NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

BALLISTIC MISSILE TRACKING USING THE INTERACTING MULTIPLE MODEL JOINT PROBABILISTIC DATA ASSOCIATION FILTER

by

Timothy M. Dunton

September 2013

Thesis Advisor:	Robert G. Hutchins
Second Reader:	Xiaoping Yun

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2013	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE BALLISTIC MISSILE TRACKING USING THE INTERACTING MULITPLE MODEL JOINT PROBABILISTIC DATA ASSOCIATION FILTER	**5. FUNDING NUMBERS**
6. AUTHOR(S) Timothy M. Dunton	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. government. IRB protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The success of interceptors used by the United States ballistic missile defense program is jeopardized by the use of hostile missile decoy deployment and evasive maneuvers. The ability to discriminate between legitimate threats and decoys is a crucial requirement for interceptor algorithms. The feasibility of the interacting multiple model joint probabilistic data association filter to effectively track a ballistic missile and detect decoys and maneuvers is the focus of this thesis. Model development and data association schemes are discussed along with optimized values for selected parameters.

Performance comparisons of the resultant algorithm to a standard Kalman filter utilizing a nearest neighbor discriminator are conducted. Scenarios include combinations of missile maneuver and decoy deployment. While the Kalman filter experiences limited success, the proposed filter tracks the missile in every scenario.

14. SUBJECT TERMS Interacting multiple model, joint probabilistic data association filter, ballistic missile, decoy	**15. NUMBER OF PAGES** 110
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

BALLISTIC MISSILE TRACKING USING THE INTERACTING MULITPLE MODEL JOINT PROBABILISTIC DATA ASSOCIATION FILTER

Timothy M. Dunton
Lieutenant, United States Navy
B.S., United States Naval Academy, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2013

Author: Timothy M. Dunton

Approved by: Robert G. Hutchins
Thesis Advisor

Xiaoping Yun
Second Reader

R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The success of interceptors used by the United States ballistic missile defense program is jeopardized by the use of hostile missile decoy deployment and evasive maneuvers. The ability to discriminate between legitimate threats and decoys is a crucial requirement for interceptor algorithms. The feasibility of the interacting multiple model joint probabilistic data association filter to effectively track a ballistic missile and detect decoys and maneuvers is the focus of this thesis. Model development and data association schemes are discussed along with optimized values for selected parameters.

Performance comparisons of the resultant algorithm to a standard Kalman filter utilizing a nearest neighbor discriminator are conducted. Scenarios include combinations of missile maneuver and decoy deployment. While the Kalman filter experiences limited success, the proposed filter tracks the missile in every scenario.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
- A. BACKGROUND....1
- B. OBJECTIVE....2
- C. PREVIOUS WORK....3
- D. THESIS ORGANIZATION....4

II. FILTERING METHODOLOGY....5
- A. KALMAN FILTER....5
- B. INTERACTING MULTIPLE MODEL....6
  - 1. Purpose....6
  - 2. Model Probabilities....7
  - 3. IMM Mixing....9
- C. JPDAF....11
- D. JPDAF AND IMM COMBINATION....14
- E. SPLIT TARGET TRACK GENERATION....15
- F. TARGET DELETION....17

III. MODEL CONSTRUCTION....19
- A. OVERVIEW....19
- B. MASTER SCRIPT FILE....19
  - 1. True Missile Motion....19
  - 2. True Split Target Motion....21
  - 3. Sensor Noise....21
  - 4. Plant Noise....22
  - 5. Model Transitions....23
- C. MODEL FUNCTIONS AND INTERACTIONS....24
  - 1. Straight-Line Motion....24
  - 2. Constant Speed....25
  - 3. Potential Split Target....25
  - 4. Split Target Tracking....26

IV. MATLAB SIMULATIONS....27
- A. SIMULATION SETUP....27
- B. INDEPENDENT EVALUATIONS....27
  - 1. IMM Missile Tracking....27
  - 2. Split Target Detection and Tracking....37
- C. COMPARATIVE ANALYSIS WITH A STANDARD KALMAN FILTER....45
  - 1. Kalman Filter Setup....45
  - 2. Scenario Setup....47
  - 3. Results....52

V. CONCLUSIONS....55
- A. SUMMARY OF RESULTS....55

1. IMM 55
2. JPDA 56
B. RECOMMENDATIONS FOR FUTURE WORK....................56
1. Probability of Detection and False Returns....................56
2. Model Modifications....................56
3. Algorithm Flexibility 57

APPENDIX. MATLAB CODE 59
A. MASTER SCRIPT FILE INCLUDING ALL INITIAZATIONS, ORDER OF FUNCTION FILE CALLS, AND GENERATION OF ALL PLOTS 59
B. MODEL ONE FUNCTION FILE....................73
C. MODEL TWO FUNCTION FILE....................73
D. MODEL THREE FUNCTION FILE....................74
E. MODEL FOUR FUNCTION FILE WITH COMPLETE JDPA FILTER ANALYSIS OF ALL AVAILABLE MEASUREMENTS..........75
F. MODEL PROBABILITY FUNCTION FILE WITH IMM MIXING 82
G. PREDICTION FUNCTION FILE 83
H. STANDARD KALMAN FILTER FUNCTION FILE WITH A NEAREST NEIGHBOR ALGORITHM FOR MEASUREMENT ASSOCIATION....................83

LIST OF REFERENCES....................85

INITIAL DISTRIBUTION LIST....................87

LIST OF FIGURES

Figure 1. A combination of sensors and platforms in a multi-layered missile defense strategy contribute to successful interception of a ballistic missile in each phase of travel. From [1]........2

Figure 2. An IMM flowchart depicts the major steps, inputs, and outputs of the IMM operation. From [4]........7

Figure 3. A typical JPDAF scenario shows the need for conflict resolution. P1 and P2 are the predicted track positions. O1, O2, and O3 are the three observations that fall within the track gates. From [4]........12

Figure 4. True target motion on an x-y plane identifying the location of areas of focus for follow-on figures........28

Figure 5. True target track, IMM estimate tracks, and measurements during straight-line motion on a Cartesian plane........29

Figure 6. True target track, IMM estimate tracks, and measurements during a constant speed maneuver in the Cartesian plane........30

Figure 7. True target track, IMM estimate tracks, and measurements during constant speed turn in a Cartesian plane........31

Figure 8. SLM mean distance errors of IMM estimates with mean measurement error as a reference........32

Figure 9. Mean distance error of IMM estimates and mean measurement error during a constant speed maneuver initiated at sample time $k = 2000$........33

Figure 10. Constant acceleration mean distance error of IMM estimates with mean measurement error as a reference........34

Figure 11. Model probability averaged over 10 samples during IMM mixing with the following transitional probabilities: $p_{11} = 0.8, p_{12} = 0.2, p_{21} = 0.45, p_{22} = 0.55$........35

Figure 12. Constant speed turn mean distance error of IMM estimates with mean measurement error as a reference........36

Figure 13. Model probability averaged over 10 samples during IMM mixing with the following transitional probabilities: $p_{11} = 0.8, p_{12} = 0.2, p_{21} = 0.6, p_{22} = 0.4$...36

Figure 14. True target motion focus area for follow-on figures in analysis of target split simulations........37

Figure 15. Split target estimates converging on split target from SLM........38

Figure 16. Model three and model four transitions during split target detection from SLM........39

Figure 17. Split target estimates converging on split target from constant acceleration...40

Figure 18. Model three and model four transitions during split target detection from constant acceleration........40

Figure 19. SLM and constant acceleration mean distance errors during target splitting track formation........41

Figure 20. SLM and constant acceleration mean distance errors observed during split target track maintenance........42

Figure 21. Combined estimate mean distance error versus measurement error seen during target splitting. 43
Figure 22. Target split during SLM with a raised number of model three samples ($t = 15$). 44
Figure 23. Target split during constant acceleration with a raised number of model three samples ($t = 15$). 44
Figure 24. Model three mean distance error versus a greater number of samples for each target split. 45
Figure 25. Comparison of Kalman and IMMJPDA filter mean distance errors during steady state SLM. 46
Figure 26. Comparison of Kalman and IMMJPDA filter mean distance error peaks during transition from SLM to constant acceleration. 46
Figure 27. Comparison of Kalman and IMMJPDA filter mean distance errors during a constant acceleration turn. 47
Figure 28. Scenario one for comparison between the Kalman and IMMJPDA filters. 48
Figure 29. Scenario two for comparison between the Kalman and IMMJPDA filters. 49
Figure 30. Scenario three for comparison between the Kalman and IMMJPDA filters. ..50
Figure 31. Scenario four for comparison between the Kalman and IMMJPDA filters.....51
Figure 32. Scenario five for comparison between the Kalman and IMMJPDA filters. 52

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. A JPDAF hypothesis matrix is constructed for the scenario presented in Figure 3 with a probability of detection equal to one. From [4]. 13

Table 2. Performance summary of the Kalman filter and the IMMJPDAF under simulated ICBM decoy deployment and maneuver scenarios. Refer to section two (Scenario Setup) for an explanation of table organization. 54

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ICBM	Intercontinental Ballistic Missile
IMM	Interacting Multiple Model
IMMJPDAF	Interacting Multiple Model Joint Probabilistic Data Association Filter
JPDA	Joint Probabilistic Data Association
JPDAF	Joint Probabilistic Data Association Filter
SLM	Straight-line Motion

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The success of interceptors used by the United States ballistic missile defense program is jeopardized by the use of hostile missile decoy deployment and evasive maneuvers. The ability to discriminate between legitimate threats and decoys is a crucial requirement for interceptor algorithms. The feasibility of the interacting multiple model joint probabilistic data association filter (IMMJPDAF) to effectively track an intercontinental ballistic missile (ICBM) and detect decoys and maneuvers is the focus of this thesis. Model development and data association schemes are discussed along with optimized values for selected parameters.

Two different models are used to describe the missile dynamics. One model describes straight-line motion (SLM), and the second model describes a constant velocity turn. The interacting multiple model (IMM) algorithm mixes the estimates from each of the models. The benefit of mixing the estimates is accurate prediction of missile position regardless of maneuvering frequency. Once a split target (e.g., decoy) is detected, the common history between the missile and decoy is exploited to generate a separate track for the decoy.

The joint probabilistic data association filter (JPDAF) matches positions from multiple targets received by the sensor to the predicted location of each existing target. All possible combinations of received positions (observations) and predicted target locations are assigned a value based on probability of occurrence. A quantitative analysis of the probability values allows the filter to distribute observations to the associated target. The successful assignments from the JPDAF allow the IMM to better predict the next target position and increase the tracking performance of the integrated IMMJPDAF.

With regard to ICBM tracking, the IMM algorithm proved to be capable during both steady-state and maneuvering conditions. For the given scenarios, the estimates resulting from the IMM mixing process received a reduction of measurement error of 50 percent on average. The degradation of the individual model estimates was minimal. The number of transitions between the models, especially during a constant velocity maneuver, was frequent and often occurred between each sample. For targets with the likelihood of transitioning between established models often, this method is appropriate. However, for the purposes of an ICBM that prefers to deploy decoys to conducting maneuvers, the necessity to account for continuous contact maneuver is reduced. In addition, the distance errors seen during model transitions are concerning due to the likelihood of decoy deployment during those periods. Distance errors at model transitions had negligible effect on the performance of the IMMJPDA filter during the scenarios tested but could become a concern under higher split target densities.

With regard to split target tracking, the IMM provided a convenient framework in which to generate a split target and allowed for a seamless transition for follow-on track maintenance. We introduced a non-traditional IMM approach, where the two models regarding the split target are not subject the standard IMM processes. Initially, stacking the combined estimates to generate the split track allowed the common history of the target and decoy to be exploited. The decoy track generation caused an increase in distance error of the combined estimates of over two and a half times measurement error. The decoupling of model three and model four with the target models prevented further degradation of the combined estimates. Additionally, the linearly increasing model three errors showed that earlier decoupling of the split target and true target was best.

Performance comparisons of the resultant algorithm to a standard Kalman filter utilizing a nearest neighbor discriminator are conducted. Scenarios included combinations of missile maneuver and decoy deployment. While the Kalman filter experiences limited success, the IMMJPDAF tracks the missile in every scenario.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I want to thank my thesis advisor, Professor Hutchins, and my second reader, Professor Yun, for their direction and patience.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Ballistic missile defense is not a new concern for the United States, but until recently the number of countries with credible threats has been limited. The availability of ballistic missile technology to unstable nations poses a legitimate threat to the security of the United States. The challenge of hitting high-speed missiles in an exo-atmospheric environment has been overcome with capable interceptors. The use of decoys released from the missile body has the potential to dramatically reduce the effectiveness of the current interceptor inventory. Solutions include development of a new platform or an upgrade to existing interceptor sensors and tracking algorithms. The more fiscally prudent solution is the latter.

In response to growing threats, the Ballistic Missile Defense Agency has adopted a multi-layered defense strategy utilizing the full spectrum of sensors and interceptors for boost, mid-course, and terminal phase interception [1]. The collection of sensors and platforms is illustrated in Figure 1. A clear distinction between the phases of ballistic missile travel helps to divide the unique challenges posed in each. The boost phase is from launch to the end of powered flight and lasts between 60 and 300 seconds [2]. Interception in this phase is preferred but challenging based on the short window of opportunity and political issues surrounding interceptor basing. Mid-course describes the trajectory from booster burnout to return to the earth's atmosphere [2]. Recognizing the vulnerability while in the midcourse phase, threat missiles deploy decoys in this phase only for protection during the 100 second to 10 minute travel time. The terminal phase is a last resort for interception and is defined where gravitational effects are noticed, typically between altitudes of 70 and 100 km [2].

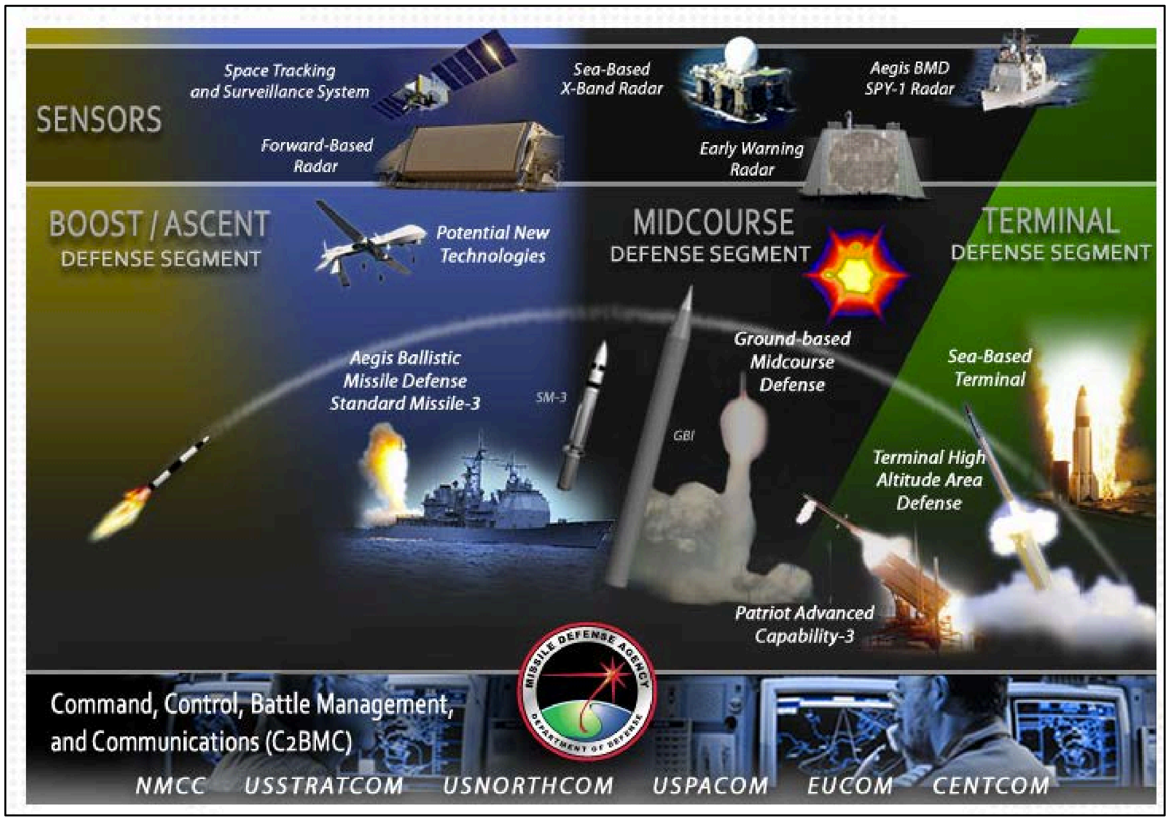
SENSORS
Space Tracking and Surveillance System
Forward-Based Radar
Sea-Based X-Band Radar
Aegis BMD SPY-1 Radar
Early Warning Radar
BOOST / ASCENT DEFENSE SEGMENT
Potential New Technologies
MIDCOURSE DEFENSE SEGMENT
TERMINAL DEFENSE SEGMENT
Aegis Ballistic Missile Defense Standard Missile-3
SM-3
GBI
Ground-based Midcourse Defense
Sea-Based Terminal
Terminal High Altitude Area Defense
Patriot Advanced Capability-3
Command, Control, Battle Management, and Communications (C2BMC)
MISSILE DEFENSE AGENCY
DEPARTMENT OF DEFENSE
NMCC
USSTRATCOM
USNORTHCOM
USPACOM
EUCOM
CENTCOM

Figure 1. A combination of sensors and platforms in a multi-layered missile defense strategy contribute to successful interception of a ballistic missile in each phase of travel. From [1].

The National Research Council submitted a report in [2] describing the current challenges of Ballistic Missile Defense. Midcourse intercept was identified as critical to the success of ballistic missile interception, even in the terminal phase [2]. If destruction of reentry vehicles that split off from the missile cannot be completed in the mid-course phase, identification and tracking must be accomplished for terminal phase interception. Mid-course discrimination is accomplished by the Ground-Based Interceptor system utilizing secure satellite communications for initial threat missile parameter evaluation and information dissemination [1].

B. OBJECTIVE

Advanced missile systems are capable of deploying decoys and maneuvering to thwart midcourse interception. Algorithms that can track through maneuvers and deal with a single target splitting into multiple independent targets when decoys are deployed are required to deal with these advanced threats. The feasibility of using the interacting multiple model (IMM) [3] and the joint probabilistic data association (JPDA) [4] tracking filters in a ballistic missile interception scenario is the focus of this research. More specifically, the ability to accurately predict the missile state during maneuvering and target split operations will be analyzed.

C. PREVIOUS WORK

Ballistic missile tracking algorithms are relied upon to efficiently process the information provided by the vast network of sensors seen in Figure 1. Effective interception requires accurate threat missile position and speed information. Application of the probabilistic data association filter (PDAF) in over-the-horizon radar systems, as discussed in [5], provides threat missile track generation from a cluttered environment. To increase confidence in target track data, the integration of additional target information such as mass, heat, and signal strength has been extensively researched. For example, Colegrove and Davey use signal-to-noise ratio in [6] to quickly formulate target tracks.

The IMM best fits applications involving fast moving and frequently maneuvering targets (e.g., air traffic control systems, GPS navigation, and radar tracking systems). Ballistic missile tracking using the IMM has been accomplished with a variety of approaches with a distinctive feature being the number of phases included. For example, Cooperman presents a model for all three phases of ballistic travel in [7] using multiple sensors. Farrell uses a similar approach in [8] by applying standard Kalman filters in each of his separate models (constant axial force, ballistic acceleration, and autocorrected acceleration). Both approaches achieved position error results that are best suited for a single phase of missile travel, while the other phases suffer from large inaccuracies. A majority of approaches, such as the one found in [9], use multiple models to describe a single phase, and most often the focus is the boost or terminal phase. Our research will focus on only mid-course missile travel, requiring missile models with reliable split target detection.

Methods for detection of split targets are less researched due to limited applications. A method for split target tracking in clutter using an IMM is presented in [10], but the assumptions used in this approach limit the number of target splits (such as weapons shot from fighter aircraft). A generalized method for integrating the interacting multiple model joint probabilistic data association filter (IMMJPDAF) with target splits is presented in [3] and is the basis for the approach used in this research. To the best of the author's knowledge, the application and parameter optimization of the IMMJPDAF for tracking a maneuvering ballistic missile during decoy deployment has not been researched.

D. THESIS ORGANIZATION

The standard Kalman filter, IMM, and JPDA equations are presented and adapted for efficiency in Chapter II, along with methods for split track generation and deletion. The generation and transition between models are discussed in Chapter III and are shaped by an estimation of the most likely scenarios. Critical initiation values for each model to include sensor and plant noise characteristics are also identified. Performance metrics and an analysis of MATLAB simulation results are given in Chapter IV. The simulations presented include a stand-alone evaluation of the IMM during target maneuver, split target detection from various target states, and a comparative analysis of the combined IMM and JPDAF to the standard Kalman filter. A summary of results and recommendations for future work are the focus of Chapter V.

II. FILTERING METHODOLOGY

A. KALMAN FILTER

A brief overview of the Kalman filter equations is given here and taken directly from [11]. The target state at time k is $x(k)$, representing the true target location. The transition of the true state of the system is described by

$$x(k+1) = F(k)x(k) + v(k), \tag{1}$$

where $F(k)$ is the state transition matrix and $v(k)$is the plant noise. The deterministic term $u(k)$, which accounts for any known inputs to the system dynamics, and the associated input gain matrix $G(k)$ are normally included in Equation (1) but are not required for our analysis. Measurements are received at each time k and are related to the true target state as follows:

$$z(k+1) = H(k+1)x(k+1) + w(k+1), \tag{2}$$

where $w(k+1)$ represents zero-mean, white measurement noise. The state estimate at time $k+1$ is

$$\hat{x}(k+1\,|\,k) = F(k)\hat{x}(k\,|\,k), \tag{3}$$

where $\hat{x}(k\,|\,k)$ is the previous state estimate. Once a measurement at $k+1$ is received, the predicted measurement is compared to the received measurement as follows:

$$\tilde{z}(k+1) = z(k+1) - \hat{z}(k+1\,|\,k), \tag{4}$$

where the predicted measurement is

$$\hat{z}(k+1\,|\,k) = H(k+1)\hat{x}(k+1\,|\,k). \tag{5}$$

The state estimate is then updated by incorporating $\tilde{z}(k+1)$ in the following manner:

$$\hat{x}(k+1\,|\,k+1) = \hat{x}(k+1\,|\,k) + W(k+1)\tilde{z}(k+1), \tag{6}$$

where $W(k+1)$ is the Kalman filter gain. The Kalman gain requires computations involving the state covariance $P(k|k)$ and the state transition matrices. The state prediction covariance is

$$P(k+1|k) = F(k)P(k|k)F(k)' + Q(k), \tag{7}$$

where $Q(k)$ is the covariance of the zero-mean white process noise. Measurement noise covariance is taken into consideration as follows:

$$S(k+1) = H(k+1)P(k+1|k)H(k+1)' + R(k+1), \tag{8}$$

where $R(k+1)$ is the zero-mean, white measurement noise covariance. The Kalman filter gain calculation is

$$W(k+1) = P(k+1|k)H(k+1)'S(k+1)^{-1}. \tag{9}$$

The state covariance is updated using $W(k+1)$ as follows:

$$P(k+1|k+1) = P(k+1|k) - W(k+1)S(k+1)W(k+1)'. \tag{10}$$

Equation (3) through Equation (10) are performed on each cycle.

B. INTERACTING MULTIPLE MODEL

1. Purpose

Kalman filter tracking errors are known to increase as a target continues to deviate from the trajectory described by the state transition matrix shown in Equation (1). Thus, a single Kalman filter is insufficient for tracking a maneuvering target or multiple targets regardless of their trajectories. The IMM approach circumvents these problems by creating multiple state models. A model is defined for each target state (e.g., straight-line motion, constant turn rate) to best describe the observed motion in the form of multiple transition matrices. Multiple Kalman filters running in parallel reduces tracking errors, assuming an adequate algorithm exists for filter transitions. “The unique feature of the IMM approach is the manner in which the state estimates and the covariance matrices from these multiple models are combined according to a Markov model for the transition between target maneuver states" [4].

The IMM flowchart in Figure 2 below appears in [4] and provides a convenient method of summarizing IMM operation. Once a measurement (observation) is received, the JPDAF is assigned the task of deciding if and to which established track it belongs, satisfying the "Gating and Data Association" block. Explanation of IMM operation begins with appropriately assigned observed data and follows the flowchart in Figure 2.

