Real-time Model Identification for Ground Vehicle Trajectory Estimation using Extended Kalman Filter Residual Analysis

by

Hyrum David Johnson

BS Mechanical Engineering
Brigham Young University, 1997

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN
MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1999

Author ....[illegible]........
Department of Mechanical Engineering
May 17, 1999

Approved By ....[illegible]........
Edward J. Lanzilotta
Thesis Supervisor, CS Draper Laboratory

Certified by ....[illegible]........
Professor Derek Rowell
Thesis Advisor, MIT

Accepted by ....[illegible]........
Professor Ain Sonin
Chairman, Department Graduate Committee

[illegible]

Real-time Model Identification for Ground Vehicle Trajectory Estimation using Extended Kalman Filter Residual Analysis

by
Hyrum David Johnson

Submitted to the Department of Mechanical Engineering on May 17, 1999 in partial fulfillment of the requirements for the Degree of Master of Science in Mechanical Engineering

Abstract
This thesis describes real-time model identification for ground vehicle trajectory estimation by means of the Automatic Dynamic Trajectory Recognition System. A theoretical approach to trajectory model identification for the ADTR system using Kalman filter residual analysis is developed. This approach selects the best trajectory model from an array of candidates by comparing residual vectors generated by a bank of Kalman filter estimators built upon the candidate models. The filter with the lowest RMS residual magnitude value is identified as containing the best trajectory model.

The system is implemented in Draper Laboratory's C-Sim simulation environment. The system simulation contains a sensor model as well as a dynamic model of the ground vehicle. The simulation results demonstrate a correlation between the RMS value of the filter residual magnitude and the RMS magnitude of the ground vehicle position prediction error of extrapolating four seconds into the future. The results support the model identification approach.

Thesis Advisor:	Derek Rowell Professor of Mechanical Engineering
Thesis Supervisor:	Edward J. Lanzilotta Charles Stark Draper Laboratory

Acknowledgments

I cannot thank all the people who have helped me to the conclusion of this thesis. I would like to thank Derek Rowell for focusing my energies towards completion. For technical direction I must thank the people of CS Draper laboratory. Foremost, Ed Lanzilotta who gave me frequent direction and countless hours of feedback during the theoretical development and document writing stages of this research. Also, Dale Landis for his guidance on Kalman filtering theory. I would also like to thank Dave Hauger and Linda Leanard for their simulation savvy.

I would also like to thank my parents, who instilled me with curiosity and ambition. All other thanks must go to my wonderful wife Katie, whose love and support have allowed me to tackle such a task, and to God.

This thesis was prepared at the Charles Stark Draper Laboratory, Inc., under CSR Project C703.

[illegible]

Hyrum David Johnson

Table of Contents

List of Figures

List of Tables

List of Symbols

A	Ground vehicle acceleration
a	Vector columns of matrix Y
b	Intermediate vector in U-D filter update algorithm
c	Intermediate vector in U-D filter update algorithm, also represents an entry in matrix C
C	Direction cosine matrix
D	Diagonal decomposition matrix of the U-D factored form
$\breve{D}$	Intermediate matrix in U-D filter propagation algorithm
f	Intermediate vector in U-D filter update algorithm
F	Matrix expression of a system of linear or linearized first-order differential equation
g	Intermediate vector in U-D filter update algorithm
H	Sensor measurement transformation matrix
I	Identity matrix
IC	Instantaneous center
J	Ground vehicle jerk (rate of change of acceleration)
K	Kalman gain matrix
L	Longitude
m	Matrix index used in U-D decomposition algorithm
n	Dimension of vector X
P	Ground vehicle position
$\mathbf{P}$	Error covariance matrix
Q	Input noise a priori covariance matrix
R	Sensor noise a priori covariance matrix
r	Kalman filter residual vector
S	Selection value, RMS value of the magnitude of the r vector
T	Minimum time period for a driver to move between acceleration extremes
t	Time
U	Upper triangular decomposition matrix of the U-D factored form
V	Ground vehicle velocity
v	Sensor noise vector

w	Input noise vector
WB	Wheelbase specification for ground vehicle
X	State vector
y	Output vector, state vector transformed to sensor measurement space
Y	Intermediate matrix in U-D filter propagation and update algorithms
z	Measurement vector
Δt	Time step of one ADTR cycle
δ	Dirac delta function
μ	Geodetic latitude
ω	Angular Velocity
Φ	State transition matrix
ϕ	Ground vehicle yaw angle, heading
τ	Time
θ	Angle of steering wheels

1 Introduction

Trajectory recognition is a part of many human tasks. As humans, we are very adept at sizing up a situation and making seat-of-the-pants estimations about how a moving object is going to behave based on experiential knowledge and upon a useful 'mental model' of the physics of a system. If a ball is thrown to us, we can anticipate well enough to catch it. If we look down the street at oncoming cars, we can ascertain the safety of crossing it.

As we migrate to autonomous systems, it is essential that we include functionality into systems that allow identification of appropriate mathematical 'mental models' of the environment they are interacting with. The concept of the 'mental model' translates very well to the subject of automation and artificial intelligence because of analogous ideas existing in estimation theory.

Utilization of an appropriate mathematical model is key to the success of many systems. An engineer does appreciable work when developing a useful mathematical model using analytical and empirical methods. The performance of a model is generally confirmed in actual operation or in simulated operation with a post-processing evaluation approach. However, robustness is given to an autonomous system if it is able to choose from a group of models that span a set of scenarios and assumptions in real-time.

1.1 Objective

In order to improve the accuracy of air-to-ground targeting systems we will identify a model that describes a moving ground vehicle's short-term trajectory in real-time operation. The objective of the Automatic Dynamic Trajectory Recognition System (ADTR) described in this thesis is to select the best trajectory model from an array of candidates by considering position measurement data. The selected trajectory model is useful in predicting the future position of the ground vehicle.

1.2 Approach

The approach taken is to sample the position of a moving ground vehicle from an Unmanned Arial Vehicle (UAV) by way of the combined sensor data of an Embedded GPS/INS navigation system (EGI) and a laser radar (LADAR). This data is processed with the Automatic Dynamic Trajectory Recognition System. The ADTR system contains a bank of Kalman filters running in parallel. Each Kalman filter uses a different vehicle trajectory model and corresponding state vector. Each filter estimates state variables (including position) of the vehicle trajectory. The residual vector is an iteration-by-iteration measure of the difference between the measured position and each filter's estimate of position. The system analyzes the performance of each Kalman filter by calculating the RMS value of the magnitude of the filter's residual vector.

The ADTR system is implemented in a simulation system. The simulation is built using the CSIM Simulation Framework. This implementation allows for system development, filter parameter selection and system demonstration using a two dimensional vehicle dynamics simulator and ADTR system software.

1.3 TOPART and Other Previous Work

Here is summarized the previous research work that has motivated the work described in this thesis. The idea for The ADTR system was developed as an outgrowth to the Tomahawk / Predator Advanced Real Time Targeting (TOPART) project. The goal is to extend the TOPART targeting capability to include maneuvering ground targets. The TOPART project, along with several other articles and reports, became the foundation for the development of the system.

The TOPART project involves the targeting of stationary objects in World Geodetic Survey 1984 (WGS-84) coordinates using a laser radar (LADAR), and an Embedded GPS/INS (EGI) navigation unit mounted on an Unmanned Aerial Vehicle (UAV). In this project the UAV is the Predator vehicle and the Tomahawk cruise missile is the intercept vehicle. The objective of this targeting task is to achieve a coordinate fix of a stationary target by combining line-of-site (LOS) position information from the LADAR and UAV position information from the EGI navigation unit on the UAV. The coordinate accuracy is degraded by atmospheric transmission delays of the GPS satellite signals, drift in the inertial sensors and calibration errors and noise in the LADAR data. The target coordinate data is then handed off to the intercept vehicle along with the specific GPS satellite constellation used. The GPS constellation information allows the intercept vehicle to use the same satellites for its EGI navigator and steer to the WGS-84 location with comparable errors. High correlation in the targeting coordinate errors and the intercept vehicle navigation errors (because the same GPS constellation was used) acts to nullify the EGI system errors if the events are closely spaced in time (10 minutes or so.) The remaining error is largely due to calibration errors and noise in the LADAR sensor. Draper Laboratory is developing an additional capability to calibrate the targeting sensor system as part of the mission but prior to targeting the unknown stationary object. This is called "Dynamic Bore-sighting." This is achieved by first targeting several accurately surveyed landmarks and generating error correction information about the sensor system.

The ADTR system is based on a similar scenario, but uses a moving ground target instead of a stationary target. In order to handle a moving target there must be a mechanism for trajectory prediction. Much work has been done in this area, especially in radar tracking and gun-siting systems.

Perkins discusses a solution for predictive gunfire control against moving ground vehicles using turret mounted guns in the article, "Sub-optimal State Estimation as Related to Predictive Fire Control System Design" [10]. The purpose of a gunfire control system is to adequately predict the position of a maneuvering ground target at the time of impact so as to move the turret to anticipate the target. The method uses a sub-optimal Kalman filter estimating the position, velocity and acceleration of the target. The vehicle states for aiming are the (LOS) coordinates: range, azimuth and elevation angle. For the Kalman filter, the LOS states are actually estimated in hybrid coordinates. Position states are in LOS coordinates but velocity and acceleration are expressed in a Cartesian coordinate frame aligned with the range vector. The state estimates for velocity and acceleration of azimuth are easily computed from estimates of range, velocity and acceleration components normal to the range vector. With this information available it is possible to extrapolate the LOS states forward to the impact time and move the turret to that location. This application is similar in concept to the random walk acceleration model found in Section 3.3.2; the difference being the mixture of LOS and Cartesian coordinate frames used in the gunfire control system.

Burke proposes a nonlinear prediction concept for curved paths called the Circular Arc Aimed Munitions (CAAM) fire control solution [1]. The CAAM concept is a modification of the standard parabolic prediction equation. The standard parabolic prediction equation extrapolates position based on velocity and constant acceleration estimates. The CAAM prediction concept extrapolates a circular arcing trajectory by adding two factors to the velocity and acceleration estimates in the standard parabolic prediction equations. These factors are functions of the rotational velocity of the vehicle. Still this prediction relies on a linear estimator to generate velocity and acceleration states.

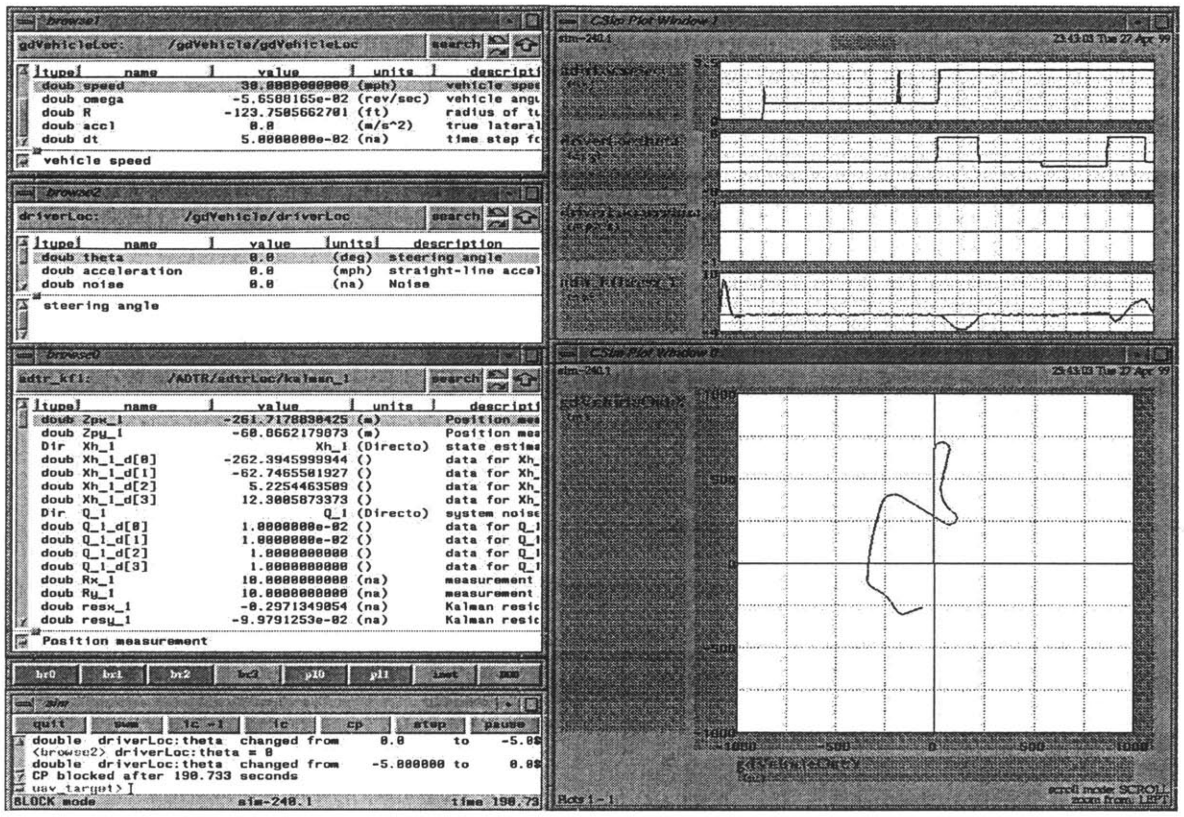
Bird develops a survey of Kalman filter approaches for tracking, estimating and predicting moving target trajectories in the report "Some Applications of Kalman Filtering in Advanced Land Fire Control Systems" [2]. He covers guidelines for choosing target models for Kalman filters. He also defines several common linear target models. These models are: random walk velocity, random walk acceleration, exponentially correlated velocity and exponentially correlated acceleration. He develops these models in both rectangular and spherical coordinates. He lays out guidelines for tuning performance of a filter design through model choice and assignment of Kalman covariance matrices.

In the paper "Identification of Maneuvering Aircraft using Class Dependent Kinematics Models" Cutaia and O'Sullivan develop multiple model tracking filters used both to track and to identify aircraft class [5]. The aircraft class models use physical aircraft attribute, allowable pilot commands and pilot behavior to discriminate between a set of classes. A multiple model tracking filter is developed for each aircraft class. This filter generates an aircraft state estimate based on radar measurements and attitude measurements from a high-resolution sensor. Class identification is achieved by choosing the class model that maximizes the probability of the sensor observations.

1.4 CSIM Simulation Framework

The CSIM Framework, developed at C. S. Draper Laboratory, is a tool designed to aid in the development and execution of real-time vehicle simulation software written in C. The CSIM Framework is well suited for human-in-the-loop and hardware-in-the-loop simulations. It consists of a database preprocessor that creates a hierarchical organization of simulation variables using nested data structures and a powerful execution environment. Simulation software development is enhanced by the existence of simulation function libraries. These include such things as simulation I\O, matrix data types/arithmetic, Gaussian noise generation.

The execution environment blends many powerful tools. A command line interface allows control of the simulation during runtime. Commands can be executed through use of command script input files, or an X-window based console. This is coupled with a Motif-based graphical user interface (GUI). This execution environment allows display and modification of simulation variables before or during runtime using the command line interpreter or simulation variable browse windows as shown in Figure 1.1. Inside the browse window the displayed units of the simulation variables can be changed in run-time. Additionally, CSIM framework configuration variables can be changed just like simulation variables.



```
gdVehicleLoc:        /gdVehicle/gdVehicleLoc
|type|      name       |       value       | units  |   descripti
 doub speed                 30.0000000000 (mph)      vehicle spee
 doub omega                -5.6508165e-02 (rev/sec)  vehicle angu
 doub R                  -123.7505662701 (ft)       radius of tu
 doub accl                  0.0           (m/s^2)    true lateral
 doub dt                    5.0000000e-02 (na)       time step fc
 vehicle speed
```

```
driverLoc:           /gdVehicle/driverLoc
|type|      name       |     value     |units|    description
 doub theta                 0.0         (deg)  steering angle
 doub acceleration          0.0         (mph)  straight-line accel
 doub noise                 0.0         (na)   Noise
 steering angle
```

```
adtr_kf1:            /ADTR/adtrLoc/kalman_1
|type|      name       |       value       | units  |   descripti
 doub Zpx_1              -261.7178830425 (m)         Position mea
 doub Zpy_1               -60.8662179073 (m)         Position mea
 Dir  Xh_1                          Xh_1 (Directo)   state estima
 doub Xh_1_d[0]          -262.3945999944 ()          data for Xh_
 doub Xh_1_d[1]           -62.7465501927 ()          data for Xh_
 doub Xh_1_d[2]             5.2254463509 ()          data for Xh_
 doub Xh_1_d[3]            12.3005873373 ()          data for Xh_
 Dir  Q_1                            Q_1 (Directo)   system noise
 doub Q_1_d[0]             1.0000000e-02 ()          data for Q_1
 doub Q_1_d[1]             1.0000000e-02 ()          data for Q_1
 doub Q_1_d[2]             1.0000000000 ()          data for Q_1
 doub Q_1_d[3]             1.0000000000 ()          data for Q_1
 doub Rx_1                10.0000000000 (na)        measurement
 doub Ry_1                10.0000000000 (na)        measurement
 doub resx_1              -0.2971349054 (na)        Kalman resic
 doub resy_1              -9.9791253e-02 (na)       Kalman resic
 Position measurement
```

```
double  driverLoc:theta  changed from     0.0      to     -5.0$
<browse2> driverLoc:theta = 0
double  driverLoc:theta  changed from  -5.000000  to      0.0$
CP blocked after 190.733 seconds
uav_target>
BLOCK mode               sim-240.1                time 190.73
```

Figure 1.1: CSIM Framework Simulation Execution Environment

Variable visibility is also enhanced through use of an on screen dynamic plotting facility. Other features include data logging and exporting of variable histories. A large part of the CSIM framework not utilized in this thesis is devoted to simulation project graphics for immersive human-in-the-loop simulations and vehicle visualization.

1.5 Thesis Content Summary

Chapter 1 is this introduction. Chapter 2 describes the components and interaction of the total targeting system and the ADTR system. Chapter 3 develops the ADTR algorithm. Chapter 4 describes the implementation of the total system in simulation. Chapter 5 describes the simulation demonstration and data collected from it. Chapter 6 draws conclusions from the demonstration and provides recommendations for future work.

2 System Description

This chapter is an overview description of the Automatic Dynamic Trajectory Recognition System. Section 2.1 describes the operational scenario used as a foundation for the system. Section 2.2 describes the coordinate frames used in the system for the expression of target position. Section 2.3 describes the sensor input to the system and the output of the system. Section 2.4 describes the assumptions made in formulating a workable problem.

