |\theta| > \epsilon_\theta_2
\end{equation}

\begin{equation}
= \left\{ \begin{array}{ll}
E[\dot{s}^2]E\left[ \frac{\sin^2 \theta \tau}{\theta^4} \right] + E[s_1^2]E\left[ \frac{\cos^2 \theta \tau}{\theta^2} \cdot \frac{1}{\theta^2} - 2E[\dot{s}s_1]E\left[ \frac{\sin \theta \cos \theta \tau}{\theta^3} - 2s_0^2 E\left[ \frac{\cos \theta \tau}{\theta^2} \right] \text{ if } |\theta| > \epsilon_\theta_1, \\
E[s_0^2 \tau^2 + \frac{1}{4}s_0^2 \tau^4] E[\sin^2 \theta \tau] \text{ otherwise} 
\end{array} \right.
\end{equation}

\begin{equation}
E[(\Delta y_1)^2] = E\left[ \left( \frac{s^2 \sin \theta \tau}{\theta^2} + s_1^2 \frac{1 - \cos \theta \tau}{\theta^4} - 2ss_1 \frac{\sin \theta (1 - \cos \theta \tau)}{\theta^3} \right) \right] \text{ if } |\theta| > \epsilon_\theta
\end{equation}

\begin{equation}
= \left\{ \begin{array}{ll}
E[\dot{s}^2]E\left[ \frac{\sin \theta \tau}{\theta^2} \right] + E[\dot{s}^2]E\left[ \frac{1 - \cos \theta \tau}{\theta^4} \right] - 2E[\dot{s}s_1]E\left[ \frac{\sin \theta (1 - \cos \theta \tau)}{\theta^3} \right] \text{ if } |\theta| > \epsilon_\theta \text{ otherwise} 
\end{array} \right.
\end{equation}

But,

\begin{equation}
E[\dot{s}s_1] = E[\dot{s}(s_0 + \dot{s}\tau)] = E[s_0\dot{s} + \dot{s}^2\tau] = s_0E[\dot{s}] + \tau E[\dot{s}^2] = \tau s_0^2
\end{equation}

So (3.92) and (3.93) become,

\begin{equation}
E[(\Delta x_1)^2]
= \left\{ \begin{array}{ll}
\sigma_x^2 E\left[ \frac{\sin^2 \theta \tau}{\theta^4} \right] + \frac{1}{4}\sigma_x^2 \left[ \frac{s^2 \sin \theta \tau}{\theta^2} + s_0^2 \frac{1}{\theta^2} - 2s_1 \frac{\sin \theta \cos \theta \tau}{\theta^3} - 2s_0 \frac{\cos \theta \tau}{\theta^2} \right] \text{ if } |\theta| > \epsilon_\theta \\
(s_0^2 \tau^2 + \frac{1}{4}s_0^2 \tau^4) E[\sin^2 \theta \tau] \text{ otherwise} 
\end{array} \right.
\end{equation}

\begin{equation}
E[(\Delta y_1)^2]
= \left\{ \begin{array}{ll}
(\tau^2\sigma_x^2 + s_0^2) E\left[ \frac{\sin \theta \tau}{\theta^2} \right] + \sigma_x^2 \left[ \frac{1 - \cos \theta \tau}{\theta^4} \right] - 2\sigma_x^2 E\left[ \frac{\sin \theta (1 - \cos \theta \tau)}{\theta^3} \right] \text{ if } |\theta| > \epsilon_\theta \\
(s_0^2 \tau^2 + \frac{1}{4}s_0^2 \tau^4) E[\cos^2 \theta \tau] \text{ otherwise} 
\end{array} \right.
\end{equation}

The remaining expected values must be solved using the definition of expected value. The integral equations for these expected values can be somewhat simplified by noting that all the integrands are even functions, and thus, symmetric to the y-axis. The expected value can be solved by integrating only over non-negative x values and doubling the result.
\[ E[(\Delta x_t)^2] = 2\sigma^2 \int_{\theta_0}^{\hat{\theta}} \frac{\sin^2 \hat{\theta} \tau}{\theta^4} d\hat{\theta} + 2(\tau^2\sigma^2 + s_0^2) \int_{\theta_0}^{\hat{\theta}} \frac{\cos^2 \hat{\theta} \tau}{\theta^2} d\hat{\theta} + 2s_0^2 \int_{\theta_0}^{\hat{\theta}} \frac{1}{\theta^2} d\hat{\theta} \]
\[- 4\tau\sigma^2 \int_{\theta_0}^{\hat{\theta}} \sin \hat{\theta} \tau \cos \hat{\theta} \tau \frac{1}{\theta^3} d\hat{\theta} - 4s_0^2 \int_{\theta_0}^{\hat{\theta}} \cos \hat{\theta} \tau \frac{1}{\theta^2} d\hat{\theta} \]
\[+ 2 \left( s_0^2 \tau^2 + \frac{1}{4} s_0^2 \tau^4 \right) \int_0^{\hat{\theta}} \sin^2 \hat{\theta} \tau d\hat{\theta} \]

\[ E[(\Delta y_t)^2] = 2(\tau^2\sigma^2 + s_0^2) \int_{\theta_0}^{\hat{\theta}} \frac{\sin \hat{\theta} \tau (1 - \cos \hat{\theta} \tau)}{\theta^3} d\hat{\theta} - 2s_0^2 \int_{\theta_0}^{\hat{\theta}} \cos^2 \hat{\theta} \tau d\hat{\theta} \]
\[- 4\tau\sigma^2 \int_{\theta_0}^{\hat{\theta}} \sin \hat{\theta} \tau (1 - \cos \hat{\theta} \tau) \frac{1}{\theta^3} d\hat{\theta} - 2 \left( s_0^2 \tau^2 + \frac{1}{4} s_0^2 \tau^4 \right) \int_{\theta_0}^{\hat{\theta}} \cos^2 \hat{\theta} \tau d\hat{\theta} \]
\[+ 2 \left( s_0^2 \tau^2 + \frac{1}{4} s_0^2 \tau^4 \right) \int_0^{\hat{\theta}} \sin^2 \hat{\theta} \tau d\hat{\theta} \]

All of the integrals arising from these expected values are solvable (based on Wolfram) but produce "messy" solutions. Thus, although a closed form solution can be found for the variance of the position, it is much too complicated to be useful. In the next section (section 3.2.5) the predicted state and covariance will be estimated using numerical methods.

3.2.5 The Mean and Variance of the Transition Density (Double Maneuver)

Now that the distribution parameters have been determined for the case when there is one maneuver over the update time interval, the case when there are two maneuvers is considered next. Recall that this was the assumption used to generate uncertainty distributions in Fig. 3-5.

For this two-step case, assume the target starts with some randomly selected acceleration and turn rate and executes that maneuver for \( \tau \) time, just as was assumed for the one-step case. But at time \( \tau \), the target randomly selects another acceleration and turn rate and executes that maneuver for the remainder of the update time interval, \( T - \tau \). This assumed process now makes \( \tau \) a random
variable as well. Maintaining consistency with the rest of the analysis, assume $\tau$ is a uniform random variable such that $0 \leq \tau \leq T$. Therefore,

\[ E[\tau] = \frac{1}{2}T \]  

\[ Var[\tau] = \sigma^2 = \frac{1}{12}T^2 \]  

To deal with the multiple maneuvers, the (randomly) selected accelerations and turn rates need to be subscripted as well. As before, assume at a specified update time, the target is at $(x_0, y_0)$. Let this update time be at time zero. The (predicted) state of the target is desired for the next update time, $T$. Let $\tau$ be time when the target finishes its first maneuver and starts its second (and last) maneuver over the time update interval. As before, the subscripts reflect target states within an update time interval. So $(x_0, y_0)$ is the position of the target at the beginning of the update time interval having initial speed $s_0$ and initial course $\theta_0$. An acceleration $\dot{s}_1$ and turn rate, $\dot{\theta}_1$, is (randomly) selected at this time. The duration of this first maneuver, $\tau$, is also (randomly) selected. The target would then have position $(x_1, y_1)$ $\tau$ time later with speed $s_1$ and course $\theta_1$. At this point in time, the target (randomly) selects a new acceleration and turn rate, $\dot{s}_2$ and $\dot{\theta}_2$, respectively. The target executes that maneuver for the remainder of the update time interval, $T - \tau$. At the end of the update time period, $T$, the target would have position $(x_2, y_2)$ with speed and course $s_2$ and $\theta_2$, respectively. Using this notation,

\[ s_1 = s_0 + \dot{s}_1 \tau \]  

\[ s_2 = s_1 + \dot{s}_2(T - \tau) = s_0 + \dot{s}_1 \tau + \dot{s}_2(T - \tau) \]  

\[ \theta_1 = \theta_0 + \dot{\theta}_1 \tau \]  

\[ \theta_2 = \theta_1 + \dot{\theta}_2(T - \tau) = \theta_0 + \dot{\theta}_1 \tau + \dot{\theta}_2(T - \tau) \]  

\[ \dot{x}_1 = s_1 \sin \theta_1 \]  

\[ \dot{x}_2 = s_2 \sin \theta_2 \]  

\[ \dot{y}_1 = s_1 \cos \theta_1 \]  

\[ \dot{y}_2 = s_2 \cos \theta_2 \]
\[ y_2 = s_2 \cos \theta_2 \]  
(3.108)

\[ x_1 = x_0 + \frac{1}{\theta_1} \left[ \dot{s}_1 \sin \theta_1 - s_1 \dot{\theta}_1 \cos \theta_1 - \dot{s}_1 \sin \theta_0 + s_0 \dot{\theta}_1 \cos \theta_0 \right] \]  
(3.109)

\[ x_2 = x_1 + \frac{1}{\theta_2} \left[ \dot{s}_2 \sin \theta_2 - s_2 \dot{\theta}_2 \cos \theta_2 - \dot{s}_2 \sin \theta_1 + s_1 \dot{\theta}_2 \cos \theta_1 \right] \]  
(3.110)

\[ y_1 = y_0 + \frac{1}{\theta_1} \left[ \dot{s}_1 \cos \theta_1 + s_1 \dot{\theta}_1 \sin \theta_1 - \dot{s}_1 \cos \theta_0 - s_0 \dot{\theta}_1 \sin \theta_0 \right] \]  
(3.111)

\[ y_2 = y_1 + \frac{1}{\theta_2} \left[ \dot{s}_2 \cos \theta_2 + s_2 \dot{\theta}_2 \sin \theta_2 - \dot{s}_2 \cos \theta_1 - s_1 \dot{\theta}_2 \sin \theta_1 \right] \]  
(3.112)

### 3.2.5.1 The Mean and Variance of the Speed and Course

Determining the mean and variance of the speed and course for the two-manuever case is fairly straightforward. To find these parameters, note that it is necessary to deal with a product of independent random variables. It is well known the variance of a product of independent random variables, A,B is given by:

\[ \text{Var}[AB] = (E[A])^2 \text{Var}[B] + (E[B])^2 \text{Var}[A] + \text{Var}[A] \text{Var}[B] \]  
(3.113)

Using this property along with \( E[\dot{s}_1] = E[\dot{s}_2] = E[\dot{\theta}_1] = E[\dot{\theta}_2] = 0 \), \( E[T - \tau] = E[\tau] = \frac{T}{2} \), \( \text{Var}[\dot{s}_1] = \text{Var}[\dot{s}_2] = \sigma_s^2 \), and \( \text{Var}[T - \tau] = \text{Var}[\tau] = \sigma_t^2 = \frac{1}{12}T^2 \), then

\[ E[s_2] = E[s_0] + E[s_1 \tau] + E[\dot{s}_2(T - \tau)] = s_0 + E[s_1]E[\tau] + E[\dot{s}_2]E[T - \tau] = s_0 \]  
(3.114)

\[ \text{Var}[s_2] = \text{Var}[s_0] + \text{Var}[s_1 \tau] + \text{Var}[\dot{s}_2(T - \tau)] \]

\[ = \{ (E[s_1])^2 \text{Var}[\tau] + (E[\tau])^2 \text{Var}[s_1] + \text{Var}[s_1] \text{Var}[\tau] \} \]

\[ + \{ (E[\dot{s}_2])^2 \text{Var}[T - \tau] + (E[T - \tau])^2 \text{Var}[\dot{s}_2] + \text{Var}[\dot{s}_2] \text{Var}[T - \tau] \} \]  
(3.115)

\[ = 2 \left[ \left( \frac{1}{2} \right)^2 \sigma_s^2 + \sigma_s^2 \left( \frac{1}{12}T^2 \right) \right] = \frac{2}{3}T^2 \sigma_s^2 \]

\[ E[\theta_2] = E[\theta_0] + E[\dot{\theta}_1 \tau] + E[\dot{\theta}_2(T - \tau)] = \theta_0 + E[\dot{\theta}_1]E[\tau] + E[\dot{\theta}_2]E[T - \tau] = \theta_0 \]  
(3.116)
\[\text{Var}[\theta_2] = \text{Var}[\theta_0] + \text{Var}[\theta_1 \tau] + \text{Var}[\dot{\theta}_2 (T - \tau)] \]
\[= \left\{ (E[\dot{\theta}_1])^2 \text{Var}[\tau] + (E[\tau])^2 \text{Var}[\dot{\theta}_1] + \text{Var}[\dot{\theta}_1] \text{Var}[\tau] \right\} \]
\[+ \left\{ (E[\dot{\theta}_2])^2 \text{Var}[T - \tau] + (E[T - \tau])^2 \text{Var}[\dot{\theta}_2] + \text{Var}[\dot{\theta}_2] \text{Var}[T - \tau] \right\} \tag{3.117} \]
\[= 2 \left[ \left( \frac{1}{2} r \right)^2 \sigma_{\dot{\theta}}^2 + \sigma_{\dot{\theta}}^2 \left( \frac{1}{12} r^2 \right) \right] = \frac{2}{3} r^2 \sigma_{\dot{\theta}}^2 \]

Comparing (3.114)-(3.117) to (3.30)-(3.33) shows the means for the speed and course for the two-maneuver case are the same as those for the one-maneuver case, but the variances for the two-maneuver case are smaller by two thirds. (Recall that \(\tau = T\) for the one-maneuver case.) The fact that the variance shrinks when the number of maneuvers increases agrees with the empirical result in section 3.2.3.