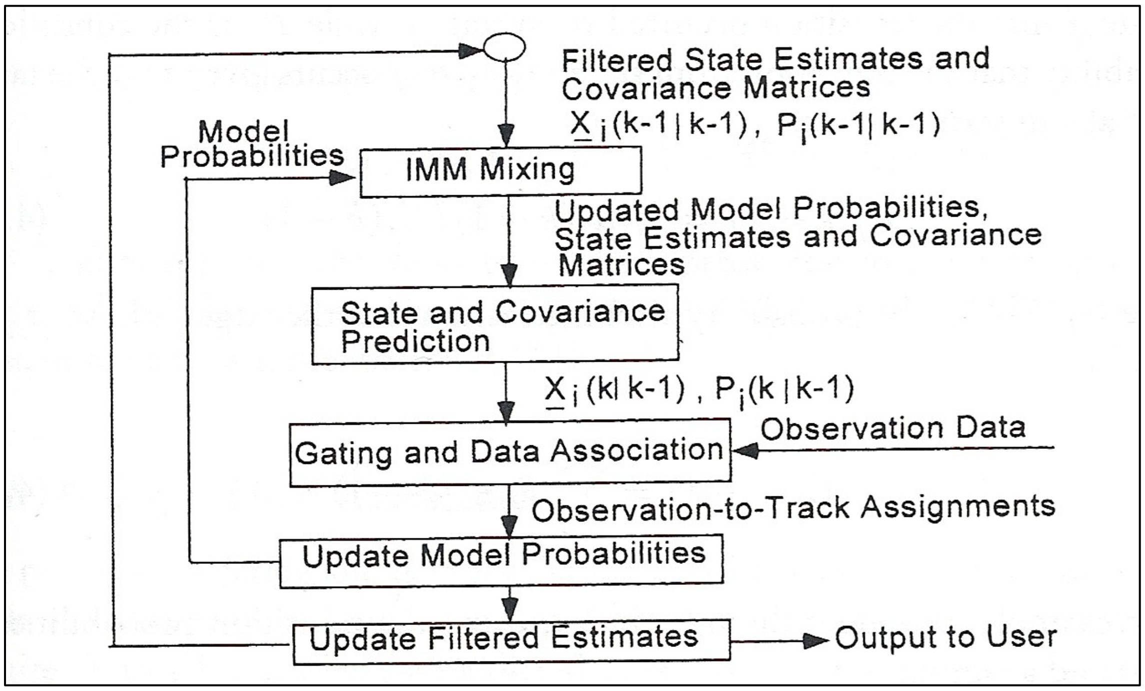
Filtered State Estimates and Covariance Matrices
$\underline{X}_i(k-1|k-1)$, $P_i(k-1|k-1)$
Model Probabilities
IMM Mixing
Updated Model Probabilities, State Estimates and Covariance Matrices
State and Covariance Prediction
$\underline{X}_i(k|k-1)$, $P_i(k|k-1)$
Gating and Data Association
Observation Data
Observation-to-Track Assignments
Update Model Probabilities
Update Filtered Estimates
Output to User

Figure 2. An IMM flowchart depicts the major steps, inputs, and outputs of the IMM operation. From [4].

2. Model Probabilities

The number of possible target states determines the number of models r required for the IMM. A sufficient collection of models is required to cover the spectrum of anticipated target motion. The transition between states is described by both conditional probabilities and transitional probabilities. Conditional probabilities u_i for $i = 1,2,...,r$ are initially assigned a priori values, which are updated at each step k. Update is based on a comparison of the received measurement $z(k)$ to each of the respective model state estimates $\hat{x}_i(k\,|\,k-1)$. The probability that the target is in model state i as computed just after measurement data are received at time $k-1$ is defined as $u_i(k-1)$ [4]. The normalized statistical distance is calculated for use in update of u_i and is

$$d_i^2 = \tilde{z}(k)' S_i^{-1}(k)\tilde{z}(k), \tag{11}$$

where the residual covariance matrix $S_i(k) = HP(k\,|\,k-1)H' + R$ is received from Equation (8). The model probability update utilizes a likelihood function based on Gaussian statistics defined as follows:

$$\lambda_i(k) = \frac{e^{-d_i^2/2}}{\sqrt{(2\pi)^M\,|S_i(k)|}}, \tag{12}$$

where M is the dimension of $S_i(k)$. The likelihood function is used to update the model probabilities by

$$\mu_i(k) = \frac{\lambda_i(k)C_i(k)}{C}, \tag{13}$$

where C is a normalizing constant from [4] and is described by

$$C = \sum_{i=1}^{r} \lambda_i(k)C_i(k-1). \tag{14}$$

The variable $C_i(k-1)$ in Equation (14) is a linear combination of both conditional and transitional probabilities and updated during the IMM mixing process. The updated model probabilities from Equation (13) assign a weighted contribution of the updated states estimates from the Kalman filter models. Combined estimates are generated that represent the most up to date estimates based on the current measurements. The combined estimates and associated covariance matrices are calculated as follows:

$$x_c(k\,|\,k-1) = \sum_{i=1}^{r} u_i(k-1)\hat{x}_i(k) \tag{15}$$

and

$$P_c(k) = \sum_{i=1}^{r} u_i (P_i(k\,|\,k-1) + [\hat{x}_i(k\,|\,k-1) - x_c(k-1)][\hat{x}_i(k\,|\,k-1) - x_c(k-1)]'), \qquad (16)$$

where $x_c(k-1)$ is the combined estimate from the previous time step, $\hat{x}_i(k\,|\,k-1)$ is the estimate from the associated model, and $P_i(k\,|\,k-1)$ is the covariance matrix produced from the standard Kalman filter equations in each respective model [11]. At this point, the model probabilities and filtered estimates have been updated based on observation data, and all required data for IMM mixing is available.

3. IMM Mixing

The IMM mixing satisfies several objectives. First, “the IMM approach provides the most effective framework for adaptive filtering.”[4] Each set of observation data allows discrimination or favoring of the model that it is most closely associated with. Additionally, “the mixing process uses the accurate filter state estimates to correct the less accurate filter state estimates” [4]. The process analyzes each possible target transition and adjusts the state estimates based on the most likely target state. Mixing occurs after update of the state estimate and covariance matrices but before prediction.

A Markov transition matrix is utilized to describe the probability p_{ij} that “the target will make the transition from model state i to model state j” [4]. For a target characterized by three states, the transition matrix is

$$M = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}. \qquad (17)$$

The probabilities contained within M are developed a priori and remain constant. Each row must sum to unity to prevent degradation of the estimates contributed by the associated model.

While the Markov matrix describes the estimated transitions between models, a parameter describing actual recent model transitions is required to update model probabilities. “The conditional probability that the target made the transition from state i to state j at time $k-1$ is $u_{ij}(k-1)$” [4]. Conditional probability differs from p_{ij} in that the transition from state i to state j occurs given that the target is initially in state i. The relationship between the conditional and transitional probability is

$$\mu_{ij}(k-1)=\frac{p_{ij}\mu_i(k-1)}{C_i(k-1)}, \tag{18}$$

where $C_j(k-1)$ is the probability after interaction that the target is in state j and can be defined as

$$C_i(k-1)=\sum_{i=1}^{r}p_{ij}\mu_i(k-1). \tag{19}$$

Once transitional probabilities are calculated for each state, a mixing process is executed to produce new filtered state estimates and Kalman filter covariance matrices. For all models $j=1,2,...,r$ the state estimates and covariance matrices are calculated using intermediate values described by

$$\hat{x}_j^o(k-1\,|\,k-1)=\sum_{i=1}^{n}\mu_{ij}(k-1)\hat{x}_i(k-1\,|\,k-1) \tag{20}$$

and

$$P_j^o(k-1\,|\,k-1)=\sum_{i=1}^{r}\mu_{ij}(k-1)P_i(k-1\,|\,k-1). \tag{21}$$

The IMM mixing is complete.

The prediction step used in the Kalman filter is then applied to produce a predicted state estimate and a Kalman filter covariance matrix as follows:

$$\hat{x}_i(k+1\,|\,k)=F_i\hat{x}_i^o \tag{22}$$

and

$$P_i(k+1\,|\,k)=F_iP_i^oF_i'+Q_i. \tag{23}$$

The “State and Covariance Prediction” block of the IMM flowchart in Figure 2 is satisfied by Equation (22) and Equation (23). Upon receipt of new observation data, the IMM process is repeated.

C. JPDAF

Ideally, no ambiguity exists in regards to which track is associated with a given observation; however, such an assumption is an oversimplification in tracking scenarios. Proper assignment of measurement data to the correct target is vital to maximizing the performance of both the Kalman filter and the IMM. To address issues of track assignment, the probabilistic data association (PDA) method associates measurement data with established tracks and removes extraneous measurements. The JPDAF is an extension of the PDA to a multiple target scenario and serves to resolve any existing correlation conflict. JPDAF integration with the IMM is described in Section D.

We will use an example posed in [4] and illustrated in Figure 3 below to describe the operation of the JPDAF. The scenario assumes two tracks have been established. The variables P1 and P2 in Figure 3 represent predicted states from the two established tracks. Both predictions are located at the center of the respective circular gates and are established along each target’s associated track. A circular gate is used in this example for ease of computation. Gate size is dependent on noise characteristics and potential target maneuver parameters defined within the target model. Three observations are received simultaneously and are denoted by O1, O2, and O3. All three observations are considered for validation since each is located within one or both track gate(s). The probability that a true target observation will fall within the correct track gate P_G is assumed to be one.

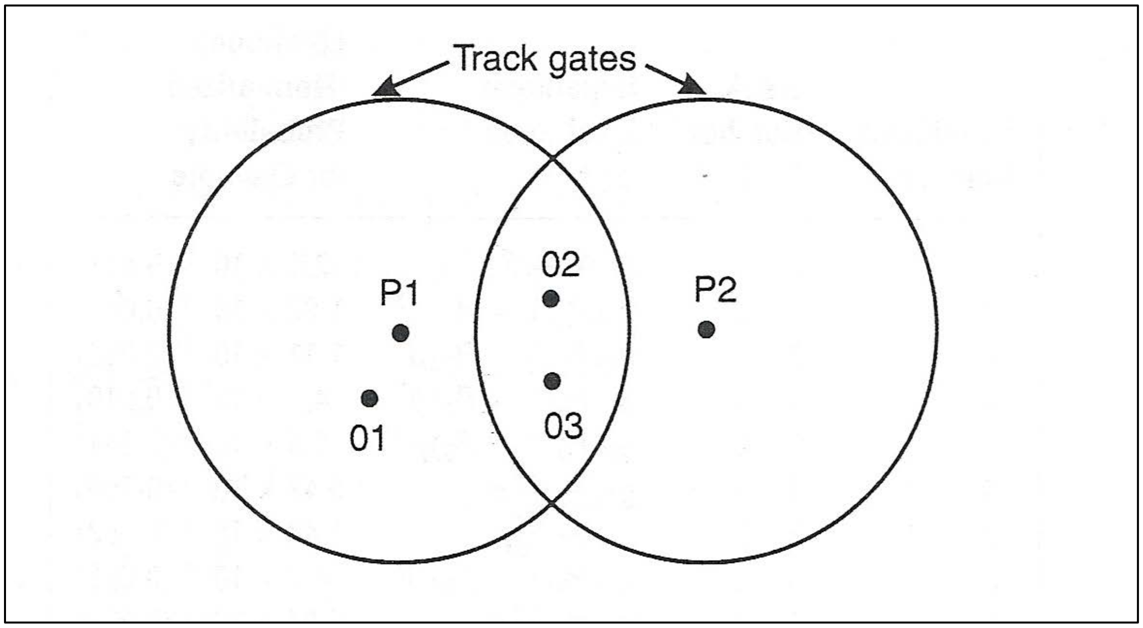

Figure 3. A typical JPDAF scenario shows the need for conflict resolution. P1 and P2 are the predicted track positions. O1, O2, and O3 are the three observations that fall within the track gates. From [4].

A hypothesis matrix is constructed to track each possible observation and track combination, including the possibility that none of the observations belong to either track. A probability is calculated for each hypothesis based on a Gaussian likelihood function. Each track i is related to each observation j by

$$g_{ij} = \frac{e^{-d_{ij}/2}}{(2\pi)^{M/2}\sqrt{|S_i|}} \tag{24}$$

where M is the measurement dimension [11]. To fully describe the probability of the hypothesis, the probability of detection P_D and a false return density function β are included. The full development of the probability equations are given in [4], but the equations used for the scenarios dealt with in this research are contained in Table 1.

Table 1. A JPDAF hypothesis matrix is constructed for the scenario presented in Figure 3 with a probability of detection equal to one. From [4].

Hypothesis Number	Track Number		Hypothesis Likelihood
	P1	P2	$h'(H_l)$
1	O1	O2	$g_{11}g_{22}P_D^2\beta$
2	O3	O2	$g_{13}g_{22}P_D^2\beta$
3	O1	O3	$g_{11}g_{23}P_D^2\beta$
4	O2	O3	$g_{12}g_{23}P_D^2\beta$

The hypotheses in Table 1 are constructed by listing all possible outcomes under each track. The scenario in Figure 3 shows that all three observations (O1, O2, and O3) may be associated with track one (P1), and only observations two and three may be associated with track two (P2). Observation one does not appear in the track two column, but all other combinations of tracks and observations are represented. For example, hypothesis one states that observation one (O1) is assigned to P1 and that observation two (O2) is assigned to track two (P2). Observation one lies outside the gate of P2 and is, therefore, not assigned to P2. We will assume that the probability of detection P_D is one, meaning that each track always receives a measurement. This is a significant assumption and greatly reduces the number of hypotheses that must be considered. For space-based sensor tracking of an exo-atmospheric ballistic missile, the assumed P_D is not unrealistic.

The hypotheses likelihoods are computed but require normalizing. Normalized probabilities are calculated by

$$h(H_l) = \frac{h'(H_l)}{\sum_{l=1}^{m} h'(H_l)}, \tag{25}$$

where m is the total number of hypotheses. The probability that O1 is associated with P1 is calculated using Table 1 as follows:

$$h_{11} = h(H_1) + h(H_3), \tag{26}$$

where h_{11} is the probability that O1 belongs to P1. The remaining probabilities for track one $(h_{12}$ and $h_{13})$ are computed in the same manner and must all sum to one. Track two probabilities are computed the same way. Notice that h_{21} is zero because O1 does not fall within the gate of P2.

The information obtained from each track's association with each observation is incorporated into the standard Kalman filter equations. To account for multiple observations in Equation (4), a weighted sum of residuals is taken by

$$\tilde{z}_i(k) = \sum_{j=1}^{N} h_{ij} \tilde{z}_{ij}(k), \tag{27}$$

where $\tilde{z}_{ij}(k) = z_j(k) - H\hat{x}_i(k \mid k-1)$. The state estimate in Equation (5) is calculated by setting $\tilde{z}_i(k)$ equal to $v(k+1)$. The Kalman covariance matrix requires inclusion of uncertainties from each of the observations. If we assume a valid observation falls within each track gate during each discrete time step, the Kalman covariance is the sum of Equation (9) with

$$dP(k) = W(k)\left[\sum_{j=1}^{N} h_{ij} \tilde{z}_{ij} \tilde{z}'_{ij} - \tilde{z}_i \tilde{z}'_i\right] W'(k). \tag{28}$$

D. JPDAF AND IMM COMBINATION

The JPDAF and the IMM integration require time phasing of their respective parameters. Model probabilities $\mu_{ij}(k-1)$, state predictions $\hat{x}_i(k \mid k-1)$, and Kalman filter covariance matrices $P_i(k \mid k-1)$ are required from the IMM during each discrete time step. The JPDAF is responsible for determining the gate regions for each model and validating observations for track update that fall in each model gate. Blackman and Papoli outline a process discussed below for integration of all parameters in [4]. Once new observations are received, the IMM submits a likelihood function for each model:

$$\lambda_i = (1 - P_D)\beta + \sum_{j=1}^{N} \frac{P_D e^{-d_{ij}^2/2}}{\sqrt{(2\pi)^M \, |S_i(k)|}}, \tag{29}$$

where N is the total number of observations. The inclusion of all hypotheses defined in the JPDAF causes Equation (24) to differ from the previously defined likelihood function in Equation (16). Model probabilities are updated using Equation (17). The state estimates and the covariance matrices are then updated for each model as discussed in the JPDAF following Equation (22). The final step includes the final steps of the IMM mixing process to produce new filtered state estimates and Kalman filter covariance matrices as described in Equation (14) and Equation (15).

E. SPLIT TARGET TRACK GENERATION

The example used above to describe the JPDAF operation dealt with track maintenance and ignored the formation of tracks. We will assume that the missile's track is reasonably developed based on external radar information and that other targets of interest are in the vicinity of the missile (e.g., deployed decoys). Thus, any measurements that occur within the missile's gate region and that cannot be associated with an existing track will be a candidate for a split target. An automatic track formation process is described in [10], but we will apply a split model within the IMM framework to support track generation.

The modeling of decoy deployment follows the methodology described in [3]. For our purposes, a decoy is synonymous with a split target. Tracking a split target adds two models to the IMM. The first model, the "just split" model, contains a stacked state vector and takes advantage of the cross-covariance terms containing the past common history of the two targets. The second model associated with the split continues track maintenance in the same manner that the missile track is being updated. All splits are contained within this model, and the JPDA is used to discriminate between split targets.

The "just split" model described below is discussed in [3] and contains several assumptions. The model begins with a single predicted measurement from the previous time step, and two validated measurements, as discussed in the JPDA, are present within the missile target gate at the current time step. Scenarios containing greater than two measurements (even if two splits are occurring simultaneously) are not covered in this research. Additionally, we will assume that once a target splits, no recombination occurs; a reasonable assumption due to the likely maneuver of the missile once a decoy is deployed. Therefore, the recombination model discussed in [3] has been ignored.

Once two measurements are received, a stacked vector consisting of a duplicated state is created, resulting in

$$x^J(k) = \begin{bmatrix} x_1^J(k) \\ x_2^J(k) \end{bmatrix}, \tag{30}$$

where J represents the "just split" model [3]. In addition to the same state, the predicted state is identical as well and is stacked in the same manner. The common predicted measurement then becomes

$$\hat{x}^J(k|k) = \hat{x}^J(k|k-1) + W^J(k)v^J(k), \tag{31}$$

where $v^J(k) = z_i(k) - \hat{z}^J(k)$, $\hat{z}^J(k) = H^J(k)\hat{x}^J(k|k-1)$, and the just split Kalman gain is

$$W^J(k) = \begin{bmatrix} W_1^J(k) & 0 \\ 0 & W_2^J(k) \end{bmatrix}. \tag{32}$$

Each of the parameters contained within the Kalman filter equations require a similar stacking for the dimensions of the matrix calculations, including the measurement noise covariance $R^J(k)$ and the measurement matrix $F^J(k)$. The covariance associated with the prediction will be constructed by

$$P^J(k|k-1) = \begin{bmatrix} P_{11}^J(k|k-1) & P_{12}^J(k|k-1) \\ P_{21}^J(k|k-1) & P_{22}^J(k|k-1) \end{bmatrix}. \tag{33}$$

The combined covariance matrix from Equation (16) serves as the initial value for each of the covariance terms listed in Equation (33). The off-diagonal covariance matrices contain the correlation between the new estimates and the past common history [3].

With the formation of the split track underway, we are now interested in decoupling the targets and tracking each independently. The validation regions discussed in the JPDA become useful metrics in determining a “no overlap” test explained in [3]. The already developed missile gate region is compared to a newly developed split target gate region. As the targets separate such that the two gate regions no longer overlap, we assume the two targets are completely decoupled. The “no overlap test can be described by

$$[\hat{z}_1^s(k|k-1)-\hat{z}_2^s(k|k-1)]'[S_1(k)+S_2(k)]^{-1}[\hat{z}_1^s(k|k-1)-\hat{z}_2^s(k|k-1)]\geq g, \tag{34}$$

where g is chosen as the tail end of a chi-squared density such as one percent [3]. Once the threshold for decoupling is met, both targets can be tracked independently with existing state estimates and covariance matrices.

F. TARGET DELETION

We have chosen to suspend split target tracking at a specified distance between the split target and the missile. The split target is likely to transit outside the interceptor field-of-view before software interaction is required. However, split target tracks may be of interest for kill vehicle adjudication or to simplify the JPDA problem should the split target return within the missile gate. For the purpose of the simulation, we choose a threshold range from the missile in which to discontinue tracking of 80,000 meters; although, split target measurements continue to be plotted.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MODEL CONSTRUCTION

A. OVERVIEW

One master script file and several function files were constructed and run in MATLAB 2012a. We assume measurement data is converted from likely range and bearing data to a Cartesian coordinate system; we ignore mathematical errors associated with such a conversion. A variable naming scheme is used throughout the MATLAB files because the same parameters are duplicated between models. In general, a number corresponding to an associated model follows each repeated variable (i.e., $\boldsymbol{F_3}$ denotes the state transition matrix for model three). All units are in meters.

We chose four models to fully describe missile and split target tracking. Assumptions include point mass modeling of both the target missile and the split target with motion described by linear control dynamics. Model one consists of a missile in straight-line motion at constant velocity. Model two simulates missile maneuver at a constant turn rate. Model three detects and analyzes for a potential target split from the missile. Lastly, model four maintains track on all confirmed target splits and discontinues tracking when the 80,000-meter threshold is reached.

B. MASTER SCRIPT FILE

The master script file defines common variables utilized by each of the models, generates true target vectors, and calls function files as required. The initialization values chosen are discussed as we present them and are embedded throughout the model construction explanations. We used a sample time of 0.01 seconds over 1,000 samples, resulting in a total simulation time of 10 seconds.

1. True Missile Motion

The missile vector is modeled as a point mass containing two degrees of freedom. At the beginning of each simulation, the true target state vector is set to

$$x_o = \begin{bmatrix} x \\ v_x \\ y \\ v_y \\ z \\ v_z \end{bmatrix} = \begin{bmatrix} 0 \\ 2500 \\ 0 \\ 2300 \\ 1000000 \\ 0 \end{bmatrix} \tag{35}$$

where x, y, z describe the position in the Cartesian plane and v_x, v_y, v_z describe the velocity in each dimension. The target vector is initialized to resemble a DF-41 Chinese ICBM with a velocity of 2670 m/s and a constant altitude of 1,000 km and is taken from open source information. The state transition matrix for straight-line motion is

$$F_1 = \begin{bmatrix} 1 & delta & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & delta & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & delta \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{36}$$

where $delta$ is the discrete step time of 0.01 seconds [12]. Thus, SLM results as each time step increments the x position vector by 25 meters and the y position vector by 23 meters. Each six by one true target motion vectors for SLM is collected in a matrix throughout the simulation. Each column represents the target state at a specified measurement time step.

Although the simulation models motion and measurements in all three dimensions, the scenarios analyzed below assume a constant value of z for true target motion. All plots are presented in the $x-y$ coordinate plane. The model for target acceleration assumes a constant speed turn in the $x-y$ plane with a state transition matrix as follows:

$$F_2 = \begin{bmatrix} 1 & \frac{\sin(ab)}{a} & 0 & \frac{1-\cos(ab)}{a} & 0 & 0 \\ 0 & \cos(ab) & 0 & -\sin(ab) & 0 & 0 \\ 0 & \frac{1-\cos(ab)}{a} & 1 & \frac{\sin(ab)}{a} & 0 & 0 \\ 0 & \sin(ab) & 0 & \cos(ab) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (37)$$

where a is a constant turn rate and b is the discrete time step [12]. The turn rate value was selected such that a 20 g turn is simulated. The turn is initiated and terminated according to specified discrete time step values. Any combination of SLM and turning in either direction is possible.

2. True Split Target Motion

Actual decoy deployment is triggered at a specified time step n as a target split. The decoy (split target) is initialized using the current value of the true target. The velocity component(s) of the decoy are then modified to create separation from the true target over time. We assume linear motion of the split targets, leading to the same transition matrix for F_3 as was used for SLM (F_1). No computational changes are made as the true split vector is analyzed in model four, but the same transition matrix is relabeled as F_4 and the true split state is relabeled as x_4 for state estimate comparison purposes.