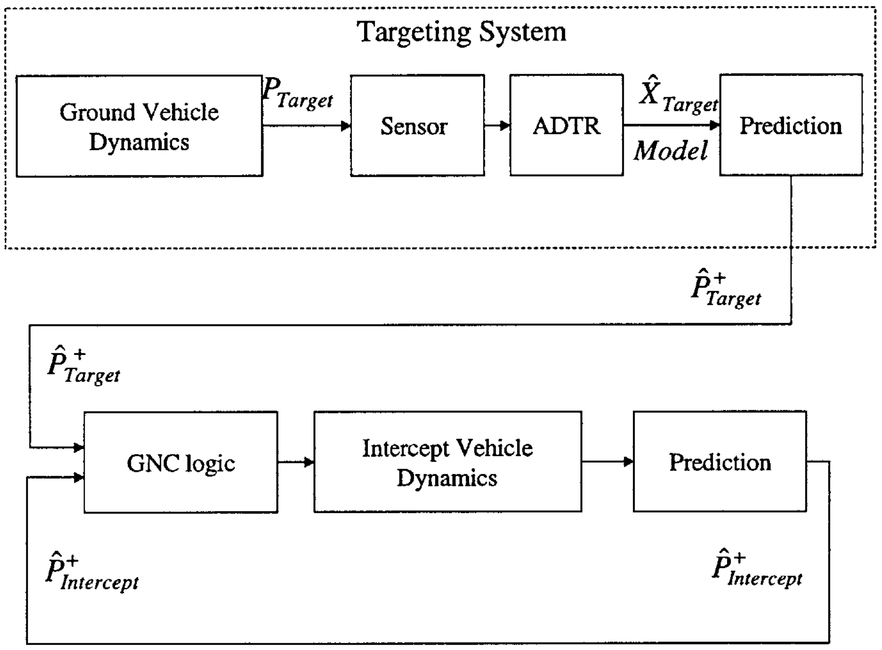


Figure 2.1: Third Party Targeting System Top Level Diagram

The Automatic Dynamic Trajectory Recognition System utilizes an Unmanned Aerial Vehicle (UAV) that act as a third party targeting system for weapons delivery. The goal of the system is to identify a short-term trajectory model for a ground vehicle based on position measurements over time. This model is useful in a targeting system for extrapolating the position of the vehicle $\hat{P}_{Target}$ into the future $\hat{P}^{+}_{Target}$. This information can be used by an intercept vehicle or weapon to control the future position of the intercept vehicle $\hat{P}^{+}_{Intercept}$ to coincide with the future position of the target.

2.1 Operational Scenario

The basic scenario is for the UAV to circle a ground target at a fixed altitude and turning rate and observe the ground vehicles position over time. Measurements are gathered by the UAV using a laser radar (LADAR) and an Embedded GPS/INS (EGI) navigation system to sense the position of the ground vehicle. The UAV observes the ground vehicles movement by taking samples with the EGI and LADAR sensors at a constant rate of 5 Hz. The LADAR measures the position of the target relative to the UAV. This measurement is represented by $P_{Target,UAV}$. This notation defines the position vector P of the target relative to the UAV. The EGI estimates the position of the UAV relative to the datum point or origin of an earth fixed inertial coordinate frame notated as P_{UAV}. Coordinate frames are discussed further in Section 2.2. These vectors add together to produce the position of the target relative to the earth-fixed inertial coordinate frame.

$$P_{Target} = P_{UAV} + P_{Target,UAV} \tag{2.1}$$

This equation is illustrated in Figure 2.2.

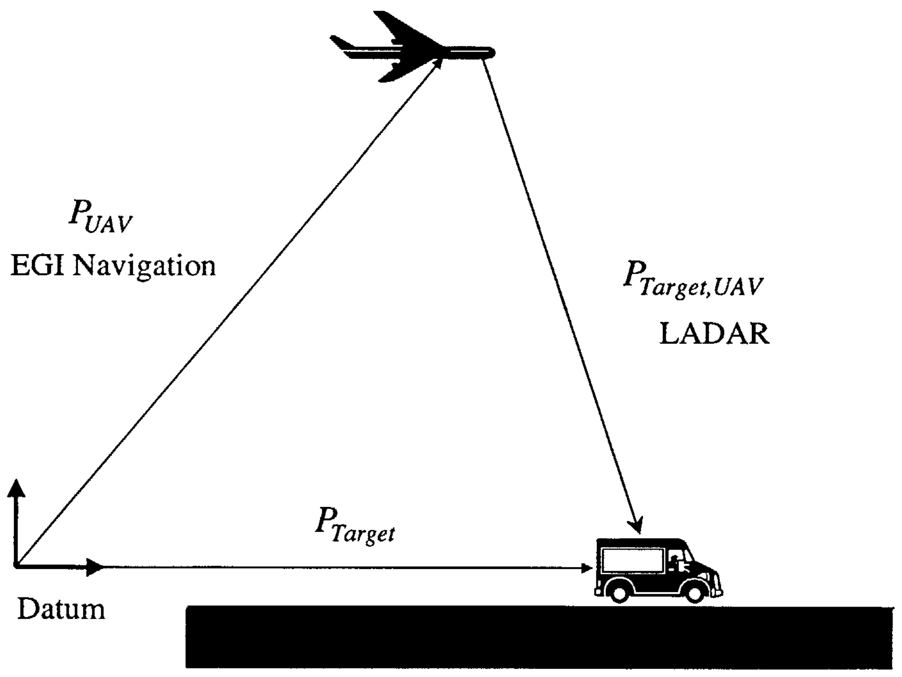


Figure 2.2: Sensor System Concept

This LADAR operates on RADAR principles but with a laser that has a much higher electromagnetic frequency. It is mounted on the UAV and pivots itself along pitch and yaw axes while sweeping the viewing area with the laser. The LADAR receives reflected laser light and makes a measurement of the line-of-site (LOS) range of the reflection and the direction relative to a coordinate frame coincident with the mounting point of the LADAR.

The EGI navigation system combines the merits of the Global Positioning System (GPS) and an Inertial Navigation System (INS) in one package. Inertial navigation uses three accelerometers and three gyros to measure linear accelerations and angular rates in six degree-of-freedom (DOF) of the body they are mounted to. The EGI integrates the measurement signals to get velocity, position and orientation of the UAV. INS sensor readings drift from true values due to imperfect sensor quality. INS is very accurate and fast for navigating in the short term, but over the long term, large errors can accumulate. The GPS system uses radio navigation principles to calculate the position of a GPS receiver relative to a worldwide constellation of GPS satellites. GPS systems may exhibit large bias errors due to atmospheric effects. These change somewhat slowly, however. The GPS and INS data are redundant, but have complimentary performance strengths. They are blended by the EGI using estimation techniques into a single integrated measure of position. This is shown in Figure 2.3

The ADTR system stores the current sampled data and processes them with four parallel Kalman filters as shown in Figure 2.4. Each of these four filters is based on a different trajectory model. Two of the filters are based on linear models that are common in target tracking applications. The remaining two filters are based on nonlinear models which use a special case discrete form of the extended Kalman filter algorithm developed in this thesis.

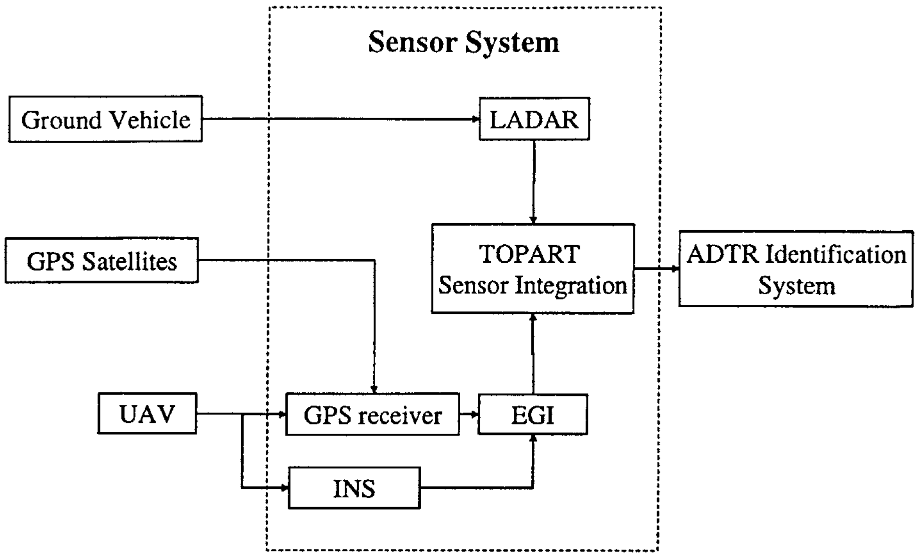


Figure 2.3: Sensor System Information Flow

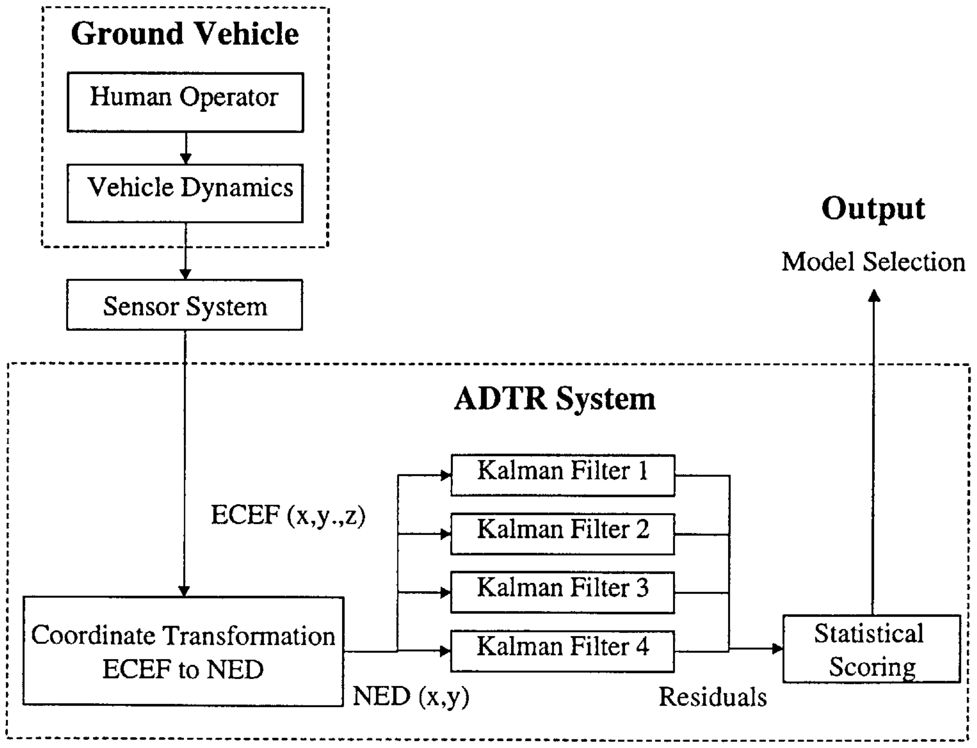


Figure 2.4: ADTR Block Diagram

The residual vector of these filters represent the difference of each position sample compared to the filter's position estimate. The filter iterates at a 5 Hz rate so that there are residuals calculated for each of the four filters for each measurement interval. Even though the position samples are corrupted with noise, the filter that is estimating position most accurately will have the smallest residual over time. The converse is that the filter with the smallest residual over time has the most valid position estimate. It follows that the filter that is estimating the position most accurately is doing so because it has the most valid model of the target's trajectory.

The ground vehicle is a free roaming wheeled vehicle that is driven by a human who controls turning and speed. The environment is flat, relatively featureless and benign. The ground vehicle's trajectory is not constrained by any known roads or terrain. It is, however, constrained by performance limits dictated by the mechanism. These constraints are of the form of acceleration limits in different directions.

2.2 Coordinate Frames

The target position data in the ADTR is expressed in two different earth fixed coordinate frames. One is the World Geodetic Survey 1984 (WGS-84) Earth-Centered-Earth-Fixed (ECEF) coordinate. The other is the North-East-Down (NED) coordinate frame. Earth fixed coordinate frames are non-inertial. We will make an assumption for inertial coordinate frames in Section 2.4

The WGS-84 ECEF frame is a Cartesian coordinate frame with the origin at the earth's center of mass. The x-axis passes through the equator at zero degrees longitude. The y-axis passes through the equator and is perpendicular to the x-axis. The z-axis is aligned with the earth's axis of rotation. The WGS-84 ECEF coordinate frame rotates with the earth as seen in Figure 2.5.

The NED coordinate frame has its origin at a datum point local to the ground vehicle's position. This frame is oriented such that the x-axis points north, the y-axis points east and the z-axis points down toward the center of the earth. In this coordinate frame, yaw angles are measured clockwise from the x-axis (which is North). This is the same convention as measuring a compass bearing.

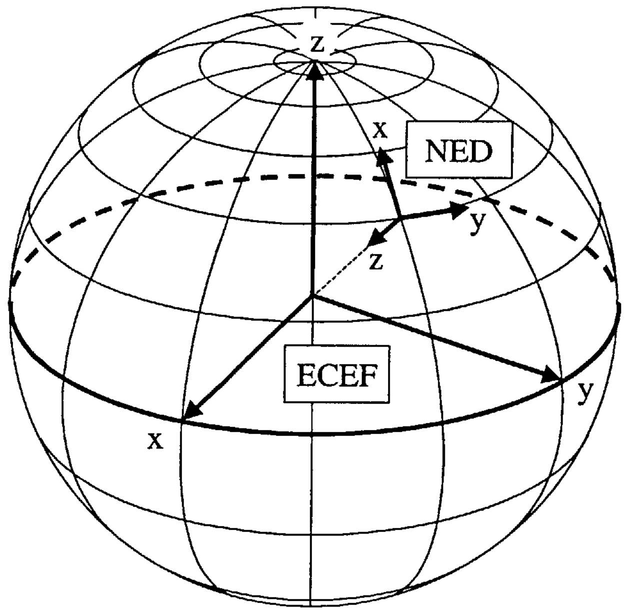


Figure 2.5: ECEF and NED coordinate frames

2.3 Input and Output Data

The input for the ADTR is an estimate of the position of the ground vehicle coming from the TOPART sensor as shown in Figure 2.4. This data is expressed in WGS-84 ECEF coordinates. A preprocessor in the ADTR system transforms the input data to NED coordinates with the origin located at the surface of the earth at the initial estimated vehicle location. This transformation aligns the x-y plane tangent to the surface of the earth. This allows the dominant components of vehicle motion to lie in the x-y plane.

The output for the ADTR is the identification number of the selected trajectory model. This is accompanied by the RMS statistics of the residuals for the selected model and the other three models.

Transformations from ECEF to NED are made by performing a coordinate rotation and then a translation. We will use a notation that expands on that previously described. $P_{Target,UAV}$ means the position of the target with respect to the UAV. P_{Target} with only one subscript means the position of the target with respect to the origin of the implied coordinate frame. For this discussion, the coordinate frame the vector is expressed in may be explicitly notated as a superscript, i.e. P_{Target}^{NED}.

A coordinate rotation can be achieved by multiplication by an appropriate matrix. This intermediate coordinate frame that is only rotated will be called NED'. The notation for the rotation matrix of frame NED' with respect to frame ECEF is $C_{ECEF}^{NED'}$. This matrix is the rotation transformation from ECEF to NED' coordinates expressed as a direction cosine matrix

$$P_{Target}^{NED'} = C_{ECEF}^{NED'} P_{Target}^{ECEF} \tag{2.2}$$

The direction cosine matrix is a 3×3 matrix whose entries represent the cosine of the angle between an axis in one frame and an axis in the other frame. For example, the direction cosine matrix entry $c_{x_A y_B}$ represents the cosine of the angle between the x-axis in the A-frame and the y-axis in the B-frame. The direction cosine matrix is defined by:

$$C_A^B = \begin{bmatrix} c_{x_A x_B} & c_{x_B y_A} & c_{x_B z_A} \\ c_{x_A y_B} & c_{y_A y_B} & c_{y_B z_A} \\ c_{x_A z_B} & c_{y_A z_B} & c_{z_A z_B} \end{bmatrix} \tag{2.3}$$

The specific direction cosign matrix for the rotation transformation from ECEF to NED' is [11]:

$$C_{ECEF}^{NED'} = \begin{bmatrix} -\sin\mu\cos L & -\sin\mu\sin L & \cos\mu \\ -\sin L & \cos L & 0 \\ -\cos\mu & -\cos\mu & -\sin\mu \end{bmatrix} \tag{2.4}$$

Where μ is the geodetic latitude of the datum point and L is the longitude. Geodetic latitude differs from geocentric latitude for an elliptical sphere like earth. Geodetic latitude is the angle formed between the equatorial plane and a line perpendicular to the elliptical surface of the earth at the datum point as seen in Figure 2.6.

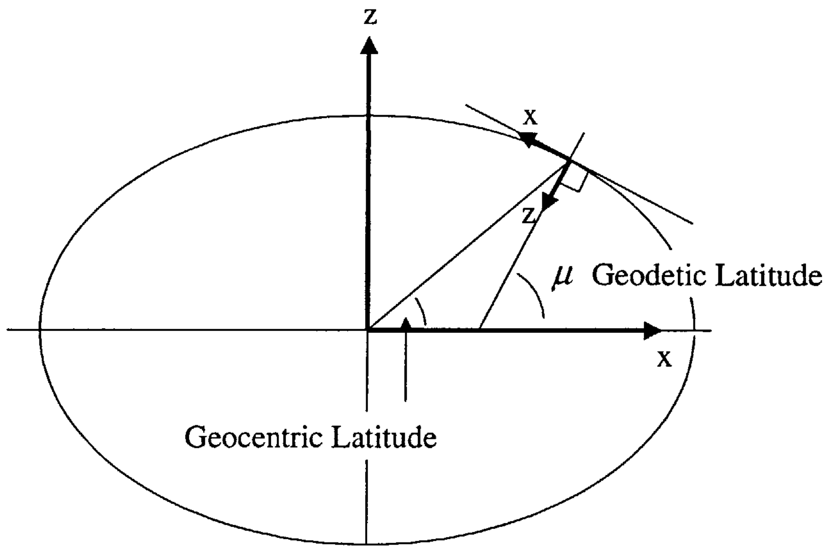


Figure 2.6 Geodetic Latitude for an Ellipsoid Earth

All that is left is a translation from the NED' coordinate frame to the NED coordinate frame. This is the same thing as expressing a NED' vector with respect to the datum point at the origin of the NED coordinate frame.

$$P_{Target}^{NED} = P_{Target,Datum}^{NED'} \tag{2.5}$$

An alternate expression for a NED' vector with respect to the datum point is:

$$P_{Target,Datum}^{NED'} = P_{Target}^{NED'} - P_{Datum}^{NED'} \tag{2.6}$$

Substituting (2.2) and (2.5) into (2.6) gives the complete transformation:

$$P_{Target}^{NED} = C_{ECEF}^{NED'}(P_{Target}^{ECEF} - P_{Datum}^{ECEF'}) \tag{2.7}$$

2.4 Assumptions

Several assumptions are made to simplify the systems development. One assumption is that the path of the ground vehicle can be sufficiently modeled on a two dimensional coordinate frame. This is to say that the ground vehicle is approximately traveling along the surface of a plane. In reality, the surface of the earth is curved. For this system we assume that the curvature of the earth has a negligible effect within the domain of distances traveled by the ground vehicle. Also, a vehicle will change altitude in any earth fixed coordinate frame due to changes of altitude in terrain. This change in altitude is neglected. It is minor in comparison to changes in latitude or longitude. Also, the driver of a ground vehicle is not able to choose an altitude trajectory, he is constrained to whatever altitude environment he is in. The behavior this system is trying to capture is the planar trajectory as controlled by the vehicle's driver.