### 3.2.5.2 The Mean and Variance of the State

Determining the mean and variance of the state when the target undergoes two (kinematically-constrained) maneuvers turns out to be difficult to do in closed-form. However, it is straightforward to approximate these parameters using Monte Carlo techniques. For all the cases, the target was initially at the origin moving due North at 20 m/s. The update time interval was \(T = 10\) s and the maximum acceleration was \(A = 2\) m/s\(^2\). Thus, \((x_0, y_0) = (0, 0), s_0 = 20,\) and \(\theta_0 = 0\). Six different maximum turn rates \((\dot{\theta})\) were used: 0.1, 1.0, 3.0, 5.0, 10.0, and 15.0 deg/s. The accelerations and turn rates were assumed to be uniformly distributed. Thus, each acceleration was randomly selected uniformly over the interval \([-A, +A]\) and each turn rate was randomly selected uniformly over the interval \([-\dot{\theta}, +\dot{\theta}]\). The results from the analysis are summarized in Table 3-1. This table compares the predicted state and covariance from the CGBF to the ones from a Kalman filter, for the six maximum turn rates. Keep in mind that a Kalman filter does not use turn rates so its predicted state and covariance do not change as the maximum turn rate is varied. Thus, there is only one column for the Kalman filter. It is straightforward to compute the predicted state and covariance for the Kalman filter. Refer to Eqs. (1.1) and (1.2)
which compute the predicted state and covariance for a Kalman filter. Using the parameters
assumed for the Monte Carlo analysis:

\[
X_{[n-1|n-1]} = \begin{bmatrix} 0 & 0 & 0 & 20 \end{bmatrix}^T
\]

(3.118)

\[
\Phi = \begin{bmatrix}
1 & 0 & T & 0 \\
0 & 1 & 0 & T \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 10 & 0 \\
0 & 1 & 0 & 10 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(3.119)

\[
Q = \begin{bmatrix}
\frac{1}{2} \sigma_d^2 T^4 & 0 & \frac{1}{2} \sigma_d^2 T^3 & 0 \\
0 & \frac{1}{2} \sigma_d^2 T & 0 & \frac{1}{2} \sigma_d^2 T^3 \\
\frac{1}{2} \sigma_d^2 T^3 & 0 & \sigma_d^2 T^2 & 0 \\
0 & \frac{1}{2} \sigma_d^2 T^3 & 0 & \sigma_d^2 T^2
\end{bmatrix}
\]

(3.120)

Since it is assumed that the initial position of the target is known exactly, \( P_{[n-1|n-1]} = 0 \).

Therefore, the mean of the predicted distribution is \( X_{[n|n-1]} = \Phi X_{[n-1|n-1]} \) and covariance of
the distribution is \( P_{[n|n-1]} = Q \). As a result, the prediction distribution is just based on the
process noise (and centered on the predicted position).

Appendix B provides the MATLAB code used for this analysis. For the analysis, 50,000,000
samples were used. However, the analysis exploited the motion symmetry (discussed later in
section 3.3.1) to effectively achieve twice the number of samples, \( i.e., \) effectively 100,000,000
samples were used, for each result.

As Table 3-1 shows, the mean and variance of the prediction distribution can vary
dramatically depending on the maximum turn rate assumed for the target. For example, when the
maximum turn rate is small, the CGBF predicted (mean) position is nearly identical to Kalman.
But as the maximum turn rate increases to 15 deg/s, the \( y \) position diminishes from 200 m to
~151, a nearly 25% reduction. The y component of the velocity reduces from 20 m/s to ~8 m/s, a ~60% reduction.

The variance terms show an interesting difference as well. As typically assumed, the Kalman filter uses a circular Gaussian distribution for the process noise. It does not usually consider the initial direction of the target. As a result, when the target has a small maximum turn rate, the Kalman filter grossly overestimates the uncertainty perpendicular to the initial course of the target. The target needs a maximum turn rate of over 10 deg/s to get the same magnitude of uncertainty that the Kalman assumes.

Table 3-1: Comparing the CGBF predicted state and covariance to Kalman as the maximum turn rate of the target is varied (accelerations are selected from uniform distribution).

<table>
<thead>
<tr>
<th>Kalman</th>
<th>CGBF (θ = 1 deg/s)</th>
<th>CGBF (θ = 3 deg/s)</th>
<th>CGBF (θ = 5 deg/s)</th>
<th>CGBF (θ = 10 deg/s)</th>
<th>CGBF (θ = 15 deg/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>0.000</td>
<td>0.000</td>
<td>-0.000</td>
<td>0.000</td>
<td>-0.000</td>
</tr>
<tr>
<td>y</td>
<td>200.000</td>
<td>199.989</td>
<td>199.736</td>
<td>197.735</td>
<td>193.774</td>
</tr>
<tr>
<td>ẋ</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Variance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xx</td>
<td>3333.333</td>
<td>0.829</td>
<td>82.609</td>
<td>728.791</td>
<td>1944.462</td>
</tr>
<tr>
<td>yy</td>
<td>3333.333</td>
<td>2444.876</td>
<td>2435.130</td>
<td>2365.718</td>
<td>2258.275</td>
</tr>
<tr>
<td>xx</td>
<td>133.333</td>
<td>0.033</td>
<td>3.323</td>
<td>28.810</td>
<td>74.241</td>
</tr>
<tr>
<td>yy</td>
<td>133.333</td>
<td>88.899</td>
<td>88.269</td>
<td>83.820</td>
<td>77.672</td>
</tr>
<tr>
<td>Covariance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xx</td>
<td>666.667</td>
<td>0.157</td>
<td>15.658</td>
<td>136.885</td>
<td>358.597</td>
</tr>
<tr>
<td>yy</td>
<td>666.667</td>
<td>444.515</td>
<td>442.027</td>
<td>424.169</td>
<td>397.230</td>
</tr>
<tr>
<td>xy</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>-0.000</td>
</tr>
<tr>
<td>ẋy</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ẋy</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Using 50 million samples with $A = 2 \text{ m/s}^2$ and $\Theta = 0.1, 1, 3, 5, 10$ and 15 deg/s with acceleration and turn rates selected from a uniform distribution.
Finally, the covariance terms show an interesting similarity and difference. The typical process noise for the Kalman filter only specifies two covariance terms: \( x \) position with the \( x \) velocity and \( y \) position with the \( y \) velocity. The other four covariance terms are assumed to be zero. As shown in the table, zeroing out the last four covariance term is a quite reasonable assumption since the CGBF estimates them to be zero as well. However, the table shows that for the two covariance terms that the Kalman filter computes, they are quite different from the CGBF. The Kalman filter uses \( \sim 667 \text{ m}^2/\text{s} \) for both \( x \) and \( y \). The CGBF estimates the \( xx \) covariance to be as small as \( \sim 0.2 \text{ m}^2/\text{s} \) for a small maximum turn rate target to as large as \( \sim 1565 \text{ m}^2/\text{s} \) for a high maximum turn rate target. Interestingly, the \( yy \) covariance does not vary much relative to the maximum turn rate. For most turn rates, it stays around \( 400 \text{ m}^2/\text{s} \), which is about \( 35\% \) smaller than the corresponding Kalman value.

However, a key thing that makes the comparison unfair is that the Kalman filter assumes the acceleration changes are normally distributed while the analysis summarized in Table 3-1 assumed the acceleration changes were uniformly distributed. To address this concern, the Monte Carlo analysis was repeated using acceleration changes that were normally distributed, truncated to four standard deviations. Thus, the maximum acceleration and turn rate were now assumed to be the four-sigma values. So now the acceleration variance, \( \sigma_a^2 \) is different. In the uniform case, \( \sigma_a^2 = \frac{3}{2} A^2 \), but now \( \sigma_a^2 = \left( \frac{1}{4} A \right)^2 \). Therefore, in addition to the accelerations being normally distributed, the acceleration variance is over five times smaller. The results of the analysis using accelerations and turn rates that are truncated normally distributed are shown in Table 3-2.

As Table 3-2 shows, using normally distributed maneuvers, the mean of the predicted state is less affected by the maximum turn rate than when using uniformly distributed maneuvers. Instead of a \( 25\% \) reduction in the \( y \) position, it is now only about \( 5\% \) reduction. The \( y \) component of the velocity reduced by only \( 15\% \), instead of by \( 60\% \).
Table 3-2: Comparing the CGBF predicted state and covariance to Kalman as the maximum turn rate of the target is varied (accelerations are selected from truncated normal).

<table>
<thead>
<tr>
<th></th>
<th>Kalman</th>
<th>CGBF ((\Theta = 1) deg/s)</th>
<th>CGBF ((\Theta = 1) deg/s)</th>
<th>CGBF ((\Theta = 3) deg/s)</th>
<th>CGBF ((\Theta = 5) deg/s)</th>
<th>CGBF ((\Theta = 10) deg/s)</th>
<th>CGBF ((\Theta = 15) deg/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x)</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>-0.000</td>
<td>-0.000</td>
<td>-0.000</td>
<td>-0.000</td>
</tr>
<tr>
<td>(y)</td>
<td>200.000</td>
<td>200.000</td>
<td>199.953</td>
<td>199.571</td>
<td>198.818</td>
<td>195.344</td>
<td>189.786</td>
</tr>
<tr>
<td>(\dot{x})</td>
<td>0.000</td>
<td>-0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>-0.000</td>
<td>-0.000</td>
<td>-0.000</td>
</tr>
<tr>
<td>(\dot{y})</td>
<td>20.000</td>
<td>20.000</td>
<td>19.987</td>
<td>19.886</td>
<td>19.686</td>
<td>18.773</td>
<td>17.349</td>
</tr>
<tr>
<td>Variance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xx)</td>
<td>625.000</td>
<td>0.142</td>
<td>14.233</td>
<td>127.295</td>
<td>349.644</td>
<td>1325.911</td>
<td>2739.034</td>
</tr>
<tr>
<td>(yy)</td>
<td>625.000</td>
<td>457.980</td>
<td>457.581</td>
<td>455.221</td>
<td>452.449</td>
<td>468.885</td>
<td>584.679</td>
</tr>
<tr>
<td>(\dot{x}\dot{x})</td>
<td>25.000</td>
<td>0.005</td>
<td>0.529</td>
<td>4.705</td>
<td>12.803</td>
<td>46.494</td>
<td>89.792</td>
</tr>
<tr>
<td>(\dot{y}\dot{y})</td>
<td>25.000</td>
<td>16.654</td>
<td>16.631</td>
<td>16.477</td>
<td>16.322</td>
<td>17.720</td>
<td>25.821</td>
</tr>
<tr>
<td>Covariance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xx)</td>
<td>125.000</td>
<td>0.026</td>
<td>2.610</td>
<td>23.283</td>
<td>63.642</td>
<td>235.900</td>
<td>469.831</td>
</tr>
<tr>
<td>(yy)</td>
<td>125.000</td>
<td>83.272</td>
<td>83.175</td>
<td>82.558</td>
<td>81.852</td>
<td>85.983</td>
<td>114.121</td>
</tr>
<tr>
<td>(xy)</td>
<td>0</td>
<td>-0.000</td>
<td>-0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>(yx)</td>
<td>0</td>
<td>-0.000</td>
<td>-0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>(\dot{x}\dot{y})</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>-0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>(\dot{y}\dot{y})</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>-0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Using 50 million samples with \(A = 2 \text{ m/s}^2\) and \(\Theta = 0.1, 1, 3, 5, 10\) and 15 degs/s with acceleration and turn rates selected from a normal distribution truncated to 4 standard deviations.

The variance in the predicted state still has significant differences. The process noise from the Kalman filter calculates the variance in \(x\) position to be 625 m\(^2\) but the CGBF shows that this variance varies from 0.14 m\(^2\) (over 4400 times smaller) for a small maximum turn rate to over 2700 m\(^2\) (over four times larger) for a large maximum turn rate. The variance in the \(x\) component of velocity is also quite different. The process noise from the Kalman filter calculates the variance in \(x\) velocity to be 25 m\(^2\)/s\(^2\) but the CGBF shows that this variance varies from nearly zero for a small maximum turn rate to ~90 m\(^2\)/s\(^2\) (nearly four times larger) for a large maximum turn rate.