3. Sensor Noise

We assumed a fixed sensor with a five-meter error in each dimension (x, y, and z). A white Gaussian process is used to generate the sensor noise $w(k)$ from Equation (2). The resultant measurement covariance matrix is

$$R = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix}, \qquad (38)$$

where σ_x^2 , σ_y^2 , and σ_z^2 are equal to 25 meters. Each of the Kalman filters operating in parallel utilizes this parameter. "Typical tracking error decreases are moderate (on the order of 25 percent)…unless very accurate, high update rate measurement data are available."[9] Based on greater than a 40 percent drop in mean distance errors with the specified tracking error and sampling rate, we assess our estimations to be reasonable.

4. Plant Noise

Plant noise $v(k)$ from Equation (1) is modeled as a white Gaussian process that is independent of sensor noise and is used to describe un-modeled parameters [12]. The plant noise covariance matrix is

$$Q(k) = q^2 \begin{bmatrix} \frac{delta^3}{3} & \frac{delta^2}{2} & 0 & 0 & 0 & 0 \\ \frac{delta^2}{2} & delta & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{delta^3}{3} & \frac{delta^2}{2} & 0 & 0 \\ 0 & 0 & \frac{delta^2}{2} & delta & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{delta^3}{3} & \frac{delta^2}{2} \\ 0 & 0 & 0 & 0 & \frac{delta^2}{2} & delta \end{bmatrix}, \qquad (39)$$

where $delta$ is the discrete time step of 0.01 seconds per sample and q^2 is a scalar value used to differentiate between the different model states [12]. The value of q^2 is maintained at 10 for SLM and 4,000 for a constant speed maneuver.

5. Model Transitions

We restrict the transition between certain models based on both physical constraints and assumptions. We first assume that from SLM (model one) the missile is able to remain in SLM with probability p_{11}, conduct a maneuver with probability p_{12}, or split with probability p_{13}. A distinction must be made concerning the "transition" from model one to model three. We are not saying that the target either proceeds in SLM or splits, but more accurately, that a split may occur while the missile proceeds down its track without maneuvering. The missile is physically capable of doing both, and missile state estimates continue to be calculated despite a split occurring.

We will also assume that from model two the missile is able to return to SLM with probability p_{21}, continue to maneuver with probability p_{22}, or conduct a target split with probability p_{23}. The same argument as above applies, and a potential target split (model three) occurs simultaneously with continued missile maneuver (model two). We considered setting p_{23} to zero considering the most likely evasion technique to be a deployed decoy followed by a missile maneuver. However, an equally feasible tactic may include deployment of a decoy during a constant turn. The approach used by Bar-Shalom, Chang, and Blom in [3] only account for target splits from SLM and not for a turn.

Assuming no recombination, transition probabilities for model three and four are straightforward. We assume potentially split targets in model three, allowing only options for the target to be a confirmed split (p_{34}) or just an erroneous measurement (p_{33}). The transitional probability p_{33} also accounts for track formation, as a finite number of measurements are required to confirm a split target. A split track will remain in model four (p_{44}) for track maintenance, resulting in a transitional probability of one.

Now that we have identified the non-zero transitional probabilities, values must be assigned to each. A lack of historical ICBM flight and decoy data, at least in open sources, forces estimated values. Assuming a dominant straight-line trajectory and low split target density, the Markov chain transition matrix was established as

$$p_{ij} = \begin{bmatrix} 0.8 & 0.1 & 0.1 & 0 \\ 0.7 & 0.2 & 0.1 & 0 \\ 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{40}$$

An unnecessary amount of complexity and restriction was realized soon after model construction when following the approach presented in [3], which led to the generation of Equation (40). The apparent sacrifice of true target accuracy as the number of splits increased would pose an issue for future work. Although only designed to handle a limited number of target splits in this research, a realistic scenario involves frequent splits or batch splits over the duration of midcourse flight. A rising split probability (p_{13} and p_{23}) results in a reduction of other model transition probabilities. Instead we decided to limit the IMM model transitions to model one and model two. Multiple observations falling within the true target gate will now trigger a transition to model three, a potential target split. Thus, the new transition matrix for the IMM becomes

$$p_{ij} = \begin{bmatrix} 0.8 & 0.2 \\ 0.6 & 0.4 \end{bmatrix}. \tag{41}$$

The “modelprob” function file utilizes the values in Equation (40) to accomplish IMM mixing.

C. MODEL FUNCTIONS AND INTERACTIONS

Each model generates estimates for its associated trajectory. The process used to generate estimates from the true target vectors is described below for each track. The master script file triggers when each model (contained within function files) is required based on the discrete time step k.

1. Straight-Line Motion

A standard Kalman filter is applied to track true target SLM. Sensor noise is added to the true position vector by multiplying each dimension of the sensor covariance by a pseudorandom, normally distributed vector of the same size, creating measurements with simulated zero mean, white noise. The state and covariance are updated in accordance with Equation (6) and Equation (10), respectively. A distance error calculation is performed following state estimate update. The model outputs the state estimate and covariance, the noisy measurements for use in model two, measurement error, and the distance error. The function file “modelprob” collects all necessary data from model one and model two to conduct IMM mixing, update model probabilities, and update the filtered estimates. The intermediate values for state estimate and covariance presented in Equation (20) and Equation (21) are sent out from “modelprob” to a separate function file, “prediction”, which generates the final model estimates from Equation (22) and Equation (23).

2. Constant Speed

The second model receives the noisy measurements generated in model one and conducts the Kalman filter update equations. Distance error is also calculated and sent as an output, along with the updated state and covariance estimates.

3. Potential Split Target

Model three requires both true target and split target vectors to generate noisy measurements for both. The model one function file is not called when analyzing a split. The noisy measurements, along with other required variables, are stacked and prepared for split target evaluation as discussed in Equation (30) through Equation (33). A JPDA calculation is conducted utilizing Equation (24) for a two observations and two track scenario. The measurement vector is stacked according to the JPDA filter results, and state and covariance updates are calculated while stacked. The state estimates and associated covariance matrices are separated out. State estimates and covariance matrix predictions are calculated for model three and are available for output. The updated model one estimates and covariance matrix are sent to “modelprob” and “prediction” just as when no split is being analyzed for.

4. Split Target Tracking

The fourth and final model receives the un-stacked state estimates from model three and continues track maintenance on all split targets. The true split target tracks are received, and sensor noise is added to the measurements. Standard Kalman filtering occurs and predicted state estimates result. When the received measurement exceeds the state estimate by greater than 80,000 meters, tracking is terminated.

IV. MATLAB SIMULATIONS

A. SIMULATION SETUP

Simulations were designed to test the ability of the multiple filter models to track a target through maneuvering turns and maintain the correct track as decoys are deployed. Mean distance errors are calculated over 100 Monte Carlo simulation runs.

B. INDEPENDENT EVALUATIONS

1. IMM Missile Tracking

Accurate tracking of the ICBM through a turn is the goal of the first simulation. The scenario establishes a true target track beginning in SLM, executing a turn from sample $k = 200$ until sample $k = 800$ and then returning to SLM. The combined estimate track is used as a measure of IMM tracking performance. To validate the additional computations in the IMM mixing process, the individual estimates from model one and model two are compared to the combined estimates. The model one and model two estimates $\hat{x}(k+1|k+1)$ are collected following the Kalman filter update (Equation 5), but the state estimates $\hat{x}(k+1|k)$ being updated were filtered from the IMM mixing process in the previous time step k. The model one and model two estimates represent the quantities at the top of the IMM flowchart in Figure 2, and the combined estimates represent the updated filtered estimates.

For the first simulation, the number of samples is 10,000 with a sample rate of 100 samples per second. The true target motion is shown in Figure 4 on an x-y coordinate plane and provided as an overview of the simulation. The boxes in Figure 4 correspond to the location in which selected focus areas in the scenario are taken from. The locations include steady state SLM, the onset of target maneuver, and steady-state maneuver. The track layouts and distance error plots are presented and discussed at each focus area.

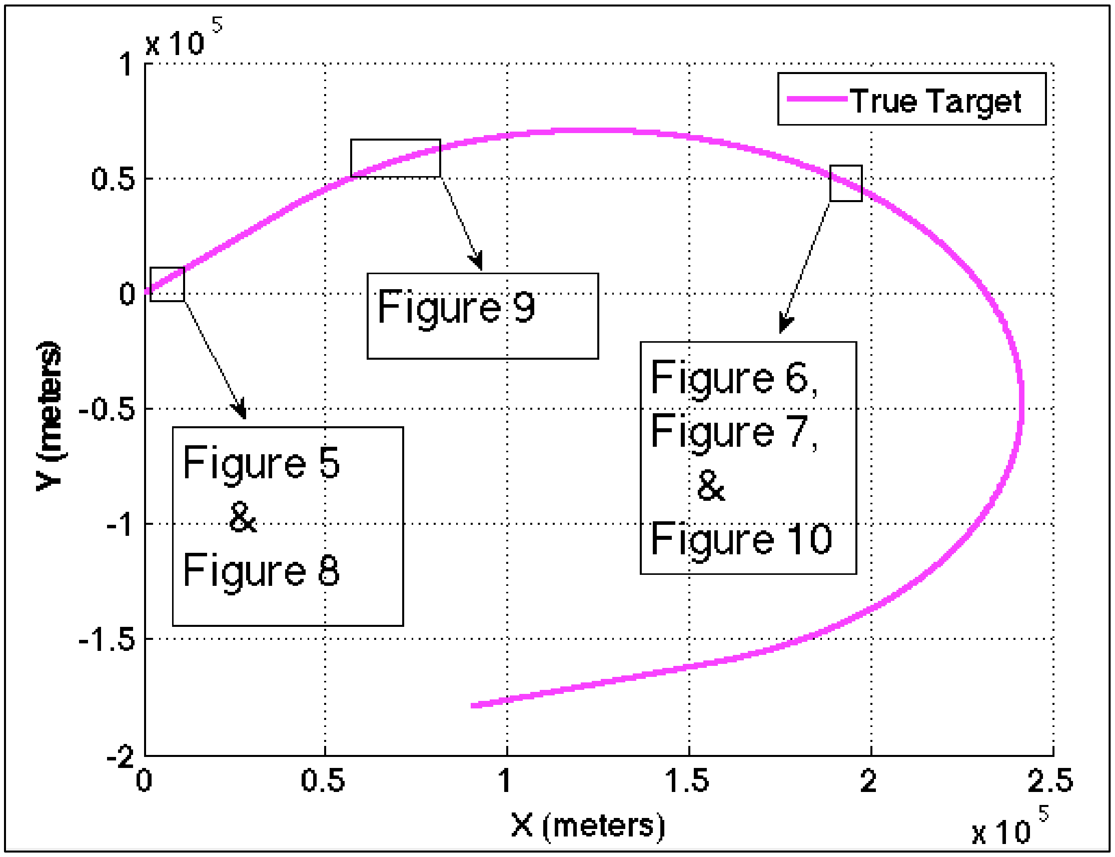

Figure 4. True target motion on an x-y plane identifying the location of areas of focus for follow-on figures.

The combined estimates provided better tracking performance in both straight-line and accelerating scenarios than the individual state estimates. The track layouts displayed in Figure 5 contain only a snapshot of the relationship between the tracks during SLM. The position of each track relative to the true target (the ICBM) track is as expected. During straight-line motion, model one is expected to be closer to the true target and model two further away. The combined estimate track sits closer than even the straight-line estimate track for a majority of the time. All tracks demonstrate an improvement over the received measurements.

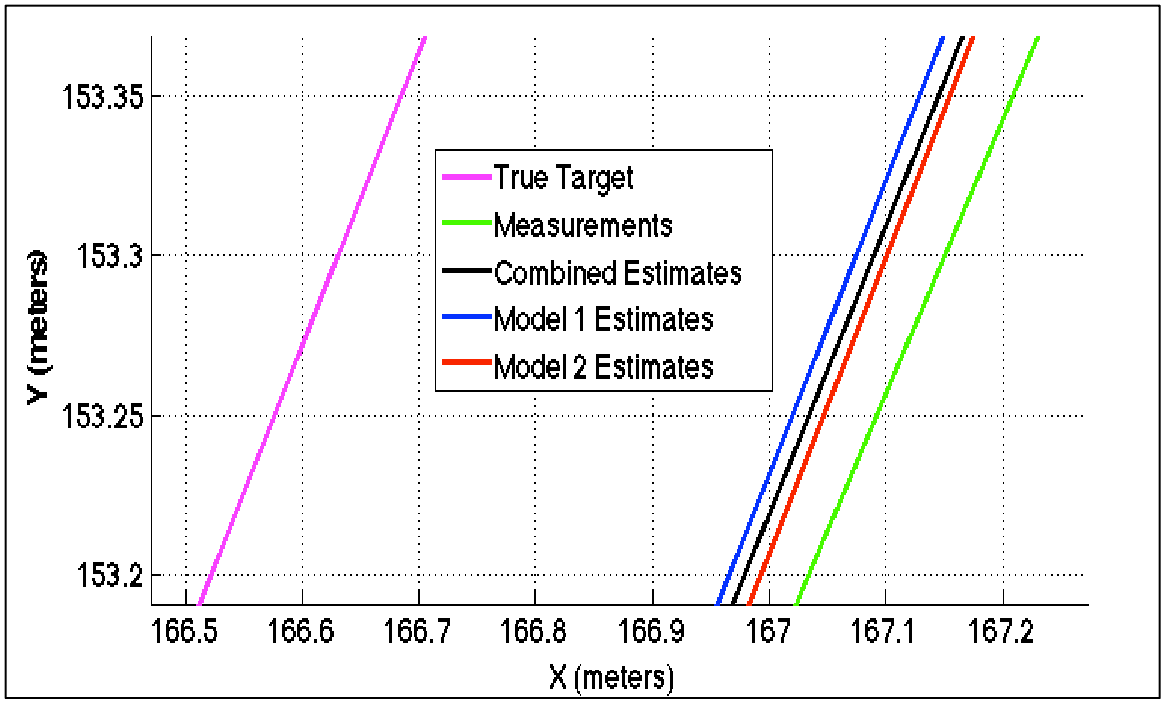

Figure 5. True target track, IMM estimate tracks, and measurements during straight-line motion on a Cartesian plane.

The track layout displayed in Figure 6 follows a similar discussion to those seen in Figure 5 except for the maneuvering scenario. Although simply a snapshot (the layout between two measurements is shown), the tracks maintain this pattern for a majority of the time. In this case, model two estimates are closer to the true target track, followed by the combined estimates, then track one estimates, and finally the measurements. The combined estimates also tend to lie closer to the model one estimates despite a target maneuver based on the probability assignments given in Equation (40).

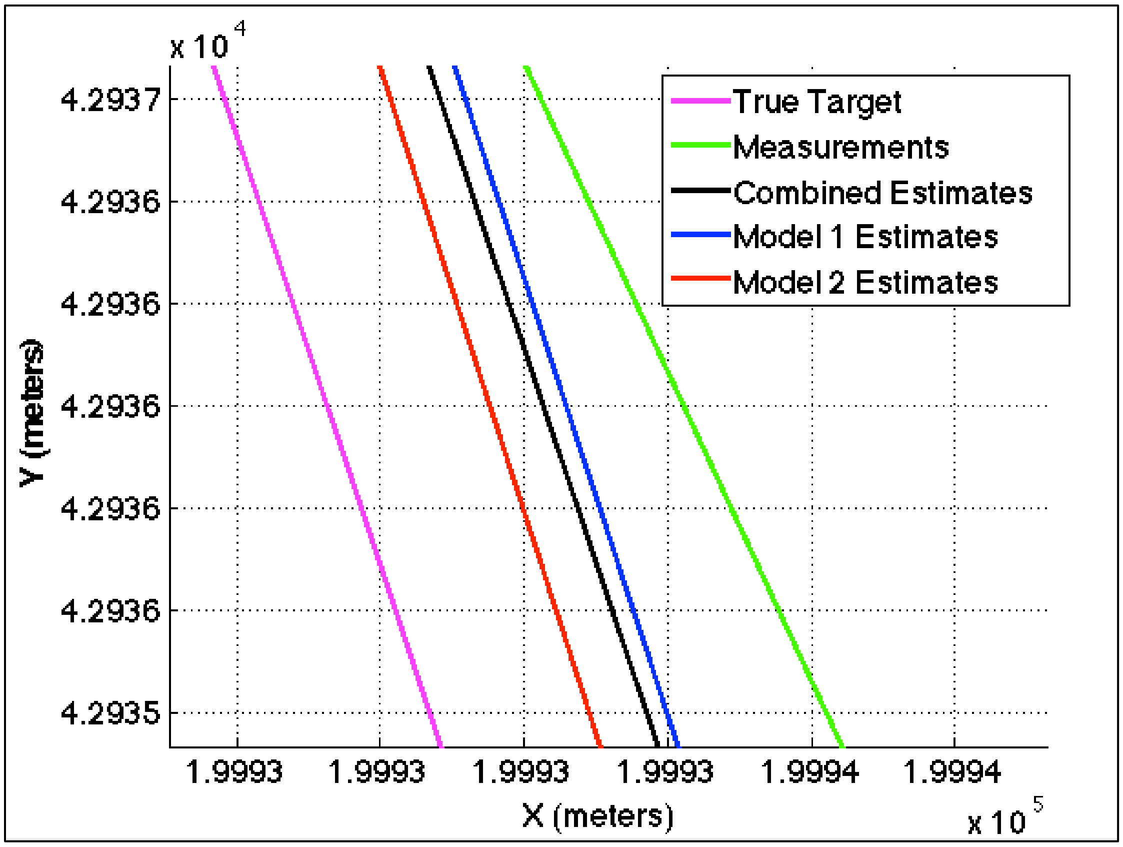

Figure 6. True target track, IMM estimate tracks, and measurements during a constant speed maneuver in the Cartesian plane.

The results contained in Figure 7 are provided to capture the relationship between the estimates and further prove the effectiveness of the IMM process. The tracks progress from the upper left corner of the figure to the lower right portion. As expected, the model one track continues to be furthest from the true target track during a constant speed turn. In the upper left portion of Figure 7, the measurements are near the true track and increase the model two probability u_2 from Equation (17). The values contained in Equation (19) and in Equation (20) are driven up as a result of u_2 going up and cause an immediate shift of the combined estimates towards the model two track. As the measurement vector diverges from the true track and toward the model one estimates, u_2 goes down and the model one probability u_1 goes up. The contribution from model one, which is further from the true track, drives the combined estimates away from the true track. In this snapshot, the measurements drive the location of the combined estimates within a channel consisting of model one and model two. This process is repeated on each time step and causes slight variations in the combined estimates.

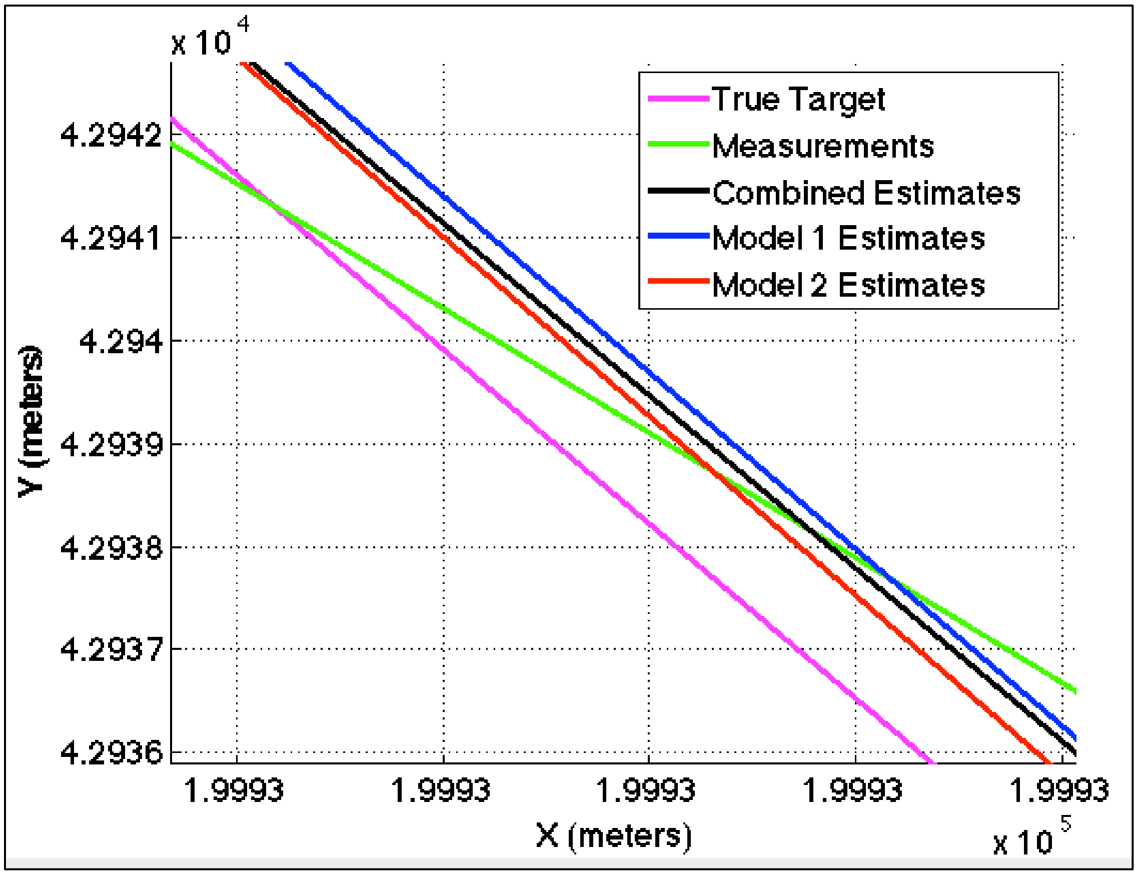

Figure 7. True target track, IMM estimate tracks, and measurements during constant speed turn in a Cartesian plane.

The relationship between the track estimates and true target track are quantified by distance error. The summation and averaging of the combined estimate variations over several (100 Monte Carlo) simulations reveals their performance. During SLM the results in Figure 8 are as expected. Model one contains the least error, followed by the combined estimates, and then model two. Estimated distance error values exhibiting a reduction of nearly 50 percent of measurement error validate performance of the IMM filter.

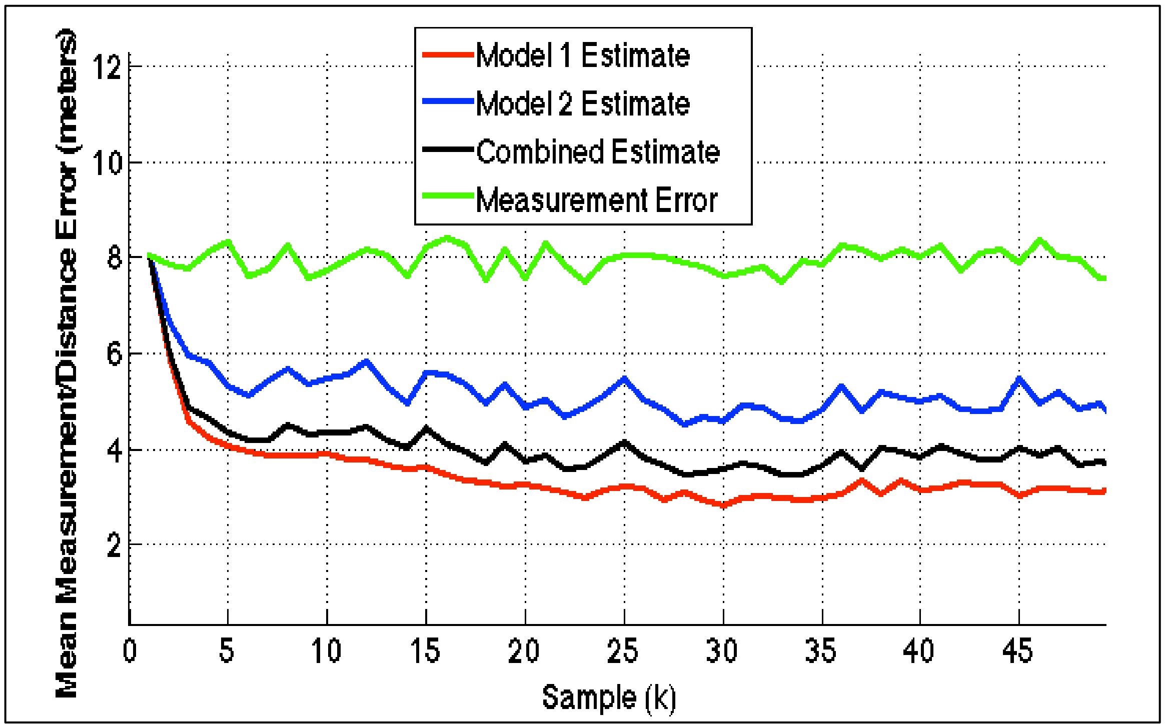

Figure 8. SLM mean distance errors of IMM estimates with mean measurement error as a reference.