The second assumption is that of a constant noise variance of the LADAR sensor. Measurements from a LADAR become noisier as the line-of-sight (LOS) distance between the vehicles increases. We will assume that the altitude and circling path of the UAV are adequate proportioned to make the angle φ relatively constant, see Figure 2.7. Thus we will assume that the LOS distance between the UAV and ground vehicle and the LADAR noise are relatively constant.

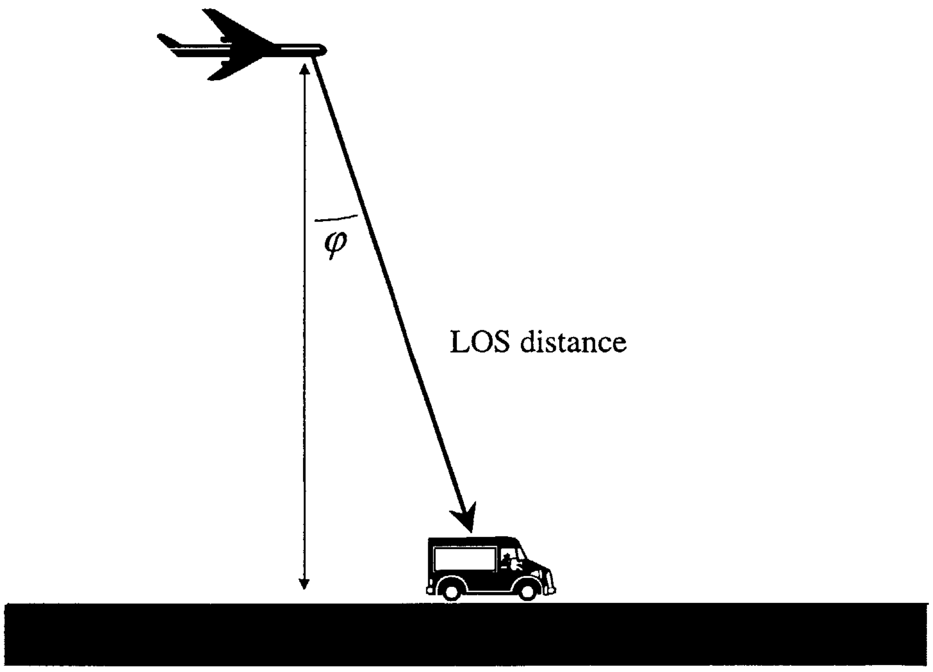


Figure 2.7: UAV and Target

A third assumption is that of unbiased noise for the integrated TOPART sensor system. The nominal target position data received from sensor system contains errors from the LADAR sensor and errors from the EGI navigation sensor. These take the form of random noise and bias errors.

$$P_{Nominal} = P_{truth} + bias_{EGI} + noise_{EGI} + bias_{LADAR} + noise_{LADAR} \tag{2.8}$$

Constant biases do not interfere with trajectory estimation since every point in the trajectory is offset the same amount. They do confound absolute coordinate prediction. To address both issues, we assume that the LADAR bias term is negligible because of the "Dynamic Bore-sighting" work for the TOPART sensor as discussed in Section 1.3. Further, we will assume that the EGI bias term is constant because the bias moves on the order of minutes and this system predicts on the order of seconds. Additionally, the third party targeting system developed for the TOPART project assumes that an intercept vehicle with a comparable EGI unit using the same GPS satellite constellation will navigate with an equivalent bias. The result is that we can assume that the true position P_{truth} is effectually corrupted by a single lumped unbiased noise component.

A final assumption is that the earth fixed coordinate frames used in this system are approximately inertial coordinate frames. This assumption neglects the rotation of the earth due to the small time periods involved in targeting compared to the rotational period of the earth.

3 Algorithm Development

This chapter describes the theoretical development of the Automatic Dynamic Trajectory Recognition System apart from simulation implementation details. Section 3.1 describes some general concepts behind the use of the Kalman filter in this system. Section 3.2 describes the discrete Kalman filter algorithm for linear models and a new special discrete approximation of the extended Kalman filter algorithm for a special class of nonlinear models. Section 3.3 describes two linear and two nonlinear state-space ground vehicle dynamics models used in the system. Section 3.4 shows the discretization and linearization of these state space models to fit the Kalman filter algorithm. Section 3.5 describes the measurement model used in all four filters. Finally, Section 3.6 describes the development of the filter performance-scoring algorithm used to select the best maneuver model.

3.1 General Concepts

In trajectory estimation it is necessary to identify behaviors of interest. For a wheeled ground vehicle, behaviors of interest include turning, accelerating and braking. Four models are chosen to represent sets of these behaviors, each model adding sophistication to the previous ones. These models will estimate:

- Straight-line travel at a constant velocity
- Travel with a constant straight line acceleration
- Travel with a constant turning acceleration and constant speed
- Travel with a constantly varying speed and constantly varying turning acceleration

Each models is expressed mathematically as a systems of first-order differential equations, each with a state vector, X , which defines the trajectory parameters that are estimated in each scenario.

$$\dot{X} = f(X(t)) + u(t) \tag{3.1}$$

$\dot{X}$ is the time derivative of the state vector. $f(X(t))$ is a vector of general functions of the continuous state vector $X(t)$. $u(t)$ is the model input vector. The differential equations are rigid-body dynamic equations, describing the relationship of the vehicle's body in a two dimensional inertial coordinate frame.

In these models, there are no deterministic inputs as in a control scenario. Inputs are unknown maneuvers. All inputs are modeled as a random disturbance vector. Specifically, lower order terms have zero values and higher order terms are modeled as random walk variables. A random walk is a stochastic process described by the two-degree-of-freedom (2-DOF) analogy of a drunken man taking steps in random directions. The random walk process is shown here in continuous scalar form with $w(t)$ representing unbiased Gaussian white noise:

$$\dot{x} = w(t) \tag{3.2}$$

The random walk process in discrete scalar form is:

$$x_{k+1} = x_k + w_k \tag{3.3}$$

The approach at modeling unknown vehicle maneuvers with a 2-DOF random walk is very simple and is common in the literature [2][10]. The simplistic models from the literature contain a mathematical admission that we don't really know very much about how the vehicle is constrained to maneuver. For example, a vehicle is commonly modeled as a 2-DOF random walk of its acceleration state. This does not include any information about a car's constraint to only travel in a relatively smooth series of tangent arcs. An attempt has been made in the system development to include models that constrain the random walk model to arcing trajectories.

Given this discussion, the models in this system fall into two forms. For the nonlinear case, the models are in the general form:

General-Gaussian Model

$$\dot{X} = f(X(t)) + w(t) \tag{3.4}$$

The linear models are in a reduced matrix form:

Linear-Gaussian Models

$$\begin{gathered} f(X) = FX(t) \\ \dot{X} = FX(t) + w(t) \end{gathered} \tag{3.5}$$

F is a square matrix of constants. Kalman filters used for this computer simulation are based on these continuous models, but ultimately require a discrete treatment of the model. For the linear models, discretization of F is accomplished by expanding a Taylor series about some point t_0.

$$x(t) = x_0 + (t - t_0)\dot{x}_0 + \tfrac{1}{2!}(t - t_0)^2 \ddot{x}_0 + \tfrac{1}{3!}(t - t_0)^3 \dddot{x}_0 + \ldots \tag{3.6}$$

From the models we know that:

$$\begin{aligned} \dot{X} &= FX \\ \ddot{X} &= F\dot{X} = F^2 X \\ \dddot{X} &= \ldots \end{aligned} \tag{3.7}$$

so then the discrete version Φ_{k-1} is found by:

$$X_k = (I + \Delta t F + \tfrac{1}{2!}\Delta t^2 F^2 + \tfrac{1}{3!}\Delta t^3 F^3 + \ldots)X_{k-1} \tag{3.8}$$

$$\Phi_{k-1} = e^{F\Delta t} = I + \Delta t F + \tfrac{1}{2!}\Delta t^2 F^2 + \tfrac{1}{3!}\Delta t^3 F^3 + \ldots \tag{3.9}$$

$$X_k = \Phi_{k-1} X_{k-1} \tag{3.10}$$

3.2 Kalman Filter

The time invariant Kalman filter equations used by the system are summarized here. The linear models are implemented in the standard form, and the nonlinear models in the extended Kalman filter form. As discussed in Section 1.2, the approach is to use four separate Kalman filters in parallel based on the vehicle models discussed in Section 3.3. For a more complete treatment of the Kalman filter derivation, see [6].

3.2.1 Discrete Linear Kalman Filter Equations

For the discrete Kalman filter the criterion of optimality is based on the assumption that the real system being estimated is truly represented by the following discrete mathematical model.

Discrete – Linear Kalman Filter Model

$$X_k = \Phi_{k-1} X_{k-1} + w_{k-1} \tag{3.11}$$

$$z_k = H_k X_k + v_k \tag{3.12}$$

The vectors w_{k-1} and v_k are both unbiased Gaussian white noise vectors and the subscript identifies the discrete time associated with it. w_{k-1} is the discrete form of the input noise $w(t)$ discussed in Section 3.1. z_k is the measure vector coming from system sensors. It is modeled as a linear combination of the state vector X_k plus sensor noise v_k. It is assumed that the noise vectors have no correlation with each other. The noise is described by covariance matrices Q_{k-1} and R_k as follows:

$$
\begin{aligned}
Q_{k-1} &= E\left\{w_{k-1}w_{k-1}^T\right\} \\
R_k &= E\left\{v_k v_k^T\right\}
\end{aligned}
\tag{3.13}
$$

It is important to note that the diagonal elements of Q_{k-1} and R_k are the variances for each member of w_k and v_k. A simplifying assumption used in this application is that there is no correlation between individual elements of the noise vectors, which leaves only a diagonal matrix.

The Kalman filter algorithm operates as shown in Figure 3.1.

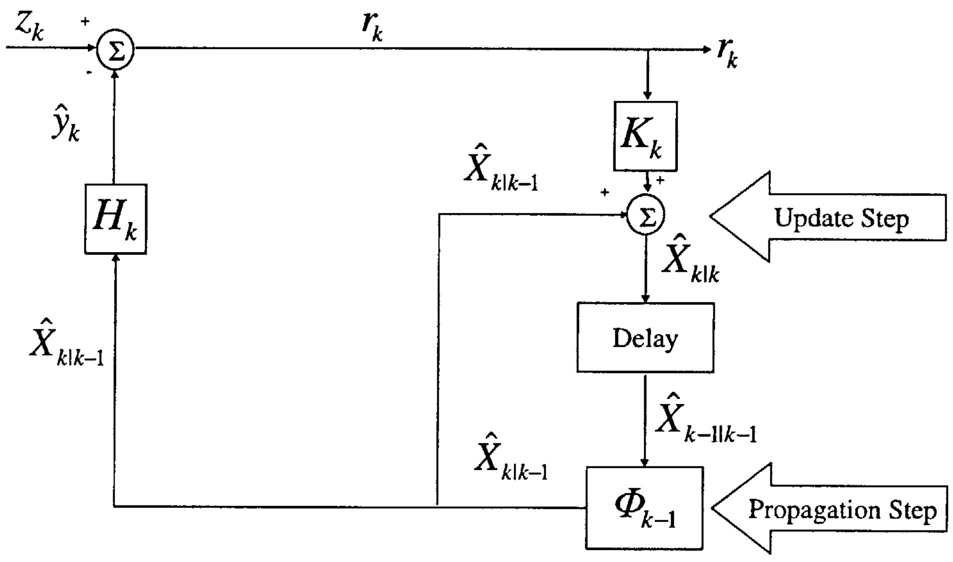


Figure 3.1: Kalman Filter Diagram

Here $\hat{X}$ represents the estimate of the state vector X. The first iteration of the filter algorithm begins at the 'Propagation Step'. This is carried out by propagating an initial state estimate $\hat{X}_0$, which is the expected value of the state vector at t_0,

$$\hat{X}_0 = E\{X_0\} \tag{3.14}$$

The state vector is propagated according to the probabilistic expectation of the truth model given by (3.11). The expected value for unbiased white noise is by definition zero. Also the expected value for a constant is the value of the constant.

$$\begin{aligned} \hat{w}_k &= E\{w_k\} = 0 \\ \Phi_{k-1} &= E\{\Phi_{k-1}\} \end{aligned} \tag{3.15}$$

Thus, the initial state estimate, $\hat{X}_0$, is propagated to the next time step by multiplying it by the associated initial state transition matrix, Φ_0. The result is the filter's prediction of the state vector at the next measurement event, and is read as the state estimate at t_k given the initial state vector.

$$\hat{X}_{k|0} = \Phi_0 \hat{X}_0 \tag{3.16}$$

A measurement vector z_k is compared to the output estimate vector $\hat{y}_k$. The error between the two is the residual vector, r_k

$$\begin{aligned} \hat{y}_k &= H_k \hat{X}_{k|0} \\ r_k &= z_k - \hat{y}_k \end{aligned} \tag{3.17}$$

The residual vector is the data source for updating the filter. It is the estimation error added to the sensor noise. This information is put to use in the 'Update Step'. The residual is multiplied by the Kalman gain, K_k and the result is added to augment the previous propagation prediction resulting in the updated current estimate. The result is the state estimate at t_k given a measurement at t_k.

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k r_k \tag{3.18}$$

The previous propagation prediction $\hat{X}_{k|k-1}$ is the general notation for $\hat{X}_{k|0}$. At this point we model the time delay of the discrete system changing the updated current estimate $\hat{X}_{k|k}$ to $\hat{X}_{k-1|k-1}$ and arrive back at the 'Propagation Step' for the beginning of the next iteration, compare (3.16).

$$\hat{X}_{k|k-1} = \Phi_{k-1} \hat{X}_{k-1|k-1} \tag{3.19}$$

For this treatment Φ_{k-1} is constant. As the algorithm iterates, the state estimate $\hat{X}_k$ converges on the value of X_k and follows state perturbations (modeled as w_k) according to the appropriateness of K_k. In order for this to occur properly it is necessary to incorporate *a priori* knowledge of covariance matrices Q_k and R_k into the selection of the gain matrix K_k. To do this, we need to be aware of another covariance matrix: the error covariance matrix $\mathbf{P}_{k|k}$.

$$\begin{aligned} \tilde{X}_{k|k} &= \hat{X}_{k|k} - X_k \\ \mathbf{P}_{k|k} &= E\{\tilde{X}_{k|k}\tilde{X}_{k|k}{}^T\} \end{aligned} \tag{3.20}$$

$\tilde{X}$ is called the estimation error. This matrix contains the expected covariance of estimation errors and describes the estimated performance of the updated state estimate $\hat{X}_{k|k}$. The error covariance matrix, like the state estimate, begins with an initial estimate $\mathbf{P}_0$ and is propagated by combining (3.11) and (3.19) into the following:

$$\begin{aligned} \tilde{X}_{k|k-1} &= \hat{X}_{k|k-1} - X_k \\ \mathbf{P}_{k|k-1} &= E\{\tilde{X}_{k|k-1}\tilde{X}_{k|k-1}{}^T\} \\ \mathbf{P}_{k|k-1} &= \Phi_{k-1}E\{\tilde{X}_{k-1|k-1}\tilde{X}_{k-1|k-1}{}^T\}\Phi_{k-1}^T + E\{w_{k-1}w_{k-1}^T\} \\ \mathbf{P}_{k|k-1} &= \Phi_{k-1}P_{k-1|k-1}\Phi_{k-1}^T + Q_{k-1} \end{aligned} \tag{3.21}$$

$\mathbf{P}_{k|k-1}$ is updated by combining (3.11), (3.12), (3.17), (3.18) and (3.20) into the following

$$\begin{aligned} \mathbf{P}_{k|k} &= (I - K_kH_k)E\{\tilde{X}_{k|k-1}\tilde{X}_{k|k-1}{}^T\}(I - K_kH_k)^T + K_kE\{v_kv_k{}^T\}K_k^T \\ \mathbf{P}_{k|k} &= (I - K_kH_k)\mathbf{P}_{k|k-1}(I - K_kH_k)^T + K_kR_kK_k^T \end{aligned} \tag{3.22}$$

The strategy in selecting the optimal or Kalman gain matrix is to do so such that the elements of the diagonal elements of the updated error covariance matrix, $\mathbf{P}_{k|k}$, is minimized [6]. The expression for the Kalman gain matrix is:

$$K_k = \mathbf{P}_{k|k-1}H_k^T(H_k\mathbf{P}_{k|k-1}H_k^T + R_k)^{-1} \tag{3.23}$$

With the Kalman gain, (3.22) is reduced to:

$$\mathbf{P}_{k|k} = (I - K_kH_k)\mathbf{P}_{k|k-1} \tag{3.24}$$

These Kalman filter equations are summarized in Table 3.1.

Discrete Linear Kalman Filter	
Propagation Equations	**Update Equations**
$\hat{X}_{k\mid k-1} = \Phi_{k-1}\hat{X}_{k-1\mid k-1}$ $\mathbf{P}_{k\mid k-1} = \Phi_{k-1}\mathbf{P}_{k-1\mid k-1}\Phi_{k-1}^T + Q_{k-1}$	$K_k = \mathbf{P}_{k\mid k-1}H_k^T[H_k\mathbf{P}_{k\mid k-1}H_k^T + R_k]^{-1}$ $r_k = z_k - H_k\hat{X}_{k\mid k-1}$ $\hat{X}_{k\mid k} = \hat{X}_{k\mid k-1} + K_k * r_k$ $\mathbf{P}_{k\mid k} = \mathbf{P}_{k\mid k-1} - K_kH_k\mathbf{P}_{k\mid k-1}$

Table 3.1: Standard Kalman Filter Equations

3.2.2 Discrete Extended Kalman Filter Equations

The extended Kalman Filter differs from the Standard Kalman filter in that it applies linear techniques to a nonlinear model through the use of iterative linearizations. Much like the standard Kalman filter, the extended Kalman filter has a criterion of optimality based on the assumption that the real process is truly represented by the following discrete-continuous mathematical model:

Discrete-Continuous Extended Kalman Filter Model

$$\begin{aligned} \dot{X}(t) &= f(X(t)) + w(t) \\ z_k &= H_kX_k + v_k \end{aligned} \tag{3.25}$$

Where input noise $w(t)$ and sensor noise v_k are zero mean white Gaussian random vectors as discussed in Section 3.2.1. The noise vectors associated with this model are described by the covariance matrices $Q(t)\delta(t-\tau)$ and R_k as follows [2]:

$$Q(t)\delta(t-\tau) = E\left\{w(t)w(\tau)^T\right\} \tag{3.26}$$

$$R_k = E\left\{v_kv_k^T\right\} \tag{3.27}$$

$\delta(t-\tau)$ is the Dirac delta function meaning that we are sampling the spectral density matrix $Q(t)$ when $t=\tau$.