Finally, the covariance terms show a trend similar to the uniform distribution case. As before, the last four covariance terms agree with the Kalman filter process noise model in that they are all zero. However, while the process noise for the Kalman filter calculates the \(xx\) covariance to be 125 m\(^2\)/s, the CGBF estimates this covariance to be as small as ~0.02 m\(^2\)/s for a small maximum
turn rate target to as large as ~470 m²/s for a high maximum turn rate target. As before, the \( y \dot{y} \) covariance does not vary much relative to the maximum turn rate. For most turn rates, it stays around 83 m²/s which, like the uniform case, is again about 35% smaller than the corresponding Kalman value.

Thus, regardless of the assumed distribution for the acceleration changes, the GCBF computed prediction distribution is much different than the Kalman filter. Furthermore, since the predicted covariance from the CGBF is tighter and more aligned to the real target state, it suggests that the CGBF should yield tighter and more accurate state estimates since these estimates are directly influenced by the predicted covariance.

3.3 Motion Update

Unfortunately, performing a motion predict is particularly computational for a GBF. Each cell in the grid must be propagated to all the other cells that the target could have transitioned to during the time to the next measurement update. These propagations are found by generating a large number of Monte Carlo samples for each cell in the grid. Since the grids tend to be large, an enormous number of Monte Carlo samples would be required for each cell to form a good approximation of the cell’s transition distribution. In [6], Aughenbaugh and La Cour state that some computational savings can be achieved by using probability thresholding. The idea is to ignore transitioning those cells that have probability less than some (threshold) value. While we agree with this idea and useable in the CGBF, it does not significantly reduce the processing requirements.

To form the target starting position for each cell, there are two approaches: 1) use the center position of the cell, or 2) randomly sample within the cell positional limits. Randomly sampling within the cell limits is usually a better approach because the true target position could be near an edge of the cell. If the cells are large compared to the distance the target moves in an update, then
the errors introduced by only using the cell center could be significant. However, random
sampling over the cell may require even more Monte Carlo samples per cell to get a good
distribution over the entire cell. Aughenbaugh and La Cour recently suggested a mix of these two
approaches by re-using the same random samples across all cells to help reduce some of the
computational complexity. They mention, however, that this approach could lead to bias in the
results [6].

An alternative motion update procedure was developed that has a few refinements over these
two approaches. The first refinement is how the mass in the cells is transitioned. As with the
current approaches, a large number of Monte Carlo samples are generated for each cell, but since
the grids are significantly smaller, many fewer samples are needed per cell. This results in
substantially fewer samples needed for each update. Furthermore, current GBF approaches move
the mass in a cell treating the cell state as a “point”, similar to a particle in a particle filter. In the
CGBF, instead of moving a point within the cell, the entire cell region is moved per Monte Carlo
sample. This is illustrated in Fig. 3-10. By moving the entire cell for each sample, there is more
assurance of obtaining a good distribution over each origination cell, without the need to
randomly sample within the cell limits.

![Figure 3-10: Each entire cell is moved per Monte Carlo sample; mass is apportioned.](image)

67
Note from Fig. 3-10 that a transitioned cell from the origination grid will generally land across boundaries of cells within the destination grid. The probability in the transitioned origination cell is apportioned out to the destination cells based on the fractional coverage it receives (from the transitioned cell). The function CellOverlapFraction in Fig. 3-1 determines this fractional coverage. Another benefit of this approach is that the cells do not need to be the same size from the origination grid to the destination grid. Each destination cell only gets the percentage of mass based on the portion of overlap. The final refinement exploits the motion symmetry to obtain two Monte Carlo samples from each sample made. This final refinement is detailed in the next section.

3.3.1 Exploiting the Motion Symmetry

As was pointed out earlier, the transition distribution is symmetric with respect to the target's initial course. As a result, this symmetry can be exploited in the generation of the random samples. Each time the target is randomly moved throughout the time interval, the mirrored state can also be used as a sample. Thus, two random samples are generated for each random walk which results in $2m$ samples being generated for the computational cost of generating $m$ samples.

Let $(x_0, y_0)$ be the (estimated) initial state of the target at the beginning of the update time interval and $(x_k, y_k)$ be the final predicted state at the end of the time interval (by undergoing $k$ maneuvers). Similarly, let $s_0$ and $\theta_0$ be the initial speed and course and $s_k$ and $\theta_k$ be the predicted speed and course at the end of the interval. The mirrored state is defined as the resulting state of the target if it simply reversed all its turns such that left turns become right turns, and vice versa. This mirrored state can be used as a second Monte Carlo sample for each sample actually computed. The mirrored state is straightforward to determine directly from the final state of the target after it completed all its maneuvers for the time interval. Refer to Fig. 3-11.
The actual target state is \((x_k, y_k)\) at the end of time interval. The mirrored target state is \((x'_k, y'_k)\) with speed \(s'_k\) and course \(\theta'_k\). Let \(\beta\) be the angle between the initial course of the target and the angle to the predicted state, \((x_k, y_k)\). Let \(\alpha = \theta_0 + \beta\). Therefore, the mirrored position can be found as:

\[
\Delta x = x_k - x_0 \tag{3.121}
\]
\[
\Delta y = y_k - y_0 \tag{3.122}
\]
\[
d = \sqrt{\Delta x^2 + \Delta y^2} \tag{3.123}
\]
\[
\sin \alpha = \frac{\Delta x}{d} \tag{3.124}
\]
\[
\cos \alpha = \frac{\Delta y}{d} \tag{3.125}
\]
\[
x'_k = x_0 + d \sin(\theta_0 - \beta) \tag{3.126}
\]
\[
y'_k = y_0 + d \cos(\theta_0 - \beta) \tag{3.127}
\]

But \(\theta_0 - \beta = \theta_0 - (\theta_0 + \beta) + \theta_0 = 2\theta_0 - \alpha\). So,
\[ x_k' = x_0 + d \sin(2\theta_0 - \alpha) = x_0 + d[\sin 2\theta_0 \cos \alpha - \cos 2\theta_0 \sin \alpha] \]
\[ = x_0 + d \left( \sin 2\theta_0 \frac{\Delta y}{d} - \cos 2\theta_0 \frac{\Delta x}{d} \right) \tag{3.128} \]
\[ = x_0 + \Delta y \sin 2\theta_0 - \Delta x \cos 2\theta_0 \]
\[ y_k' = y_0 + d \cos(2\theta_0 - \alpha) = y_0 + d[\cos 2\theta_0 \cos \alpha + \sin 2\theta_0 \sin \alpha] \]
\[ = y_0 + d \left( \cos 2\theta_0 \frac{\Delta y}{d} + \sin 2\theta_0 \frac{\Delta x}{d} \right) \tag{3.129} \]
\[ = y_0 + \Delta y \cos 2\theta_0 + \Delta x \sin 2\theta_0 \]

Computing the mirrored speed and course is straightforward as well. By definition, the mirrored speed does not change; only the target's course. Let \( \Delta \theta = \theta_k - \theta_0 \) be the course difference from the initial course to the final course. Thus, the final course can be thought of as \( \theta_k = \theta_0 + \Delta \theta \). The mirrored course must be \( \theta_k' = \theta_0 - \Delta \theta \). Therefore the mirrored state is:

\[ x_k' = x_0 + (y_k - y_0) \sin 2\theta_0 - (x_k - x_0) \cos 2\theta_0 \tag{3.130} \]
\[ y_k' = y_0 + (y_k - y_0) \cos 2\theta_0 + (x_k - x_0) \sin 2\theta_0 \tag{3.131} \]
\[ s_k' = s_k \tag{3.132} \]
\[ \theta_k' = \theta_0 - \Delta \theta = \theta_0 - (\theta_k - \theta_0) = 2\theta_0 - \theta_k \tag{3.133} \]

3.3.2 Using Uniform RVs to Approximate Normal RVs

Although many computer systems provide the ability to generate normal random variables, they typically require much more computations than the generation of uniform random variables. When performing the Monte Carlo sampling, the random variables must be drawn from a normal distribution (see Fig. 3-1). Fortunately, the CGBF sampling process does not need “true” normal random variables; only that they be approximately normal. A simple approximation was developed for the CGBF to exploit this idea. The normal-like random variables are generated using the following procedure. Suppose a total of \( m \) random samples are needed. Each random
sample is a value from a normal distribution with a specified standard deviation. The number of random samples is divided into thirds. Let \( k = m/3 \). Then:

* generate \( k \) uniform random samples that are within one sigma,

* generate \( k \) uniform random samples that are within two sigma, and finally,

* generate \( k \) uniform random samples that are within three sigma.

For obvious reasons, this method is called the \textit{thirds procedure}. For a “true” (1D) normal random variable, \(~68.2\%\) of the samples would be within one sigma, \(~95.4\%\) are within two sigma, and \(~99.7\%\) are within three sigma\cite{1}. Using the approximating procedure just described, \( k + \frac{1}{2}k + \frac{2}{3}k = \frac{k}{6} (6 + 3 + 2) = \frac{11}{6} \left( \frac{m}{3} \right) = (61.1\%) \) samples will be within one sigma, \( k + k + \frac{2}{3}k = \frac{k}{3} (3 + 3 + 2) = \frac{8}{3} \left( \frac{m}{3} \right) = (88.8\%) \) samples will be within two sigma, and of course, 100\% of the samples will be within three sigma. These percentages are close enough to get the proper sampling. Thus, using this thirds procedure with uniform random variables, the needed normal random variables are efficiently obtained.
Chapter 4 - Trade Study: Comparison Performance

A trade study was conducted to compare the performance of a CGBF to a Kalman filter. As mentioned earlier, conducting trade studies is an important systems engineering endeavor. Of course, in addition to deciding to conduct a trade study, engineering principles must be applied to determine how the study will be conducted and how the results will be evaluated. For this comparison, the approach was to implement the algorithms in MATLAB®. A maneuvering target model and detection model was developed and implemented in MATLAB as well. Monte Carlo analysis was conducted, running the comparisons many times to determine which filter provided more accurate state estimates on the average. The block diagram of the paradigm used to compare the performance is shown in Fig. 4-1. As the figure shows, the parameters for the target and sensor were specified. The parameters for the target were its maximum acceleration (and deceleration), maximum speed, and maximum turn rate. The sensor parameters were the measurement error and update rate.

The attempt was to make the comparison as fair as possible. However, due to the different assumptions made by the filters, some bias was unavoidable. The key difference between the filters to be examined is the benefit derived from using a more realistic, non-Gaussian prediction distribution. For the comparison, the same sensor measurements were provided to both filter methods and the track estimates from these filters were then compared to the true target state. The sensor measurements were also a source of comparison bias as well. The Kalman filter assumes the measurement errors are (zero-mean) normally distributed, but the CGBF requires the measurement errors to be bounded by some known (finite) limit. There are two straightforward ways of addressing both filters: 1) use uniform random variables (as was done for most of the analysis in this report), or 2) use truncated normal random variables. Using truncated normal random variables does put a slight bias towards favoring CGBF since the Kalman filter cannot exploit the known bounds. However, using uniform random variables would tend to bias the
comparison results even more towards the CGBF since the Kalman filter treats all errors as Gaussian. When the errors are uniformly distributed (i.e., non-Gaussian), the Kalman filter becomes the best linear state estimator [11]. However, both types of measurement errors will be tested.

Decisions needed to be made as to what the target maneuvering scenarios should be like. It was decided that two different approaches should be used to compare the filter performance. The first approach assumes the target executes the same behavior over the duration of the scenario. This approach is called the uniform behavior scenario. This approach is further broken down into two types of targets: a constant velocity target and a maneuvering target. For the constant velocity target, the target maintains constant velocity over the duration of the scenario. The maneuvering target is allowed to execute a random path over the duration of the scenario, limited only by its kinematic constraints. The target uses the maneuvering motion model described in section 3.2. Of course, the filters are not “told” which type of target they are tracking. They are only given the maneuvering capability of the target. The details of the comparison using the uniform behavior scenario are given in section 4.1.

The second approach uses the scenario provided by Kirubarajan and Bar-Shalom in [55]. This scenario is referred to as the K&B scenario. Details of this scenario are presented in section 4.2. The comparison using the K&B scenario is presented in section 4.3.

Figure 4-1: Block diagram of the comparison methodology.
4.1 Comparison Using Uniform Behavior Scenario

A CGBF was compared to a Kalman filter using a uniform behavior scenario. As previously described, in a uniform behavior scenario, the target maintains the same maneuvering behavior over the duration of the scenario. The target was assumed to have known maximum limits on its speed, acceleration and turn rate. All target maneuvers were restricted to these limits. If the target reached its maximum speed, it remained at this maximum speed until a negative acceleration was selected to slow it down. For all the scenarios in this study, the target was limited to a maximum speed of 55 m/s. Six different targets were explored where they differed only by their maximum acceleration and maximum turn rate. A (standard) Kalman filter, however, does not use turn rates. To make a fair comparison, the turn rate was set so that it was equivalent to its acceleration. But how can this be achieved? It turns out that there is a straightforward solution. From Newtonian physics, the maximum acceleration can be obtained from the maximum turn rate using the centripetal force equation, \( F = m \frac{v^2}{r} \), where \( m \) is the mass, \( v \) is the tangent velocity, and \( r \) is the radius of the circle [72]. But also from Newtonian physics, it is known that \( F = ma \), where \( a \) is the acceleration. Setting these equations equal yields \( a = \frac{v^2}{r} \). Using the terms from section 3, \( v = s_{k-1} = s_0 \), so \( a = \frac{s_0^2}{r} \) and \( r \) is known using (3.15), so \( a = \frac{s_0^2}{s_0} \frac{\theta}{\theta} = s_0 \frac{\theta}{\theta} \). Therefore, the acceleration is maximized when the speed and turn rate are maximized. Let \( \theta \) be the maximum turn rate (in deg/s). Since it is assumed that the target has a maximum speed of 55 m/s, then the (equivalent) maximum acceleration is \( A = (55) \frac{\theta}{180} \approx 0.96 \theta \approx \theta \). Therefore, the maximum acceleration induced by the maximum turn rate is approximately equal to the maximum acceleration itself. Indeed, the maximum speed was specially selected to make this true. As a result, the process noise matrix for the Kalman filter can be based on the maximum acceleration parameter alone. Six target models were selected to sample increasing levels of target...
maneuvering. The accelerations were selected such that the maneuver index, $\lambda$, would be low maneuvering (0.25, 0.50), high maneuvering (0.75, 1.0), and very high maneuvering (2.0, 3.0). The 0.50 threshold value between low and high maneuvering was established in [55]. The very high maneuvering values were inferred from that same study. The 3.0 very high maneuvering case is 0.5 larger than the largest value considered in that study.