The transition between SLM and a constant speed turn sees a reduction in error when using an IMM filter, but the magnitude is still significant. The combined estimates distance error lies directly on top of the distance error for model two in Figure 9. The estimates take approximately thirty-five samples (0.35 seconds with a 0.01 sampling rate) to return below measurement error and another 40 samples (0.4 seconds) before returning to steady-state. The transition back to SLM duplicates the results seen in Figure 9. A later comparison with the Kalman filter proves these transition times in both directions to be very reasonable.

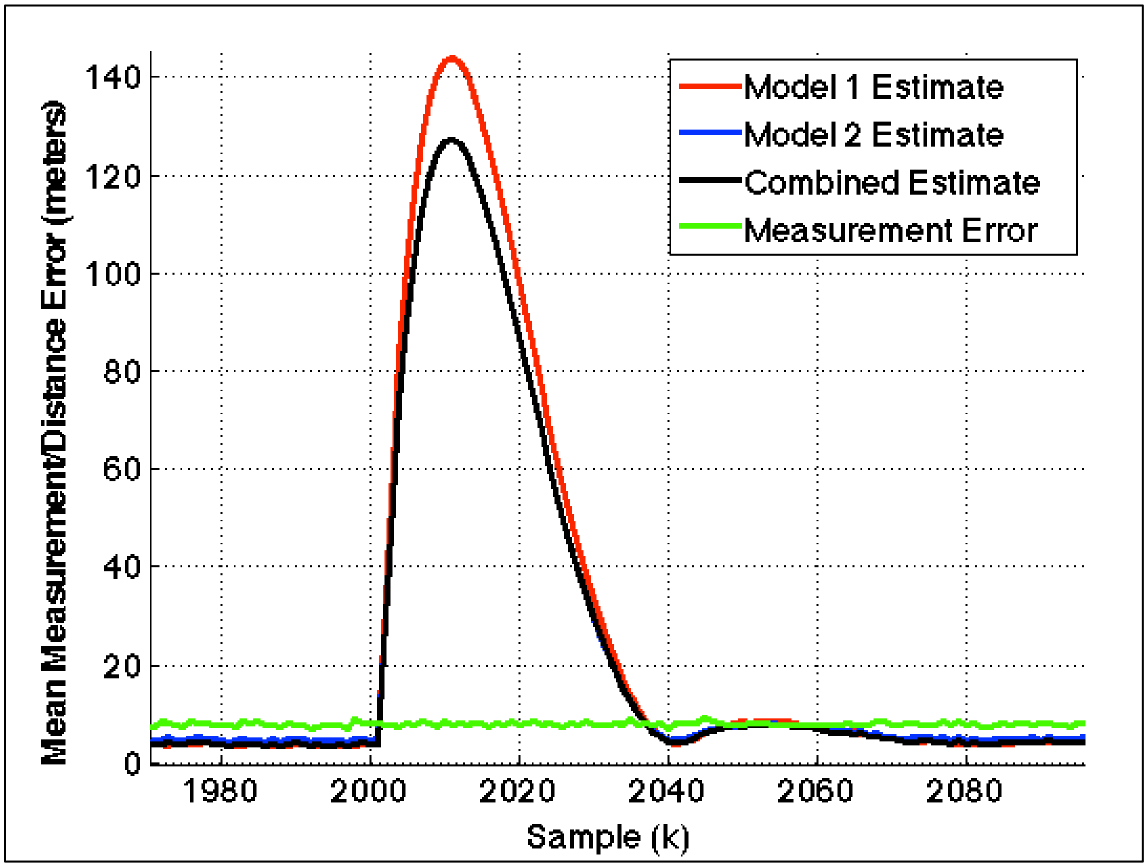

Figure 9. Mean distance error of IMM estimates and mean measurement error during a constant speed maneuver initiated at sample time $k = 2000$

The distance errors associated with a constant speed turn replicate those seen during straight-line motion and exhibit only half of a meter up-shift towards measurement error. A combination of transition probabilities ($p_{12} = 0.2$ and $p_{21} = 0.4$) favoring model one, IMM based prediction methods, and high process noise ($q_2 = 4000$) during the turn compared to SLM ($q_1 = 10$) take away from model two estimates replicating a true turn. These effects also drive the model probability for model two down. The result is seen in Figure 10 as model one retains the lowest distance errors.

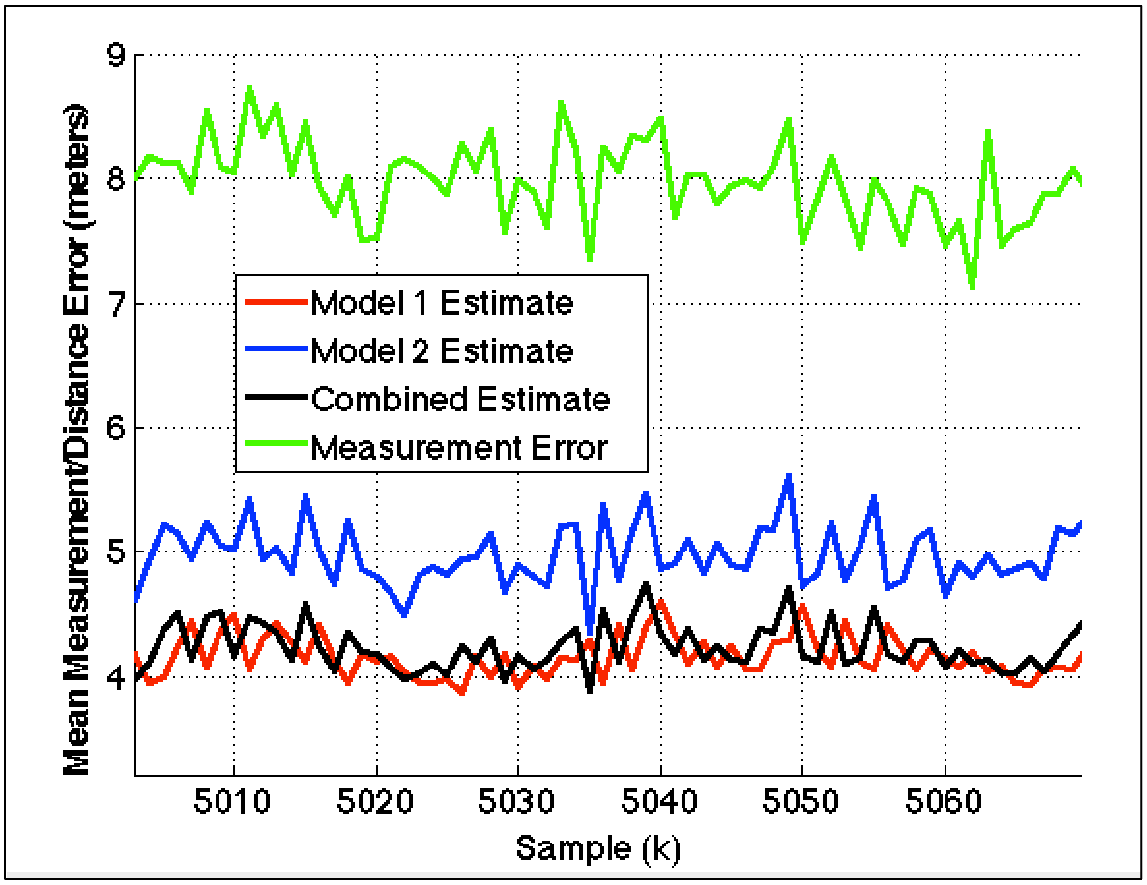

Figure 10. Constant acceleration mean distance error of IMM estimates with mean measurement error as a reference.

Another useful tool in assessing IMM performance is the model probability values and variations. These parameters give a clear indication of how well the IMM reflects reality. The probabilities shown in Figure 11 are averaged over 100 samples due to the vast amount of model probability value switching that occurred, particularly during the turn. Prior to the turn, model one probability was greater but not drastically. At the onset of the turn (at sample 2,000, displayed in Figure 11 as sample 20), model two is assured. However, as the transition to steady-state occurs, the IMM becomes less certain that a turn is occurring. The transition back to SLM is falsely identified as an assured target turn, when, in fact, completion of the turn is being identified.

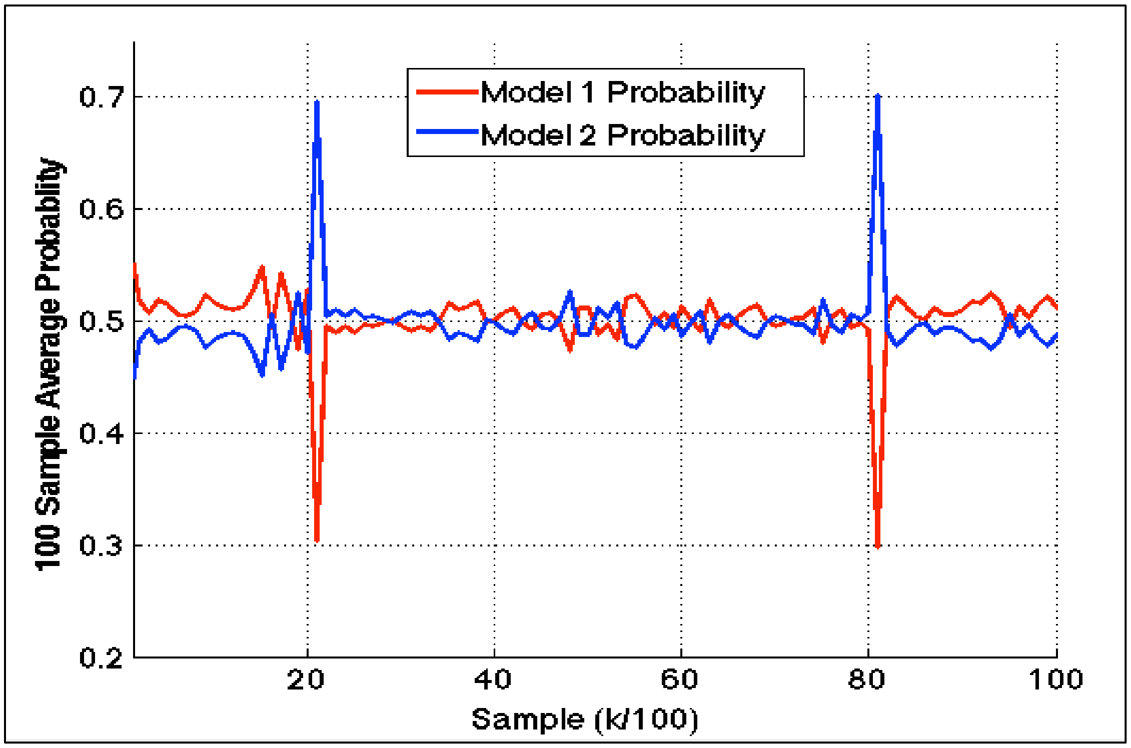

Figure 11. Model probability averaged over 10 samples during IMM mixing with the following transitional probabilities: $p_{11} = 0.8, p_{12} = 0.2, p_{21} = 0.45, p_{22} = 0.55$.

The transitional probabilities used in the simulation described above were modified based on the inaccuracy of those original estimated in Equation (40). A second simulation is presented to show the significance of the adjustment. The only parameters adjusted from the above simulation were p_{21} and p_{22}. We assumed that maneuvering would be minimal and estimated the values accordingly ($p_{21} = 0.6$ and $p_{22} = 0.4$). The simulation resulted in higher distance errors in each of the estimates during the turn, as seen in Figure 12. The combined estimates became dominated by model one. The SLM and peak model transition errors were unaffected, but an analysis of the model probabilities indicated a major problem that can be seen in Figure 13. The IMM assumed the target remained in model one during the entire simulation with the exception of the transitions.

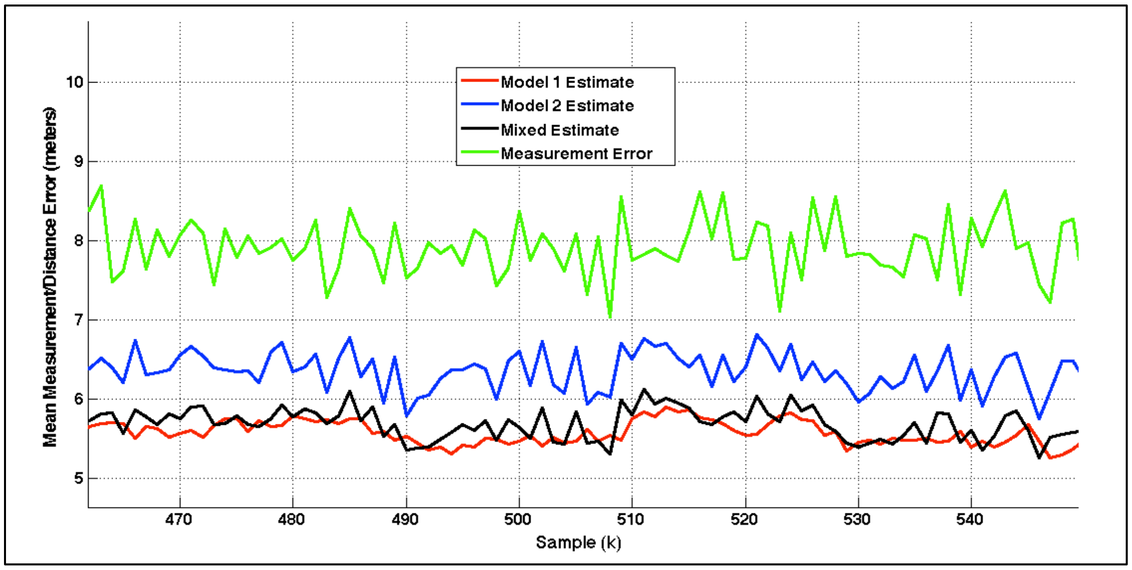

Figure 12. Constant speed turn mean distance error of IMM estimates with mean measurement error as a reference.

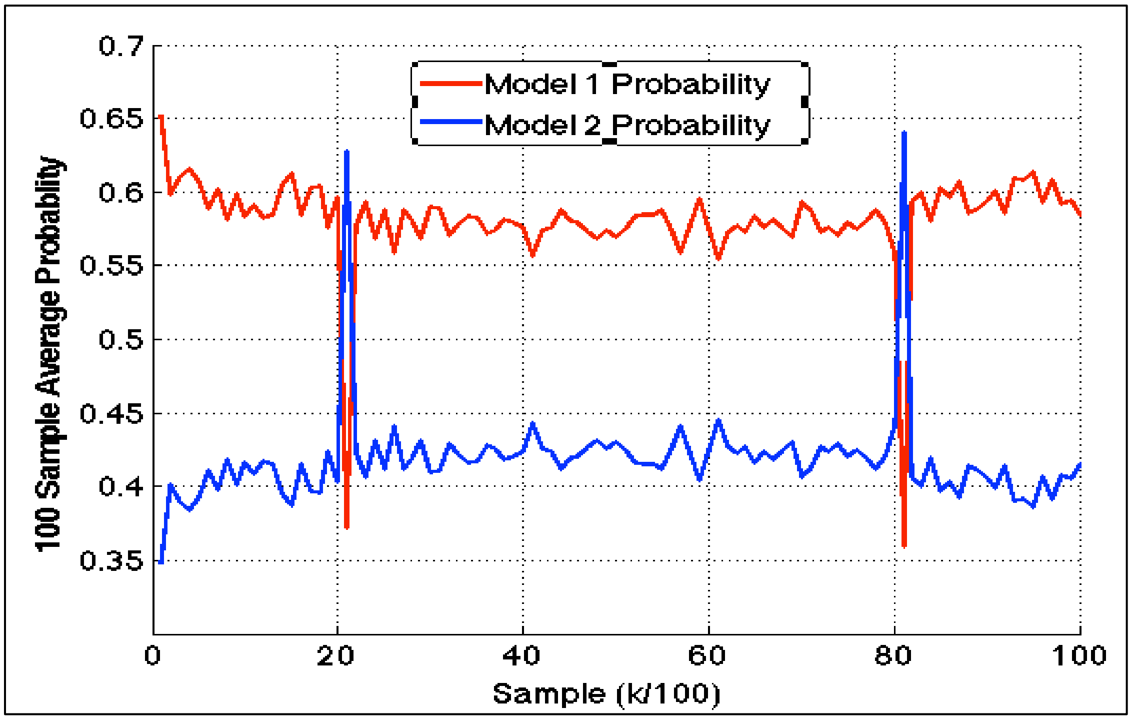

Figure 13. Model probability averaged over 10 samples during IMM mixing with the following transitional probabilities: $p_{11} = 0.8, p_{12} = 0.2, p_{21} = 0.6, p_{22} = 0.4$.

Further adjustments to the transition probabilities proved only marginally effective. The values used in the first simulation were adequate to track the missile through a turn. We assessed the IMM performance to be sufficient and, when paired with the JPDAF, capable of tracking the missile during split target tracking. All future simulations will ignore the model one and model two estimates and be evaluated based on combined estimates only.

2. Split Target Detection and Tracking

True split target tracks and model parameters were adjusted in the following simulation to display split track generation ability. The conditional probabilities, process and sensor noise, sampling frequency, sample length, and number of simulation runs remain unchanged. The generation of a true split track drives model three to generate measurements for the split and attempt track formation. The transition between models three and four is the focus, and a parameter t represents the number of samples model three processes, initially set to five.

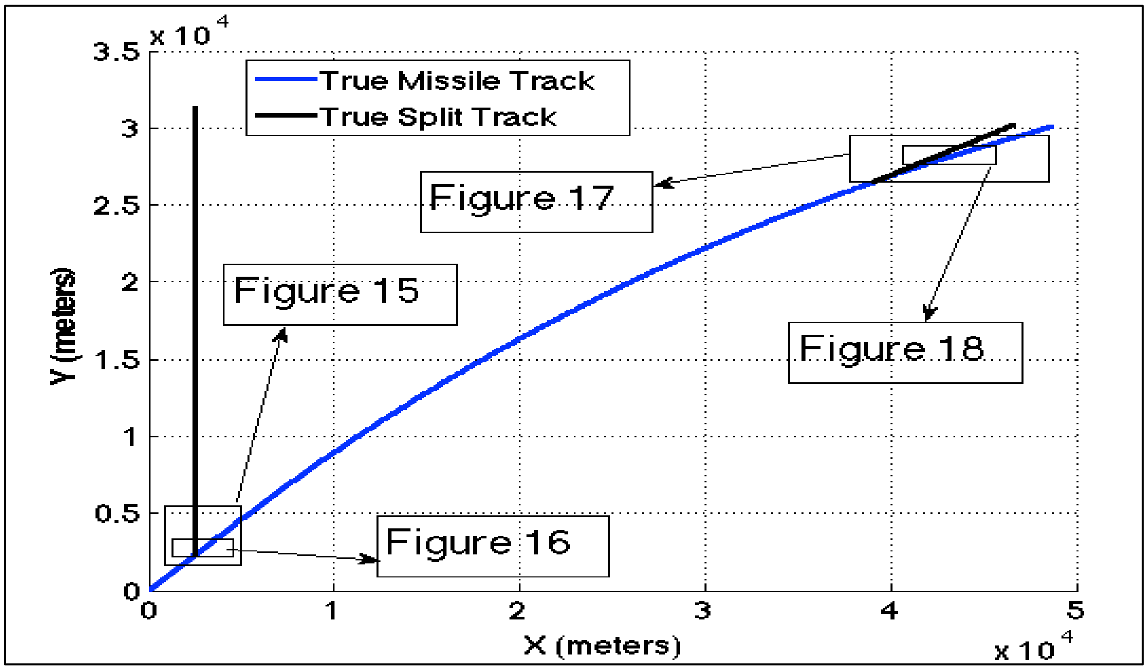

Figure 14. True target motion focus area for follow-on figures in analysis of target split simulations.

The second simulation begins in SLM with target splits occurring at sample $k = 100$ and at sample $k = 800$. The entire simulation of 1,000 samples (sampling rate remains at 100 samples per second) is contained within Figure 14. The locations in which the follow-on figures are taken are also displayed in Figure 14.

The convergence of the filter estimates on the split target is shown in Figure 15. Although an unlikely decoy deployment scenario, the simulated split shows the ability of the filter to correct to a drastic change of original estimates (nearly 90 degrees in this case). The transition to the red line corresponds to the transition to model four. The lower tracks represent the missile and combined estimates and are where model three estimates begin. The results contained in Figure 16 show an enlargement of the conditions in Figure 15 at the split junction. Clear divergence of the model three estimates occurs after three samples. Model four takes over track maintenance and is indicated by the solid blue line following the green line in Figure 16.

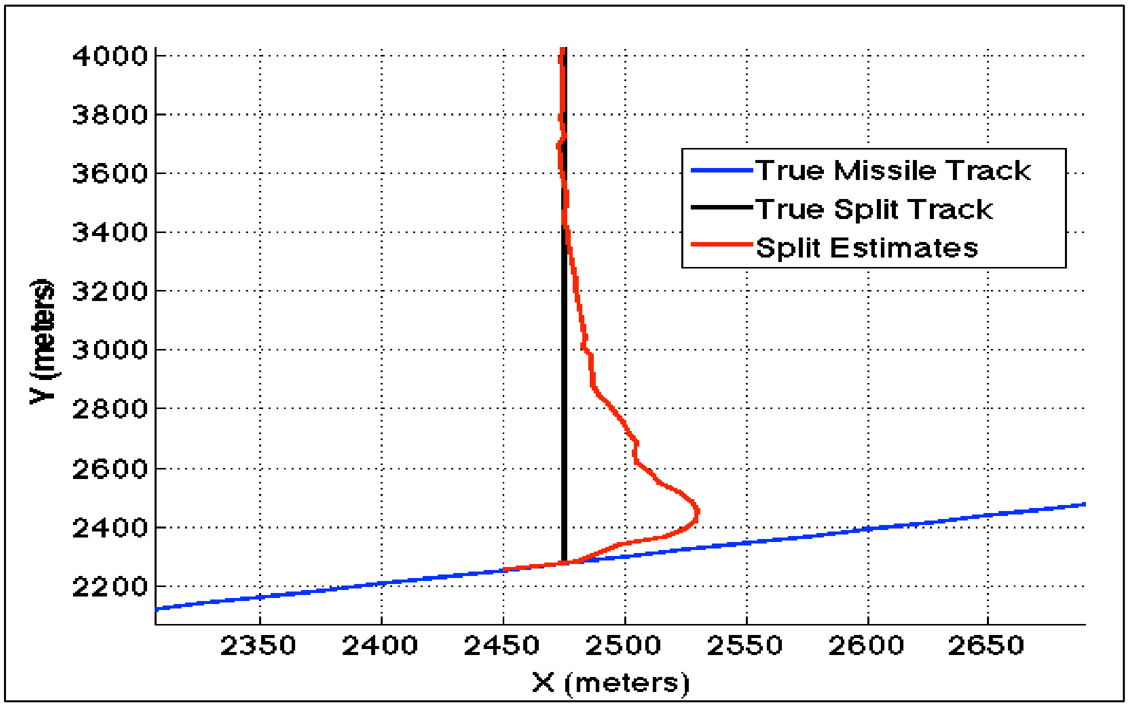

Figure 15. Split target estimates converging on split target from SLM.