As with the Discrete Linear Kalman filter, we first propagate an initial estimate of the state vector $\hat{X}$ and the error covariance matrix $\mathbf{P}$. This is stated in continuous form [6]:

$$\dot{\hat{X}} = f(\hat{X}(t))$$
$$\dot{\mathbf{P}}(t) = F(\hat{X}(t))\mathbf{P}(t) + \mathbf{P}(t)F(\hat{X}(t))^T + Q(t) \qquad (3.28)$$

The rest of the algorithm: the residual calculation, 'Update Step' and Kalman gain calculation are identical to the standard linear implementation.

The discrete approximation of the propagation equations (3.28) are found by first defining the transition matrix associated with $f(X)$:

$$X_k = \Phi_{k|\tau} X_\tau \qquad (3.29)$$

Where $\Phi_{k|\tau}$ represents the transition matrix between time τ and t_k. With these intermediate definitions, (3.28) and (3.26) can be written alternatively as [6][8]:

$$\hat{X}_{k|k-1} = \hat{X}_{k-1} + \int_{t_{k-1}}^{t_k} f(\hat{X}(\tau))d\tau \qquad (3.30)$$

$$\mathbf{P}_{k|k-1} = \Phi_{k|k-1}\mathbf{P}_{k-1|k-1}\Phi_{k|k-1}^T + Q_{k-1} \qquad (3.31)$$

$$Q_{k-1} = \int_{t_{k-1}}^{t_k} \Phi_{k|\tau} Q(t) \Phi_{k|\tau}^T d\tau \qquad (3.32)$$

Where $\Phi_{k|k-1}$ represents the transition matrix between t_{k-1} and t_k. These equations are an intermediate step toward discretization of the extended Kalman filter.

Next we linearize by calculating the partial derivative matrix of $f(X)$:

$$F(t) = \left.\frac{\partial f(X)}{\partial X}\right|_{X(t)} \qquad (3.33)$$

$F(t)$ is a linearization of $f(X)$ evaluated at the state vector $X(t)$. For a time invariant $F(t)$ the transition matrix is:

$$\Phi_{k|\tau} = e^{F[t_k - \tau]} \qquad (3.34)$$

Notice that for a linear system $F(t)$ is constant, $\Phi_{k|\tau}$ becomes calculable, and the above integrations become straight forward. In the case when $F(\tau)$ is changing over the time step interval neither $\Phi_{k|k-1}$ or $\Phi_{k|\tau}$ is easily calculated by exact analytical methods. Novel approximations can be made for $\Phi_{k|k-1}$ and $\Phi_{k|\tau}$ by substituting $F(\tau)$ with $F(t_{k-1})$ for the entire time interval. This is a satisfactory approximation for a special class of nonlinear models used in this system where:

Discrete Extended Kalman Filter Approximation Criterion

$$f(X(t)) = F(t)X(t) \tag{3.35}$$

The calculation of $F(t)$ for those models will be shown Section 3.4.

This linearization, as required by the Kalman filter's $\mathbf{P}$ propagation formula, has the bonus of an exact matrix representation $F(t)$ of the nonlinear model although it changes with the state. Because this matrix representation is exact for the evaluation point $X(t)$, it is a good approximation for small deviations from the evaluation point. Thus it is a useful approximation to use a discretized version of $F(t)$ as the transition matrix Φ_{k-1} for a judiciously small time step ahead of the evaluation point.

$$\Phi_{k-1} = \Phi_{k|k-1} \approx e^{F(t_{k-1})[t_k - t_{k-1}]} \tag{3.36}$$

Once this approximation has been made, the other equations become:

$$\hat{X}_{k|k-1} = \hat{X}_{k-1} + \int_{t_{k-1}}^{t_k} F(\tau)\hat{X}(\tau)d\tau \approx \Phi_{k-1}\hat{X}_{k-1} \tag{3.37}$$

$$\mathbf{P}_{k|k-1} \approx \Phi_{k-1}\mathbf{P}_{k-1}\Phi_{k-1}^T + Q_{k-1} \tag{3.38}$$

$$Q_{k-1} \approx \int_{t_{k-1}}^{t_k} \Phi_{k-1}Q(t)\Phi_{k-1}^T d\tau \tag{3.39}$$

In this approximation we have lost the terms associated with $\dot{F}$, $\ddot{F}$ etc as follows. Returning to the Taylor series expansion of the state vector, we approximate F_{k-1} as being appropriate for the whole time step. This means we approximate $\dot{F}$ to be zero for that step. Returning to the Taylor series expansion (3.6), the terms are given as:

$$\begin{aligned} \dot{X} &= FX \\ \ddot{X} &= \dot{F}X + F\dot{X} \approx F^2X \\ \dddot{X} &= \ldots \end{aligned} \tag{3.40}$$

This takes us to the same state propagation approximation as the previous argument:

$$\begin{aligned} X_k &\approx (I + \Delta t F_{k-1} + \tfrac{1}{2!}\Delta t^2 F_{k-1}^2 + \tfrac{1}{3!}\Delta t^3 F_{k-1}^3 + \ldots) X_{k-1} \\ \Phi_{k-1} &= e^{F_{k-1}\Delta t} = I + \Delta t F_{k-1} + \tfrac{1}{2!}\Delta t^2 F_{k-1}^2 + \tfrac{1}{3!}\Delta t^3 F_{k-1}^3 + \ldots \\ X_k &\approx \Phi_{k-1} X_{k-1} \end{aligned} \tag{3.41}$$

This special approximation makes the extended Kalman filter algorithm very close to the standard linear 'Propagation Step' with the addition of iterative calculation of Φ_{k-1}. The extended Kalman filter equations are summarized in Table 3.2

Discrete -Continuous	
Propagation Equations	Update Equations
$\dot{\hat{X}}(t) = f(\hat{X}(t))$	$K_k = \mathbf{P}_{k\vert k-1} H_k^T [H_k \mathbf{P}_{k\vert k-1} H_k^T + R_k]^{-1}$
$\dot{\mathbf{P}}(t) = F(\hat{x}(t))\mathbf{P}(t) + \mathbf{P}(t) F(\hat{x}(t))^T + Q(t)$	$r_k = z_k - H_k \hat{X}_{k\vert k-1}$
$F(\hat{X}(t)) = \left.\dfrac{\partial f(X)}{\partial X}\right\vert_{\hat{X}(t)}$	$\hat{X}_{k\vert k} = \hat{X}_{k\vert k-1} + K_k * r_k$
	$\mathbf{P}_{k\vert k} = \mathbf{P}_{k\vert k-1} - K_k H_k \mathbf{P}_{k\vert k-1}$
Discrete Approximation for: $f(X(t)) = F(t)X(t)$	
Propagation Equations	Update Equations
$\hat{X}_{k\vert k-1} \approx \Phi_{k-1} \hat{X}_{k-1\vert k-1}$	$K_k = \mathbf{P}_{k\vert k-1} H_k^T [H_k \mathbf{P}_{k\vert k-1} H_k^T + R_k]^{-1}$
$\mathbf{P}_{k\vert k-1} \approx \Phi_{k-1} \mathbf{P}_{k-1\vert k-1} \Phi_{k-1}^T + Q_{k-1}$	$r_k = z_k - H_k \hat{X}_{k\vert k-1}$
$F(\hat{X}(t)) = \left.\dfrac{\partial f(X)}{\partial X}\right\vert_{\hat{X}(t)}$	$\hat{X}_{k\vert k} = \hat{X}_{k\vert k-1} + K_k * r_k$
$\Phi_{k-1} \approx e^{F(t_{k-1})[t_k - t_{k-1}]}$	$\mathbf{P}_{k\vert k} = \mathbf{P}_{k\vert k-1} - K_k H_k \mathbf{P}_{k\vert k-1}$

Table 3.2: Extended Kalman Filter Equations

3.3 State Space Models for Ground Vehicle Dynamics

The models are based on two classic linear-Gaussian target-tracking models [2][10] and two new nonlinear-Gaussian target-tracking models. Note that four models are used instead of one definitive model that incorporates the full set of expected behaviors. This is done because the unknown driver actions are obscured by noise from the sensor. In the absence of noise, a definitive model would return a good estimate of a simple maneuver [2]. In the presence of noise, however, the more complex models are sensitive to the sensor noise and states that should be zero are estimated as non-zero values simply because of their inclusion in the model. For example, if the vehicle is going straight, the model that just includes enough states to describe straight-line travel will perform the best. The models are expressed in the forms shown by (3.4) and (3.5):

General-Gaussian Model

$$\dot{X} = f(X(t)) + w(t)$$

Linear-Gaussian Models

$$f(X) = F(t)X(t)$$
$$\dot{X} = F(t)X(t) + w(t)$$

The following model expressions and state vectors will use the state variable notations: P for position, V for velocity, A for acceleration and J for jerk. The following subscripts define the direction of the state variable Cartesian components: subscripts x and y specify the components in a North-East-Down (NED) earth fixed coordinate frame. Subscripts T and L define the components in a rotating velocity fixed coordinate frame and stand for 'Tangential' and 'Lateral' respectively [3].

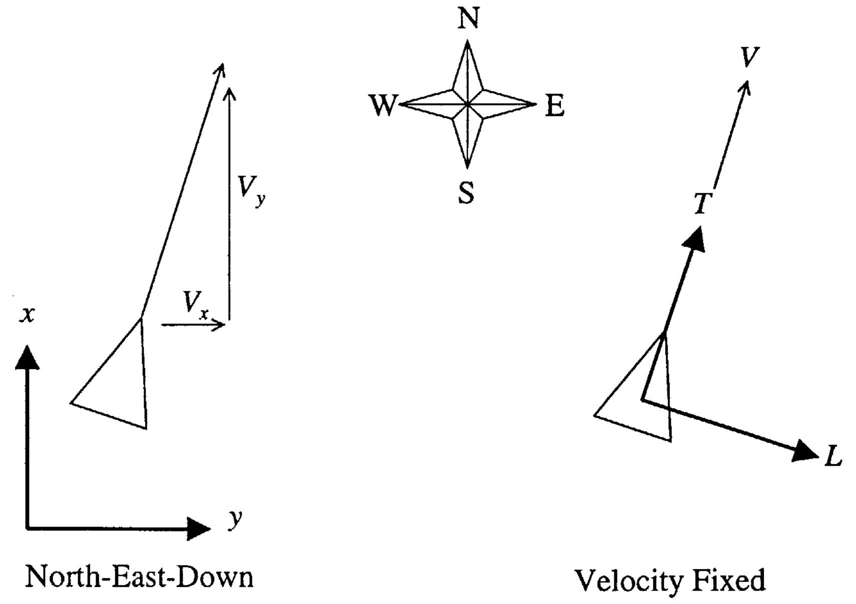


Figure 3.2: Coordinate Frames

The NED coordinate frame has the origin fixed at some local point and the orientation fixed to the compass rose as shown in Figure 3.2 with the z-axis pointing down toward the center of the earth. The velocity fixed coordinate frame has the origin fixed on the vehicle and the orientation defined with the T-axis fixed on the vehicle's velocity vector. Following the right hand rule: T is the "x-axis", L is the "y-axis" and the z-axis is pointing down toward the center of the earth.

3.3.1 Random Walk Velocity Model

Here the Vehicle is modeled as traveling at an approximately constant velocity in magnitude and direction. The random walk portion models the uncertainty of deviations from the constant velocity by unknown driver maneuvering. These maneuvers are analogous to using the accelerator/ brake pedal and the steering wheel, but are expected to be within a Gaussian envelope of zero mean.

The Random Walk Velocity Model or (RWV) is:

$$\dot{X} = \begin{pmatrix} \dot{P}_x \\ \dot{P}_y \\ \dot{V}_x \\ \dot{V}_y \end{pmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} P_x \\ P_y \\ V_x \\ V_y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ w_x \\ w_y \end{pmatrix} \tag{3.42}$$

3.3.2 Random Walk Acceleration Model

Here the Vehicle is modeled as traveling at an approximately constant acceleration in magnitude and direction. This builds on (3.42). The random walk portion models the uncertainty of deviations from the constant acceleration by unknown driver maneuvering. These are changes in the acceleration rate (jerk) which are analogous to changes in the rate of acceleration/ braking combined with steering acceleration changes. Again, these are expected to be within a Gaussian envelope of zero mean.

The Random Walk Acceleration Model or (RWA) is:

$$\dot{X}=\begin{pmatrix}\dot{P}_x\\ \dot{P}_y\\ \dot{V}_x\\ \dot{V}_y\\ \dot{A}_x\\ \dot{A}_y\end{pmatrix}=\begin{bmatrix}0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\\0&0&0&0&0&0\\0&0&0&0&0&0\end{bmatrix}\begin{pmatrix}P_x\\ P_y\\ V_x\\ V_y\\ A_x\\ A_y\end{pmatrix}+\begin{pmatrix}0\\0\\0\\0\\w_x\\w_y\end{pmatrix} \tag{3.43}$$

3.3.3 Random Walk Velocity and Lateral Acceleration Model

Here the Vehicle is modeled as traveling approximately in a circle prescribed by a constant lateral acceleration and constant speed or velocity magnitude. The lateral acceleration $\mathrm{A_L}$ that a vehicle experiences is also equivalent to instantaneous centripetal acceleration, which can be expressed as a function of speed $\|\mathrm{V}\|$ and turn radius R.

$$A_L = Centripetal\,Acceleration = \frac{\|V\|^2}{R} \tag{3.44}$$

For a standard wheeled vehicle the lateral acceleration R is a function of vehicle speed, the angle of the front wheels θ, and wheel base length WB, see Figure 3.3.

$$R=\frac{\mathrm{WB}}{\tan\theta} \tag{3.45}$$

The equation for a model for lateral acceleration for an idealized-wheeled vehicle without tire slippage is shown.

$$A_\mathrm{L}=\frac{\|\mathrm{V}\|^2\tan\theta}{\mathrm{WB}} \tag{3.46}$$

A_L is chosen as the state variable over R or θ, because of its simplicity, consistency with (3.43) and because it is divorces the model from vehicle specifics.

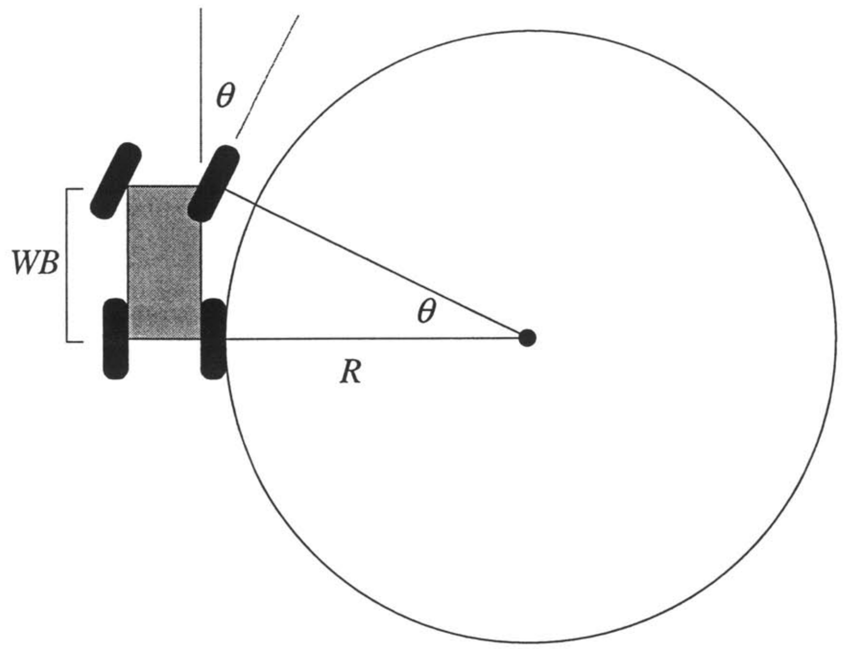


Figure 3.3: Ground Vehicle Lateral Acceleration

The nonlinearities introduced into this model arise from the transformation of A_L, expressed in the non-inertial velocity fixed coordinate frame, to the NED coordinate frame expressed as the vector $\dot{V}$. The velocity fixed component T is always perpendicular to the component L. Thus the coordinate transformation occurs by multiplying A_L times a unit vector that is perpendicular to V since in fixed to the T-axis, see Figure 3.2. This is summarized by the equation:

$$\begin{pmatrix} \dot{V}_x \\ \dot{V}_y \end{pmatrix} = \frac{1}{\|V\|} \begin{pmatrix} -V_y \\ V_x \end{pmatrix} A_L \tag{3.47}$$

The sign convention for the Velocity Fixed coordinate frame makes a constant positive velocity combined with a constant positive lateral acceleration describes a clockwise circle, see Figure 3.2 and Figure 3.3.

For this state space representation the random walk process models the uncertainty of deviations from constant lateral acceleration and constant speed. The uncertainty in speed is not expressed in a tangential direction although the intent of its inclusion is to allow the tangential velocity component room to breath.

Applying (3.47) to (3.42) results in the Random Walk Velocity and Lateral Acceleration Model or (RWVLA):

$$\dot{X} = \begin{pmatrix} \dot{P}_x \\ \dot{P}_y \\ \dot{V}_x \\ \dot{V}_y \\ \dot{A}_L \end{pmatrix} = \begin{bmatrix} V_x \\ V_y \\ \frac{-V_y}{\|V\|} A_L \\ \frac{V_x}{\|V\|} A_L \\ 0 \end{bmatrix} + \begin{pmatrix} 0 \\ 0 \\ w_x \\ w_y \\ w_L \end{pmatrix} \tag{3.48}$$

The model constrains the unknown maneuvering jerk to be in terms of steering changes only. This is a good way to constrain the random walk process to something more realistic than expecting jerk in any random direction, as discussed in Section 3.1. This model constraint is useful since much driving is done at a constant speed but with steering adjustments. The complexity of mixing coordinate frames in this model is justified by the fact that a constant turning trajectory can now be estimated. This is a useful expression since the lateral acceleration can be a constant value even though it is rotating smoothly with respect to the earth.

As an alternative to the form in (3.48), the RWVLA model may be expressed with the velocity vector in a polar coordinate NED form[1]. However, the model approach presented above has an advantage in its linearized form. This will be discussed further in Section 3.4.