As stated earlier, the attempt was to make the comparison as fair as possible and to stay true to the Kalman filter assumptions with the exception of the prediction distribution. Therefore, the comparison was to only focus on the benefit the CGBF has by using a bounded, non-Gaussian prediction distribution. However, due to the differences in the two filters, some compromises needed to be made. The main compromise concerns the measurements. Although the Kalman filter assumes Gaussian distributed measurement errors, the CGBF requires the errors to be bounded by some known limit. As a result, for the comparison, the measurement errors were assumed to be truncated Gaussian, i.e., Gaussian distributed but truncated to a specified number of sigmas (i.e., standard deviations). For this analysis, all measurements were in $(x,y)$ but the measurement errors were truncated to 3 sigma. Therefore, the measurements can be written as:

$$z(t_k) = Hx(t_k) + \varepsilon(t_k)$$  \hspace{1cm} (4.1)

where $x(t_k)$ is the true state of the target at time $t_k$ and $\varepsilon(t_k) \sim N_3(0,\sigma)$. The notation $N_m(.)$ means normally distributed but truncated to $m$ standard deviations. The time between measurement updates was $T=10$ s, and remained constant over the scenario.

The following procedure was used to generate truncated normal random variables. For each measurement, a normal random variable was drawn to determine the measurement error. If this random variable exceeded 3 sigma, the variable was discarded and a new normal random variable was drawn. The standard deviations of these errors were $\sigma_x = \sigma_y = 100$ m for all runs. The measurement error was assumed to be independent of target range. It should be pointed out that
since the normal distribution was truncated to three sigma, the effective standard deviation is slightly less. In fact, the effective standard deviation can be shown to be:

\[
\sigma_{\text{eff}} = \sigma_x \sqrt{1 - \left( \frac{2}{m} \right) \frac{m}{\sqrt{\pi}} \text{erf} \left( \frac{m}{\sqrt{2}} \right)} e^{-\frac{1}{2} m^2}
\]  

(4.2)

where \( m \) is the maximum number of standard deviations and \( \text{erf}(.) \) is the error function. For \( m = 3 \), the multiplier in (4.2) is 0.9866, which is approximately equal to 1.

With the maneuver indexes and sensor parameters specified, the process noise, \( \sigma_a \) can be determined using (1.12). The Kalman filter assumes, however, that the acceleration changes are normally distributed. Since any acceleration change is possible under that assumption, it would violate the GCBF assumption that the maximum acceleration is bounded and known. To address this concern, the acceleration changes were also drawn from a truncated normal distribution. To encompass more of the possible acceleration values, this normal distribution was truncated to 4 standard deviations, so the accelerations were \( a \sim N(0, \sigma_a) \). As a result, the maximum acceleration is four sigma, \( i.e., A = 4\sigma_a \). The fact that \( \sigma_a \) is slightly smaller due to the truncated distribution was ignored. From (4.2), the multiplier would now be 0.9995, which justifies ignoring the use of the smaller standard deviation. Likewise, selection of the turn rate was also drawn from a normal distribution, truncated to four standard deviations. The six levels of maneuvering are summarized in Table 4-1. The table was created using the maneuver index equation in (1.12) and solving for the \( \sigma_a \). The maximum acceleration is then \( A = 4\sigma_a \). From (1.12), it should be clear that \( \sigma_a = \lambda \frac{\sigma_{\beta}}{T^2} \). Thus, as an example, when \( \lambda = 1 \), \( \sigma_a = \lambda \frac{\sigma_{\beta}}{T^2} = 1 \cdot \frac{100}{10^2} = 1 \) and the maximum acceleration is \( A = 4 \cdot 1 = 4 \).
Table 4-1: Six levels of maneuvering target tested.

<table>
<thead>
<tr>
<th>Man Index</th>
<th>Max Accel (m/s²)</th>
<th>Accel Std Dev (m/s²)</th>
<th>Max # Accel Std Devs</th>
<th>Max Turn Rate (deg/s)</th>
<th>Max # Turn Rate Std Devs</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ</td>
<td>Λ</td>
<td>σₐ</td>
<td>Θ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>1.00</td>
<td>0.25</td>
<td>4</td>
<td>1.00</td>
<td>4</td>
</tr>
<tr>
<td>0.50</td>
<td>2.00</td>
<td>0.50</td>
<td>4</td>
<td>2.00</td>
<td>4</td>
</tr>
<tr>
<td>0.75</td>
<td>3.00</td>
<td>0.75</td>
<td>4</td>
<td>3.00</td>
<td>4</td>
</tr>
<tr>
<td>1.00</td>
<td>4.00</td>
<td>1.00</td>
<td>4</td>
<td>4.00</td>
<td>4</td>
</tr>
<tr>
<td>2.00</td>
<td>8.00</td>
<td>2.00</td>
<td>4</td>
<td>8.00</td>
<td>4</td>
</tr>
<tr>
<td>3.00</td>
<td>12.00</td>
<td>3.00</td>
<td>4</td>
<td>12.00</td>
<td>4</td>
</tr>
</tbody>
</table>

A four-state vector was used for the Kalman filter of the form $[x, y, \dot{x}, \dot{y}]^T$. The standard deviations of the measurement errors were used to construct the appropriate measurement error matrix, $R$, for the Kalman filter, given in (4.3).

$$R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$  \hspace{1cm} (4.3)

The maximum acceleration (and turn rate) were used to construct the process noise matrix, $Q$, for the Kalman filter. The process noise matrix is given in (4.4).

$$Q = \begin{bmatrix} \frac{1}{4} \sigma_a^2 T^4 & 0 & \frac{1}{2} \sigma_a^2 T^3 & 0 \\ 0 & \frac{1}{4} \sigma_a^2 T^4 & 0 & \frac{1}{2} \sigma_a^2 T^3 \\ \frac{1}{2} \sigma_a^2 T^3 & 0 & \sigma_a^2 T^2 & 0 \\ 0 & \frac{1}{2} \sigma_a^2 T^3 & 0 & \sigma_a^2 T^2 \end{bmatrix}$$  \hspace{1cm} (4.4)

Since the measurements were in $xy$, the observation matrix, $H$ is straightforward. It is given in (4.5). The (linear) transition matrix is given in (4.6).
\[
H = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}
\]  \hspace{1cm} (4.5)

\[
\Phi = \begin{bmatrix}
1 & 0 & T & 0 \\
0 & 1 & 0 & T \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  \hspace{1cm} (4.6)

For the CGBF, grids of size 11 x 11 (i.e., so actually 11 x 11 x 5) were used. Each cell generated 3,000 Monte Carlo samples for a total of 11 x 11 x 3,000 = 363,000 samples per update. The maximum kinematic limits were provided to the CGBF. (In contrast, the Kalman filter does not use kinematic limits.)

Each scenario started with the target initially at the origin moving due west with a speed of 12 m/s. Each scenario lasted 190 s with a measurement update occurring every 10 s (i.e., \(T = 10\)). Therefore, there were 20 measurements per scenario, with the first measurement at time 0. Two types of targets were considered: a constant velocity target and a maneuvering target (limited by its assumed maximum kinematic limits). Section 4.1.1 will discuss the results using a constant velocity target. Section 4.1.2 will discuss the results using a maneuvering target.

### 4.1.1 Comparison Using a Constant Velocity Target

The first comparisons used a target moving with constant velocity over the duration of the scenario. The target moved along the (negative) \(x\)-axis. Since the target was moving with constant velocity, it remained 12 m/s for the entire scenario. Note that the maneuver index is unaffected by the fact that the target does not maneuver. The index is computed based on what the target can do, not what the target actually does. Of course, what the target actually does is unknown \textit{a priori} to the filter anyway.
All the assumptions to ensure optimality of the Kalman filter were basically upheld except the target motion uncertainty was not Gaussian [11]. As a result, the Kalman filter should perform reasonably well. Keep in mind that the prediction distribution is not normally distributed due to imposing realistic kinematic constraints on the target motion, as described in section 3.2. Thus, by allowing for more realistic target motion, one of the key assumptions of the Kalman filter has been violated, thereby forfeiting the optimality guarantee of the Kalman filter. So how much does the non-normal prediction distribution affect the results of the Kalman filter as compared to the CGBF against a constant velocity target? It might seem that this would have only a small effect, especially compared to a CGBF which needs to discretize a continuous solution space. Thus, one might suspect that the CGBF will perform no better, or maybe even worse than Kalman filter. The results will show otherwise.

To show how the CGBF works, an arbitrary sample run was conducted. The CGBF is executing the algorithm presented in Fig. 3-1. Figure 4-2 shows the first four updates of both filters for this arbitrary run. Each grid is centered on the measurement and scaled to include the maximum error ($3\sigma_R$). The green dotted axis lines at the center of the grid are centered on the measurement. The white dotted axis lines are centered on the true target location (which of course is not given to the filters). The black dashed axis lines are centered on the Kalman (position) estimate. The black dotted circles are the 1, 2, and 3 sigma ellipses from the Kalman filter. The magenta dashed axis lines are centered on the CGBF (position) estimate. The colored squares denote the probability distribution from the CGBF. The cell colors range from dark red for highest probability to deep blue for low probability. Dark blue denotes zero probability. Note that, as per the bounded measurement error assumption, the target is always located somewhere within the grid.

The grid in Fig. 4-2a shows the initialization from the first measurement. Like the Kalman filter, the CGBF uses the first measurement as its first position estimate. Therefore the position
and cell colors align with the position and sigma circles from the Kalman filter. Figure 4-3 shows the updates after measurement updates 5, 10, 15, and 20 for the same arbitrary run.

Figure 4-2: Constant velocity target; first four updates of the two filters.
Figure 4-3: Constant velocity target; the 5th, 10th, 15th, and 20th updates.

Figure 4-4 is the $xy$ comparison plot for the arbitrary run in Figs 4-2 and 4-3. The green line is the true target path. The blue line is the path from the Kalman filter and the red line is from the CGBF. For this run, the plots do not differ by much but it does show that the path from the CGBF is often closer to truth.

The differences in the errors are easier to see in Fig. 4-5, which plots the estimation errors over the duration of the scenario. The left-hand plot in Fig. 4-5 compares the position estimation
errors and the right-hand plot compares the velocity estimation errors. The dotted horizon lines are the mean errors over the entire scenario. The blue dotted line is the root mean squared error (RMSE) for the Kalman filter and the red dotted line is the RMSE for the CGBF. As the figure shows, the CGBF had less error than the Kalman filter for both position and especially velocity for this particular run. In fact, the velocity error from the CGBF was about 30% smaller than the error from the Kalman filter.

![XY Plot](image)

**Figure 4-4:** Position comparison for a constant velocity target.

![Plot](image)

**Figure 4-5:** Position and velocity error plots for constant velocity target.
An alternative way to view the position and velocity errors is via a state error plot as shown in Fig. 4-6. A state error plot looks similar to a position error plot but actually plots the error after combining both the position and velocity estimates. When plotting position estimation error, it is often plotted only at the update times, as was done in Fig. 4-5. This can yield deceiving performance since it is often necessary to know the position of the target at any arbitrary point in time, not just when a measurement update was received. Therefore, the state error plot computes the position estimation error at a higher rate than the update rate. The velocity estimate is used to determine the position of the target between updates. For example, in Fig. 4-6 the errors were calculated every second, while the measurement updates occurred every 10 s. Since the measurements arrive every 10 s, 9 out of every 10 position estimates are based on using the entire target state. The state errors for the Kalman filter are plotted in solid blue. The dashed red lines are the state errors from the CGBF. For this run, the estimation errors from the CGBF were generally smaller than those from Kalman. The two flat dotted lines are the average RMS errors, one for each filter.

![State Error Plot](image)

**Figure 4-6: State error plot for a constant velocity target.**

State error plots are useful, but for this comparison analysis, they fail to capture the breakout of the velocity error in terms of the contribution of error from speed and course. Recall that the
CGBF works with course and speed, rather than component velocity. Figure 4-7 plots the speed and course errors for the same arbitrary run. With these plots it is much easier to see the contributions the errors in the speed and course estimates bring to the overall velocity error. For example, notice how much spikier the course errors are for the Kalman filter than for the CGBF. To better illuminate the differences between the filters, the results will be summarized using position, velocity, speed, and course estimation error plots for the comparison.