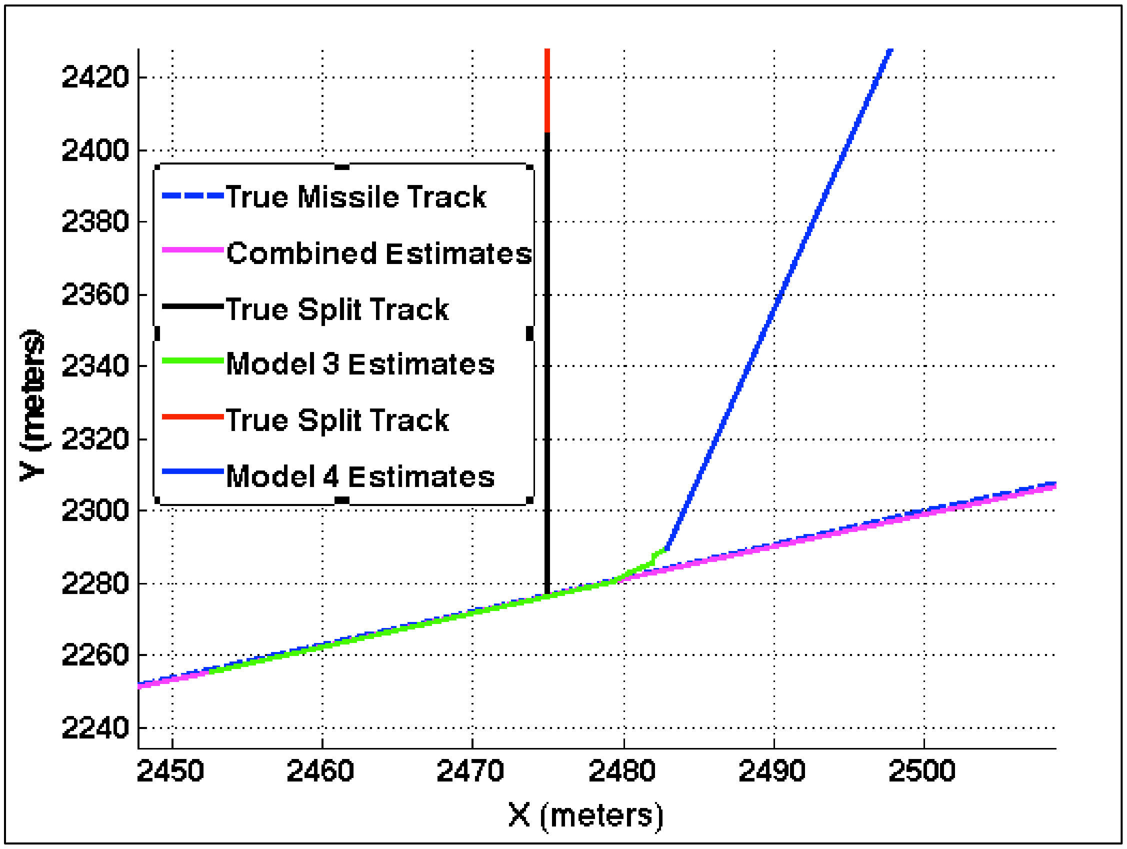

Figure 16. Model three and model four transitions during split target detection from SLM.

The simulation continued by implementing a turn from sample $k = 200$ to $k = 900$ and generating a second split at $k = 800$. The same color scheme as above is applied to the true and estimated tracks for the split a from constant speed maneuver. The small angle of separation shown in Figure 17 allows the split estimates to converge more quickly on the true target track. On the other hand, the small divergence of the split track causes the model three estimates to not noticeably diverge in Figure 18 until about the fifth sample.

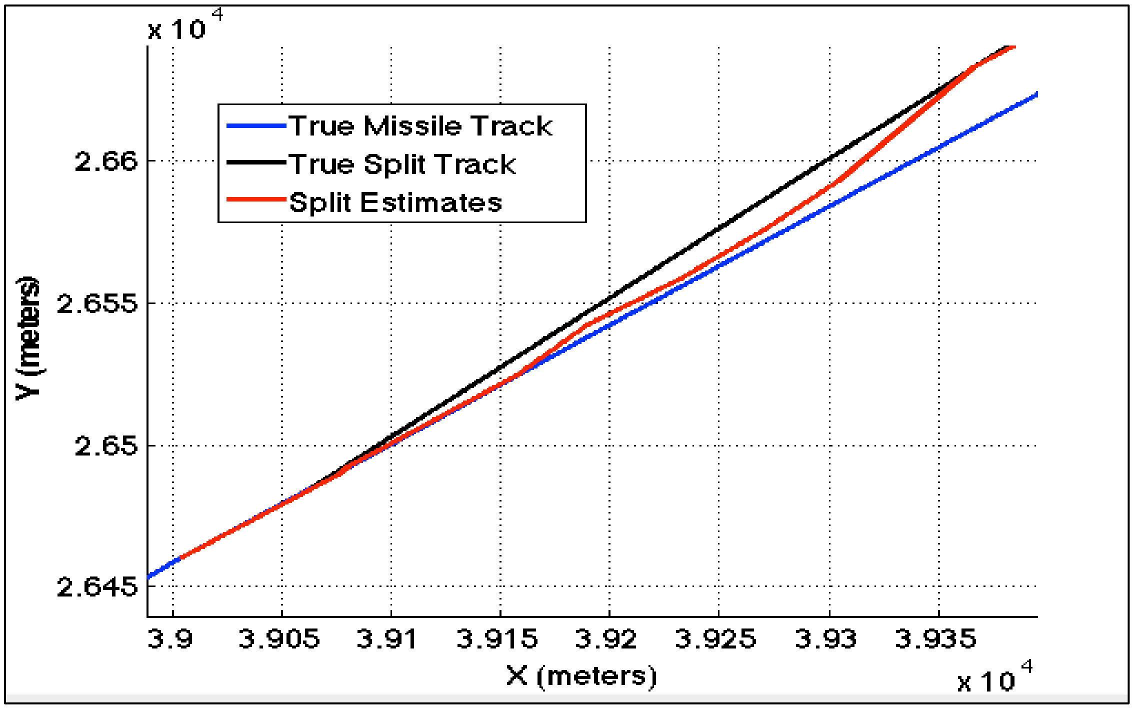

Figure 17. Split target estimates converging on split target from constant acceleration.

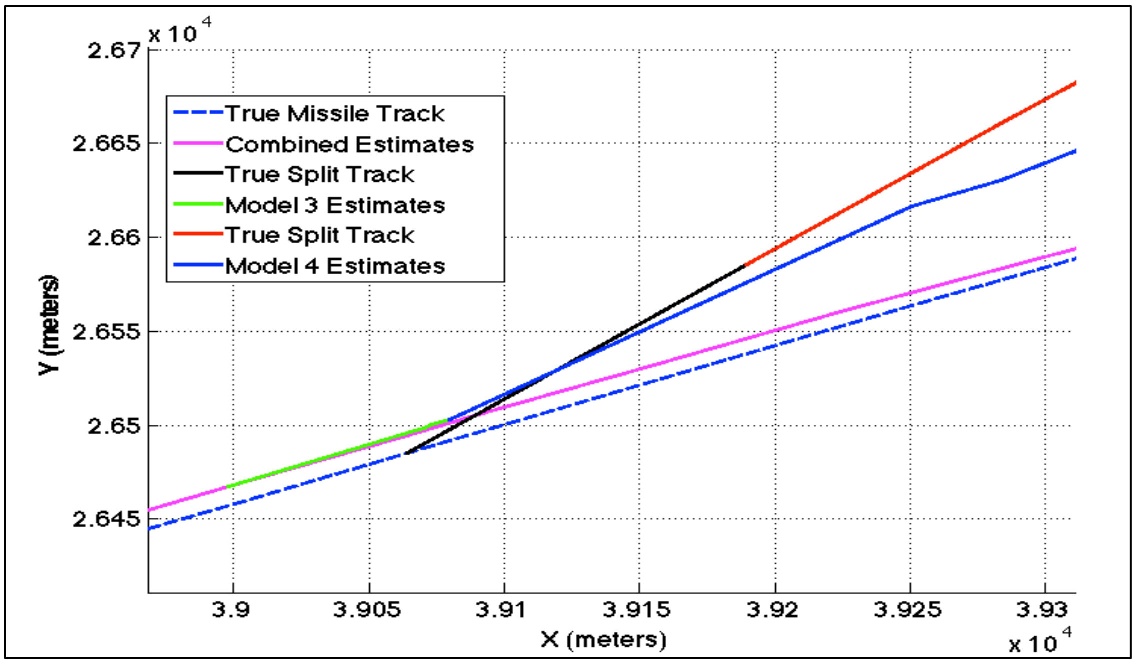

Figure 18. Model three and model four transitions during split target detection from constant acceleration.

The stacked vector method used in model three causes a linear increase in mean distance error over a short period of samples, as seen in Figure 19. We experienced three major factors based on simulations that affect the magnitude of model three error: the angle of separation between the true target track and the split target track, the missile state, and the relative speed of the split to the missile. The model three estimates are passed to model four, which receives an additional measurement and performs a Kalman filter update. Hence, the maximum mean distance error seen from model three is reduced in one estimate to the initial mean distance error seen in Figure 19. Both split tracks require about 25 samples to match measurement error and another 25 samples to reach a common steady-state error value, totaling a half of a second from receipt of model three estimates to nearly one-quarter of measurement error.

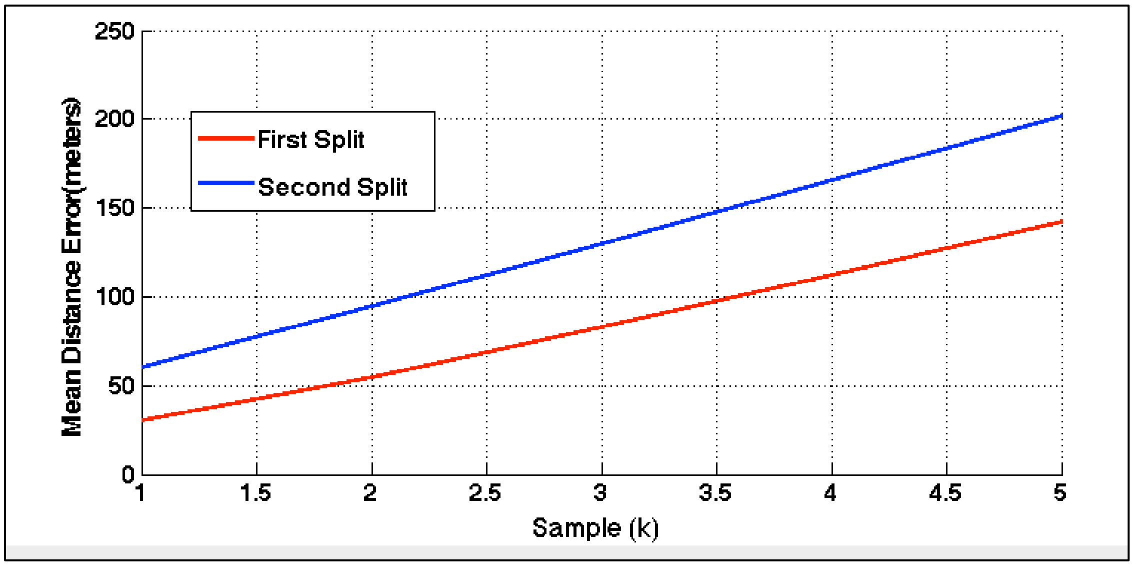

Figure 19. SLM and constant acceleration mean distance errors during target splitting track formation.

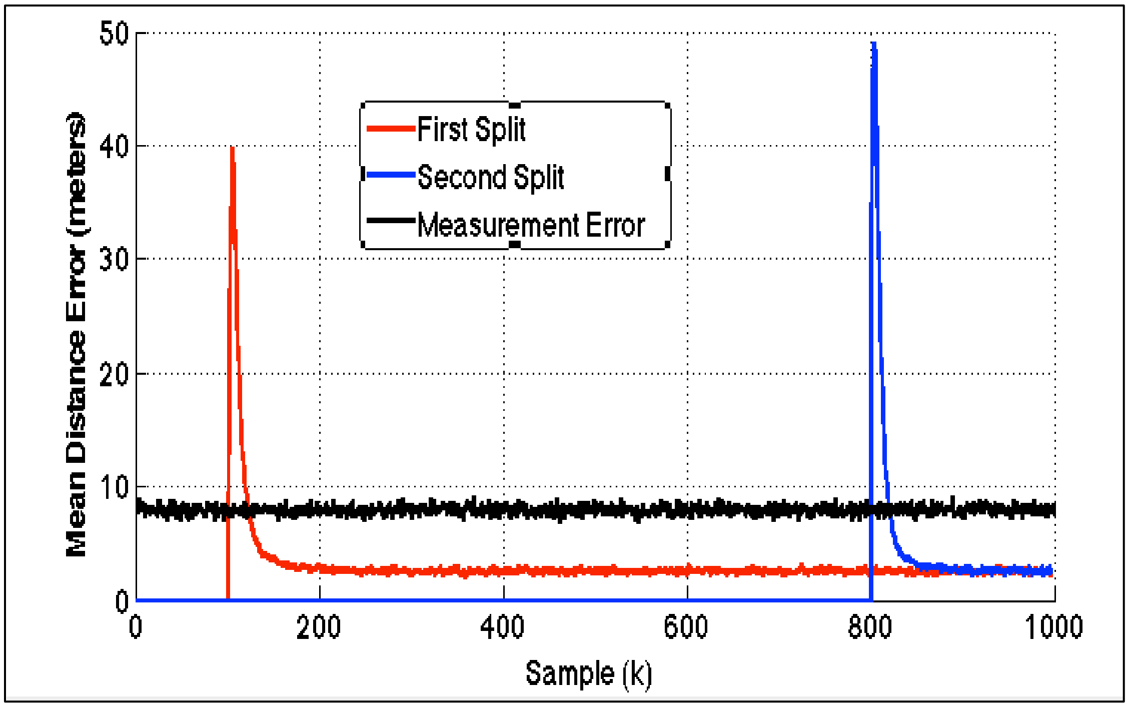

Figure 20. SLM and constant acceleration mean distance errors observed during split target track maintenance.

The combined estimate performance is reduced as a result of the model three stacked vector method. Although difficult to ascertain from the track layout during the simulation, the mean distance errors shown in Figure 20 show a 20-meter increase for straight-line motion and thirty-five meter increase for constant acceleration. The combined estimates require approximately 60 samples to return to steady-state following each split evolution. The steady-state errors are consistent on either side of the respective splits; however, the mean distance error during the turn is raised to only half of a meter below measurement error on average. The model one and model two transition errors seen in Figure 9 are duplicated in Figure 21.

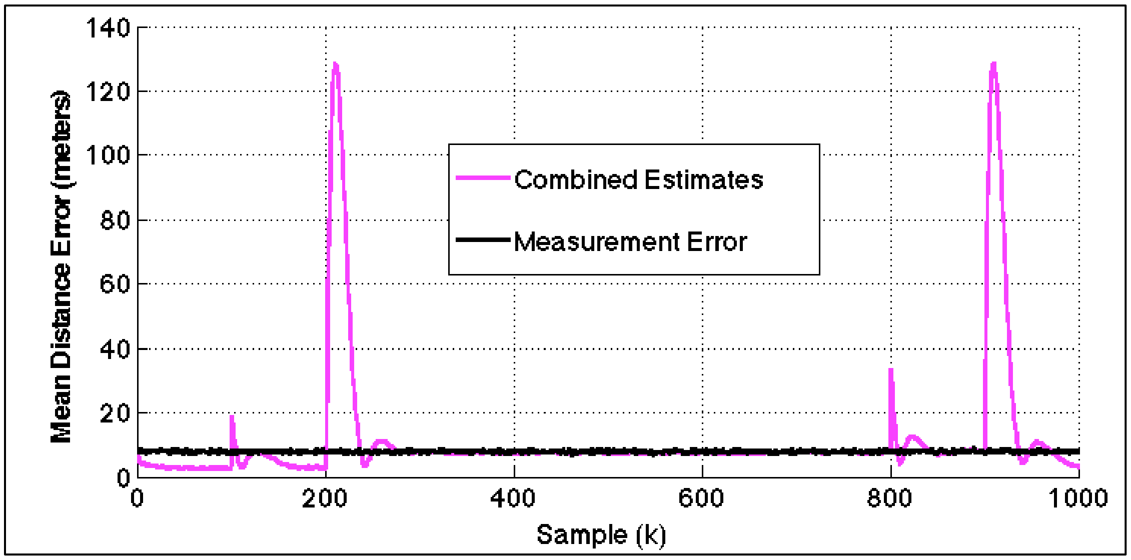

Figure 21. Combined estimate mean distance error versus measurement error seen during target splitting.

The next simulation analyzes the effects of raising the number of measurements that model three filters before transitioning to model four. For a split target track that remains significantly close to the true target track (i.e., within measurement error), further model three analyses may be required to ensure sufficient track separation before split target confirmation. The recorded common past history maintained by the stacked covariance matrix would likely be more accurate than relying on the JPDA with no cross-correlation between the measurements.

The number of measurements analyzed by model three t was raised to fifteen with all other parameters being the same as the $t = 5$ simulation. Track formation in Figure 22 (analogous in position to Figure 16) and Figure 23 (analogous in position to Figure 18) are improved but increasingly lags the true split vector. The mean distance error continues to increase linearly as seen in Figure 24. The time required for the split estimates to converge on the split target were minimally effected and resulted in plots identical to Figure 15 and Figure 27. Therefore, to minimize the linearly expanding error and still be able to generate a track, a value of $t = 2$ will be used.

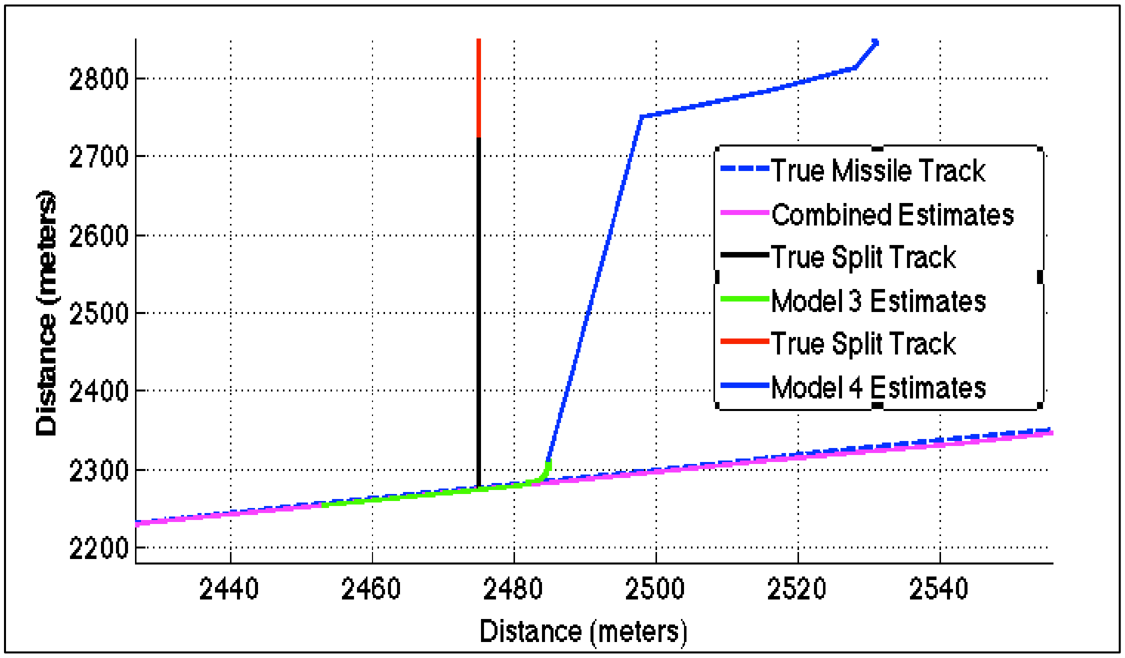

Figure 22. Target split during SLM with a raised number of model three samples ($t = 15$).

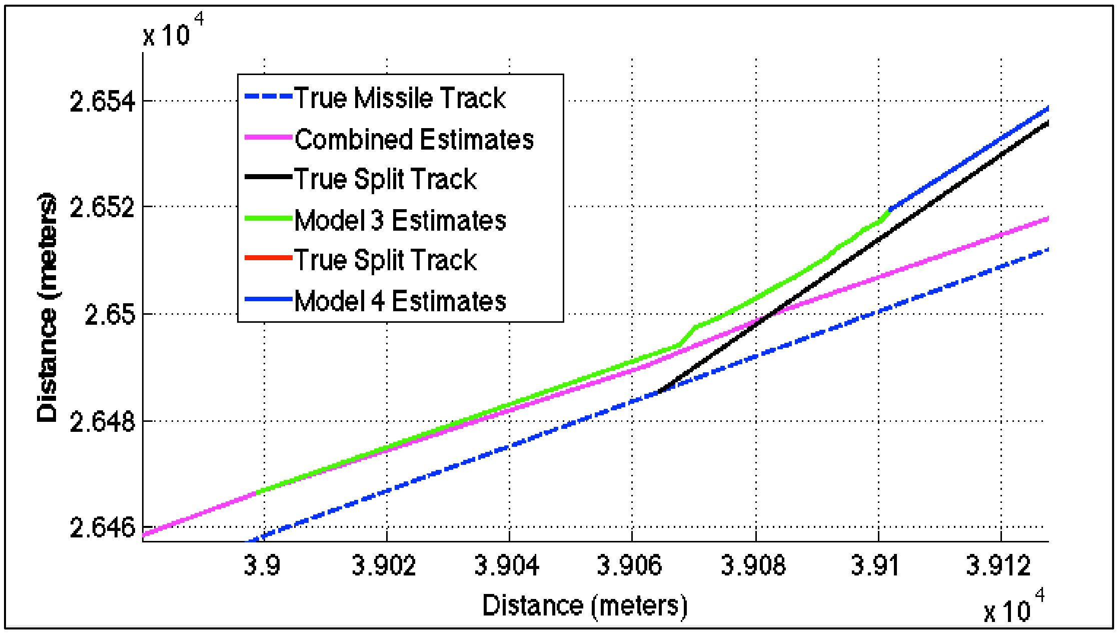

Figure 23. Target split during constant acceleration with a raised number of model three samples ($t = 15$).

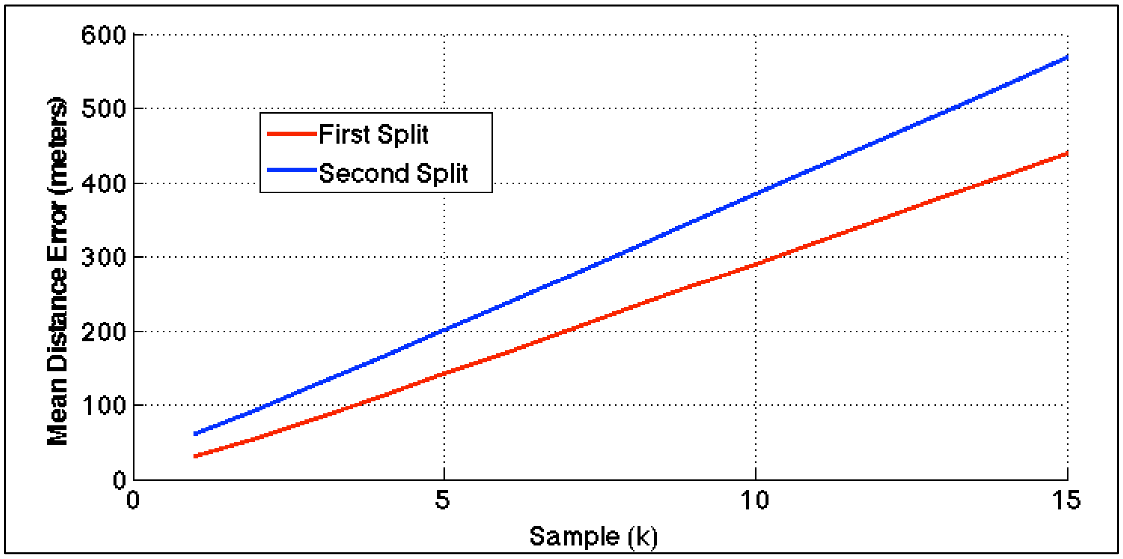

Figure 24. Model three mean distance error versus a greater number of samples for each target split.

C. COMPARATIVE ANALYSIS WITH A STANDARD KALMAN FILTER

1. Kalman Filter Setup

A standard Kalman filter was created and run in parallel with the IMMJPDA filter for split target comparison purposes. The comparison is intended to evaluate if the additional computing of the IMMJPDA results in increased split detection performance over a basic algorithm. All parameters of the Kalman filter are completely independent from the IMM and JPDA operations. The Kalman filter discriminates between the measurements by choosing the one with the smallest distance from the predicted track position. Mean distance error is calculated following state and covariance update using the norm between the estimated value and the measurement. The plant noise was adjusted to allow suboptimal performance during SLM in exchange for reasonable tracking capability during target maneuver. The Kalman can be adjusted to perform better in either of the target states but not both. The steady-state mean distance errors can be seen for SLM in Figure 25 and for a constant velocity maneuver in Figure 27. The peaks of the distance errors in Figure 26 were matched as closely as possible for comparable maneuvering tracking.