1 A polar coordinate expression replaces V_x and V_y with states for velocity magnitude $\|V\|$ and orientation angle ϕ. Polar form allows speed uncertainty to be expressed correctly in the tangential direction. It is also straightforward to express the effect of the lateral acceleration, pure rotation of the velocity vector, as the differential equation: $\dot{\phi} = A_L / \|V\|$.

3.3.4 Random Walk Lateral Jerk and Tangential Acceleration Model

This model is an extension of (3.48). Here the vehicle is modeled as traveling with approximately a constant lateral jerk and a constant tangential acceleration. Again, nonlinearities are introduced into this model from the transformation of states expressed in velocity fixed coordinates to NED coordinates. In this case A_L and A_T are the states being transformed. The transformation is similar to that in Section 3.3.3. It occurs by multiplying A_L by a unit vector perpendicular to the velocity vector, and multiplying A_T by a unit vector parallel to the velocity vector. This is summarized by the equation:

$$\begin{pmatrix} \dot{V}_x \\ \dot{V}_y \end{pmatrix} = \frac{1}{\|V\|}\begin{pmatrix} -V_y \\ V_x \end{pmatrix} A_L + \frac{1}{\|V\|}\begin{pmatrix} V_x \\ V_y \end{pmatrix} A_T \tag{3.49}$$

The random walk process now models the uncertainties of deviations from constant lateral jerk and constant tangential acceleration. Again the lateral and tangential terms are components of a vehicle fixed coordinate frame, see Figure 3.2. The maneuvering uncertainties are analogous to changes in the rate of acceleration/braking or changes in steering acceleration.

Applying (3.49) to (3.42) results in the Random Walk Lateral Jerk and Tangential Acceleration Model or (RWLJTA):

$$\dot{X} = \begin{pmatrix} \dot{P}_x \\ \dot{P}_y \\ \dot{V}_x \\ \dot{V}_y \\ \dot{A}_L \\ \dot{J}_L \\ \dot{A}_T \end{pmatrix} = \begin{pmatrix} V_x \\ V_y \\ -\frac{V_y}{\|V\|}A_L + \frac{V_x}{\|V\|}A_T \\ +\frac{V_x}{\|V\|}A_L + \frac{V_y}{\|V\|}A_T \\ J_L \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ w_L \\ w_T \end{pmatrix} \qquad X = \begin{pmatrix} P_x \\ P_y \\ V_x \\ V_y \\ A_L \\ J_L \\ A_T \end{pmatrix} \tag{3.50}$$

This model, however, is inadequate to describe a constant radius turn R with a constant tangential acceleration A_T.

Given (3.44):

$$A_L = \frac{\|V\|^2}{R}$$

The time derivative of lateral acceleration for a constant radius turn is:

$$\dot{R} = 0$$

$$\dot{A}_L = J_L = \frac{2\|V\|\|\dot{V}\|}{R} \tag{3.51}$$

Tangential acceleration is defined as:

$$A_T \equiv \|\dot{V}\| \tag{3.52}$$

By substituting (3.52) into (3.51).

$$J_L = \frac{2\int A_T dt\ A_T}{\mathrm{R}} \tag{3.53}$$

This shows that the two states A_T and J_L are coupled when $\dot{R} = 0$. Thus, for a constant tangential acceleration and constant radius turn, there is a linearly changing lateral jerk. This model is not adequate to propagate such a scenario. It is an improvement for such a scenario over previously discussed models, but it is not complete. This is a considerable drawback of using this model since accelerating through a constant radius turn is a scenario that is likely to occur.

3.4 Model Discretization and Linearization

The $F(t)$ part of the linear models and the $f(X(t))$ part of the nonlinear models derived in Section 3.3 form the basis for the Kalman filter algorithm. In order to be implemented in the Kalman filter algorithms, these parts must be discretized and the nonlinear models must also be linearized. That is accomplished in the following sections.

It is also important that the $w(t)$ vector be discretized and its discrete covariance description Q_{k-1} be found for the Kalman filter algorithm. This can be generalized for all models as follows. The continuous input noise covariance matrix $Q(t)\delta(t-\tau)$ is related to Q_{k-1} as in (3.32):

$$Q_{k-1} = \int_{t_{k-1}}^{t_k} \Phi_{k|\tau} Q(t) \Phi_{k|\tau}^T d\tau$$

Referring to [6] and [2], for a sufficiently small time step Δt an approximation for Q_{k-1} is given by:

$$\begin{aligned} \Delta t &= t_k - t_{k-1} \\ Q_{k-1} &\approx Q(t)\Delta t \end{aligned} \tag{3.54}$$

Thus Q_{k-1} has corresponding nonzero entries with $Q(t)$ and we know which entries of Q_{k-1} to calculate. Also since Q_{k-1} is defined in (3.13) as:

$$Q_{k-1} = E\{w_{k-1}w_{k-1}^T\}$$

We know that the vector w_{k-1} has corresponding nonzero entries with vector $w(t)$. Since we now know the form of w_{k-1} and Q_{k-1} we can concentrate on calculating their values in lieu of a discretization from $w(t)$ and $Q(t)$ values. This is done by the Singer method [3]. The following sections detail the formulation of an analytical starting point for the covariance matrices Q_{k-1}.

3.4.1 Discrete Random Walk Velocity Model

The random walk velocity model is very simple to discretize by evaluating (3.9). It is an exact discretization since the terms F^2 and higher are zero matrices. This results in the state transition matrix:

$$\Phi_{k-1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.55}$$

Q_{k-1} values are chosen for the RWV model by the following analytical means. The state vector X represents a decoupled velocity pair driven by uncorrelated noises w_x and w_y. Since in this model the vehicle is equally likely to maneuver in any direction in the NED coordinate frame, we will say that their Q_{k-1} values are equivalent for the pair. So from the discussion at the beginning of Section 3.4:

$$Q_{k-1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & Q_V & 0 \\ 0 & 0 & 0 & Q_V \end{bmatrix} \tag{3.56}$$

The general discrete equation for the velocity random walk is:

$$V_{k+1} = V_k + w_{V_k} \tag{3.57}$$

The meaning of w_{V_k} comes from a truncated Taylor expansion:

$$V_{k+1} = V_k + \Delta t A_k \tag{3.58}$$

$$w_{V_k} = \Delta t A_k \tag{3.59}$$

The noise term is a model for the change in velocity due to an unknown constant maneuvering acceleration over the time step Δt. The value Q_v represents the variance for velocity change due unknown vehicle acceleration. All the state space models were constructed with this end in mind [2] and [10].

$$E[w_{V_k}^2] = \Delta t^2 E[A_k^2] \tag{3.60}$$

We shall take for this expected variance of A_k, the maximum value that a ground vehicle's tires can likely achieve.

$$Q_V = \Delta t^2 A_{MAX}^2 \tag{3.61}$$

This is the Singer method [3]. Since w_{V_k} is unbiased, this is equivalent to saying that $\Delta t A_{MAX}$ is equal to one standard deviation of the Gaussian distribution function that describes w_{V_k}, see Figure 3.4.

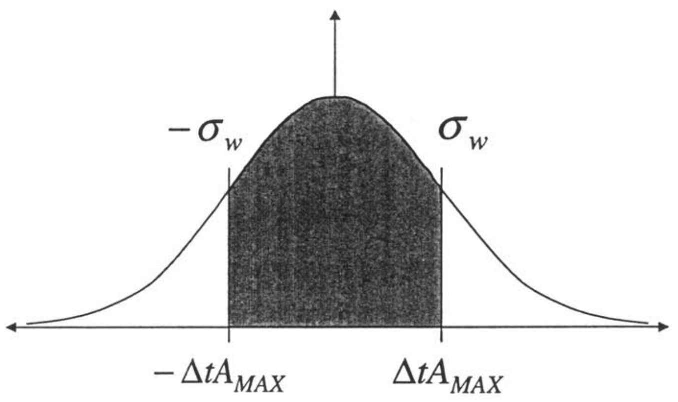


Figure 3.4: Gausisan Noise Modeling – Singer Method

This is a rough guideline and is very conservative since much of the area under the Gaussian curve is beyond the capability of ground vehicle limits. The final value for Q_V may be perturbed from this analytic starting point to tune the performance of the total filter.

3.4.2 Discrete Random Walk Acceleration Model

The random walk acceleration model discretization by evaluating (3.9) is also very simple since the terms F^3 and higher are similarly zero matrices. This results in the state transition matrix:

$$\Phi_{k-1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.62}$$

For the calculation of Q_{k-1}, we will again consider the w_x, w_y noise pair to have equivalent variances Q_A in the NED coordinate frame.

$$Q_{k-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_A & 0 \\ 0 & 0 & 0 & 0 & 0 & Q_A \end{bmatrix} \tag{3.63}$$

As discussed in Section 3.1, the modeling intent for the random walk process is to cover uncertainties in the derivative of the random walk variable. For this model we see the general discrete equation for the acceleration random walk is:

$$A_{k+1} = A_k + w_{V_k} \tag{3.64}$$

The noise variable w_{A_k} is:

$$w_{A_k} = \Delta t J_k \tag{3.65}$$

So the noise term is a model for the change in acceleration due to an unknown constant maneuvering jerk of the vehicle over the time period Δt. The value Q_A equals the variance of vehicle jerk we might approximate to be experienced.

$$Q_A = \Delta t^2 E[J_k^2] = \Delta t^2 J_{MAX}^2 \tag{3.66}$$

One approach in determining J_{MAX} is to approximate a derivative by dividing $2A_{MAX}$ by how long T it would take to move from one A_{MAX} extreme to the other using only the steering wheel.

$$J_{MAX} = \frac{2A_{MAX}}{T} \tag{3.67}$$

3.4.3 Discrete Random Walk Lateral Acceleration Model

Finding the discrete version of this model is much more involved because the basis model is both continuous and nonlinear. Linearization is a simple process of taking the partial derivative matrix of the $f(X(t))$ vector. The is from (3.33):

$$F(t) = \left. \frac{\partial f(X)}{\partial X} \right|_{X(t)}$$

For each of the functions we find the partial derivatives. For the difficult partial derivatives of $\dot{V}_x$ and $\dot{V}_y$ this is most easily done by finding a general expression for the intermediary partial derivative for $\|V\|$:

$$\begin{aligned} \|V\| &= \sqrt{V_x^2 + V_y^2} \\ \frac{\partial}{\partial V_i}\left(\frac{1}{\|V\|}\right) &= \frac{-V_i}{\|V\|^3} \end{aligned} \tag{3.68}$$

We use this to find the first partial derivative of $\dot{V}_x$ with respect to V_x:

$$\begin{aligned} \frac{\partial f(\dot{V}_x)}{\partial V_x} &= -V_y A_L \frac{\partial}{\partial V_x}\left(\frac{1}{\|V\|}\right) \\ \frac{\partial f(\dot{V}_x)}{\partial V_x} &= \frac{V_x V_y A_L}{\|V\|^3} \end{aligned} \tag{3.69}$$

Next we find the partial derivative with respect to V_y using the product rule:

$$\begin{aligned} \frac{\partial f(\dot{V}_x)}{\partial V_y} &= -A_L\left[V_y \frac{\partial}{\partial V_y}\left(\frac{1}{\|V\|}\right) + \frac{1}{\|V\|}\right] \\ \frac{\partial f(\dot{V}_x)}{\partial V_y} &= \frac{-V_x^2 A_L}{\|V\|^3} \end{aligned} \tag{3.70}$$

leading to a partial derivative matrix:

$$F(t) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{V_x V_y A_L}{\|V\|^3} & \frac{-V_x^2 A_L}{\|V\|^3} & \frac{-V_y}{\|V\|} \\ 0 & 0 & \frac{V_y^2 A_L}{\|V\|^3} & \frac{-V_x V_y A_L}{\|V\|^3} & \frac{V_x}{\|V\|} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.71}$$

It is important to note that when multiplied by the current state vector there is cancellation in terms that results in the original nonlinear model $f(X(t))$. This allows us to use the special discrete version of the extended Kalman filter developed in Section 3.2.2.

$$F(t)X(t) = \begin{bmatrix} V_x \\ V_y \\ \dfrac{-V_y}{\|V\|}A_L \\ \dfrac{V_x}{\|V\|}A_L \\ 0 \end{bmatrix} = f(X(t)) \tag{3.72}$$

The discretization by evaluating (3.41) is carried to the F^3 term. The discretization has no series truncation error since the term F^4 and higher are zero matrices. This results in the state transition matrix:

$$\Phi_{k-1} = I + \Delta t F_{k-1} + \tfrac{1}{2!}\Delta t^2 F_{k-1}^2 + \tfrac{1}{3!}\Delta t^3 F_{k-1}^3 \tag{3.73}$$

$$\Phi_{k-1} = \begin{bmatrix} 1 & 0 & \Delta t + \frac{1}{2}\Delta t^2 \dfrac{V_x V_y A_L}{\|V\|^3} & -\frac{1}{2}\Delta t^2 \dfrac{V_x^2 A_L}{\|V\|^3} & -\frac{1}{2}\Delta t^2 \dfrac{V_y}{\|V\|} - \frac{1}{6}\Delta t^3 \dfrac{(V_x V_y^2 + V_x^3)A_L}{\|V\|^4} \\ 0 & 1 & \frac{1}{2}\Delta t^2 \dfrac{V_y^2 A_L}{\|V\|^3} & \Delta t - \frac{1}{2}\Delta t^2 \dfrac{V_x V_y A_L}{\|V\|^3} & \frac{1}{2}\Delta t^2 \dfrac{V_x}{\|V\|} - \frac{1}{6}\Delta t^3 \dfrac{(V_y V_x^2 + V_y^3)A_L}{\|V\|^4} \\ 0 & 0 & 1 + \Delta t \dfrac{V_x V_y A_L}{\|V\|^3} & \dfrac{-V_x^2}{\|V\|^3}\Delta t A_L & -\Delta t \dfrac{V_y}{\|V\|} - \frac{1}{2}\Delta t^2 \dfrac{(V_x V_y^2 + V_x^3)A_L}{\|V\|^4} \\ 0 & 0 & \dfrac{V_y^2}{\|V\|^3}\Delta t A_L & 1 - \dfrac{V_x V_y}{\|V\|^3}\Delta t A_L & \Delta t \dfrac{V_x}{\|V\|} - \frac{1}{2}\Delta t^2 \dfrac{(V_y V_x^2 + V_y^3)A_L}{\|V\|^4} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{\hat{x}_k}$$

For the calculation of discretized noise covariance matrix Q_{k-1}, as with the RWV model, we consider the NED coordinate frame velocity pair to be driven by noise w_x and w_y. Unlike the RWV model, this input noise is included to model only the tangential uncertainties. However, since velocity is still expressed in NED coordinates; the model can't discriminate tangential direction and so w_x and w_y are represented as equivalent variances Q_V. These covariance matrix values Q_V are assumed to be the same as in the RWV model.

$$Q_{k-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Q_V & 0 & 0 \\ 0 & 0 & 0 & Q_V & 0 \\ 0 & 0 & 0 & 0 & Q_{A_L} \end{bmatrix} \tag{3.74}$$

New to this model is the concept of the velocity fixed coordinate frame. Now the noise w_L is constrained to the lateral direction with respect to the vehicle. The discrete equation for this lateral acceleration random walk is:

$$A_{L_{k+1}} = A_{L_k} + w_{L_k} \tag{3.75}$$

in comparison to a truncated Taylor series:

$$A_{L_{k+1}} = A_{L_k} + \Delta t J_{L_k} \tag{3.76}$$

it is seen that w_{L_k} is :

$$w_{L_k} = \Delta t J_{L_k} \tag{3.77}$$

The discrete noise is a model for the change in lateral acceleration due to an unknown constant lateral maneuvering jerk. Q_{A_L} is equal to the expected value of the noise.

$$E[w_{L_k}^2] = \Delta t^2 E[J_{L_k}^2] \tag{3.78}$$

By the Singer method, we will take the variance for the lateral jerk to be the maximum lateral jerk squared.

$$Q_{A_L} = \Delta t^2 J_{L_{MAX}}^2 \tag{3.79}$$

The value for $J_{L_{MAX}}$ is subject to approximation.

3.4.4 Discrete Random Walk Lateral Jerk and Tangetial Acceleration Model

As shown for the previous continuous nonlinear model, the partial derivative matrix must first be calculated to linearize the model before a discrete approximation can be calculated. Again using (3.68) we find that the partial derivative of $\dot{V}_x$ with respect to V_x:

$$\begin{aligned} \frac{\partial f(\dot{V}_x)}{\partial V_x} &= -V_y A_L \frac{\partial}{\partial V_x}\left(\frac{1}{\|V\|}\right) + A_T\left[V_x \frac{\partial}{\partial V_x}\left(\frac{1}{\|V\|}\right) + \frac{1}{\|V\|}\right] \\ \frac{\partial f(\dot{V}_x)}{\partial V_x} &= \frac{V_y(V_x A_L + V_y A_T)}{\|V\|^3} \end{aligned} \tag{3.80}$$

Next we find the partial derivative with respect to V_y using the product rule:

$$\begin{aligned} \frac{\partial f(\dot{V}_x)}{\partial V_y} &= -A_L\left[V_y \frac{\partial}{\partial V_y}\left(\frac{1}{\|V\|}\right) + \frac{1}{\|V\|}\right] + V_x A_T \frac{\partial}{\partial V_y}\left(\frac{1}{\|V\|}\right) \\ \frac{\partial f(\dot{V}_x)}{\partial V_y} &= \frac{-V_x(V_x A_L + V_y A_T)}{\|V\|^3} \end{aligned} \tag{3.81}$$

Leading to a partial derivative matrix:

$$F(t) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{V_y(V_x A_L + V_y A_T)}{\|V\|^3} & \frac{-V_x(V_x A_L + V_y A_T)}{\|V\|^3} & -\frac{V_y}{\|V\|} & 0 & \frac{V_x}{\|V\|} \\ 0 & 0 & \frac{-V_y(-V_y A_L + V_x A_T)}{\|V\|^3} & \frac{V_x(-V_y A_L + V_x A_T)}{\|V\|^3} & \frac{V_x}{\|V\|} & 0 & \frac{V_y}{\|V\|} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.82}$$

As with the previous continuous nonlinear model, this partial derivative matrix exactly represents the original nonlinear form:

$$F(t)X(t) = f(X(t)) \tag{3.83}$$

From here we make the same argument that the matrix representation $F(t)$ is approximate for small deviation from the evaluation point and so the transition matrix Φ_{k-1} corresponding to F_{k-1} is a valid approximation of the transition matrix for a small time step ahead of the evaluation point. This allows us to use the special discrete version of the extended Kalman filter developed in Section 3.2.2.