The example run shown was for just one arbitrary run. Since filters are estimating a stochastic process, basing comparison performance on a single run is useless. To address this concern, a Monte Carlo analysis was performed using 200 sample runs for each of the six levels of maneuvering target. For each target, the average position, velocity, speed, and course RMS errors over the duration of the scenario were plotted. Figures 4-8 - 4-13 show the averaged results over these 200 sample runs for each of the six levels of maneuvering target. Tables 4-2 and 4-3 summarize the RMS errors over the entire scenario for the 200 Monte Carlo runs, i.e., the value of the horizontal lines in Figs. 4-8 - 4-13. The tables also show the percentage reduction in the estimation errors the CGBF provides over the Kalman filter. As can be seen from the plots and tables, when the target has low maneuverability, both filters do equally well. However, as the
maneuver index increases above 0.25, the improvement from the CGBF becomes more dramatic, especially for the course estimate. Note that once the maneuver index is around two or higher, the CGBF provides course estimates that have about 60% less error than the Kalman filter estimate.

Looking at the figures reveals more about the quality of the CGBF estimates. In general, the CGBF converges slightly slower than the Kalman filter, but converges to a lower error. Therefore, the CGBF performance can be characterized as “slower, but lower” than the Kalman filter. In addition, a couple other interesting trends emerge. The shape of the position and speed errors between the two filters looks the same except the CGBF is shifted lower than the Kalman filter. It is not clear why the shapes of these plots are so similar. The other interesting observation concerns the course estimates. The course errors for the CGBF remain smooth over time even as the target becomes more maneuverable while the Kalman filter course estimates become spikier. Thus, the CGBF must be achieving this improvement by taking advantage of the target's kinematic constraints.
Figure 4-8: Constant velocity target, maneuver index = 1/4.
Figure 4-9: Constant velocity target, maneuver index = 1/2.
Figure 4-10: Constant velocity target, maneuver index = 3/4.
Figure 4-11: Constant velocity target, maneuver index = 1.
Figure 4-12: Constant velocity target, maneuver index = 2.
Figure 4-13: Constant velocity target, maneuver index = 3.
The results of the Monte Carlo analysis showed that the CGBF consistently outperformed the Kalman filter against a constant velocity target. An important point to keep in mind is that this was a benign scenario where the Kalman filter assumptions held except that the process noise for target motion was not normally distributed. The CGBF was able to exploit that information and by doing so, it did significantly better. In the next section, the filters will be compared against a maneuvering target using the same six levels of maneuvering.

### Table 4-2: Averaged RMS position and velocity errors for constant velocity target (200 runs).

<table>
<thead>
<tr>
<th>Maneuver Index</th>
<th>Kalman Position Error (m)</th>
<th>CGBF Position Error (m)</th>
<th>Position Error Reduction</th>
<th>Kalman Velocity Error (m/s)</th>
<th>CGBF Velocity Error (m/s)</th>
<th>Velocity Error Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>100.8</td>
<td>102.1</td>
<td>-1.3%</td>
<td>4.3</td>
<td>4.8</td>
<td>-11.6%</td>
</tr>
<tr>
<td>0.50</td>
<td>108.2</td>
<td>104.0</td>
<td>3.9%</td>
<td>5.8</td>
<td>5.3</td>
<td>8.6%</td>
</tr>
<tr>
<td>0.75</td>
<td>114.2</td>
<td>106.0</td>
<td>7.2%</td>
<td>7.3</td>
<td>6.0</td>
<td>17.8%</td>
</tr>
<tr>
<td>1.00</td>
<td>118.0</td>
<td>108.0</td>
<td>8.5%</td>
<td>8.7</td>
<td>6.5</td>
<td>25.3%</td>
</tr>
<tr>
<td>2.00</td>
<td>125.0</td>
<td>111.2</td>
<td>11.0%</td>
<td>13.4</td>
<td>8.3</td>
<td>38.1%</td>
</tr>
<tr>
<td>3.00</td>
<td>129.7</td>
<td>116.1</td>
<td>10.5%</td>
<td>17.4</td>
<td>9.8</td>
<td>43.7%</td>
</tr>
</tbody>
</table>

### Table 4-3: Averaged RMS speed and course errors for constant velocity target (200 runs).

<table>
<thead>
<tr>
<th>Maneuver Index</th>
<th>Kalman Speed Error (m/s)</th>
<th>CGBF Speed Error (m/s)</th>
<th>Speed Error Reduction</th>
<th>Kalman Course Error (deg)</th>
<th>CGBF Course Error (deg)</th>
<th>Course Error Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>2.9</td>
<td>3.2</td>
<td>-10.3%</td>
<td>16.1</td>
<td>16.2</td>
<td>-0.6%</td>
</tr>
<tr>
<td>0.50</td>
<td>3.9</td>
<td>3.8</td>
<td>2.6%</td>
<td>22.2</td>
<td>17.0</td>
<td>23.4%</td>
</tr>
<tr>
<td>0.75</td>
<td>4.9</td>
<td>4.5</td>
<td>8.2%</td>
<td>28.5</td>
<td>18.1</td>
<td>36.5%</td>
</tr>
<tr>
<td>1.00</td>
<td>5.8</td>
<td>5.0</td>
<td>13.8%</td>
<td>35.1</td>
<td>18.2</td>
<td>48.1%</td>
</tr>
<tr>
<td>2.00</td>
<td>8.8</td>
<td>6.7</td>
<td>23.9%</td>
<td>55.5</td>
<td>21.4</td>
<td>61.4%</td>
</tr>
<tr>
<td>3.00</td>
<td>11.7</td>
<td>7.7</td>
<td>34.2%</td>
<td>63.6</td>
<td>26.2</td>
<td>58.8%</td>
</tr>
</tbody>
</table>
4.1.2 Comparison Using a Maneuvering Target

It was shown that a CGBF performs better than a Kalman filter against a constant velocity target when the maneuver index exceeds 0.25. The important question is whether the CGBF still performs better than a Kalman filter against a maneuvering target. For this analysis, the same six levels of target maneuvering were used, but now the targets continually maneuvers over the duration of the scenario. The maneuvering target motion model used was as described in section 3.2. To use that model, it must be decided at what times the target maneuvers. This was addressed by assuming the time between maneuvers was uniformly distributed between zero and a specified maximum time. The maximum time was set to 10 s. Keep in mind that although this maximum time is equal to the time between measurement updates, they are independent. Although possible, it is unlikely for the target to maneuver at the same time as an update. In fact since the maximum time between updates is 10 s, then, on the average, the target maneuvers about every 5 s.

At each maneuver time, an acceleration and turn rate within the target kinematic limits were each randomly selected from a normal distribution, truncated to four standard deviations. Maneuver duration was also randomly selected between 0 and 10 s, uniformly distributed. The target moves with the randomly selected acceleration and turn rate for the maneuver duration. At the end of the duration, the target maneuver process repeats. Using this process, it should be clear that the target is effectively moving via a (continuous-time) white noise acceleration model with maneuver times independent of the measurement updates.

Like the analysis for the constant velocity target, Monte Carlo analysis was performed using 200 sample runs for each of the six targets. For each target, the average position, velocity, speed, and course RMS errors over the duration of the scenario were plotted. Figures 4-14 - 4-19 show the averaged results over these 200 sample runs for each of the six levels of maneuvering. Tables 4-4 and 4-5 summarized the RMS errors over the entire scenario for the 200 Monte Carlo runs, i.e., the value of the horizontal lines in Figs. 4-14 - 4-19. The tables also provide the percentage
reduction in the estimation errors the CGBF provides over the Kalman filter. As can be seen from the plots and tables, when the target has low maneuverability, both filters do equally well. However, once again as the target has a maneuver index above 0.25, the improvement from the CGBF becomes more dramatic, especially for the course estimate. Note that once the maneuver index is around two or higher, the CGBF provides course estimates with about half the error of the Kalman filter estimate.

Figure 4-14: Maneuvering target, maneuver index = 1/4.
Figure 4-15: Maneuvering target, maneuver index = 1/2.
Figure 4-16: Maneuvering target, maneuver index = 3/4.
Figure 4-17: Maneuvering target, maneuver index = 1.
Figure 4-18: Maneuvering target, maneuver index = 2.
Figure 4-19: Maneuvering target, maneuver index = 3.
Table 4-4: Averaged RMS position and velocity errors for maneuvering target (200 runs).

<table>
<thead>
<tr>
<th>Maneuver Index</th>
<th>Kalman Position Error (m)</th>
<th>CGBF Position Error (m)</th>
<th>Position Error Reduction</th>
<th>Kalman Velocity Error (m/s)</th>
<th>CGBF Velocity Error (m/s)</th>
<th>Velocity Error Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>102.2</td>
<td>102.1</td>
<td>0%</td>
<td>5.0</td>
<td>5.4</td>
<td>-8.0%</td>
</tr>
<tr>
<td>0.50</td>
<td>109.0</td>
<td>105.3</td>
<td>3.4%</td>
<td>7.0</td>
<td>6.9</td>
<td>1.4%</td>
</tr>
<tr>
<td>0.75</td>
<td>116.5</td>
<td>110.2</td>
<td>5.4%</td>
<td>9.0</td>
<td>8.4</td>
<td>6.7%</td>
</tr>
<tr>
<td>1.00</td>
<td>119.7</td>
<td>112.5</td>
<td>6.0%</td>
<td>10.7</td>
<td>9.6</td>
<td>10.3%</td>
</tr>
<tr>
<td>2.00</td>
<td>126.8</td>
<td>118.4</td>
<td>6.6%</td>
<td>16.3</td>
<td>13.8</td>
<td>15.3%</td>
</tr>
<tr>
<td>3.00</td>
<td>131.0</td>
<td>121.6</td>
<td>7.2%</td>
<td>21.0</td>
<td>16.8</td>
<td>20.0%</td>
</tr>
</tbody>
</table>

Table 4-5: Averaged RMS speed and course errors for maneuvering target (200 runs).

<table>
<thead>
<tr>
<th>Maneuver Index</th>
<th>Kalman Speed Error (m/s)</th>
<th>CGBF Speed Error (m/s)</th>
<th>Speed Error Reduction</th>
<th>Kalman Course Error (deg)</th>
<th>CGBF Course Error (deg)</th>
<th>Course Error Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>3.9</td>
<td>4.1</td>
<td>-5.1%</td>
<td>24.0</td>
<td>18.7</td>
<td>22.1%</td>
</tr>
<tr>
<td>0.50</td>
<td>5.4</td>
<td>5.5</td>
<td>-1.8%</td>
<td>41.0</td>
<td>26.2</td>
<td>36.1%</td>
</tr>
<tr>
<td>0.75</td>
<td>6.9</td>
<td>6.8</td>
<td>1.4%</td>
<td>49.1</td>
<td>29.9</td>
<td>39.1%</td>
</tr>
<tr>
<td>1.00</td>
<td>8.2</td>
<td>7.8</td>
<td>4.9%</td>
<td>53.2</td>
<td>26.0</td>
<td>51.1%</td>
</tr>
<tr>
<td>2.00</td>
<td>12.3</td>
<td>10.8</td>
<td>12.2%</td>
<td>53.9</td>
<td>29.9</td>
<td>44.5%</td>
</tr>
<tr>
<td>3.00</td>
<td>15.5</td>
<td>12.6</td>
<td>18.7%</td>
<td>58.2</td>
<td>38.8</td>
<td>33.3%</td>
</tr>
</tbody>
</table>

4.2 The Kirubarajan & Bar-Shalom Comparison Scenario

Although it was shown that the CGBF outperforms the Kalman filter using the uniform behavior scenario, these scenarios were created somewhat arbitrarily. In order to gain more confidence in the true performance improvement, a more standardized and formalized method should be employed. Unfortunately, true standardized methods to compare track filters do not
exist. An alternative would be to base the comparison on an accepted or respected method used in
the literature. Several years ago, Kirubarajan and Bar-Shalom (K&B) conducted a performance
comparison of an interacting motion model filter (IMM) to a Kalman filter against a maneuvering
target. Their scenario was used to show the relative state estimation performance between an
IMM filter and a Kalman filter as the target becomes more maneuverable [55]. They defined the
target to have periods of maneuvering followed by periods of nearly constant velocity (NCV). As
mentioned earlier, a NCV target is one that moves with small perturbations to its velocity (via
small acceleration changes) to yield more realistic target motion. K&B assumed the acceleration
changes were normally distributed with zero mean and specified a standard deviation on these
acceleration changes. They defined a scenario lasting seven minutes (of simulated time) as shown
in Fig. 4-20. During the first minute, the target moves with nearly constant velocity by randomly
choosing accelerations in $x$ and $y$ from a normal distribution with standard deviation, $\sigma_{aL}$. This
selection was (artificially) synchronized with the measurement update time. K&B used the time
between measurement updates to be $T = 5$ s. Therefore, every 5 s, the target randomly chooses a
new acceleration. Then, during the next minute, the target maneuvers by randomly selecting
accelerations in $x$ and $y$ from a normal distribution with standard deviation, $\sigma_{aH}$. The target then
continues to alternate between moving with NCV and maneuvering for each subsequent minute.
The target changes between these two motions such that the first and last minute of the scenario
time have the target moving with NCV.