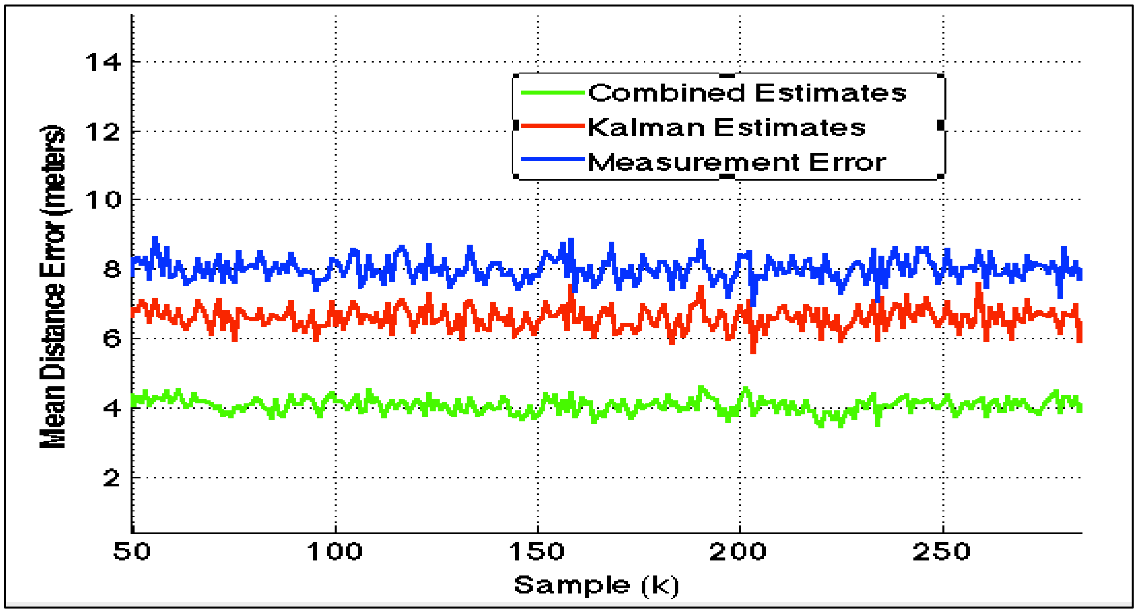

Figure 25. Comparison of Kalman and IMMJPDA filter mean distance errors during steady state SLM.

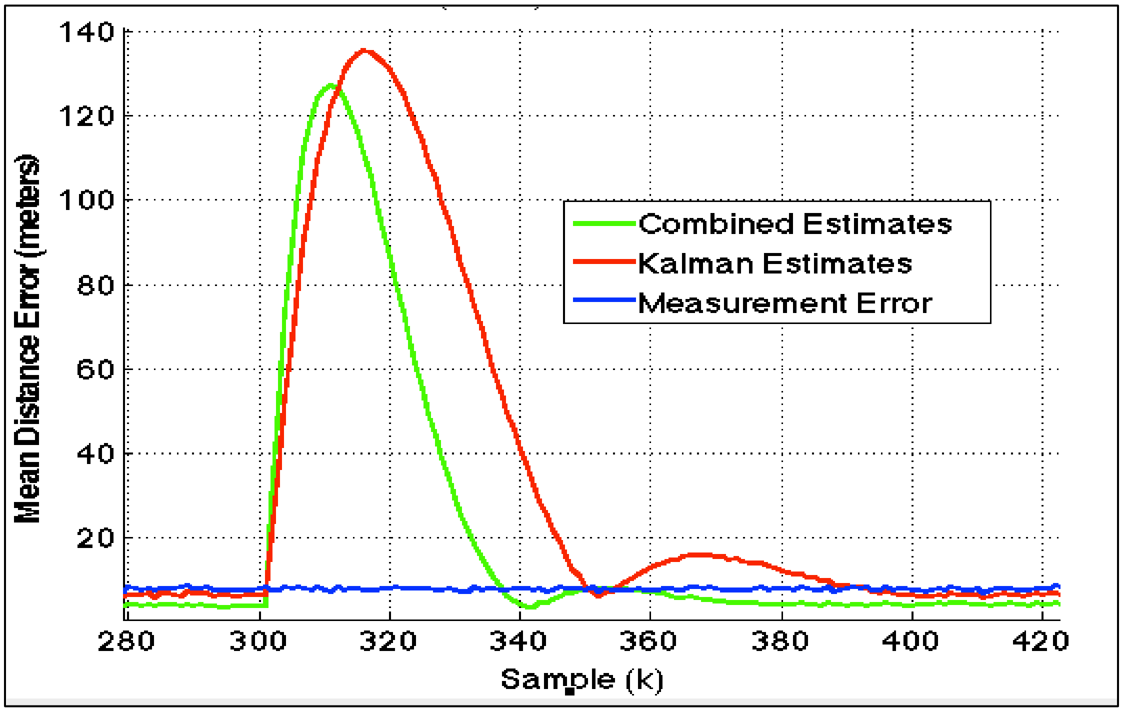

Figure 26. Comparison of Kalman and IMMJPDA filter mean distance error peaks during transition from SLM to constant acceleration.

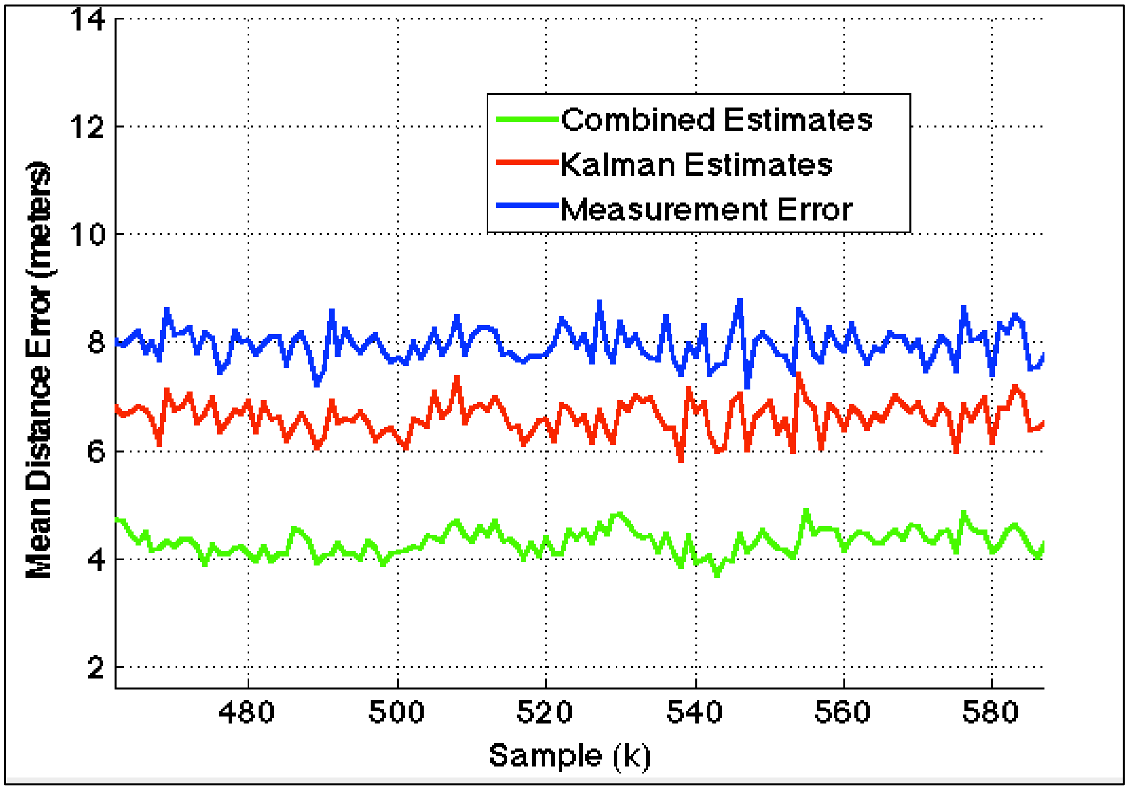

Figure 27. Comparison of Kalman and IMMJPDA filter mean distance errors during a constant acceleration turn.

2. Scenario Setup

The scenarios designed for comparison are intended to replicate decoy deployment and follow-on ballistic missile evasion tactics. The exo-atmospheric environment allows decoys to replicate true target motion. The true target and split target vectors are presented and the results are summarized in Table 2. A single simulation consists of 1,000 samples. Each scenario is used to generate 100 Monte Carlo simulation runs. The filters are evaluated based on their ability to maintain true target track.

Two performance criteria are used. First, if the final mean distance error value of the simulation exceeds a threshold value (approximately five times the peak values seen in Figure 26), we assume the decoy has tricked the filter. The percentage of success out of 100 runs is annotated in Table 2. Secondly, a counter is established to determine when each filter chooses split target measurements over true target measurements, representing the filter being tricked by the decoy at that time step. The measurements associated with the filter tracking the decoy are removed. The remaining number of faulty measurements associations represents isolated instances during successful target tracking where decoy measurement association was chosen. Two splits occur in each scenario. Successful tracking percentage and number of faulty measurement associations are recorded per simulation (1,000 samples) for each split separately. In general, divergence from the true target is about 20 degrees for the first split and by less than 10 degrees for the second split. The distinguishing features between the scenarios are the split locations and the varying target state before and after the split.

Scenario one is SLM with target splits at $k=200$ and $k=600$. The second split is intended to parallel the true target track. The layout of true target tracks is illustrated in Figure 28.

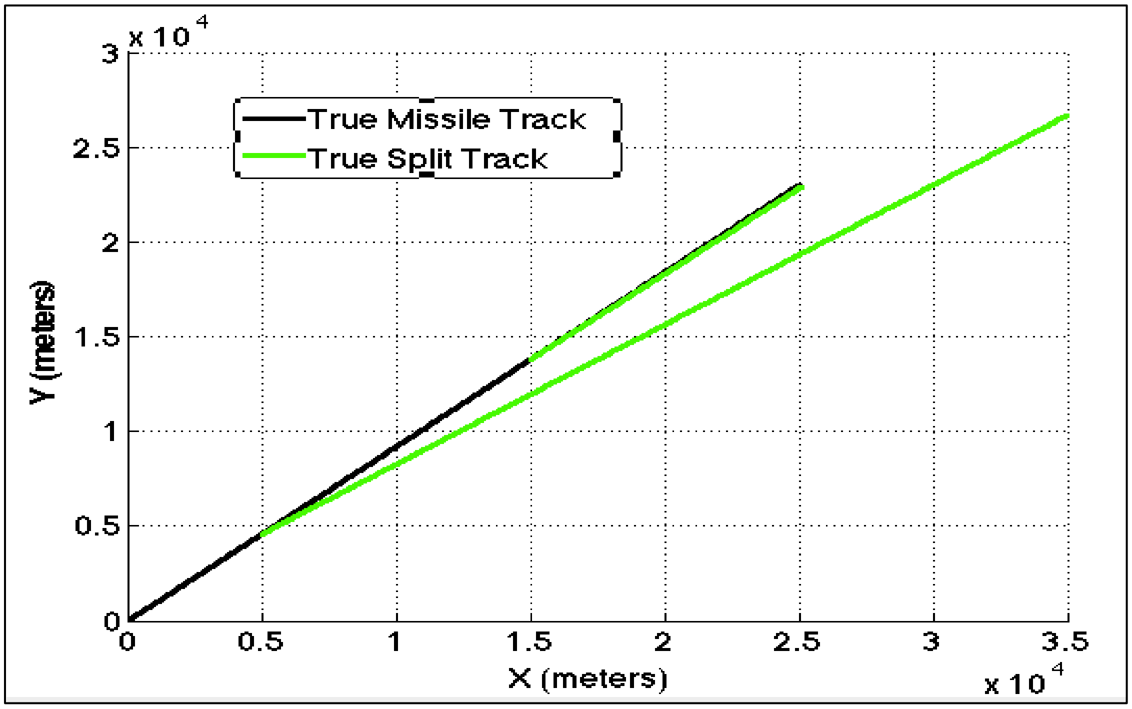

Figure 28. Scenario one for comparison between the Kalman and IMMJPDA filters.

The second scenario contains the most likely method of decoy deployment and missile maneuver. The split occurs at $k = 200$, followed by a turn from $k = 205$ to $k = 500$. The second decoy is deployed at $k = 600$ and followed by a turn from $k = 605$ to $k = 1000$. The true target and split vectors are shown in Figure 29.

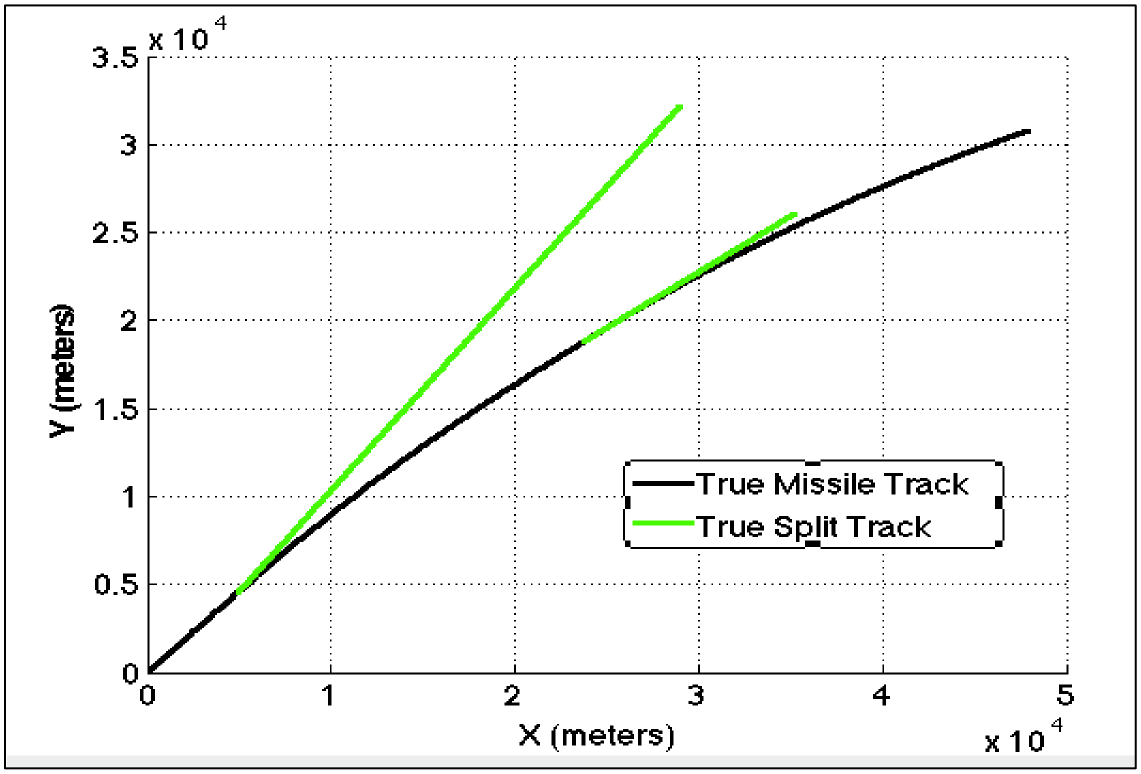

Figure 29. Scenario two for comparison between the Kalman and IMMJPDA filters.

Although a constant missile maneuver is detrimental to fuel consumption and increases vulnerability to tracking systems, the third scenario is included for a complete analysis of each target state. The missile deploys decoys and maintains constant velocity throughout the simulation while maneuvering, as shown in Figure 30.

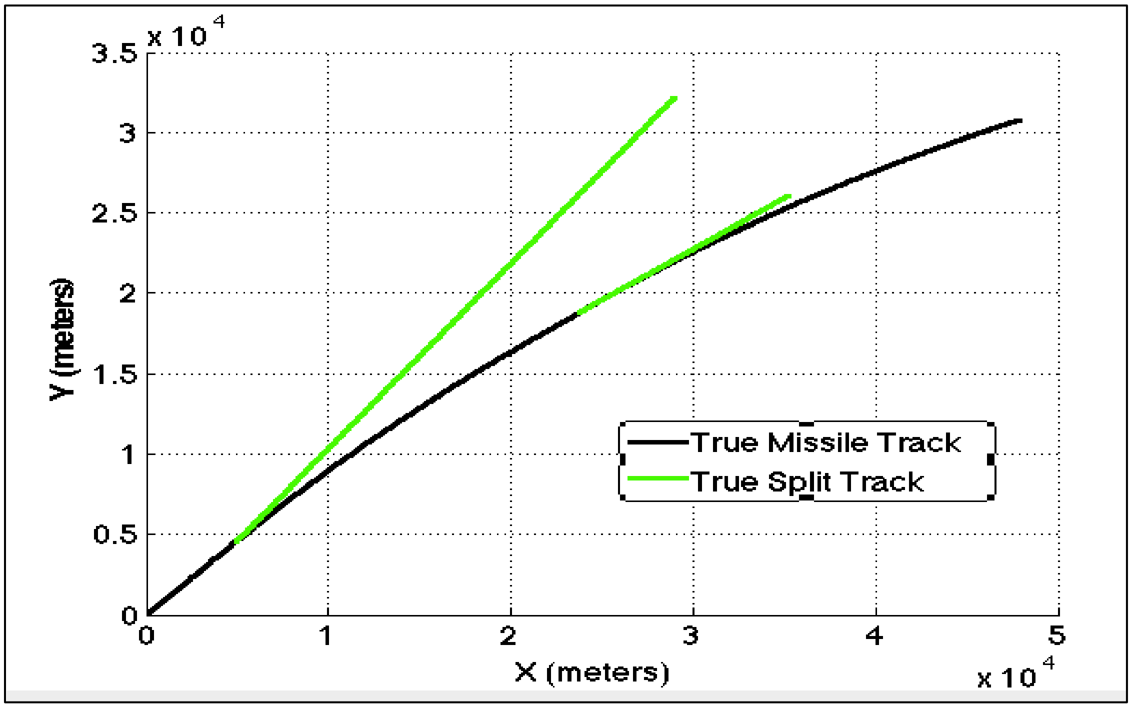

Figure 30. Scenario three for comparison between the Kalman and IMMJPDA filters.

The fourth scenario tests the feasibility of tracking a decoy during the transition from SLM to constant speed acceleration. The first split occurs 10 samples following the commencement of a turn. The ten-sample delay (instead of a 25-sample delay) was chosen to prevent an advantage to either filter and was chosen based on the results of Figure 26. The turn is maintained for 200 samples before returning to SLM. The second split is deployed under the same conditions but with significantly less divergence from the missile. The true target vectors can be seen in Figure 31.

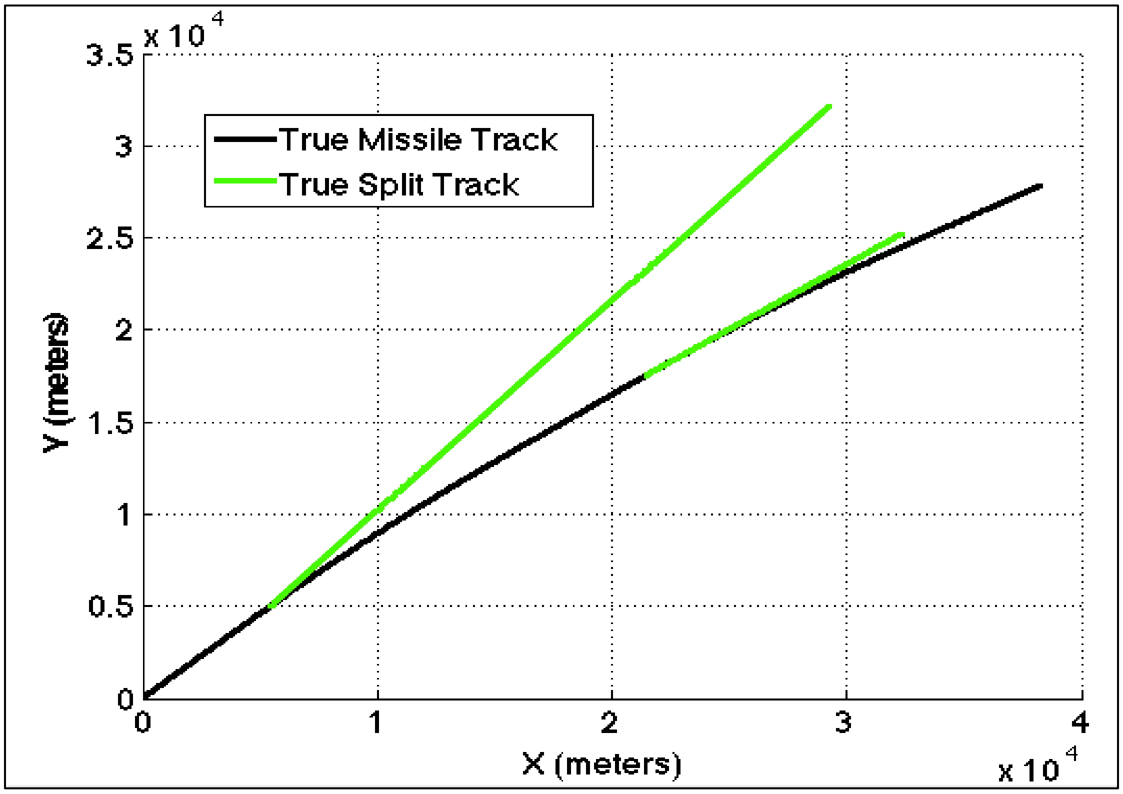

Figure 31. Scenario four for comparison between the Kalman and IMMJPDA filters.

Similar to the previous conditions, the fifth scenario analyzes the ability to track a decoy through a transition from constant velocity acceleration to SLM. Distance error spikes occur during model transitions despite the direction as seen in Figure 21. The missile begins in a turn and transitions to straight-line motion followed by a split 10 samples later. The second split is under similar conditions, and both split locations can be seen in Figure 32.

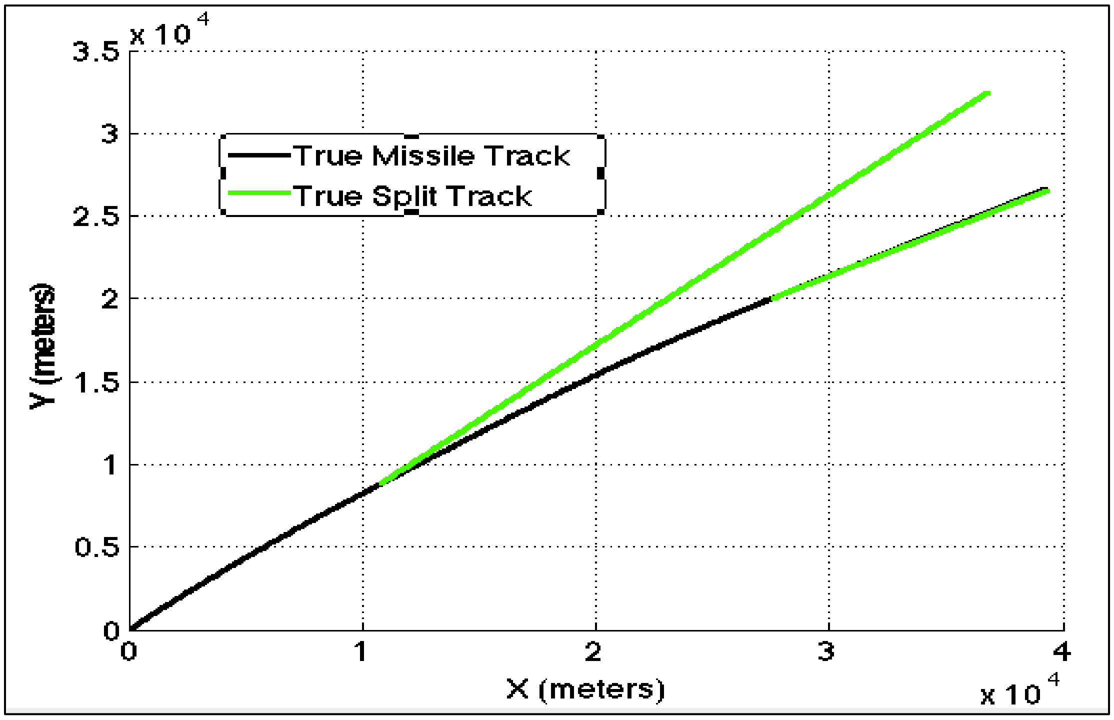

Figure 32. Scenario five for comparison between the Kalman and IMMJPDA filters.