Referring to (3.41):

$$\Phi_{k-1} = e^{F_{k-1}\Delta t} \approx I + \Delta t F_{k-1} + \tfrac{1}{2!}\Delta t^2 F_{k-1}^2 + \tfrac{1}{3!}\Delta t^3 F_{k-1}^3 \tag{3.84}$$

In this case the series expression for Φ_{k-1} has an infinite number of nonzero terms. It is necessary to truncate the series for calculation of a reasonable estimate. The powers of F above F^3 involve the same 4×4 nonzero entities, as seen in (3.85) and (3.86).

$$F = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & F_i & F_i & F_i & 0 & F_i \\ 0 & 0 & F_i & F_i & F_i & 0 & F_i \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} F^2 = \begin{bmatrix} 0 & 0 & F_i^2 & F_i^2 & F_i^2 & 0 & F_i^2 \\ 0 & 0 & F_i^2 & F_i^2 & F_i^2 & 0 & F_i^2 \\ 0 & 0 & F_i^2 & F_i^2 & F_i^2 & F_i^2 & 0 \\ 0 & 0 & F_i^2 & F_i^2 & F_i^2 & F_i^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.85}$$

$$F^3 = \begin{bmatrix} 0 & 0 & F_i^3 & F_i^3 & F_i^3 & F_i^3 & 0 \\ 0 & 0 & F_i^3 & F_i^3 & F_i^3 & F_i^3 & 0 \\ 0 & 0 & F_i^3 & F_i^3 & F_i^3 & F_i^3 & 0 \\ 0 & 0 & F_i^3 & F_i^3 & F_i^3 & F_i^3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} F^4 = \begin{bmatrix} 0 & 0 & F_i^4 & F_i^4 & F_i^4 & F_i^4 & 0 \\ 0 & 0 & F_i^4 & F_i^4 & F_i^4 & F_i^4 & 0 \\ 0 & 0 & F_i^4 & F_i^4 & F_i^4 & F_i^4 & 0 \\ 0 & 0 & F_i^4 & F_i^4 & F_i^4 & F_i^4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.86}$$

We will truncate the series at the F^3 term in this case. This is the same number of Taylor series terms as the RWVLA model. Although the series is truncated, all the matrix entries that will ever be filled are filled with at least one term. Beyond this approximations terms consistently decrease on the order of an exponential rate due to the Δt^n portion of the term. Refer to the appendix to see the full algebraic expression of Φ_{k-1}.

The discretization of the noise covariance matrix Q_{k-1} leaves us with a sparse matrix describing the expected variance of the discrete expression of noises w_L and w_T.

$$Q_{k-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & Q_{J_L} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Q_{A_T} \end{bmatrix} \tag{3.87}$$

The equations for the random walk part of this model are:

$$J_{L_{k+1}} = J_{L_k} + w_{L_k} \tag{3.88}$$

$$A_{T_{k+1}} = A_{T_k} + w_{T_k} \tag{3.89}$$

We compare these equations to a truncated Taylor series for the propagated state as seen for the discretization of previous models:

$$J_{L_{k+1}} = J_{L_k} + \Delta t \dot{J}_{L_k} \tag{3.90}$$

$$A_{T_{k+1}} = A_{T_k} + \Delta t J_{T_k} \tag{3.91}$$

We can equate these equations to find the expression for the noise terms:

$$w_{L_k} = \Delta t \dot{J}_{L_k} \tag{3.92}$$

$$w_{T_k} = \Delta t J_{T_k} \tag{3.93}$$

The noise terms describe the state change over the time interval. This state change is due to an unknown constant rate applied over that interval. The entries in the covariance matrix Q are defined as the expected values of the square of the noise vector. This results in:

$$E[w_{L_k}^2] = \Delta t^2 E[\dot{J}_{L_k}^2] \tag{3.94}$$

$$E[w_{T_k}^2] = \Delta t^2 E[J_{T_k}^2] \tag{3.95}$$

The Singer method is applied to relate the expected values to a maximum value for that parameter. The final forms of the analytic expressions for the entries of the Q matrix are:

$$Q_{J_L} = \Delta t^2 \dot{J}_{LMAX}^2 \tag{3.96}$$

$$Q_{A_T} = \Delta t^2 J_{TMAX}^2 \tag{3.97}$$

The values for these covariance entries are subject to approximation. For example, the maximum value for tangential jerk is very large. The time that it takes for a driver and vehicle to change from zero acceleration to full braking in very small. We may approximate that such an action could be completed on the order of one time step of the algorithm or .2 seconds. Thus the value for $J_{T_{MAX}}$ would be very large. The maximum value for the time rate of lateral acceleration is also potentially large. We may approximate that a driver using a steering wheel may able to go from a fixed wheel position to a constant hand-over-hand rate on the order of one time step. This action maps approximately to a step change in lateral jerk. This demonstrates the very high value that $\dot{J}_{L_{MAX}}$ may be.

3.5 Measurement Model

The measurement model is common for all of the Kalman filters. All of the filters receive the same input, target position, and they all use the same coordinate frame expression of position in their state vector. Further, we carry out coordinate transformations from a global 3-dimensional form to a local 2-dimensional form before it enters the Kalman filter algorithm, see Section 2.3. This rids the measurement vector z_k of the extraneous altitude component and allows the expectation for sensor noise v_k to be specified in an intuitive 2-dimentional form. This gives a very straightforward linear model of the form specified by (3.12)

Linear-Measurement Model

$$z_k = H_k X_k + v_k$$

$$z_k = \begin{bmatrix} 1 & 0 & 0 & 0\ldots \\ 0 & 1 & 0 & 0\ldots \end{bmatrix} \begin{pmatrix} P_{x_k} \\ P_{y_k} \\ V_{x_k} \\ V_{y_k} \\ \vdots \end{pmatrix} + \begin{pmatrix} v_k \\ v_k \end{pmatrix} \tag{3.98}$$

The H_k matrix varies in dimension for each filter. The number of terminal zero columns must change to match the dimension required by the vector multiplication of each state vector. Essentially, the model is identical for each filter.

3.6 Statistical Performance Scoring

An important part of the system operation is that of selecting the model that best represents the dynamics of the ground target. This is accomplished by comparing the statistics of the residual vector r_k of each Kalman filter as given by (3.17)

$$\begin{aligned} \hat{y}_k &= H_k \hat{X}_k \\ r_k &= z_k - \hat{y}_k \end{aligned}$$

The residual vector represents the difference between the estimated output vector $\hat{y}_k$ and the actual measurement vector z_k. This is apart from the estimate error $\tilde{y}_k$, which compares the estimated output vector to the true output vector.

$$\begin{aligned} y_k &= H_k X_k \\ \tilde{y}_k &= \hat{y}_k - y_k \end{aligned} \tag{3.99}$$

The measurement vector can be described by:

$$z_k = H_k X_k + v_k = y_k + v_k \tag{3.100}$$

Thus, perfect estimation would result in $\tilde{y}_k$ equal to zero and residuals that are identical to the sensor noise and no smaller.

$$r_k = z_k - H_k \hat{X}_k = v_k - \tilde{y}_k \tag{3.101}$$

Individual residual values that are less than or greater than sensor noise values are so because of estimation error and chance.

3.6.1 Statistical Measures

The filter with the 'best behaved' residual vector has the best model of the ground target. 'Best behaved' in a statistical sense can involve different measures. Analyzing the residual magnitude mean gives an approximation of how the residual vector is biased from a desired zero value. Analyzing the residual magnitude variance approximates how precise the measurement may be.

Analyzing the Root-Mean-Square (RMS) of the magnitude of the residual vector approximates both bias and precision and gives a good indication of overall accuracy within a single number.

$$E[r_k^2] = E[v_k^2] - 2E[v_k \tilde{y}_k] + E[\tilde{y}_k^2] \tag{3.102}$$

Since v_k is white noise:

$$\begin{aligned} &E[v_k \tilde{y}_k] = 0 \\ &RMS_{r_k} \equiv \sqrt{E[r_k^2]} \\ &RMS_{r_k} = \sqrt{E[v_k^2] + E[\tilde{y}_k^2]} \end{aligned} \tag{3.103}$$

Since the filters all experience the same sensor noise, the filter with the smallest estimation error will also have the smallest RMS value. Smallest estimation error does not automatically mean 'best behaved'. A system that has a relatively constant bias error of +1 will have the same RMS value as one that alternates between -1 and +1, but the former would seem to be better behaved. Since the derivation of the Kalman filter assumes the estimation error is unbiased, we claim that RMS analysis is adequate under the assumption.

3.6.2 Scoring System

The RMS scoring approach is used in the scoring system. For implementation purposes it is implemented in a fixed window recursive form. Using a fixed window idea removes the effect of the oldest residual term out of the computation at each iteration resulting in the RMS calculation of a fixed number of most recent terms. Additionally a recursive approach does not require the actual storage of previous residual values but updates the previous RMS value with respect to the new residual term.

The recursive RMS algorithm is derived as follows from the definition of RMS:

$$\begin{aligned} &RMS_{r_k} \equiv \sqrt{E[r_k^2]} \approx \sqrt{\frac{1}{k}\sum_{i=1}^{k} r_i^2} \\ &RMS_{r_{k+1}} \approx \sqrt{\frac{1}{k+1}\sum_{i=1}^{k+1} r_i^2} = \sqrt{\frac{k}{k+1}\frac{1}{k}\left(r_{k+1}^2 + \sum_{i=1}^{k} r_i^2\right)} \\ &RMS_{r_{k+1}} \approx \sqrt{\frac{k}{k+1} RMS_{r_k}^2 + \frac{1}{k+1} r_{k+1}^2} \end{aligned} \tag{3.104}$$

This residual algorithm is used to calculate the RMS from all the residual samples. This algorithm is also useful in calculating RMS initially until the sample size reaches the window size when a switch is made to the fixed window recursive form. This is done so that early poor initial RMS values aren't exaggerated in the recursive propagation.

The fixed window recursive RMS algorithm substitutes the window size ‘N’ for the sample count ‘k’.

$$RMS_{r_{k+1}} \approx \sqrt{\frac{N}{N+1} RMS_{r_k}^2 + \frac{1}{N+1} r_{k+1}^2} \tag{3.105}$$

The choice of ‘N’ is a design decision finding an acceptable tradeoff between accuracy of RMS value and agility in registering changes in statistical trends. Since this application has a dimensional residual vector, we will find the RMS value of the magnitude of the residual vector

$$\|r\|_{k+1} = \sqrt{r_x^2 + r_y^2} \tag{3.106}$$

$$S = RMS_{\|r\|_{k+1}} \tag{3.107}$$

4 System Simulation

This chapter describes the implementation of the ADTR in a targeting system simulation as shown in Figure 4.1.

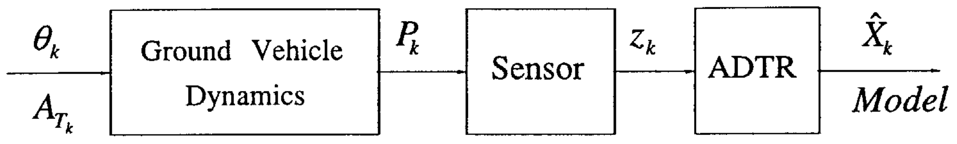


Figure 4.1: ADTR Simulation Implementation

Section 4.1 describes the equations for dynamics of the ground vehicle target. It receives inputs of steering angle θ_k and tangential acceleration A_{T_k} and outputs position P_k to the sensors. Section 4.2 describes the software simulating the sensor system. The software generates the measurement z_k from the position input P_k. Section 4.3 describes the ADTR software containing the algorithms discussed in the previous chapter. It takes input measurements from the sensor z_k and outputs the identification of the best trajectory model and the associated state vector estimate $\hat{X}_k$.

4.1 Ground Vehicle Simulation

4.1.1 Ground Vehicle Dynamics Simulation Model

The ground vehicle simulation software is responsible for the generation of the dynamics data that is measured by the sensors and fed to the ADTR system as shown in Figure 4.1. The software propagates the position of the vehicle P_k according to the driver inputs of front wheel angle θ_k and acceleration from the accelerator / brake pedals A_{T_k}. The simulation calculates the following vehicle states relative to an NED coordinate frame as shown in Figure 4.2: position P_k, and speed $\|V\|_k$, heading ϕ_k, yaw velocity ω_k, and yaw acceleration $\dot{\omega}_k$.

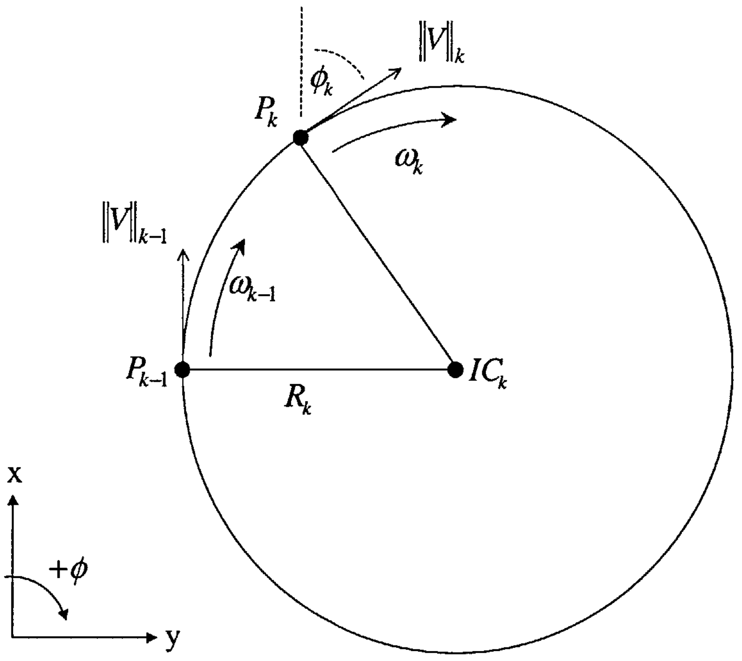


Figure 4.2: Ground Vehicle Simulations Model

Equation (3.45) shows that from the single input θ_k and the wheel base parameter WB we can specify what the instantaneous radius of a turn R_k as shown in Figure 3.3.

$$R_k = \frac{WB}{\tan(\theta_k)}$$

In the case of straight-line travel, R_k is infinite. In that case the position P_k is easily propagated by the equations:

$$\begin{aligned} P_{x_k} &= P_{x_{k-1}} + \left(\Delta t \|V\|_{k-1} + \tfrac{1}{2}\Delta t^2 A_{T_k}\right)\cos\phi_k \\ P_{y_k} &= P_{y_{k-1}} + \left(\Delta t \|V\|_{k-1} + \tfrac{1}{2}\Delta t^2 A_{T_k}\right)\sin\phi_k \end{aligned} \tag{4.1}$$

For the arced line travel case when R_k is not infinite, the angular rate is given by:

$$\omega_{k-1} = \frac{\|V\|_{k-1}}{R_k} \tag{4.2}$$

Further, the angular acceleration is given by:

$$\dot{\omega}_{k-1} = \frac{A_{T_k}}{R_k} \tag{4.3}$$

In order to propagate the position, the instantaneous center is found as an intermediate vehicle state. The instantaneous center IC_k is expressed in NED coordinates and is constant throughout the time step Δt. IC_k can be calculated with current state values as a function of the radius R_k, the current position P_{k-1} also expressed in NED coordinates, and the vehicle's heading ϕ_{k-1}.

$$\begin{aligned} IC_{x_k} &= P_{x_{k-1}} - R_k \sin\phi_{k-1} \\ IC_{y_k} &= P_{y_{k-1}} + R_k \cos\phi_{k-1} \end{aligned} \tag{4.4}$$

We now propagate the heading state according to the inputs given

$$\phi_k = \phi_{k-1} + \Delta t \omega_{k-1} + \tfrac{1}{2}\Delta t^2 \dot{\omega}_{k-1} \tag{4.5}$$

Once the final heading ϕ_k is found, we can relate it back to the sought after position P_k

$$\begin{aligned} P_{x_k} &= IC_{x_k} + R_k \sin\phi_k \\ P_{y_k} &= IC_{y_k} - R_k \cos\phi_k \end{aligned} \tag{4.6}$$

We must also propagate the speed state

$$\|V\|_k = \|V\|_{k-1} + A_{T_k}\Delta t \tag{4.7}$$

4.1.2 Vehicle Parameters and Limitations

Values for vehicle states spoken of in the previous chapter are bounded for any real vehicle. Mechanical limitations have been derived according to some data for a Hummer vehicle as published on the Internet at www.hummer.com and www.consumerreports.org. The published performance specifications are summarized in Table 4.1.

Parameter		English	Metric
Wheel Base		130 in	3.302 m
Minimum Turning Radius		26.5 ft	8.1 m
Max Speed		83 mph	38.0 m/s
Maximum Acceleration		0-30 mph / 4.6 sec	0-13.4 m/s / 4.6 s
		0-60 mph / 17.0 sec	0-26.8 m/s / 17.0 s
Maximum Braking	Cold	60-0 mph in 176 ft.	26.8 – 0 m/s in 53.6 m
	Warm	60-0 mph in 165 ft.	26.8 – 0 m/s in 50.3 m
	Hot	60-0 mph in 237 ft.	26.8 – 0 m/s in 72.2 m

Table 4.1: Published Hummer Specifications

We calculate the maximum deceleration A_{MAX}^{-} the tires would allow for this vehicle from the maximum braking specification. In order to develop a relation for average velocity for a braking scenario we will assume a constant acceleration over the entire distance. This assumption allows us to make the following relations

$$\ddot{P} = A_{MAX}^{-} \tag{4.8}$$

By integrating this relation we can derive the following equations that contain the specification start position P_0, end position $P(t)$, start velocity V_0 and end velocity $V(t)$.

$$\begin{aligned} V(t) &= A_{MAX}^{-}\, t + V_0 \\ P(t) &= \tfrac{1}{2} A_{MAX}^{-}\, t^2 + V_0\, t + P_0 \end{aligned} \tag{4.9}$$

These equations are substituted into each other to get the final expression:

$$A_{MAX}^{-} = \frac{V(t)^2 - V_0{}^2}{2(P(t) - P_0)} \tag{4.10}$$

For the single deceleration limiting value A_{MAX}^{-} we take the worst case scenario of braking while hot. For the maximum tangential acceleration A_{MAX}^{+} possible we simply used an average of the published maximum acceleration values. To calculate the maximum wheel angle θ_{MAX} the vehicle is mechanically capable of, we can use the following relation from (3.45):

$$\theta_{MAX} = \arctan\left(\frac{WB}{R_{MIN}}\right) \tag{4.11}$$

Parameter Name	Symbol	Value
Max Speed	$\|V\|_{MAX}$	38.0 m/s
Max Braking	A^{-}_{MAX}	4.98 m/ s^2
Max Acceleration	A^{+}_{MAX}	2.2 m/s^2
Max steering angle	θ_{MAX}	22.1°
Wheel Base	WB	3.3 m

Table 4.2: Derived Ground Vehicle Simulation Parameters

Additionally, θ should be limited to the angle that would cause a skid from an excessive lateral acceleration requirement. This is developed from (3.44) and (3.45) as follows:

$$\theta_{SKID_k} = \arctan\left(\frac{WB * A^{-}_{MAX}}{\|V\|_k^2}\right) \tag{4.12}$$

4.1.3 Implementation in C

The ground vehicle simulation software is written in C for use in the CSIM simulation framework. The ground vehicle simulation is accessed from the function 'gdVehicle_main'. This simulation function calls subfunctions: 'gdVehicle_init', 'driver_model' and 'propogate_state'. 'gdVehicle_init' performs all required variable initialization. 'driver_model' gets input for driver controls θ_k and A_{T_k}. There is a branch point in this function to get inputs either from a stochastic model or a joystick. 'propogate_state' uses the equations in Section 4.1.1 to propagate the position of the ground vehicle according to driver inputs θ_k and A_{T_k} which is made available to the sensor simulation software. The function structure is summarized in a Parent-child hierarchy in Table 4.3.