With this K&B comparison scenario, it is possible to compare the accuracy of the two filters
during the maneuvering and the non-maneuvering intervals, in addition to using the entire
scenario. This is exactly what Kirubarajan and Bar-Shalom did in their study. For example, to
compare the filters during the time the target was maneuvering, only the estimates during the
second, fourth, and sixth minutes are used. This same detailed analysis will be carried out for the
CGBF versus Kalman filter comparison.
4.3 Performance Comparison Using the K&B Scenario

The K&B comparison scenario was used for this CGBF/Kalman filter comparison. Monte Carlo analysis was used to compare the accuracy of state estimates between the CGBF and Kalman filter.

Besides the artificiality of having the target maneuvers perfectly synchronized with the measurement update times, there are other problems using this model directly for the CGBF/Kalman filter comparison. First, the CGBF assumes known limits on the target's speed, acceleration and turn rate. The K&B scenario assumes no limits on the target kinematics. Indeed, it does not even use turn rates, only accelerations. This problem is easy to fix using the mapping from turn rates to accelerations described in section 4.1. A maximum speed was also imposed on the target. As was done for the uniform behavior scenario, once the maximum speed was reached, the target remained at this maximum until a negative acceleration slowed it down. As before, the intent was to set the maximum speed high enough such that it would be possible but not often achieved. With this intent, the maximum speed was set to 180 m/s.

The second problem with the K&B scenario arises from selecting accelerations from a normal distribution. Since the accelerations are being selected from a normal distribution, there are no
real limits on the possible accelerations used or the resulting speeds the target could obtain. This lack of kinematic constraints on the target contradicts the basic assumptions of the CGBF (and of real targets). To address this issue, the acceleration must be drawn from a bounded distribution such as a truncated Gaussian distribution (as was done for the uniform behavior model) or a uniform distribution. The third problem is that the CGBF assumes known bounds on the measurement errors. Since the K&B scenario assumes Gaussian measurements (in $x$ and $y$), there are no limits. Like the accelerations, the measurement errors must be drawn from a bounded distribution as well.

Since the uniform behavior scenario used truncated Gaussian random variables, it was decided to use uniform random variables for the accelerations, turn rates, and measurement errors for the K&B scenarios. Given the standard deviation of a distribution, the range of values is easy to obtain. If the standard deviation is $\sigma$, the corresponding uniform random variable must be in the interval $[-\sigma\sqrt{3}, +\sigma\sqrt{3}]$.

A summary of the K&B comparison parameters is given in Table 4-5. The table also compares the parameters used for this study to those used in the original K&B study. The same standard deviations from the Kirubarajan and Bar-Shalom study were used, except now all random variables were uniformly distributed. For example, the K&B scenario used a standard deviation of $\sigma_R = 100$ m in each dimension for the measurement errors but now the errors were drawn from the corresponding uniform distribution.

Using the Kirubarajan and Bar-Shalom parameters, the initial position of the target was at $(25,000$ m, $10,000$ m) with an initial speed of $120$ m/s and an initial course of -90 deg (measured clockwise from North). The speed of the target they used is somewhat strange. Although the target is moving in 2D, its speed is much too fast to be a boat or car. Regardless, the same initial speed was used, but a maximum speed of $180$ m/s was imposed.
Table 4-6. Comparison of K&B study parameters.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Original K&amp;B Study</th>
<th>This Study</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensor</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measurement Space</td>
<td>(x,y)</td>
<td>same</td>
</tr>
<tr>
<td>Error Distribution</td>
<td>Normal(0, (\sigma_R))</td>
<td>Uniform ([-\sigma_R\sqrt{3}, +\sigma_R\sqrt{3}]))</td>
</tr>
<tr>
<td><strong>Update Time Interval (s)</strong></td>
<td>5</td>
<td>same</td>
</tr>
<tr>
<td><strong>Target</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Position (m, m)</td>
<td>(25,000, 10,000)</td>
<td>same</td>
</tr>
<tr>
<td>Initial Speed (m/s)</td>
<td>120</td>
<td>same</td>
</tr>
<tr>
<td>Initial Course (deg)</td>
<td>-90</td>
<td>same</td>
</tr>
<tr>
<td>NCV Acceleration Distribution</td>
<td>Normal(0, (\sigma_{al}))</td>
<td>Uniform ([-\sigma_{al}\sqrt{3}, +\sigma_{al}\sqrt{3}]))</td>
</tr>
<tr>
<td>Maneuver Acceleration Distribution</td>
<td>Normal(0, (\sigma_{ah}))</td>
<td>Uniform ([-\sigma_{ah}\sqrt{3}, +\sigma_{ah}\sqrt{3}]))</td>
</tr>
<tr>
<td>Max Acceleration (m/s²)</td>
<td>None</td>
<td>(\sigma_{ah}\sqrt{3})</td>
</tr>
<tr>
<td>Max Turn Rate (deg/s)</td>
<td>Not used</td>
<td>(\sigma_{ah})</td>
</tr>
<tr>
<td>Max Speed (m/s)</td>
<td>None</td>
<td>180</td>
</tr>
<tr>
<td>Time Between Maneuvers (s)</td>
<td>5, synchronized with sensor updates</td>
<td>same</td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scoring Time Interval</td>
<td>5 s, immediately after measurement update</td>
<td>same</td>
</tr>
<tr>
<td>Monte Carlo Runs per (\lambda)</td>
<td>100</td>
<td>1000</td>
</tr>
</tbody>
</table>

During the first, third, fifth, and seventh minute, the target moves with nearly constant velocity using process noise standard deviation \(\sigma_{al} = 0.2\) m/s² (which is the same process noise that Kirubarajan and Bar-Shalom used). However, as previously mentioned, the accelerations are assumed to be uniformly distributed instead of normally distributed. During the second, fourth, and sixth minute the target maneuvers with a process noise level that is varied as the independent variable. Kirubarajan and Bar-Shalom used \(\sigma_{ah} = 0.4\) m/s² to 10.0 m/s², in increments of 0.4 m/s², but for this study, the values were slightly modified to \(\sigma_a = 0.5\) m/s² to 12.0 m/s², in increments of 0.5 m/s². Although the process noise was varied, the average state estimation errors are plotted as a function of the maneuver index, which as can be seen from (1.12) is directly
related to the process noise. Kirubarajan and Bar-Shalom considered a maximum maneuver index of 2.5, but it was increased to 3.0 in this study. This was the reason behind the slight modification of the process noise values for this study. Again, these accelerations were assumed to now be uniformly distributed.

Since all measurement errors are now uniformly distributed, there are now two key assumptions of the Kalman filter violated: non-normally distributed prediction distribution (via imposing kinematic constraints on the target) and non-normally distributed measurement errors. As a result, the Kalman filter is not guaranteed to be optimal, but still the best linear filter. As a result, the comparison is more to show how much the Kalman filter degrades relative to the CGBF.

The results of the comparison will be summarized as a set of graphs plotting the average position and velocity estimation RMS errors from the Monte Carlo analysis. The speed and course estimation plots will not be presented since they were not done in the original study. The average RMS error, $E_{avg}$ is given in (4.7) where $m =$ number of Monte Carlo iterations, $n =$ number of measurement updates per scenario, and $e_{ij} = (\tilde{x}_{ij} - x_{ij})^2 + (\tilde{y}_{ij} - y_{ij})^2$ is the computed squared error of the $i^{th}$ measurement update on the $j^{th}$ Monte Carlo iteration.

$$E_{avg} = \frac{1}{m} \sum_{j=1}^{m} \left[ \frac{1}{n} \sum_{i=1}^{n} e_{ij}^2 \right]$$

The original study used 100 Monte Carlo iterations for each process noise standard deviation (i.e., maneuver index), but 1000 Monte Carlo iterations were used in this study to make the plots smoother.
4.3.1 The Kalman Filter Setup

The setup for the Kalman filter in this study was identical to the one in the K&B study. As mentioned, the measurement errors were now uniformly distributed (instead of Gaussian), but the standard deviations of these errors remained the same: $\sigma_x = \sigma_y = 100$ m for all runs. Like the K&B study, all measurement errors were assumed to be independent of target range. These standard deviations were used to construct the (constant) measurement error matrix, $R$ for the Kalman filter, given in (4.8).

$$R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$  (4.8)

The maximum accelerations and turn rates were used to construct the process noise matrix, $Q$ for the Kalman filter. A (standard) Kalman filter, however, does not use turn rates. Given $\sigma_a$, the corresponding maximum turn rate in deg/s is:

$$\theta_{\text{max}} = \frac{\sigma_a \sqrt{3}}{s} \left( \frac{180^\circ}{\pi} \right) \approx \frac{100 \sigma_a}{s}$$  (4.9)

where $s$ is the speed of the target. Therefore, if the average speed of the target is $\sim 100$ m/s, then the maximum turn rate of the target (in deg/s) is $\theta_{\text{max}} \approx \sigma_a$.

The process noise $Q$ matrix is given in (4.10), where $\sigma_a$ is set to 0.8 of the (true) process noise standard deviation used during the maneuvering intervals. The 0.8 adjustment factor was the value Kirubarajan and Bar-Shalom determined to be the best value for tuning the Kalman filter for these seven-minute scenarios.

$$Q = \begin{bmatrix} \frac{1}{4} \sigma_a^2 T^4 & 0 & \frac{1}{2} \sigma_a^2 T^3 & 0 \\ 0 & \frac{1}{4} \sigma_a^2 T^4 & 0 & \frac{1}{2} \sigma_a^2 T^3 \\ \frac{1}{2} \sigma_a^2 T^3 & 0 & \sigma_a^2 T^2 & 0 \\ 0 & \frac{1}{2} \sigma_a^2 T^3 & 0 & \sigma_a^2 T^2 \end{bmatrix}$$  (4.10)
4.3.2 The CGBF Setup

The CGBF that was used for the comparison employed a 5 x 5 grid with 1,000,000 samples per update. The CGBF was given the maximum speed of the target, which was set to 180 m/s. Since the measurement errors were uniform, the maximum measurement error was $E = \sigma_r \sqrt{3} = 100\sqrt{3}$, in each dimension.

4.3.3 Comparison Results

The results from the 1000 Monte Carlo runs are shown in Figures 4-21 – 4-24. Figures 4-21a and 4-21b plot the peak position estimation RMS errors and peak velocity estimation RMS errors respectively, over the seven-minute scenario as a function of the maneuver index. The red line is the RMS errors from the CGBF and the blue line is the RMS errors from the Kalman filter. As can be seen from Fig. 4-21a, the peak position errors for the CGBF are about the same as those for the Kalman filter. However, Fig. 4-21b shows that the peak velocity errors for the CGBF are slightly less than the Kalman filter, especially as the target becomes more maneuverable.

Figure 4-21: Peak RMS errors for CGBF and Kalman filter. (a) Position errors; (b) Velocity errors.

Figures 4-22a and 4-2b plot the average position estimation RMS errors and average velocity estimation RMS errors respectively, over the seven-minute scenario. Both the positional errors and the velocity errors from the CGBF are less than the Kalman filter. The GBF continually
reduces the errors over the Kalman filter as the target becomes more maneuverable. When the maneuver index is 3, the CGBF provides ~5% reduction in position error and ~13% reduction in velocity error.

![Figure 4-22: Overall RMS errors for CGBF and Kalman filter. (a) Position errors; (b) Velocity errors.](image)

Figures 4-23a and 4-23b plot the average position estimation RMS errors and the average velocity estimation RMS errors respectively, during the NCV (a.k.a., uniform motion) intervals of the scenario. Again, both the positional errors and the velocity errors from the CGBF are less than those from the Kalman filter. Like the overall average errors of the previous plots, the CGBF continually reduces the errors over the Kalman filter as the target becomes more maneuverable. When the maneuver index is 3, the CGBF again provides ~5% reduction in position error but now ~23% reduction in velocity error.

The average position estimation RMS errors and the average velocity estimation RMS errors respectively, during the maneuvering intervals of the scenario are plotted in Figures 4-24a and 4-24b. The CGBF outperforms the Kalman filter again, especially as the target becomes more maneuverable. When the maneuver index is 3, the CGBF provides ~5% reduction in both position and velocity error.
The results of this analysis show that the Kalman filter still performs well even when some of its key assumptions are violated, but the CGBF performs even better. Furthermore, the improvement that the CGBF provides increases as the target becomes more maneuverable.
Chapter 5 - Computational Considerations

The CGBF used in the K&B paradigm comparison study used only a 5 x 5 grid with 1,000,000 samples per update. This grid may seem rather small. It might seem that perhaps even better performance would have emerged if a larger grid (and/or more samples) were used. Fortunately, the CGBF is fairly insensitive to grid size and the number of samples used. Fig. 5-1 shows the average percentage reduction in position error, speed error, and course error, as a function of the grid size. The average was based on 200 Monte Carlo iterations for each data point in the plot. As can be seen from the figure, the reduction in errors from the CGBF are essentially flat, varying only ~2 - 3% between grid sizes as small as 3 x 3 to grids as large as 15 x 15. In addition, the figure shows the error reduction as the number of samples per update is increased from 100,000 to 1,000,000. Again, very little difference (~1% for position error) in the estimation performance of the CGBF results as the samples per update is increased. This is good news for the CGBF method. It means that although the CGBF is much more computational than a Kalman filter, it does not take large grids or large amounts of samples to obtain performance gains.