The final scenario involves the missile maintaining SLM throughout and deploying both splits within five samples of each other. The first split is deployed from the left side of the missile at $k = 500$, and the second is ejected at $k = 505$ from the right side of the missile. The vicinity of both splits to the true target is identical to the second split in scenario one. No visual plot is included for this scenario, but the results are included in Table 2. The results for this scenario are expressed as a single event and are not differentiated between the two splits.

3. Results

The IMMJPDA filter displayed improved missile tracking performance and decoy avoidance over the tuned standard Kalman filter during the specified scenarios. The results between the two splits of the first scenario show the benefits of the JPDA filter over a simple nearest neighbor algorithm.

The second and fourth scenarios validate the additional computations required of the IMMJPDAF over the standard Kalman filter. We expected a significant drop in the ability to maintain true target track given the large distance error during model switching in comparison to sensor error. The successful tracking of the true target by the IMMJPDAF in scenario two and scenario four discredit the concerns about IMMJPDAF degradation between model transitions. The number of faulty measurement associations in both scenarios highlights a need for model improvements. However, these results prove that the combination of the IMM and the JPDA can compensate for simple target models.

The ability to track a decoy deployment through a turn is proven in scenario three. A repeating trend throughout each simulation is the effect of the non-acting model estimate upon the existing model. The increased number of faulty samples over the standard Kalman filter is explained by the model one estimates that tend to associate more with the split target in this scenario. The flawless success percentage proves adequate model transition probabilities to allow the filter to self-correct with the model two estimates.

Despite scenario five being a non-traditional decoy deployment strategy, the IMMJPDAF performed as expected and similarly to scenarios two and four. The results of these conditions show consistent performance through model transitions.

The results of the final scenario suggest the IMMJPDAF may be suitable for ICBM tracking in a high decoy density scenario. The decoys maintained essentially a constant track (each velocity component was offset by merely 0.01 meters per second in each direction for both decoys) to the missile, yet tracking accuracy was maintained. The combination of target track gates and hypothesis scoring yielded significant improvements, especially when compared to the standard Kalman filter.

Table 2. Performance summary of the Kalman filter and the IMMJPDAF under simulated ICBM decoy deployment and maneuver scenarios. Refer to section two (Scenario Setup) for an explanation of table organization.

Scenario	Split Number	Kalman Filter		IMMJPDA Filter	
		Success (%)	Faulty Measurement Associations Per Successful Simulation	Success (%)	Faulty Measurement Associations Per Successful Simulation
1	1	100	0.25	100	0.14
	2	71	1.3	100	0.35
2	1	0	■	100	254.6
	2	0	■	100	22.1
3	1	100	0.04	100	0.21
	2	100	0	100	0.16
4	1	0	■	100	23.1
	2	0	■	100	147.8
5	1	0	■	100	122.1
	2	0	■	100	176.4
6	1	59	0.1	100	0.71
	2				

V. CONCLUSIONS

A. SUMMARY OF RESULTS

1. IMM

With regard to ICBM tracking, the IMM proved to be capable during both steady-state and maneuvering conditions. For the given conditions, the estimates resulting from the IMM mixing process received a reduction from measurement error of 50 percent on average. The degradation of the individual model estimates was minimal. A significant amount of iterations was required to find optimal values for transition probabilities $\boldsymbol{p_{ij}}$. Even with the determined values for $\boldsymbol{p_{ij}}$, the model probability $\boldsymbol{u_i}$transitions, especially during a constant velocity maneuver, were frequent and often occurred between each sample. For targets with the likelihood of transitioning between established models often, this method is appropriate. However, for the purposes of an ICBM that prefers deploy decoys to conducting maneuvers, the necessity to account for continuous contact maneuver is reduced. In addition, the distance errors seen during model transitions are concerning due to the likelihood of decoy deployment during those periods. The IMMJPDA filter performed well during these transitions for the scenarios tested but could become a concern under higher split target densities.

With regard to split target tracking, the IMM provided a convenient framework in which to generate a split target and allowed for a seamless transition for follow-on track maintenance. We introduced a non-traditional IMM approach where the two models regarding the split target are not subject the standard IMM processes. Initially stacking the combined estimates to generate the split track allowed the common history of the target and decoy to be exploited. The decoy track generation caused an increase in distance error of the combined estimates of over two and a half times measurement error. The decoupling of model three and model four with the target models prevented further degradation of the combined estimates. Additionally, the linearly increasing model three errors showed that decoupling of the split target and true target sooner was best.

Prolonged tracking of the split target was detrimental to both the combined estimates and the split track maintenance.

2. JPDA

The JPDA was able to discriminate between true target and decoy measurements that fell within established track gates with minimal faulty associations. Although dependent on the quality of estimates provided and difficult to evaluate on its own, the JPDA's comparison to the nearest neighbor algorithm implemented by the Kalman filter proved very effective. The generation of hypothesis is simplistic for the scenarios presented, but computational requirements drastically increase with additional measurements.

B. RECOMMENDATIONS FOR FUTURE WORK

1. Probability of Detection and False Returns

The assumption that the probability of detection P_D is one requires a sensor that produces a valid measurement for each target, including all decoys, at every measurement time. The assumption on no false alarms (e.g., clutter or spurious measurements) reduces the number of hypotheses that the JPDA filter must consider. Measurements may be included that do not belong to any track, or an established track may not receive an associated measurement at any given time interval. The expansion of the JPDA filter equations to include these parameters is discussed in [3]. This would result in a more complicated but potentially more flexible JPDA filter and is worth future exploration.

2. Model Modifications

The established models to describe the ICBM dynamics do not fully describe a body traveling in exo-atmospheric conditions. An inclusion of the gravitational forces on the missile on the z-dimension, rotating Earth effects [2], and the associated filter performance should be explored.

As decoys become more sophisticated, the expansion of a decoy model beyond mimicking the trajectory of the missile may be necessary. Application of a separate IMM process, complete with its own set of models, for a split target may be appropriate for splits determined to be a reentry vehicles.

3. Algorithm Flexibility

Given the existing MATLAB code as a guide, expansion to include any number of splits would be essential for further simulation testing. As is, the algorithms can support zero, one, or two splits. Any additional splits written in the same manner will require an increasing amount of code for each one. The additional hypothesis required for the JPDA and the gate-related calculations contribute to the majority of the required code expansion.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. MATLAB CODE

A. MASTER SCRIPT FILE INCLUDING ALL INITIAZATIONS, ORDER OF FUNCTION FILE CALLS, AND GENERATION OF ALL PLOTS

```
close all;
clear all;
clc;

delta=0.01;
nsamples=1000;
nloops=100;

%Turn start and stop times
t_start1=300;
t_stop1= 700;
t_start2=1400;
t_stop2=1600;

%Split locations
n=[1200;1503];

md1=[];
md2=[];
mdo=[];
md3=[];
md4a=[];
md4b=[];
mdk=[];

for ii=1:nloops

   %Measurement matrix
H=[1 0 0 0 0 0;
   0 0 1 0 0 0;
   0 0 0 0 1 0];

%Sensor Error
se=5;
sigmax = se;
sigmay = se;
sigmaz = se;
R=diag([sigmax^2;sigmay^2;sigmaz^2]);

%Target state transition matrix
F1 = [1 delta  0    0    0 0;
      0   1    0    0    0 0;
      0   0    1  delta  0 0;
      0   0    0    1    0 0;
      0   0    0    0    1 0;
      0   0    0    0    0 1];



% %Initialize the target vectors
x =     [     0;                          %x position
           2500;                        %x velocity in m/s
              0;                          %y position
           2300;                        %y velocity in m/s
           10000;                      %z position
             0 ];                         %z velocity in m/s

P1 = 10^4.*    [1 0 0 0 0 0;        %Covariance matrix
                0 1 0 0 0 0;
                0 0 1 0 0 0;
                0 0 0 1 0 0;
                0 0 0 0 1 0;
                0 0 0 0 0 1];
 q1=10;
Q1 = q1 .*  [(delta^3)/3  (delta^2)/2       0           0           0
0;
             (delta^2)/2    delta           0           0           0
0;
             0                  0       (delta^3)/3 (delta^2)/2     0
0;
             0                  0       (delta^2)/2   delta         0
0;
             0                  0           0           0
(delta^3)/3 (delta^2)/2;
             0                  0           0           0
(delta^2)/2   delta];

% Q1= [0  0       0   0   0   0;
%      0  q12     0   0   0   0;
%      0  0       0   0   0   0;
%      0  0       0   q12 0   0;
%      0  0       0   0   0   0;
%      0  0       0   0   0   q12];

Qk=200*Q1;



spd= 3397.06;     %in m/s
g=-20;          % # of g's
omega=g*9.80665/spd;                    %turn rate in radians/sec

F2= [1  sin(omega*delta)/omega       0   (1-cos(omega*delta))/omega  0
```

```
0;
     0  cos(omega*delta)             0    -sin(omega*delta)          0
0;
     0  (1-cos(omega*delta))/omega   1   sin(omega*delta)/omega      0
0;
     0  sin(omega*delta)             0   cos(omega*delta)            0
0;
     0       0                       0        0                      1
0;
     0       0                       0        0                      0
1];


P2 = 10^4.*    [1 0 0 0 0 0;        %Covariance vector
                0 1 0 0 0 0;
                0 0 1 0 0 0;
                0 0 0 1 0 0;
                0 0 0 0 1 0;
                0 0 0 0 0 1];
q2=4000;
Q2 = q2 .* [(delta^3)/3  (delta^2)/2       0           0           0
0;
             (delta^2)/2    delta           0           0           0
0;
             0                  0       (delta^3)/3 (delta^2)/2     0
0;
             0                  0       (delta^2)/2   delta         0
0;
             0                  0           0           0           1
0;
             0                  0           0           0           0
1];
F3 = [1   delta       0    0           0 0;
      0   1           0    0           0 0;
      0   0           1    delta       0 0;
      0   0           0    1           0 0;
      0   0           0    0           1 0;
      0   0           0    0           0 1];

Q3 = q1 .* [(delta^3)/3  (delta^2)/2       0           0           0
0;
             (delta^2)/2    delta           0           0           0
0;
             0                  0       (delta^3)/3 (delta^2)/2     0
0;
             0                  0       (delta^2)/2   delta         0
0;
             0                  0           0           0           1
0;
             0                  0           0           0           0
1];
```

```
P3 = 10^4.*    [1 0 0 0 0 0;        %Covariance vector
                0 1 0 0 0 0;
                0 0 1 0 0 0;
                0 0 0 1 0 0;
                0 0 0 0 1 0;
                0 0 0 0 0 1];

K1 = P1*H'*inv(H*P1*H' + R);
K3 = P3*H'*inv(H*P3*H' + R);
Ks = [K1          zeros(6,3);
      zeros(6,3)         K3];



F4 = [1   delta  0    0       0 0;
      0   1      0    0       0 0;
      0   0      1    delta   0 0;
      0   0      0    1       0 0;
      0   0      0    0       1 0;
      0   0      0    0       0 1];

Q4 = q1 .* [(delta^3)/3  (delta^2)/2       0           0           0
0;
             (delta^2)/2    delta           0           0           0
0;
             0                  0       (delta^3)/3 (delta^2)/2     0
0;
             0                  0       (delta^2)/2   delta         0
0;
             0                  0           0           0           1
0;
             0                  0           0           0           0
1];

P4 = 10^4.*    [1 0 0 0 0 0;        %Covariance vector
                0 1 0 0 0 0;
                0 0 1 0 0 0;
                0 0 0 1 0 0;
                0 0 0 0 1 0;
                0 0 0 0 0 1];

x1h=x;
x2h=x;
xk=x;
Pk=P1;
x4a=[0;0;0;0;0;0];
x4b=[0;0;0;0;0;0];
tcorrect=0;
xo=x;
```

```
Po=P1;

%Intial Likelihood for each state
u1 = 0.7; %More likely to remain in straight line motion than to turn
u2 = 0.3; %More likely to transition out of turn and into stright line
motion

%Probability of changing state
    p11 = 0.75;
    p12 = 0.25;             %Probability of changing state
    p21 = 0.45;
    p22 = 0.55;

    %Model 1 and Model 2 mixing
    c1b = p11*u1 + p21*u2;
    c2b = p12*u1 + p22*u2;

%Matrix Initialization
x1hm=[];
zn1=[];
em1=[];
x2hm=[];
zn2=[];
em2=[];
x3hm=[];
zn3=[];
em3=[];
xsh=[];
zn4=[];
x4hm=[];
em4a=[];
em4b=[];
xm=[];
x3m=[];
x4am=[];
x4bm=[];
xom=[];
znk=[];
xkm=[];
d1m=[];
d2m=[];
dkm=[];
dom=[];
d3m=[];
d4am=[];
d4bm=[];
znkm=[];
znc=zeros(6,1);
mitrick=[];
mktrick=[];
```

```
%samples in model 3 before transition to 4
t=1;



    for k=1:nsamples


        %Measurement matrix
        H=[1 0 0 0 0 0;
           0 0 1 0 0 0;
           0 0 0 0 1 0];

        %Sensor Error
        sigmax = se;
        sigmay = se;
        sigmaz = se;
        R=diag([sigmax^2;sigmay^2;sigmaz^2]);

        %Measuring Target Position
        ztrue = H*x;                %True position

        %Add sensor noise to the measurements and calculate measurement
error
        w=randn(3,1).*[sigmax; sigmay; sigmaz];
        znu = ztrue + w;
        em = sqrt(w'*w);
        zn1=[zn1, znu];
        em1=[em1,em];



        %Target motion step for split target



         %Generate the first split vector
         if k==n(1)
            x3 = x + [0; 0.2*x(2); 0; +0.5*x(4); 0; 0];
            x3h=xo;
            xsh=[xo;x3h];
            x3hm=[x3hm,x3h];
            x3m=[x3m,x3];

            Ko = Po*H'*inv(H*Po*H' + R);
            K3 = P3*H'*inv(H*P3*H' + R);
            Ks = [Ko          zeros(6,3);
                  zeros(6,3)         K3];
            Ps = [Po    Po;
                  Po    Po];
         end

         if k==n(2)
            x3 = x + [0; 0.1*x(2); 0; -0.1*x(4); 0; 0];
            x3h=xo;
            xsh=[xo;x3h];
            x3hm=[x3hm,x3h];
            x3m=[x3m,x3];
            P3 = 10^4.*     [1 0 0 0 0 0;        %Covariance vector
                             0 1 0 0 0 0;
                             0 0 1 0 0 0;
                             0 0 0 1 0 0;
                             0 0 0 0 1 0;
                             0 0 0 0 0 1];

                Ko = Po*H'*inv(H*Po*H' + R);
                K3 = P3*H'*inv(H*P3*H' + R);
                Ks = [Ko          zeros(6,3);
                      zeros(6,3)         K3];

                Ps = [Po    Po;
                      Po    Po];

         end

         %Generate true split motion for x3
        if k > n(1) && k<=n(1)+t
            x3=F3*x3;
            x3m=[x3m,x3];
        end
        if k > n(2) && k<=n(2)+t
            x3=F3*x3;
            x3m=[x3m,x3];

        end


        %Generate true target position for x4, two splits
        if k==n(1)+t
            x4a=x3;
            x4=[x4a;x4b];
            x4am=[x4am,x4a];


        elseif k>=n(1)+t
            x4a=F4*x4a;
```

```
            x4=[x4a;x4b];
            x4am=[x4am,x4a];
        end

        if k==n(2)+t
            x4b=x3;
            x4=[x4a;x4b];
            x4bm=[x4bm,x4b];

        elseif k>n(2)+t
            x4b=F4*x4b;
            x4=[x4a;x4b];
            x4bm=[x4bm,x4b];
        end


        if k>n(1)+t
             [ x4h,P4,znu,znc,d4a,d4b,itrick ] = modelfour(
delta,x4h,x4,P4,Q4,x1h,H,R,se,znc,znu,P1,P3 );
             zn4=[zn4,znc];
             x4hm=[x4hm,x4h];
             d4am=[d4am,d4a];
             d4bm=[d4bm,d4b];
             mitrick=[mitrick,itrick];

        else

             x4=zeros(12,1);
             x4h=zeros(12,1);

             [ x4h,P4,znu,znc,d4a,d4b,itrick ] = modelfour(
delta,x4h,x4,P4,Q4,x1h,H,R,se,znc,znu,P1,P3 );
             zn4=[zn4,znc];
             x4hm=[x4hm,x4h];
             d4am=[d4am,d4a];
             d4bm=[d4bm,d4b];
             mitrick=[mitrick,itrick];


         end

         if k >n(1) && k<=n(1)+t
            [P1,P3,Ps,Ks,zns,x1h,x3h,d1,d3,itrick] =
modelthree(delta,x,x3,xsh,Q3,Ps,H,R,F3,Ks,P1,se,znu);
            mitrick=[mitrick,itrick];
            zn3=[zn3,zns];
            x3hm=[x3hm,x3h];
            d1m=[d1m,d1];
            d3m=[d3m,d3];
            [x2h,P2,d2] = modeltwo(x2h,x,P2,znu,H,R);
            d2m=[d2m,d2];

            znk=[znu;zns;zeros(3,1)];
            znkm=[znkm,znk];
            [xk,Pk,dk,ktrick] = kalman( delta,xk,x,Pk,znk,Qk,se);
            mktrick=[mktrick,ktrick];
            xkm=[xkm,xk];
            dkm=[dkm,dk];



[x01h,x02h,P01,P02,u1,u2,xo,Po,do,c1b,c2b]=modelprob(x1h,x2h,P1,P2,u1,u
2,znu,H,R,x,c1b,c2b);
            xom=[xom,xo];
            dom=[dom,do];


[x1h,x2h,P1,P2]=prediction(x01h,x02h,P01,P02,F1,F2,Q1,Q2,u1,u2);
            x1hm=[x1hm,x1h];
            x2hm=[x2hm,x2h];

         elseif k >n(2) && k<=n(2)+t

            [P1,P3,Ps,Ks,zns,x1h,x3h,d1,d3,itrick] =
modelthree(delta,x,x3,xsh,Q3,Ps,H,R,F3,Ks,P1,se,znu);
            mitrick=[mitrick,itrick];

            zn3=[zn3,zns];
            x3hm=[x3hm,x3h];
            d1m=[d1m,d1];
            d3m=[d3m,d3];

            [x2h,P2,d2] = modeltwo(x2h,x,P2,znu,H,R);
            d2m=[d2m,d2];

            znk=[znu;znc];
            znkm=[znkm,znk];

            [xk,Pk,dk,ktrick] = kalman( delta,xk,x,Pk,znk,Qk,se);
            mktrick=[mktrick,ktrick];
            xkm=[xkm,xk];
            dkm=[dkm,dk];



[x01h,x02h,P01,P02,u1,u2,xo,Po,do,c1b,c2b]=modelprob(x1h,x2h,P1,P2,u1,u
2,znu,H,R,x,c1b,c2b);
```

```
            xom=[xom,xo];
            dom=[dom,do];


[x1h,x2h,P1,P2]=prediction(x01h,x02h,P01,P02,F1,F2,Q1,Q2,u1,u2);
            x1hm=[x1hm,x1h];
            x2hm=[x2hm,x2h];

         else
           [ x1h,P1,d1,R] = modelone( x1h,x,P1,H,R,znu);
            d1m=[d1m,d1];

            [x2h,P2,d2] = modeltwo(x2h,x,P2,znu,H,R);
            d2m=[d2m,d2];

            if k<n(1)
                znk=[znu;zeros(6,1)];
            else
                znk=[znu;znc];
            end

            znkm=[znkm,znk];
            [xk,Pk,dk,ktrick] = kalman( delta,xk,x,Pk,znk,Qk,se);
            mktrick=[mktrick,ktrick];
            xkm=[xkm,xk];
            dkm=[dkm,dk];


[x01h,x02h,P01,P02,u1,u2,xo,Po,do,c1b,c2b]=modelprob(x1h,x2h,P1,P2,u1,u
2,znu,H,R,x,c1b,c2b);
            xom=[xom,xo];
            dom=[dom,do];


[x1h,x2h,P1,P2]=prediction(x01h,x02h,P01,P02,F1,F2,Q1,Q2,u1,u2);
            x1hm=[x1hm,x1h];
            x2hm=[x2hm,x2h];

         end



         if k==n(1)+t

            x4(1:6,1)=x3;
            x4(7:12,1)=x4b;
            %x4h(1:6,1)=x3h;
            %x4h(7:12,1)=zeros(6,1);
            x4h=[x3h;zeros(6,1)];
            x4hm=[x4hm,x4h];
            znc=zeros(6,1);

         elseif k==n(2)+t

            x4(7:12,1)=x3;
            x4h(7:12,1)=x3h;
            x4hm=[x4hm,x4h];

         end

          %Target Motion Step for Straight line motion
         x=F1*x;

        %Target motion step for constant acceleration
        if k>t_start1 && k<t_stop1
            x=F2*x;
        elseif k>t_start2 && k<t_stop2
            x=F2*x;
        end

        %Collect true target vectors
        xm=[xm,x];

    end


    if dkm(size(dkm,2))>600
        ttk=1;
    else
        ttk=0;
    end

    if dom(size(dom,2))>600
        tto=1;
    else
        tto=0;
    end


%Keep a running total of distance errors
if ii==1
    md1=d1m;
    md2=d2m;
    mdo=dom;
    md3=d3m;
    md4a=d4am;
    md4b=d4bm;
    mdk=dkm;
    mem1=em1;
    txom=xom;
    tzn1=zn1;
    txkm=xkm;
    tzn3=zn3;
    tx3hm=x3hm;
    tzn4=zn4;
    tx4hm=x4hm;
    txkm=xkm;
    tottk=ttk;
    totto=tto;
    sitrick=sum(mitrick);
    sktrick=sum(mktrick);
else
    md1=md1+d1m;
    md2=md2+d2m;
    mdo=mdo+dom;
    md3=md3+d3m;
    md4a=md4a+d4am;
    md4b=md4b+d4bm;
    mdk=mdk+dkm;
    mem1=mem1+em1;
    txom=txom+xom;
    tzn1=tzn1+zn1;
    txkm=txkm+xkm;
    tzn3=tzn3+zn3;
    tx3hm=tx3hm+x3hm;
    tzn4=tzn4+zn4;
    tx4hm=tx4hm+x4hm;
    txkm=txkm+xkm;
    tottk=tottk+ttk
    sktrick=sktrick+sum(mktrick)
    totto=totto+tto
    sitrick=sitrick+sum(mitrick)

end


end

%Calculate average distance errors
md1=md1/nloops;
md2=md2/nloops;
mdo=mdo/nloops;
md3=md3/nloops;
md4a=md4a/nloops;
md4b=md4b/nloops;
mdk=mdk/nloops;
mem1=mem1/nloops;
mxom=txom/nloops;
```

```
mzn1=tzn1/nloops;
mxkm=txkm/nloops;
mzn3=tzn3/nloops;
mx3hm=tx3hm/nloops;
mzn4=tzn4/nloops;
mx4hm=tx4hm/nloops;
mxkm=txkm/nloops;
titrick=sitrick/nloops;
tktrick=sktrick/nloops;



figure;
hold all;
plot(mdo,'g');
plot(mdk,'r');
plot(mem1,'b');
title('Combined(Model 1/2) vs Kalman Estimate Mean Error');
set(gcf,'color','w');
legend ('Combined Estimates','Kalman Estimates','Measurement Error');
grid on;
set( findobj(gca,'type','line'), 'LineWidth', 1.5);
xlabel('Sample (k)','fontsize',18,'fontweight','bold');
ylabel('Mean Distance Error
(meters)','fontsize',18,'fontweight','bold');
set( findobj(gca,'type','line'), 'LineWidth', 1.5);



figure;
title('IMM and JDPA')
hold all;
plot(xm(1,:),xm(3,:),'b');
plot(xkm(1,:),xkm(3,:),'r');
plot(xom(1,:),xom(3,:),'m');

First split
plot(zn3(1,1:size(zn3,2)/2),zn3(2,1:size(zn3,2)/2),'g');
plot(x3m(1,1:size(x3m,2)/2),x3m(3,1:size(x3m,2)/2),'g');
plot(x3hm(1,1:size(x3hm,2)/2),x3hm(3,1:size(x3hm,2)/2),'r');
plot(mzn4(1,1:size(mzn4,2)),zn4(2,1:size(zn4,2)),'r');
plot(x4am(1,1:size(x4am,2)),x4am(3,1:size(x4am,2)),'g');
plot(x4hm(1,1:size(x4hm,2)),x4hm(3,1:size(x4hm,2)),'r');

Second split
plot(zn3(1,size(zn3,2)/2+1:size(zn3,2)),zn3(2,size(zn3,2)/2+1:size(zn3,
2)),'g');
plot(x3m(1,size(x3m,2)/2+1:size(x3m,2)),x3m(3,size(x3m,2)/2+1:size(x3m,
2)),'g')
plot(x3hm(1,size(x3hm,2)/2+1:size(x3hm,2)),x3hm(3,size(x3hm,2)/2+1:size
(x3hm,2)),'r');
plot(mzn4(4,n(2)-n(1)+1:size(mzn4,2)),zn4(5,n(2)-
n(1)+1:size(mzn4,2)),'r');
plot(x4bm(1,1:size(x4bm,2)),x4bm(3,1:size(x4bm,2)),'g');
plot(x4hm(7,n(2)-n(1)+1:size(x4hm,2)),x4hm(9,n(2)-
n(1)+1:size(x4hm,2)),'r');

set(gcf,'color','w');
legend ('True Missile Track','True Split Track','Split Estimates');

grid on;
set( findobj(gca,'type','line'), 'LineWidth', 1.5);
xlabel('X (meters)','fontsize',18,'fontweight','bold');
ylabel('Y (meters)','fontsize',18,'fontweight','bold');



figure;
hold all;
title('Attempt to Trick Kalman')
plot(xm(1,:),xm(3,:),'k');
plot(xom(1,:),xom(3,:),'b');
plot(xkm(1,:),xkm(3,:),'r');
plot(x3m(1,1:size(x3m,2)/2),x3m(3,1:size(x3m,2)/2),'g');
plot(x4am(1,1:size(x4am,2)),x4am(3,1:size(x4am,2)),'g');
plot(x3m(1,size(x3m,2)/2+1:size(x3m,2)),x3m(3,size(x3m,2)/2+1:size(x3m,
2)),'g');
plot(x4bm(1,1:size(x4bm,2)),x4bm(3,1:size(x4bm,2)),'g');
legend ('True Missile Track','True Split Track');
set(gcf,'color','w');
grid on;
set( findobj(gca,'type','line'), 'LineWidth', 1.5);
xlabel('X (meters)','fontsize',18,'fontweight','bold');
ylabel('Y (meters)','fontsize',18,'fontweight','bold');

figure;
hold all;
plot(md3(1:size(md3,2)/2),'r');
plot(md3(size(md3,2)/2+1:size(md3,2)),'b');
set(gcf,'color','w');
legend ('First Split','Second Split');
grid on;
set( findobj(gca,'type','line'), 'LineWidth', 1.5);
xlabel('Sample (k)','fontsize',18,'fontweight','bold');
ylabel('Mean Distance Error
(meters)','fontsize',18,'fontweight','bold');

figure;
hold all;
plot([zeros(1,100),md4a],'r');
plot([zeros(1,800),md4b(n(2)-n(1)+1:size(md4a,2))],'b');
plot(mem1,'k');
set(gcf,'color','w');
```

```
legend ('First Split','Second Split','Measurement Error');
grid on;
set( findobj(gca,'type','line'), 'LineWidth', 1.5);
xlabel('Sample (k)','fontsize',18,'fontweight','bold');
ylabel('Mean Distance Error
(meters)','fontsize',18,'fontweight','bold');

figure;
hold all;
plot(mdo,'m');
plot(mem1,'k');

set(gcf,'color','w');
legend ('Combined Estimates','Measurement Error');
grid on;
set( findobj(gca,'type','line'), 'LineWidth', 1.5);
xlabel('Sample (k)','fontsize',18,'fontweight','bold');
ylabel('Mean Distance Error
(meters)','fontsize',18,'fontweight','bold');
set( findobj(gca,'type','line'), 'LineWidth', 1.5);
```