- gdVehicle_main ( ) -- main interface for the ground vehicle model
  - gdVehicle_init ( ) -- initialize state variables
  - driver_model ( ) -- driver model for vehicle dynamics input
    - stochastic_driver ( ) -- generates stochastic driver commands for vehicle mode
    - live_driver ( ) -- this mode is for joystick input
  - propogate_state ( ) -- propogates vehicle state according driver inputs

Table 4.3: Ground Vehicle Simulation Code Function Hierarchy

4.2 Sensor Simulation

The sensor simulation software is very simplistic based on the assumptions made in Section 2.4. Based on that discussion the simulation of the sensor follows the exact formulation of the analytical model given by (3.98):

$$z_k = \begin{pmatrix} P_{x_k} \\ P_{y_k} \end{pmatrix} + \begin{pmatrix} v_{x_k} \\ v_{y_k} \end{pmatrix}$$

z_k is the sensor measurement, P_k is the position in NED coordinates and v_k is the noise vector that corrupt the signal. The noise vector v_k is evaluated using a Gaussian generator. The sensor simulation is accessed through 'sensor_main'. This function performs a function call to a CSIM library to generate the Gaussian noise with a 10-meter standard deviation for both components.

- sensor_main ( ) -- main interface for the sensor model

Table 4.4: Sensor Simulation Code Function

4.3 ADTR Simulation

The Automatic Dynamic Trajectory Recognition System simulation software takes measurement inputs z_k from the sensor simulation software and identifies the trajectory model that best fits the it. The outputs of the simulation are the model identification number and the state vector $\hat{X}_k$ associated with it. The software functions follow the development described in Chapter 3.

4.3.1 U-D Covariance Factorization Kalman filter formulation

The Kalman filters for the ADTR are implemented in the U-D covariance factorization form. The primary reason for this is the existence of a well-tested U-D Kalman filter library. In this method the covariance matrix $\mathbf{P}$ is factored into D, a diagonal matrix, and U, an upper triangular matrix with ones on the diagonal. This factored form is shown here:

$$\mathbf{P} = UDU^T = \begin{bmatrix} 1 & U & U & \cdots & U \\ & 1 & U & \cdots & U \\ & & 1 & \cdots & U \\ & & & \ddots & \vdots \\ & & & & 1 \end{bmatrix} \begin{bmatrix} D & & & & \\ & D & & & \\ & & D & & \\ & & & \ddots & \\ & & & & D \end{bmatrix} \begin{bmatrix} 1 & & & & \\ U & 1 & & & \\ U & U & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ U & U & U & \cdots & 1 \end{bmatrix} \quad (4.13)$$

The U-D covariance factorization filter is a method developed to improve the numerical conditioning of the standard Kalman filter algorithm. The U-D filter guarantees positive definiteness for covariance matrix $\mathbf{P}$ and increases numerical accuracy and stability as is comparable to square root filtering methods[2] [7]. The upper triangular and diagonal matrices of the U-D filter require less storage memory than the standard matrix form. The U-D filter, however, requires more computations than the standard filter. This application uses relatively small matrices and consequently requires little memory or computations. The U-D filter implementation is justified on the basis of good numerical conditioning and the convenience of availability.

The form stated in (4.13) does not define a unique set of matrices U and D. The entries for the matrices will be defined explicitly here [7] in algorithmic form for a matrix $\mathbf{P}$ of dimensions $n \times n$. The subscripts denote the row and column for each U or D matrix entry. It is expressed

[2] Square root filter methods propagate and use $\sqrt{\mathbf{P}}$ for Kalman gain calculation where $\sqrt{\mathbf{P}}\sqrt{\mathbf{P}}^T = \mathbf{P}$

in pseudo code to show the iteration sequence. The definition algorithms start with the entry at the lower right corner given by $j = n \ldots 1$. For the first iteration, nonexistent entries (i.e. D_{mm}) are set to zero.

```
for $(j = n \text{ to } 1)$ {
    $D_{mm} = \mathbf{P}_{jj} - \sum_{m=j+1}^{n} D_{mm} U_{jm}^2$
    $U_{ij} = \begin{cases} 0 & \mathrm{i > j} \\ 1 & \mathrm{i = j} \\ \mathbf{P}_{ij} - \sum_{m=j+1}^{n} D_{mm} U_{im} U_{jm} & \mathrm{i = j\text{-}1 \ldots 1} \end{cases}$    (4.14)
}
```

This definition is useful for defining an initial condition of the factored matrices U_0 and D_0 given the initial condition $\mathbf{P}_0$, see Section 4.3.4.

As an intermediate step to U and D propagation, we build two new matrices $Y_{k|k-1}$ and $\breve{D}_{k|k-1}$. $Y_{k|k-1}$ includes the state transition matrix Φ_{k-1} and identity sub-matrix I that is of size $n \times n$. The columns of $Y_{k|k-1}$ are also expressed as a series of vectors named a_i.

$$Y_{k|k-1} = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} = \begin{bmatrix} \Phi_{k-1} U_{k-1|k-1} & \mathbf{I} \end{bmatrix} \tag{4.15}$$

$$\breve{D}_{k|k-1} = \begin{bmatrix} D_{k-1|k-1} & \\ & Q_{k-1} \end{bmatrix} \tag{4.16}$$

The propagation for each entry of the decomposition matrices U and D is given in algorithmic form [7]. It is expressed in pseudo code to express the nested iterations required.

```
for ($j = n$ to 1) {
    $D_{jj_{k|k-1}} = a_j \breve{D}_{k|k-1} a_j^T$

    for ($i = 1$ to j-1) {                                   (4.17)
        $U_{ij_{k|k-1}} = \frac{a_i^T \breve{D}_{k|k-1} a_j}{D_{jj_{k|k-1}}}$
        $a_i = a_i - U_{ij_{k|k-1}}$
    }
}
```

As an intermediate step to updating U and D, we build two new vectors. The definition for f includes the measurement model matrix H_k.

$$f = U_{k|k-1}^T H_k \tag{4.18}$$

$$g = D_{k|k-1} f \tag{4.19}$$

The update for the decomposition matrices U and D is also expressed in pseudo code to express the nested iterations required.

```
for ($j = 1$ to $n$) {
    $a_j = a_{j-1} + f_j g_j$
    $D_{jj_{k|k}} = D_{jj_{k|k-1}} \frac{a_j}{a_{j-1}}$
    $b_j = g_j$
    $c = \frac{-f_j}{a_{j-1}}$
                                                             (4.20)
    for ($i = 1$ to $j$-1) {
        $U_{ij_{k|k}} = U_{ij_{k|k-1}} + b_i c$
        $b_i = b_i - U_{ij_{k|k-1}} g_j$
    }
}
```

The Kalman gain is calculated at this point

$$K_k = \frac{b}{a_n} \tag{4.21}$$

Note the special equivalences attached to a_n and b [9].

$$a_n = H_k^T P_{k|k-1} H_k + R_k \tag{4.22}$$

$$b = \mathbf{P}_{k|k-1} H_k^T \tag{4.23}$$

This shows how the algorithms for U, D and K_k replace the equations for $\mathbf{P}$ and K_k seen in Table 3.1 and Table 3.2. All other equations remain the same for the U-D covariance factorization filter.

The implementation of this algorithm uses a matrix data structure. The matrices are passed into Kalman filter library functions: 'KF_prp_cov', 'KF_prp_est', 'KF_innov', 'KF_upd_cov', 'KF_upd_est'. These functions were developed previous to this thesis [9]. They are described briefly in Table 4.6 as a key parts of the simulation implementation.

4.3.2 *A Priori* Covariance Matrices

The covariance matrices Q and R describe the probabilistic behavior of the input disturbance and sensor noise respectively. In order to achieve an optimal filter, these must be rigorously defined to match the phenomena they describe. This would require the disturbance and noise to be have statistics of a Gaussian function. Under the assumptions made in Section 2.4, the sensor noise will be close to a Gaussian model and the covariance matrix R is analytically chosen.

Admittedly, the behavior of the driver is not modeled very closely by a Gaussian random walk variable. Driver behavior are highly correlated in time and are not characterized by random Gaussian white noise. Additionally, the behavior of the driver most likely has distinct periods of different statistical behavior. Regardless, the Gaussian model attempts to envelop the actual behavior with a conservative description in Q. This is a method proposed by Singer that was discussed in Section 3.4.1. The result is a sub-optimal estimator that is useful in this application.

The values gained from the Singer method are a starting point. The targeting filter requires heuristic tuning of the Q and R matrices. There is a performance trade-off with the selection of the relative weighting of the Q and R matrices. With R held fixed a Q matrix with large values will result in a noisy estimate in steady state, but will be very quick to accommodate maneuvers into the estimate.

The approach for tuning multiple filters used in this system is to keep the R matrix constant and common for all while the Q matrices are tuned for each filter individually. The approach for each Q is to increase the values uniformly for each filter until the position estimate error due steady state is comparable to the position estimate error due to sudden maneuvers.

In addition to finding the values for the nonzero diagonal entries of Q, it was found useful to give nominally small nonzero values to the normally zero diagonal entries. This effects the filter by adding uncertainty to the model where there was none previously. This added uncertainty is known as pseudo-noise [8]. The nominal value of .01 in Q represents the variance for the pseudo-noise. The pseudo-noise reduces the oscillations of the filter at the sacrifice of a noisier estimate

4.3.2.1 'Q' Values for Filter #1

The covariance matrix Q for the RWV model is calculated by using (3.61):

$$Q_V = \Delta t^2 A_{MAX}^2$$

The value for A_{MAX} comes from the maximum acceleration a tire can transmit. This is approximated to be around one half that of gravity or 4.90 m/s^2. This value is close to the value shown for a Hummer in ideal conditions shown in Table 4.2. The value for Q_v is rounded off to one significant digit because the A_{MAX} value is a conservative number representative of any vehicle the system may encounter.

$$Q_V = 1 \tag{4.24}$$

4.3.2.2 'Q' Values for Filter #2

The covariance matrix Q for the RWA model is calculated by using (3.66) and (3.67):

$$Q_A = \frac{4\Delta t^2 A_{MAX}^2}{T^2} \tag{4.25}$$

The value for A_{MAX} is the same as in the RWV case just described. The value T is the least amount of time it takes for a human to change from full steering acceleration in one direction to the full steering acceleration in the other. This value is estimated at 2 seconds. The covariance value is calculated as:

$$Q_A = 1 \tag{4.26}$$

4.3.2.3 'Q' Values for Filter #3

The covariance matrix Q for the RWVLA model is calculated by using the values calculated already for Q_V and Q_A:

$$\begin{aligned} Q_V &= 1 \\ Q_{A_L} &= 1 \end{aligned} \tag{4.27}$$

4.3.2.4 'Q' Values for Filter #4

The covariance matrix Q for the RWLJTA model is calculated by using (3.96) and (3.97):

$$\begin{aligned} Q_{J_L} &= \Delta t^2 \dot{J}_{L_{MAX}}^2 \\ Q_{A_T} &= \Delta t^2 J_{T_{MAX}}^2 \end{aligned}$$

The values for $\dot{J}_{L_{MAX}}$ and $J_{T_{MAX}}$ are potentially very large. These values do not fit the well into the method employed by the Singer method. These values are scaled back and arbitrarily set to equal the nonzero Q values calculated thus far.

$$\begin{aligned} Q_{J_L} &= 1 \\ Q_{A_T} &= 1 \end{aligned} \tag{4.28}$$

4.3.2.5 'R' Values

The value for the noise covariance matrix R is common for all four filters. The diagonal elements are the only nonzero elements based on the assumption that there is no correlation between the noise sources. The two diagonal terms are equivalent because the LADAR is rotating with respect to the NED coordinate frame giving either component equal chance to have error. It is representative of the expected variance of the sensor noise, which has a good analytical basis value of 1^2 as related to a priori knowledge of the constant noise statistics given in Section 4.2. In practice, this changing this value is an easy way to tune all the filters in uniform manner. By heuristic methods the value are as given in Table 4.5.

RWV (Filter #3)	RWA (Filter #2)
$Q = \begin{bmatrix} .01 & 0 & 0 & 0 \\ 0 & .01 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$Q = \begin{bmatrix} .01 & 0 & 0 & 0 & 0 & 0 \\ 0 & .01 & 0 & 0 & 0 & 0 \\ 0 & 0 & .01 & 0 & 0 & 0 \\ 0 & 0 & 0 & .01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
RWVLA(Filter #3)	**RWLJTA (Filter #4)**
$Q = \begin{bmatrix} .01 & 0 & 0 & 0 & 0 \\ 0 & .01 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	$Q = \begin{bmatrix} .01 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
RWV, RWA, RWVLA and RWLJTA Filters	
$R = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	

Table 4.5: Final Q and R Covariance Matrix Values

4.3.3 Scoring System

The scoring system is implemented exactly as given by equations (3.105) and (3.107)

$$S = RMS_{\|r\|_{k+1}}$$

$$RMS_{\|r\|_{k+1}} \approx \sqrt{\frac{N}{N+1} RMS_{\|r\|_k}^2 + \frac{1}{N+1} \|r\|_{k+1}^2}$$

The final value for the window parameter is set at :

$$N = 10 \tag{4.29}$$

This value equals a 2 second RMS window period. A longer window period results in a sluggish reaction to new maneuvers. A smaller window period results in a jumpy selection process.

4.3.4 Initialization

The method for state vector initialization is to set all states to zero except for the position state P, which is set to the first sensor measurement received. The initialization in the simulation sets the position to zero since the origin of the NED coordinate frame is defined at the initial measurement position. The initialization method for the covariance matrix $\mathbf{P}$ is to set all the diagonal elements to one. This is equivalent to setting the decomposition matrix D entries to one and the decomposition matrix U entries to zero. The states and covariance matrix $\mathbf{P}$ must converge on an estimate without a calculated initial condition. For this reason, a ten-second start-up period is given to the system to let the state estimates and the covariance matrix $\mathbf{P}$ converge on appropriate values.

After ten seconds, the scoring algorithm begins. The RMS function begins in a non-windowed mode as given in (3.104), which replaces the window parameter N with the current iteration count k. This continues until one window interval of data has been collected and then it switches to windowed RMS operation as given in Section 4.3.3.

4.3.5 Implementation in C

The ADTR simulation software is written in C for use in the CSIM simulation framework. The software implements the algorithm described in Chapter 3. The ADTR simulation is accessed from the 'ADTR_main' function. This function calls subfunctions: 'ADTR_init', 'sensor_main', ‘getZ’, ‘PHImodel_3’, ‘PHImodel_4’, ‘ADTR_filters’, ‘ADTR_scoring’ and ‘ADTR_predict’. The purpose of each sub-function and the Parent-Child hierarchy is summarized in Table 4.6.

The sub-functions that build the state transition matrices are called from several places. This is due to the constant need to update the state transition matrices for filters three and four. Thus the sub-functions are called in ‘ADTR_main’ and in the prediction function ‘ADTR_predict’. The prediction function runs like a Kalman filter with no update information. It propagates over a foursecond interval and stores the state vector in memory. The purpose of this prediction is to collect results for the performance of the system as described in Chapter 5.

- ADTR_main ( ) – main interface for the ADTR targeting system emulation
  - ADTR_init( ) – initializes all necessary variables in ADTR module
    - KalmanInit – initialize matrices and vectors for Kalman filters
    - PHImodel_1( ) – builds state transition matrix for filter #1
    - PHImodel_2( ) – builds state transition matrix for filter #2
    - PHImodel_3( ) – builds dynamic state transition matrix for filter #3
    - PHImodel_4( ) – builds dynamic state transition matrix for filter #4
  - sensor_main( ) – main interface for the sensor model
  - getZ( ) – get sensor measurement in NED coordinates
  - PHImodel_3( ) – builds dynamic state transition matrix for filter #3
  - PHImodel_4( ) – builds dynamic state transition matrix for filter #4
  - ADTR_filters( ) – Executes Kalman filter propagation and update
    - KF_prp_cov ( ) – propagates the covariance matrices U and D
    - KF_prp_est ( ) – propagates the state vector estimate
    - KF_innov ( ) – calculates the filter residual
    - KF_upd_cov ( ) – updates the covariance matrices U and D according to the filter residual
    - KF_upd_est ( ) – updates the state vector estimate according to the filter residual
  - ADTR_scoring( ) – Calculates the RMS value for each filter and selects the lowest value
    - rms ( ) – Calculates the combined recursive RMS value
  - ADTR_predict( ) –Makes a prediction based on current state estimate
    - KF_prp_est ( ) – propagates the state vector estimate
    - PHImodel_3( ) – builds dynamic state transition matrix for filter #3
    - PHImodel_4( ) – builds dynamic state transition matrix for filter #4

Table 4.6: ADTR System Simulation Code Function Hierarchy

5 Demonstration and Results

This chapter describes the results of a demonstration of the ADTR system implemented as described in the previous chapter. Section 5.1 describes the objective of the demonstration and the approach used. Section 5.2 presents the results of the demonstration for several simple maneuvers.

5.1 Demonstration Method

The results in this chapter are prepared to demonstrate the operation of the ADTR system. The results will show the effects of three simple maneuvers on the real-time model identification process. The filter identification number returned by the system is corresponds to the identification number of the selected model. The random walk velocity model is filter #1, the random walk acceleration model is filter #2, the random walk velocity and lateral acceleration model is filter #3 and the random walk lateral jerk and tangential acceleration model is filter #4.

The correct performance of the ADTR is scrutinized by performing a position prediction four seconds into the future for each filter's state estimate. If a filter is selected as the best fit then the predicted position should be the closest to the actual future value. This method is used because position prediction is a component of the full targeting system shown in Figure 2.1. The position prediction is performed at every filter iteration and is compared to the real-time output after run-time.

The prediction error is defined as the difference between the predicted position and the true position value four seconds later. Knowledge of the true position value is a valuable feature readily available in a simulation environment. Good correlation between the position prediction error and the residual magnitude supports the residual analysis approach for model identification used in the system.