Although it does not take very large grids or very large sample sizes to produce quality estimates, the CGBF is still more computational than a Kalman filter. However, the bulk of the computations for the CGBF are from the Monte Carlo sampling. These computations are inherently independent which means the CGBF is quite amenable to parallel processing. With the large number of processors in today's CPUs and GPUs (graphics processing units), the time to compute the state estimates can be even further reduced. Thus, between the relatively small grids and sample sizes needed, along with the multi-processors of today, the processing time of a CGBF can be possibly reduced to processing times that are close to those needed for a Kalman filter.
Figure 5-1: Percentage reduction in (a) position error, (b) speed error, and (c) course error that CGBF offers over Kalman filter as grid size increases.
Chapter 6 - Future Work / Other Considerations

Although this research has shown that the CGBF outperforms the Kalman filter, there are a few areas where further studies could be focused. Presented here is a list of areas that would be interesting to further pursue.

Since the Kirubarajan and Bar-Shalom study showed that the IMM filter also outperforms the Kalman filter, it would be interesting to compare the CGBF to the IMM filter. This has been partially addressed in [76]. In that study, the Kalman filter versus the IMM study was repeated, but with using a Perfect IMM instead. The purpose of using a Perfect IMM was to form the lower bound on the errors from any IMM. Different results were obtained than those from Kirubarajan and Bar-Shalom. The results from this study suggest that the IMM can outperform the CGBF only during the non-maneuvering intervals since that is when the IMM exploits the low target maneuvering to provide better estimates. However, perhaps the equivalent of an IMM can be imposed on this CGBF. If so, then the CGBF should outperform an IMM.

The K&B analysis scored the state estimates immediately after the measurement update. This provides overly optimistic estimates of the real estimation performance. The total estimation error is really the “area under curve”, so the scoring should be done at a much higher rate than just at the measurement update rate. This was looked at briefly in the uniform behavior comparison with the use of the state error plots. It was decided that using speed and course error plots was more illuminating for this study than using state error. By doing so, it showed that the course errors were substantially reduced using the CGBF. This improvement becomes somewhat masked in the K&B study by only comparing velocity instead. It would be interesting to see how the filters compare in the K&B paradigm using the more inclusive comparison.

The scenarios used in this K&B comparison assumed an update time interval of 5 s. As can be seen from (1.12), the maneuver index increases as the square of the update time interval. Therefore, larger interval times lead to much harder tracking problems. As was pointed out, for
many real-world tracking systems, the times between measurement updates are much longer than 5 s. Based on the results from the uniform behavior comparison, it is suspected that the CGBF will outperform a Kalman (or even an IMM) filter even more as the time between updates increases. Namely, in addition to a larger maneuver index, as the time between updates increases, there is more reliance on the state estimate to do the predict. Since the CGBF produces better state estimates (especially in course), it should become more superior to a Kalman filter as the time interval increases. Of course the process noise and/or measurement errors would need to be adjusted accordingly to keep the maneuver index from getting too large. It was already shown that once the index gets to three, the Kalman filter is doing little more than simply using the last measurement (see section 1.4.1).

Another benefit of the CGBF is that the uncertainty distribution of the state estimate is directly captured into the grid structure. For this single target study, these distributions played only a minimal role. However, it would be interesting to compare the CGBF to the Kalman filter, or even to the IMM filter, in a multi-target environment. It is suspected that the CGBF would lead to significant overall multi-target tracking improvement by exploiting its improved uncertainty information.

Finally, the research focused on targets moving in 2D. It is suspected that this method could be extended to targets moving in 3D as well. In this case, three-dimensional grids would be used and each cell would need to maintain the distribution parameters for 3D motion.
Chapter 7 - Conclusions

This research focused on improving the filter method required for tracking maneuvering targets with particular emphasis on maritime targets. Currently, all current filter methods fail to cover the required operating range needed for tracking all maritime targets. Although grid-based filters are mostly ignored, they offer a good match for tracking maritime targets except they fail to remain computationally practical when addressing a 4D filter problem which is required for tracking maritime targets. To address this shortfall, a novel implementation of a grid-based filter was developed and presented, called a constrained grid-based filter (CGBF). It was shown that by employing realistic assumptions and a series of computational shortcuts, key surveillance processing solution constraints can be relaxed. This provides system engineers with a deeper solution trade space and thus, more flexibility in determining surveillance system requirements allocation. Specifically, for targets that are more maneuverable, yet still under conditions consistent with the operational constraints of maritime surveillance, the CGBF offers a promising new computational approach that improves tracking performance over the standard Kalman filter-based approaches. This new filter not only simplifies the allocation problem but allows for a greater range of potential operational considerations, target types, or the servicing of higher fidelity tracking requirements.

This CGBF works efficiently for the 4D tracking problem and offers several implementation improvements over the current GBFs. The following are the key improvements that this research introduced:

- Using small specialized 2D grids instead of large 4D grids

Typical implementations of a GBF place each grid over some large defined region. By doing so, it requires massive amounts of storage and exponential processing. Placing the grid over a region also causes the boundary-edge problem. In the CGBF, each grid is placed over the extent of the measurement instead, which eliminates the boundary-edge problem. By
placing the grids over the measurement, much smaller grids are needed. In fact, the research shows that quality state estimates emerge from as few as three cells in each dimension.

- **Folding velocity into the 2D grid instead of using higher dimension grids**

  The CGBF uses $n \times n \times 5$ grids instead of a 4D grids. The speed and course are folded into this 2D grid by dealing with their distributions parametrically. In addition to storing the probability in each cell, the CGBF also stores the mean speed and course and their variances.

- **Using speed and course instead of velocity components**

  Like the Kalman filter, grid-based filters tend to work with the rectilinear velocity components. Instead, the CGBF works with speed and course. This allows the CGBF to directly exploit the kinematic constraints of the target since the constraints are typically specified in terms of maximum speeds and turn rates.

- **Using grid cell-sized particles instead of point-sized particles**

  During each Monte Carlo sample, each “particle” is sized to the size of the grid cell to ensure uniform sampling over the entire cell without the need to increase the number of Monte Carlo samples. The mass in the particle is apportioned out to all the destination cells it overlaps.

- **Using “mirroring” to generate twice as many samples**

  For each Monte Carlo sample, two samples are generated using the mirrored state of the one computed. Thus, $2m$ samples are generated for the same computational effort of computing $m$ of them.

  A trade study was conducted to compare this novel CGBF against a Kalman filter. The results showed that the CGBF consistently outperformed the Kalman filter, especially as the target became more maneuverable. Two different methods were used to empirically compare a CGBF to
a Kalman filter: using a uniform behavior scenario and using the Kirubarajan & Bar-Shalom scenario. The CGBF consistently outperformed the Kalman filter for both approaches. For example, when using the uniform behavior scenario, the CGBF reduced the position and speed estimate errors by 12% and the course errors by approximately half.

The motion predict of the CGBF was studied analytically by determining the mean and variance of the state distribution. It was shown that the CGBF has a more focused distribution oriented along the target’s presumed initial course and speed. Furthermore, all movements were confined to the target’s presumed kinematic constraints, i.e., all motion is kinematically constrained. This imposed bounds on the distance the target could travel during a time update interval. A Kalman filter on the other hand, by virtue of assuming the distribution is Gaussian, imposes no bounds.

The research also showed that the CGBF provides highly accurate state estimates even when using very small grids and small sample sizes. This makes the CGBF very attractive from a computational standpoint as well. The results from this study should help re-vitalize interest in the grid-based filters since the CGBF is much less computational than the current grid-based approaches and the CGBF consistently produces better state estimates than a Kalman filter. Since the CGBF was shown to provide better state estimates than a Kalman filter, tracking systems using this filter should provide better tracking and surveillance of maneuvering targets. Thus, track filtering problems such as those that arise from tracking all types of maritime targets can be greatly improved using this CGBF. A possible extension of this research to consider and compare the CGBF to an IMM filter may further extend awareness of track filter solution capabilities to make the systems engineering of maritime surveillance and tracking systems more systematic and repeatable.
References


[25] Sora Choi, Peter Willett, Fred Daum, and Jim Huang, "Discussion and application of the


[39] Fun-Bin Duh and Chin-Teng Lin, "Tracking a maneuvering target using neural fuzzy


Appendix A

Mex code (i.e., C code) to generate predicted state and update.

/*
   * FastMCnext.c
   *
   * This is the update part of the Monte Carlo algorithm. This version uses
   * the mean and std deviation of the course and speed to form a uniform
   * distribution which is then sampled from.
   *
   * Program written by Mark Silbert, Oct 2010
   *
   * Revisions
   * 03 Nov 10 - Modified to use uniform distribution, instead of Gaussian
   * 18 Feb 11 - Instead of randomly sampling with the cell position, move
   *   each fraction of the entire cell probability to each
   *   transitioned cell.
   * 19 Feb 11 - Fixes/cleanup of prior changes.
   * 20 Feb 11 - Minor fix; only would be an error for range/az msmt cases.
   * 14 May 11 - Incrementally build up the number of sigmas.
   *
   * This is a MEX-file for MATLAB.
   */

#include "mex.h"
#include "math.h"

#define NSIGS 3

double *mass1M, rot1, xcell1_m, ycell1_m;
double *mass2M, rot2;
double x1L_m, y1L_m;
double x2L_m, x2H_m, y2L_m, y2H_m;

double *spd1M, *spd1devM, *crs1M, *crs1devM;

mwSize nrows, ncols, ncells;
int nSamples;

#include "FastTrueMoveCODE.c"
#include "FastRandomWalkCODE.c"

#ifndef max
#define max( a, b ) ( ((a) >= (b)) ? (a) : (b) )
#endif

#ifndef min
#define min( a, b ) ( ((a) <= (b)) ? (a) : (b) )
#endif

/*
   * FastMCnext - returns the next Monte Carlo update for a 2D target.
   *
   * Function Usage:
   *  [mass2M spd2M spd2devM crs2M crs2devM] = ...
   *  FastMCnext(mass1M, nSamples, duration_s, dtMan_s
   *  rot1, x1L_m, y1L_m, xcell1_m, ycell1_m, ...
void FastMCnext(double duration_s, double dtMan_s,
                double spdMax_mps, double spddotMax_mpss, double crsdotMax_rps)
{
    mwSize i,j, k = -1;
    double xcell1_m, ycell1_m = xy size of each cell (meters);
    double xcell2_m = (x2H_m - x2L_m)/ncols;
    double ycell2_m = (y2H_m - y2L_m)/nrows;
    double sinDrot = sin(rot2 - rot1);
    double cosDrot = cos(rot2 - rot1);
    double sinRot2 = sin(rot2);
    double cosRot2 = cos(rot2);
    double x1cellL_m,x1cellH_m, y1cellL_m,y1cellH_m;
    double spd1_mps, crs1;
    double spd1L_mps,spd1H_mps, dspd1_mps, crs1L,crs1H, dcrs1;
    double prob1, probcell2, prob2;
    double x1LLr2_m, y1LLr2_m, x1LHr2_m, y1LHr2_m;
    double x1HLr2_m, y1HLr2_m, x1HHr2_m, y1HHr2_m;
    double x1Lr2_m, y1Lr2_m, x1Hr2_m, y1Hr2_m;
    double x2Lr2_m, y2Lr2_m, x2Hr2_m, y2Hr2_m;
    double dx_m[2], dy_m[2], spd2_mps[2], crs2[2];
    double dxr2_m, dyr2_m;
    double totalMass = 0.0;
    double crs2offset, s, c;
    double xjL,xjH, yiL,yiH;
    double pxj,pyi;
    int kSamples = (int) (nSamples/NSIGS);
    double nSamples2 = 2.0*NSIGS*kSamples;
    mwSize iyL,iyH, jxL,jxH, iy, jx, kxy;
    int nsig;
int m, p;

for(i=0; i<ncells; i++) {
    mass2M[i] = 0.0;
    spd2M[i] = 0.0;
    crs2M[i] = 0.0;
    spd2devM[i] = 0.0;
    crs2devM[i] = 0.0;
}
crs2MM = (double *) malloc(ncells*sizeof(double));