B. MODEL ONE FUNCTION FILE

```
function [ x1h,P1,d1,R] = modelone( x1h,x,P1,H,R,znu)

K1 = P1*H'*inv(H*P1*H' + R);
x1h = x1h + K1*(znu - H*x1h);

d1=norm(H*(x1h-x));

%Covariance Update
n=max(size(x1h));
K11 = (eye(n) - K1*H);
P1 = K11*P1*K11' + K1*R*K1';

end
```

C. MODEL TWO FUNCTION FILE

```
function [x2h,P2,d2] = modeltwo(x2h,x,P2,znu,H,R)

    K2 = P2*H'*inv(H*P2*H' + R);
    x2h = x2h + K2*(znu - H*x2h);

    d2=norm(H*(x2h-x));

    %Covariance Update
    n=max(size(x2h));
    K22 = (eye(n) - K2*H);
    P2 = K22*P2*K22' + K2*R*K2';

end
```

D. MODEL THREE FUNCTION FILE

```
function [P1,P3,Ps,Ks,zns,x1h,x3h,d1,d3,itrick] =
modelthree(delta,x,x3,xsh,Q3,Ps,H,R,F3,Ks,P1,se,znu);

    sigmax=se;
    sigmay=se;
    sigmaz=se;

    zstrue = H*x3;
    t=randn(3,1).*[sigmax; sigmay; sigmaz];
    zns = zstrue + t;
    esm = sqrt(t'*t);

    %Generate stacked vectors and matrices
    Hs = [H          zeros(3,6);
          zeros(3,6)         H];

    Rs = [R zeros(3,3);zeros(3,3) R];

    Fs = [F3           zeros(6,6);
          zeros(6,6)           F3];

    Qs = [Q3           zeros(6,6);
          zeros(6,6)          Q3];



     %Model 3 Target assignment
    z1=[znu;zns];
    z2=[zns;znu];

    z1t = z1 - Hs*xsh;
    z2t = z2 - Hs*xsh;

    S3 = Hs*Ps*Hs' + Rs;

    n1=(z1t)'*inv(S3)*z1t;
    n2=(z2t)'*inv(S3)*z2t;

    omega1 = (exp(-(n1/2)))/(det(2*pi*S3)^(1/2));
    omega2 = (exp(-(n2/2)))/(det(2*pi*S3)^(1/2));

    %Use omega values to associate target with measurement
    if omega1>=omega2
        zs=z1;
        itrick=0;
    else
       zs=z2;
       itrick=1;
    end

    %Update
    Ks = Ps*Hs'*inv(Hs*Ps*Hs' + Rs);
    xsh = xsh + Ks*(zs - Hs*xsh);

    %Convariance Update
    m=max(size(xsh));
    Kss = (eye(m) - Ks*Hs);
    Ps = Kss*Ps*Kss' + Ks*Rs*Ks';

    x1h=xsh(1:6);
    x3h=xsh(7:12);
    P1 = Ps(1:6,1:6);
    P3 = Ps(7:12,7:12);

    d1 = norm(H*(x1h-x));
    d3 = norm(H*(x3h-x3));

    xsh = Fs*xsh;
    Ps = Fs*Ps*Fs' + Qs;

end
```

E. MODEL FOUR FUNCTION FILE WITH COMPLETE JDPA FILTER ANALYSIS OF ALL AVAILABLE MEASUREMENTS

```
function [ x4h,P4,znu,znc,d4a,d4b,itrick ] = modelfour(
delta,x4h,x4,P4,Q4,x1h,H,R,se,znc,znu,P1,P3 )

%Target state transition matrix
    F4 = [1   delta  0    0       0 0;
          0   1      0    0       0 0;
          0   0      1    delta   0 0;
          0   0      0    1       0 0;
          0   0      0    0       1 0;
          0   0      0    0       0 1];
    %Measurement matrix
    H=[1 0 0 0 0 0;
       0 0 1 0 0 0;
```

```
   0 0 0 0 1 0];

sigmax=se;
sigmay=se;
sigmaz=se;
itrick=0;
x4a=x4(1:6);
zstrue = H*x4a;
t=randn(3,1).*[sigmax; sigmay; sigmaz];
zna = zstrue + t;

x4b=x4(7:12);
zstrue = H*x4b;
v=randn(3,1).*[sigmax; sigmay; sigmaz];
znb = zstrue + v;

znc=[zna;znb];

%JPDA

znk=[znu;znc];


if znk(7,1)>0             %Two split scenario
    z1=znk(1:3);
    z2=znk(4:6);
    z3=znk(7:9);

    x1=x1h;
    x2=x4(1:6);
    x3=x4(7:12);

    z11 = z1 - H*x1;
    z12 = z1 - H*x2;
    z13 = z1 - H*x3;
    z21 = z2 - H*x1;
    z22 = z2 - H*x2;
    z23 = z2 - H*x3;
    z31 = z3 - H*x1;
    z32 = z3 - H*x2;
    z33 = z3 - H*x3;

    S1 = H*P1*H' + R;
    S2 = H*P3*H' + R;
    S3 = H*P3*H' + R;

   g11 = (exp(-(z11)'*inv(S1)*z11/2))/(det(2*pi*S1)^(1/2));
   g12 = (exp(-(z12)'*inv(S1)*z12/2))/(det(2*pi*S1)^(1/2));
   g13 = (exp(-(z13)'*inv(S1)*z13/2))/(det(2*pi*S1)^(1/2));
```

```
g21 = (exp(-(z21)'*inv(S2)*z11/2))/(det(2*pi*S2)^(1/2));
g22 = (exp(-(z22)'*inv(S2)*z22/2))/(det(2*pi*S2)^(1/2));
g23 = (exp(-(z23)'*inv(S2)*z23/2))/(det(2*pi*S2)^(1/2));
g31 = (exp(-(z31)'*inv(S3)*z31/2))/(det(2*pi*S3)^(1/2));
g32 = (exp(-(z32)'*inv(S3)*z32/2))/(det(2*pi*S3)^(1/2));
g33 = (exp(-(z33)'*inv(S3)*z33/2))/(det(2*pi*S3)^(1/2));

gnorm = g11 + g12 + g13 + g21 + g22 + g23 + g31 + g32 + g33;

h1 = (g11*g22*g33)/gnorm;
h2 = (g11*g23*g32)/gnorm;
h3 = (g13*g22*g31)/gnorm;
h4 = (g12*g21*g33)/gnorm;
h5 = (g31*g12*g23)/gnorm;
h6 = (g12*g23*g31)/gnorm;

p11 = h1+h2;
p12 = h4+h6;
p13 = h3+h5;
p21 = h4+h5;
p22 = h1+h3;
p23 = h2+h6;
p31 = h3+h6;
p32 = h2+h5;
p33 = h1+h4;

if p11>p12 && p11>p13
    znu=znk(1:3);
    itrick=0;
end
if p12>p11 && p12>p13
    znu=znk(4:6);
    itrick=1;
end
if p13>p11 && p13>p12
    znu=znk(7:9);
    itrick=1;
end

if p21>p22 && p21>p23
    znc(1:3)=znk(1:3);
     itrick=1;
end
if p22>p11 && p22>p13
    znc(1:3)=znk(4:6);
     itrick=0;
end
if p23>p11 && p23>p12
    znc(1:3)=znk(7:9);
     itrick=1;
```

```
   end

   if p31>p12 && p31>p13
       znc(4:6)=znk(1:3);
        itrick=1;
   end
   if p32>p11 && p32>p13
       znc(4:6)=znk(4:6);
        itrick=1;
   end
   if p33>p11 && p33>p12
       znc(4:6)=znk(7:9);
        itrick=0;
   end

elseif znk(4,1)>0              %One split scenario

    z1=znk(1:3);
    z2=znk(4:6);

    x1=x1h;
    x2=x4(1:6);

    z11 = z1 - H*x1;
    z12 = z1 - H*x2;
    z21 = z2 - H*x1;
    z22 = z2 - H*x2;

    S1 = H*P1*H' + R;
    S2 = H*P3*H' + R;

   g11 = (exp(-(z11)'*inv(S1)*z11/2))/(det(2*pi*S1)^(1/2));
   g12 = (exp(-(z12)'*inv(S1)*z12/2))/(det(2*pi*S1)^(1/2));
   g21 = (exp(-(z21)'*inv(S2)*z21/2))/(det(2*pi*S2)^(1/2));
   g22 = (exp(-(z22)'*inv(S2)*z22/2))/(det(2*pi*S2)^(1/2));

   gnorm = g11 + g12 + g21 + g22;

   h1 = (g11*g22)/gnorm;
   h2 = (g21*g12)/gnorm;

   p11 = h1;
   p12 = h2;
   p21 = h2;
   p22 = h1;

   if p11>p12
       znu=znk(1:3);
       znc(1:3)=znk(4:6);
```

```
            znc(4:6)=zeros(3,1);
            itrick=0;
        else
            znc(1:3)=znk(1:3);
            znu=znk(4:6);
            znc(4:6)=zeros(3,1);
            itrick=1;
        end
    else
             znu=znk(1:3);
             znc=znk(4:9);
             itrick=0;
    end

    zna=znc(1:3);
    znb=znc(4:6);

if x4(7) > 0

    x4a=x4(1:6);
    x4ha=x4h(1:6);
    P4a=P4(1:6,1:6);

    %Drop split target estimates if > 50 km from missile estimates
    if norm(H*(x1h-x4ha)) < 80*10^3

%          zstrue = H*x4a;
%          t=randn(3,1).*[sigmax; sigmay; sigmaz];
%          zna = zstrue + t;

        %Update
        K4a = P4a*H'*inv(H*P4a*H' + R);
        x4ha = x4ha + K4a*(zna - H*x4ha);

        d4a=norm(H*(x4ha-x4(1:6)));

        %Covariance Update
        n=max(size(x4ha));
        K44a = (eye(n) - K4a*H);
        P4a = K44a*P4a*K44a' + K4a*R*K4a';

        %Prediction
        x4ha=F4*x4ha;
        P4a = F4*P4a*F4' + Q4;

    else
        P4a=P4a;
        x4ha=x4ha;
        d4a=norm(H*(x4ha-x4(7:12)));
    end


    x4b=x4(7:12);
    x4hb=x4h(7:12);
    P4b=P4(7:12,1:6);

   if norm(H*(x1h-x4hb)) < 80*10^3

%         zstrue = H*x4b;
%         v=randn(3,1).*[sigmax; sigmay; sigmaz];
%         znb = zstrue + v;
%
       %Update
        K4b = P4b*H'*inv(H*P4b*H' + R);
        x4hb = x4hb + K4b*(znb - H*x4hb);

        d4b=norm(H*(x4hb-x4(7:12)));

        %Covariance Update
        n=max(size(x4hb));
        K44b = (eye(n) - K4b*H);
        P4b = K44b*P4b*K44b' + K4b*R*K4b';

        %Prediction
        x4hb=F4*x4hb;
        P4b = F4*P4b*F4' + Q4;
    else
        P4b=P4b;
        x4hb=x4hb;
        d4b=norm(H*(x4hb-x4(7:12)));
    end

 elseif x4(1) > 0

    x4a=x4(1:6);
    x4ha=x4h(1:6);
    P4a=P4(1:6,1:6);
    P4b=zeros(6,6);
    x4hb=zeros(6,1);
    d4b=0;

    %Drop split target estimates if > 50 km from missile estimates
    if norm(H*(x1h-x4ha)) < 80*10^3

%          zstrue = H*x4a;
%          t=randn(3,1).*[sigmax; sigmay; sigmaz];
%          zna = zstrue + t;
```

```
        %Update
        K4a = P4a*H'*inv(H*P4a*H' + R);
        x4ha = x4ha + K4a*(zna - H*x4ha);

        d4a=norm(H*(x4ha-x4(1:6)));

        %Covariance Update
        n=max(size(x4ha));
        K44a = (eye(n) - K4a*H);
        P4a = K44a*P4a*K44a' + K4a*R*K4a';

        %Prediction
        x4ha=F4*x4ha;
        P4a = F4*P4a*F4' + Q4;
    else
       P4a=P4a;
       x4ha=x4ha;
       d4a=norm(H*(x4ha-x4(1:6)));
    end
else
    x4a=zeros(6,1);
    x4ha=zeros(6,1);
    zna=zeros(3,1);
    d4a=0;
    P4a=10^4.*    [1 0 0 0 0 0;        %Covariance vector
                   0 1 0 0 0 0;
                   0 0 1 0 0 0;
                   0 0 0 1 0 0;
                   0 0 0 0 1 0;
                   0 0 0 0 0 1];

    x4b=zeros(6,1);
    x4hb=zeros(6,1);
    znb=zeros(3,1);
    P4b=10^4.*    [1 0 0 0 0 0;        %Covariance vector
                   0 1 0 0 0 0;
                   0 0 1 0 0 0;
                   0 0 0 1 0 0;
                   0 0 0 0 1 0;
                   0 0 0 0 0 1];

    d4b=0;
end

    znc=[zna;znb];
    x4h=[x4ha;x4hb];
    P4=[P4a;P4b];

end
```

F. MODEL PROBABILITY FUNCTION FILE WITH IMM MIXING

```
function
[x01h,x02h,P01,P02,u1,u2,xo,Po,do,c1b,c2b]=modelprob(x1h,x2h,P1,P2,u1,u
2,znu,H,R,x,c1b,c2b)

    %Score the Association
    z1t = znu - H*x1h;
    z2t = znu - H*x2h;
    S1 = H*P1*H' + R;
    S2 = H*P2*H' + R;
    if sqrt(det(S1))<10^-5
        score1 = 10^-5;
    else
    score1 = (exp(-(z1t)'*inv(S1)*z1t/2))/(det(2*pi*S1)^(1/2));
    end
    if sqrt(det(S2))<10^-5
        score2 = 10^-5;
    else
        score2 = (exp(-(z2t)'*inv(S2)*z2t/2))/(det(2*pi*S2)^(1/2));
    end

    %Update Model 1 and 2 Probabilities
    c = score1*c1b + score2*c2b;
    u1 =  score1*(c1b/c);
    u2 =  score2*(c2b/c);

    %Combined Estimates
    xo = u1*x1h + u2*x2h;
    Po = u1*(P1+[x1h-xo]*[x1h-xo]') + u2*(P2+[x2h-xo]*[x2h-xo]');

    do=norm(H*(xo-x));

    %Probability of changing state
    p11 = 0.75;
    p12 = 0.25;
    p21 = 0.45;
    p22 = 0.55;

    %Model 1 and Model 2 mixing
    c1b = p11*u1 + p21*u2;
    c2b = p12*u1 + p22*u2;

    x01h = x1h*((p11*u1)/c1b) + x2h*((p21*u2)/c1b);
    x02h = x1h*((p12*u1)/c2b) + x2h*((p22*u2)/c2b);

    u11=(p11*u1)/c1b;
    u21=(p21*u2)/c1b;
    u12=(p12*u1)/c2b;
    u22=(p22*u2)/c2b;
    x11til=x1h-x01h;
    x21til=x2h-x01h;
    x12til=x1h-x02h;
    x22til=x2h-x02h;

    P01=u11*[P1+x11til*x11til']+u21*[P2+x21til*x21til'];
    P02=u12*[P1+x12til*x12til']+u22*[P2+x22til*x22til'];
    P01=P1;
    P02=P2;
end
```

G. PREDICTION FUNCTION FILE

```
function
[x1h,x2h,P1,P2]=prediction(x01h,x02h,P01,P02,F1,F2,Q1,Q2,u1,u2)

    %Predicted target motion after mixing of Model 1 and 2
    x1h = F1*x01h;
    x2h = F2*x02h;
    P1 = F1*P01*F1' + Q1;
    P2 = F2*P02*F2' + Q2;

end
```

H. STANDARD KALMAN FILTER FUNCTION FILE WITH A NEAREST NEIGHBOR ALGORITHM FOR MEASUREMENT ASSOCIATION

```
function [ xk, Pk,dk,ktrick] = kalman( delta,xk,x,Pk,znk,Qk,R)

%Target state transition matrix
F1 = [1 delta  0    0    0 0;
      0   1    0    0    0 0;
      0   0    1  delta  0 0;
      0   0    0    1    0 0;
      0   0    0    0    1 0;
      0   0    0    0    0 1];

%Measurement matrix
H=[1 0 0 0 0 0;
   0 0 1 0 0 0;
   0 0 0 0 1 0];

%Determine which measurement is closest to the Kalman estimate

d1=sqrt((znk(1)-xk(1))^2 + (znk(2)-xk(3))^2);
d2=sqrt((znk(4)-xk(1))^2 + (znk(5)-xk(3))^2);
d3=sqrt((znk(7)-xk(1))^2 + (znk(8)-xk(3))^2);
```

```
ktrick=0;

if abs(znk(4))>0 && abs(znk(7))>0

    if d2<d1 && d2<d3
      znc=znk(4:6);
      ktrick=1;
    elseif d3<d1 && d3<d2
      znc=znk(7:9);
      ktrick=1;
    else
        znc=znk(1:3);
        ktrick=0;
    end

elseif abs(znk(4))>0
    if d2<d1
      znc=znk(4:6);
      ktrick=1;
    else
      znc=znk(1:3);
      ktrick=0;
    end

else
    znc=znk(1:3);
    ktrick=0;
end

%Update
Kk = Pk*H'*inv(H*Pk*H' + R);
xk = xk + Kk*(znc - H*xk);

dk = norm(H*(xk-x));

%Covariance Update
n=max(size(xk));
Kkk = (eye(n) - Kk*H);
Pk = Kkk*Pk*Kkk' + Kk*R*Kk';

%Predicted Target Motion
xk = F1*xk;

Pk = F1*Pk*F1' + Qk;

end
```

LIST OF REFERENCES

[1] United States Department of Defense. (2013, August 18). *Missile Defense Agency*. [Online]. Available: htttp://www.mda.mil.

[2] National Research Council, *Making Sense of Ballistic Missile Defense: An Assessment of Concepts and Systems for U.S. Boost–Phase Missile Defense in Comparison to Other Alternatives*. Washington, DC: The National Academies Press, 2012.

[3] Y. Bar–Shalom et al., "Tracking of splitting targets in clutter using an interacting multiple model joint probabilistic data association filter," in *Proc. of the 30th IEEE Conf. on Decision and Control*, Storrs, CT, pp. 2043–2048, 1991.

[4] S. Blackman and R. Papoli, *Design and Analysis of Modern Tracking Systems*. Boston, MA: Artech House, 1999.

[5] S.B.Colegrove and S.J. Davey, "PDAF with multiple clutter regions and target models," in *IEEE Transactions on Aerospace and Electronic Systems,* vol. 39, no.1, pp. 110–124, Jan., 2003.

[6] S. B. Colegrove and S. J. Davey, "The probabilistic data association filter with multiple nonuniform clutter regions," in *IEEE Int. Radar Conf.*, Alexandria, VA, pp. 65–70, 2000.

[7] Cooperman, Robert. "Tactical ballistic missile tracking using the interacting multiple model algorithm," in *Proc. of the 5th Int. Conf. on Information Fusion,* Annapolis, MD, pp. 824–831, 2002.

[8] W.J. Farrell, "Interacting multiple model filter for tactical ballistic missile tracking," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no.2, pp. 418–426, Apr., 2008.

[9] J. Lu et al., "Simulation study for observation of space reentry vehicles based on UKF," in *2011 Int. Conf. on Network Computing and Information Security,* Guilin, China, pp. 410–414, 2011.

[10] D. Sigalov et al., "Tracking a splitting target in clutter using the IMM methodology," in *2012 IEEE 27th Conv. of Electrical and Electronics Engineers in Israel*, pp. 1–5, 2012.

[11] Y. Bar–Shalom and X. Li, "Introduction," in *Multitarget–Multisensor Tracking: Principles and Techniques*. New Orleans, LA: University of New Orleans, pp. 1–84, 1995.

[12] R. G. Hutchins, “Optimal estimation: Sensor and data association,” notes for EC3310, Naval Postgraduate School, 1997.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California