x1cellH_m = x1L_m;
for(j=0; j<ncols; j++) {
    x1cellL_m = x1cellH_m;
    x1cellH_m = x1cellH_m + xcell1_m;
    y1cellH_m = y1L_m;
    for(i=0; i<nrows; i++) {
        y1cellL_m = y1cellH_m;
        y1cellH_m = y1cellH_m + ycell1_m;
        k++;
        prob1 = *(mass1M+k);
        if(prob1 == 0.0) continue;
        x1LLr2_m = x1cellL_m*cosDrot - y1cellL_m*sinDrot;
        y1LLr2_m = x1cellL_m*sinDrot + y1cellL_m*cosDrot;
        x1LRr2_m = x1cellL_m*cosDrot - (y1cellL_m + ycell1_m)*sinDrot;
        y1LRr2_m = x1cellL_m*sinDrot + (y1cellL_m + ycell1_m)*cosDrot;
        x1HRr2_m = (x1cellL_m + xcell1_m)*cosDrot - y1cellL_m*sinDrot;
        y1HRr2_m = (x1cellL_m + xcell1_m)*sinDrot + y1cellL_m*cosDrot;
        x1HRr2_m = (x1cellL_m + xcell1_m)*cosDrot
                   - (y1cellL_m + ycell1_m)*sinDrot;
        y1HRr2_m = (x1cellL_m + xcell1_m)*sinDrot
                   + (y1cellL_m + ycell1_m)*cosDrot;
        x1Lr2_m = min(x1LLr2_m, x1LRr2_m);
        x1Hr2_m = max(x1HRr2_m, x1HRr2_m);
        y1Lr2_m = min(y1LRr2_m, y1HRr2_m);
        y1Hr2_m = max(y1HRr2_m, y1HRr2_m);
        probcell2 = prob1*xcell2_m*ycell2_m
                   /(nSamples2*(x1Hr2_m-x1Lr2_m)*(y1Hr2_m-y1Lr2_m));
    }
    for(nsig=1; nsig<=NSIGS; nsig++) {
        dpsd1_mps = spd1devM[k]*nsig;
        spd1L_mps = spd1M[k] - dpsd1_mps;
        spd1H_mps = spd1M[k] + dpsd1_mps;
        if(spd1L_mps < 0.0) spd1L_mps = 0.0;
        if(spd1H_mps > spdMax_mps) spd1H_mps = spdMax_mps;
        dcrs1 = crs1devM[k]*nsig;
        if(dcrs1 < M_PI) {
            crs1L = crs1M[k] - dcrs1;
            crs1H = crs1M[k] + dcrs1;
        } else {
            crs1L = -M_PI;
            crs1H = +M_PI;
        }
        spd1L_mps = spd1H_mps - spd1L_mps;
        dcrs1 = crs1H - crs1L;
        for(m=0; m<kSamples; m++) {
            spd1_mps = spd1L_mps + dpsd1_mps*rand()/(RAND_MAX + 1.0);
crs1 = AngleFix(crs1L + dcrs1*rand()/(RAND_MAX + 1.0));
FastRandomWalk(spd1_mps, crs1, duration_s, dtMan_s,
spdMax_mps, spddotMax_mps, crsdotMax_rps,
dx_m, dy_m, spd2_mps, crs2);
for(p=0; p<2; p++) {
    dxr2_m = dx_m[p]*cosRot2 - dy_m[p]*sinRot2;
    dyr2_m = dx_m[p]*sinRot2 + dy_m[p]*cosRot2;
    x2Lr2_m = x1Lr2_m + dxr2_m;
    x2Hr2_m = x1Hr2_m + dxr2_m;
    if((x2Hr2_m < x2L_m) || (x2Lr2_m >= x2H_m)) continue;
    y2Lr2_m = y1Lr2_m + dyr2_m;
    y2Hr2_m = y1Hr2_m + dyr2_m;
    if((y2Hr2_m < y2L_m) || (y2Lr2_m >= y2H_m)) continue;
    if(x2Lr2_m > x2L_m)
        xjL = (x2Lr2_m - x2L_m)/xcell2_m;
    else xjL = 0.0;
    xjH = (x2Hr2_m - x2L_m)/xcell2_m;
    if(xjH > ncols) xjH = (double) ncols;
    if(y2Lr2_m > y2L_m)
        yiL = (y2Lr2_m - y2L_m)/ycell2_m;
    else yiL = 0.0;
    yiH = (y2Hr2_m - y2L_m)/ycell2_m;
    if(yiH > nrows) yiH = (double) nrows;
    jxL = (mwSize) xjL;
    jxH = (mwSize) xjH;
    if(jxH >= ncols) jxH = ncols - 1;
    iyL = (mwSize) yiL;
    iyH = (mwSize) yiH;
    if(iyH >= nrows) iyH = nrows - 1;
    for(jx=jxL; jx<=jxH; jx++) {
        pxj = min(pxj, jx+1.0) - max(pxj, jx);
        for(iy=iyL; iy<=iyH; iy++) {
            pyi = min(pyi, iy+1.0) - max(pyi, iy);
            kxy = nrows*jx + iy;
            if(*((mass2M+kxy) > 0.0))
                //compute crs offset from the "0" crs
                crs2offset = AngleFix(crs2[kxy]-crs2ZM[kxy]);
            else {
                //need to set the "0" crs at initial crs
                crs2ZM[kxy] = crs2[kxy];
                crs2offset = 0.0;
            }
            prob2 = probcell2*pxj*pyi;
            *(mass2M+kxy) += prob2;
            s = prob2*spd2_mps[p];
            c = prob2*crs2offset;
            *(spd2M+kxy) += s;
            *(crs2M+kxy) += c;
            *(spd2devM+kxy) += s*spd2_mps[p];
            *(crs2devM+kxy) += c*crs2offset;
        }
    }
}
for(k=0; k<ncells; k++)
if(*(mass2M+k) > 0.0) {
    totalMass += mass2M[k];
    spd2M[k] /= mass2M[k];
    crs2M[k] /= mass2M[k];
    s = spd2devM[k]/mass2M[k] - spd2M[k]*spd2M[k];
    if(s < 1.0) spd2devM[k] = 1.0;
    else spd2devM[k] = sqrt(s);
    c = crs2devM[k]/mass2M[k] - crs2M[k]*crs2M[k];
    if(c < 0.0004) crs2devM[k] = 0.02; //about 1 deg
    else crs2devM[k] = sqrt(c);
    crs2M[k] = AngleFix(crs2M[k] + crs2ZM[k]);
}
free(crs2ZM);

if(totalMass > 0.0) {
    prob2 = 1.0/totalMass;
    for(k=0; k<ncells; k++)
        *(mass2M+k) *= prob2;
} else *mass2M = -1.0; //set first cell negative to indicate failure
}
Appendix B

MATLAB code to generate the mean and covariance of the predicted state for double maneuvers.

%******************************************************************************
%* This file estimates the predicted state and covariance for a target
%* executing up to 2 maneuvers within a time interval.
%* Written by Mark Silbert, Dec 2011
%******************************************************************************
initialize = true; %flag to re-use the same random variables

small = 1.0e-6; %smallest value for turn rate. Any turn rate smaller
%than this value just moves target along constant course

n = 50000000; %number of Monte Carlo samples
useSymmetry = true %doubles the number of samples by using the symmetric state
%NOTE: crsInit must be 0 to use symmetry

maxSigmas = 4 %use >0 for truncated normal random variable
% =0 for uniform random vars

maxturnrate_dps = 15 %max turn rate of target (degs/sec)
timespan = 10; %duration of the time interval (secs)
maxacc = 2; %max acceleration of the target (m/s^2)

spdInit = 20; %assumed initial speed of the target (m/s)
crsInit = 0; %assumed initial course of the target (radians)

maxturnrate = maxturnrate_dps*pi/180;

if initialize
    Tau = timespan*rand(1,n); %pick the duration of the 1st maneuver
    if maxSigmas > 0 %use truncated normal
        spdVar = (maxacc/maxSigmas)^2;
        Rs = TruncatedNormal(2,n, maxSigmas)/maxSigmas;
        Rc = TruncatedNormal(2,n, maxSigmas)/maxSigmas;
    else %assume accelerations are uniformly distributed
        spdVar = maxacc^2/3;
        Rs = 2*rand(2,n) - 1;
        Rc = 2*rand(2,n) - 1;
    end
    SpdDot = maxacc*Rs; %pick accels
    CrsDot = maxturnrate*Rc; %pick turn rates
end

Spd = zeros(2,n);
Crs = zeros(2,n);
Spd(1,:) = spdInit + SpdDot(1,:).*Tau;
Crs(1,:) = crsInit + CrsDot(1,:).*Tau;
Spd(2,:) = Spd(1,:) + SpdDot(2,:).*(timespan - Tau);
Crs(2,:) = Crs(1,:) + CrsDot(2,:).*(timespan - Tau);
Xdot2 = Spd(2,:).*sin(Crs(2,:));  %vel x component at end of time interval
Ydot2 = Spd(2,:).*cos(Crs(2,:));  %vel y component at end of time interval

sinCrsInit = sin(crsInit);
cosCrsInit = cos(crsInit);
SinCrs = sin(Crs);
CosCrs = cos(Crs);

X2 = zeros(1,n);     %x position at end of time interval
Y2 = zeros(1,n);     %y position at end of time interval
for i=1:n
    spd0 = spdInit;
    sinCrs0 = sinCrsInit;
    cosCrs0 = cosCrsInit;
    dt = Tau(i);
    for k=1:2
        if abs(CrsDot(k,i)) > small %target is turning
            X2(i) = X2(i) + (SpdDot(k,i)*SinCrs(k,i) -
                Spd(k,i)*CrsDot(k,i)*CosCrs(k,i) -
                SpdDot(k,i)*sinCrs0 + spd0*cosCrs0*CrsDot(k,i))/(CrsDot(k,i)^2);
            Y2(i) = Y2(i) + (SpdDot(k,i)*CosCrs(k,i) +
                Spd(k,i)*CrsDot(k,i)*SinCrs(k,i) -
                SpdDot(k,i)*cosCrs0 - spd0*sinCrs0*CrsDot(k,i))/(CrsDot(k,i)^2);
        else %target moving along a constant course
            X2(i) = X2(i) + 0.5*(spd0 + Spd(k,i))*SinCrs(k,i)*dt;
            Y2(i) = Y2(i) + 0.5*(spd0 + Spd(k,i))*CosCrs(k,i)*dt;
        end
        spd0 = Spd(1,i);
        sinCrs0 = SinCrs(1,i);
        cosCrs0 = CosCrs(1,i);
        dt = timespan - Tau(i);
    end
end

if useSymmetry
    X2 = [X2 -X2];
    Y2 = [Y2 Y2];
    Spd2 = [Spd(2,:) Spd(2,:)];
    Crs2 = [Crs(2,:) -Crs(2,:)];
    Xdot2 = [Xdot2 -Xdot2];
    Ydot2 = [Ydot2 Ydot2];
else
    Spd2 = Spd(2,:);
    Crs2 = Crs(2,:);
end

XXdot2 = X2.*Xdot2;
YYdot2 = Y2.*Ydot2;

xdotkalmean = spdInit*sinCrsInit;
ydotkalmean = spdInit*cosCrsInit;
xxkalmean = xdotkalmean*timespan;
yykalmean = ydotkalmean*timespan;

xdotkalvar = spdVar*(timespan^2);
ydotkalvar = xdotkalvar;
xxkalvar = spdVar*(timespan^4)/4;
ykalvar = xxkalvar;
\[ \text{xxdotkalcov} = \text{spdVar} \times (\text{timespan}^3)/2; \]
\[ \text{yydotkalcov} = \text{xxdotkalcov}; \]
\[ \text{spd2mean} = \text{mean(Spd2);} \]
\[ \text{spd2var} = \text{var(Spd2);} \]
\[ \text{crs2mean} = \text{mean(Crs2);} \]
\[ \text{crs2var} = \text{var(Crs2);} \]
\[ \text{x2mean} = \text{mean(X2);} \]
\[ \text{y2mean} = \text{mean(Y2);} \]
\[ \text{x2var} = \text{var(X2);} \]
\[ \text{y2var} = \text{var(Y2);} \]
\[ \text{xdot2mean} = \text{mean(Xdot2);} \]
\[ \text{ydot2mean} = \text{mean(Ydot2);} \]
\[ \text{xdot2var} = \text{var(Xdot2);} \]
\[ \text{ydot2var} = \text{var(Ydot2);} \]

\[ \text{Spd2_mean_var} = [\text{spd2mean} \text{ spd2var}] \]
\[ \text{Crs2_mean_var} = [\text{crs2mean} \text{ crs2var}] \]
\[ \%\text{Xdot2_mean_var} = [\text{mean(Xdot2)} \text{ var(Xdot2)} \text{ spd0x}] \]
\[ \%\text{Ydot2_mean_var} = [\text{mean(Ydot2)} \text{ var(Ydot2)} \text{ spd0y}] \]
\[ \%\text{X2_mean_var} = [\text{mean(X2)} \text{ var(X2)} \text{ spd0x*timespan}] \]
\[ \%\text{Y2_mean_var} = [\text{mean(Y2)} \text{ var(Y2)} \text{ spd0y*timespan}] \]
\[ \%\text{XXdot2_mean_var} = [\text{mean(XXdot2)} \text{ var(XXdot2)} 0 0] \]
\[ \%\text{YYdot2_mean_var} = [\text{mean(YYdot2)} \text{ var(YYdot2)} 0 0] \]

\[ \%\text{MEAN} \quad X \quad Y \quad Xdot \quad Ydot \quad XXdot \quad YYdot \]
\[ \%\text{Kalman} 123456.890 123456.890 123456.890 123456.890 123456.890 \]
\[ \%\text{CGBF} \quad 123456.890 \quad 123456.890 \quad 123456.890 \quad 123456.890 \quad 123456.890 \]
\[ \%\text{Variance} \quad XX \quad YY \quad XdotXdot \quad YdotYdot \]
\[ \%\text{Kalman} 123456.890 123456.890 123456.890 123456.890 \]
\[ \%\text{CGBF} \quad 123456.890 \quad 123456.890 \quad 123456.890 \quad 123456.890 \quad 123456.890 \]
\[ \%\text{Covariance} \quad XXdot \quad YYdot \quad XY \quad XYdot \quad XdotY \]
\[ \%\text{Kalman} 123456.890 123456.890 123456.890 123456.890 \]
\[ \%\text{CGBF} \quad 123456.890 \quad 123456.890 \quad 123456.890 \quad 123456.890 \quad 123456.890 \]

\[ \text{beep}; \text{pause}(0.1); \text{beep} \%\text{let user know we're finished} \]