Three maneuvers were chosen to highlight the particular maneuver scenario each model is best suited for. The demonstration of these maneuvers is not intended to verify optimal filter design nor to target any predetermined performance specifications.

5.2 Results

The first maneuver demonstrates a period of constant tangential acceleration bracketed by straight-line travel. This maneuver is intended to highlight the RWV model (filter #1) and the RWA model (filter #2), by creating a trajectory with constant velocity and constant acceleration respectively. Figure 5.1 through Figure 5.4 show the results for this maneuver.

Figure 5.1 shows the inputs made to the vehicle dynamics model and the output of the ADTR algorithm. The first ten seconds are set aside to let the filter state estimates and state covariance matrix converge to a steady state. After the ten-second initialization period, the driver continues in straight line constant velocity travel until the twenty second mark. The ADTR system selects the RWV model (filter #1) as the best model during this time. At the twenty second mark, the driver inputs a constant tangential acceleration of 3 mph/sec (1.34 m/s^2). After a brief period, the RMS value of the residual change enough that the RWA model (filter #2) is selected. At the thirty second mark, the acceleration goes to zero once again and the vehicle continues in straight line constant velocity travel.

Figure 5.2 shows the predictions and estimates made by each filter overlaid on the actual vehicle path. Small lines connect the position estimate point at which the prediction was made to the predicted position. They are primarily overlapping in this maneuver. Filters three and four show some fringing evident of a noisy prediction. Filter one has a large magnitude error not visible in this prediction plot, but revealed in Figure 5.4.

Figure 5.3 shows the time history of the combined residual magnitudes for each filter. The differences caught by the RMS algorithm and expressed in the selected filter value are not visibly evident in the residual histories. There is evidence of agreement between the selected filter and the smallest RMS of the prediction errors shown in Figure 5.4. Both filter one and filter three have pronounced 'humps' during the period that filter two is selected. Filter four is very noisy, due to its sensitivity to and assimilation of sensor noise, to ever be selected. During the periods that filter was selected, the prediction from filter one was clearly the least in error. This would suggest a strong correlation between the statistics of the residuals and that of the predictions.

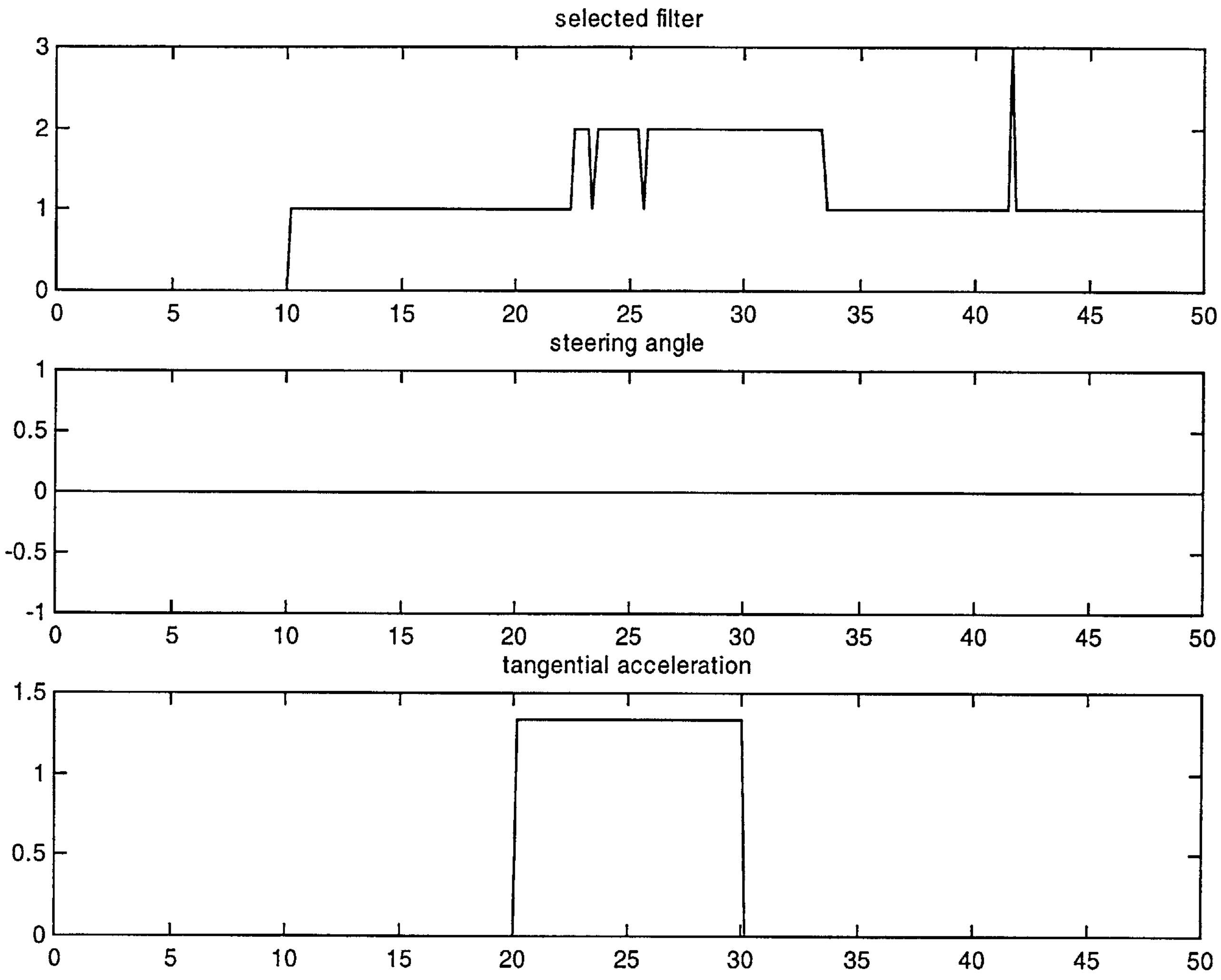


Figure 5.1: Selection Output and Driver Input for Constant Tangential Acceleration Maneuver

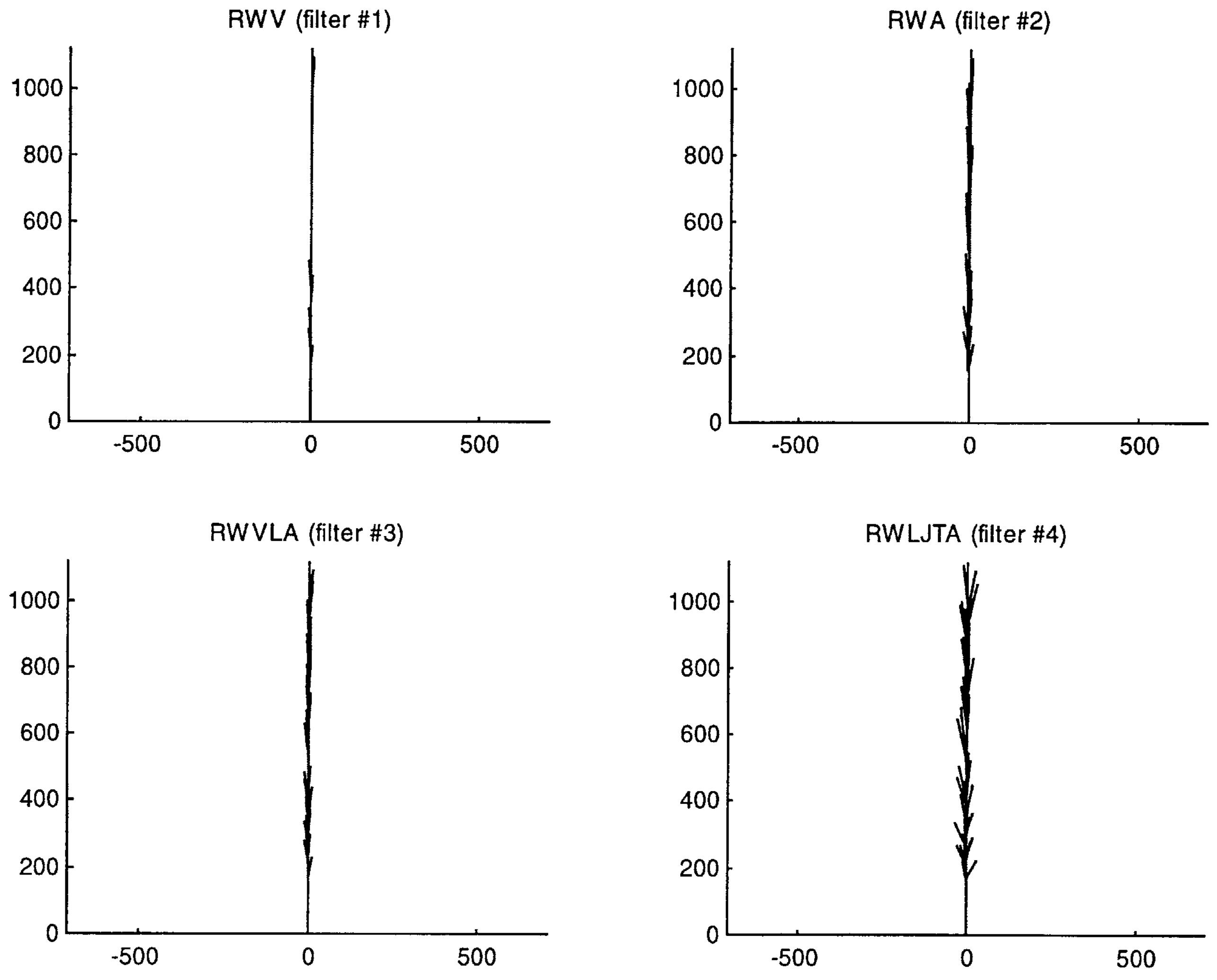


Figure 5.2: Prediction Plot for Constant Tangential Acceleration Maneuver

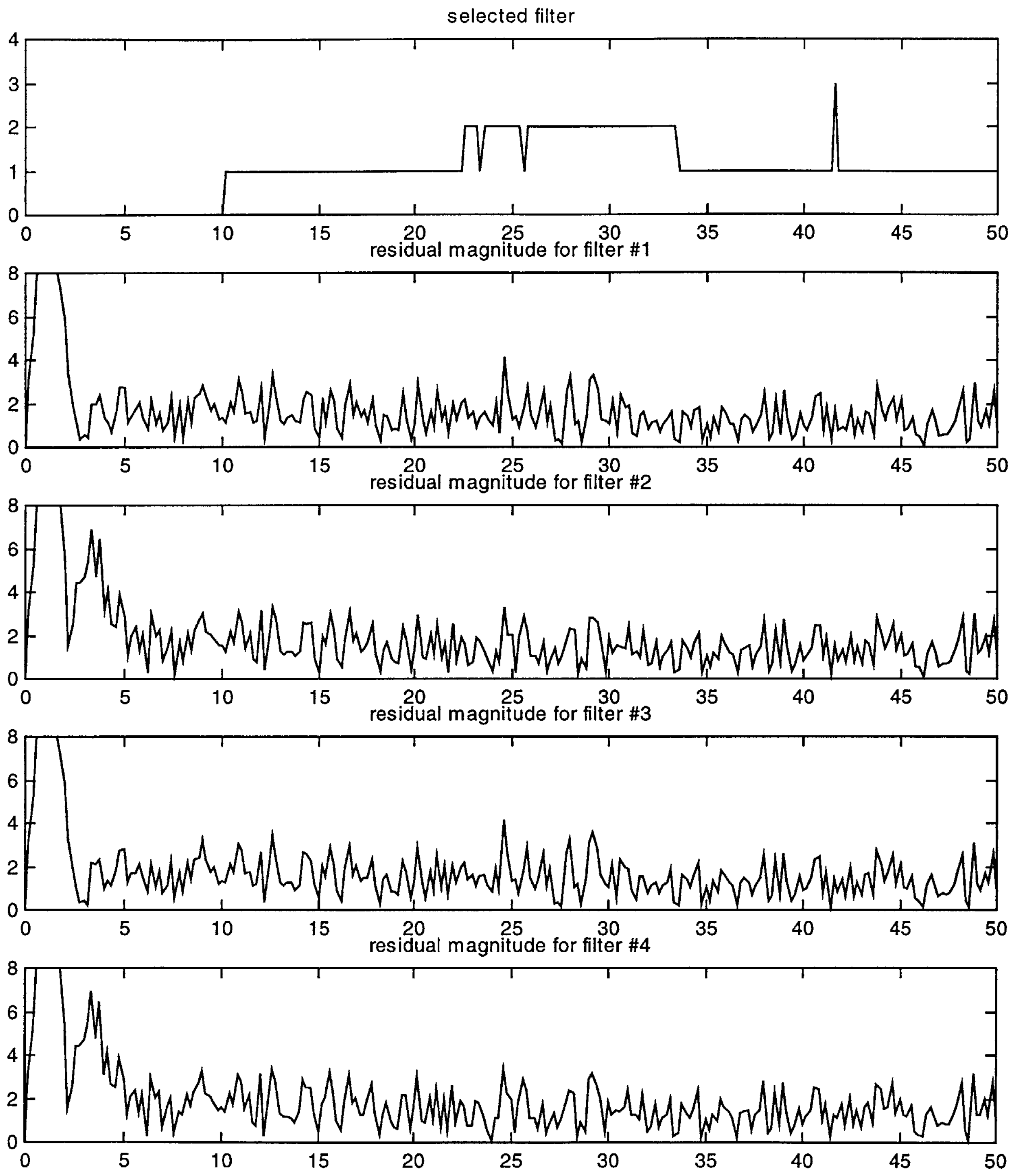


Figure 5.3: Selection and Residual Magnitude Time Histories for Constant Tangential Acceleration Maneuver

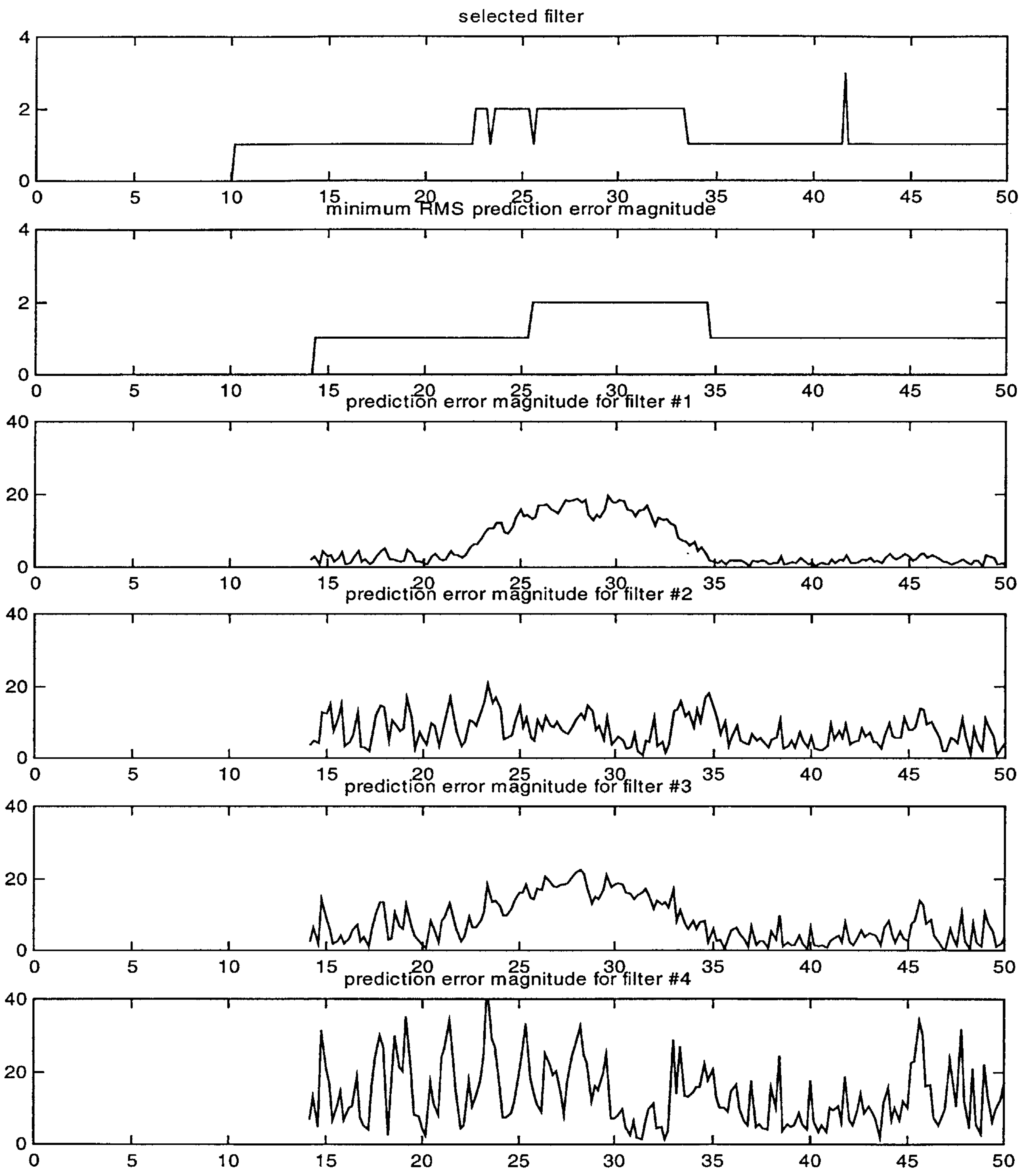


Figure 5.4: Prediction Error Plot for Constant Tangential Acceleration Maneuver

The second maneuver demonstrates a period of constant lateral acceleration bracketed by straight-line travel. This maneuver is intended to highlight the RWVLA model. Figure 5.5 shows the driver suddenly turning his steering wheels to an angle of 1° (.017 radians) between the twenty second and fifty second time points. It also shows a switch to select filter three with a slight time lag behind the driver. Filter four is selected briefly near the sudden maneuver change when there is a high value of lateral jerk.

Figure 5.6 shows predictions made by each filter. Small line segments connect the predicted point and the estimated point from which the prediction was extrapolated. Filter one has a bias consistent with predicting a tangent constant velocity during the constant turn. Filter two and three produce predictions that lie closer to the actual curve. Filter four is very sensitive to the sensor noise but has predictions that straddle the true path.

Figure 5.7 illustrates more clearly how well each filter is predicting. Each plot shows the actual path the vehicle took with small line segments connecting the predicted points to the point the vehicle actually reached. The same trends are evident with better clarity.

Figure 5.8 shows the time history of the combined residual magnitudes for each filter. The bias shown in the prediction from filter one is also evident in the residuals. The other three filters are close enough to merit the RMS scoring algorithm to distinguish their performance.

Figure 5.9 clearly shows the trends that correlate with the residual time history. The bias in the prediction error attributed to filter one is approximately forty meters in magnitude. It is also evident that the prediction of filter three is the smallest over the period that the filter was selected.

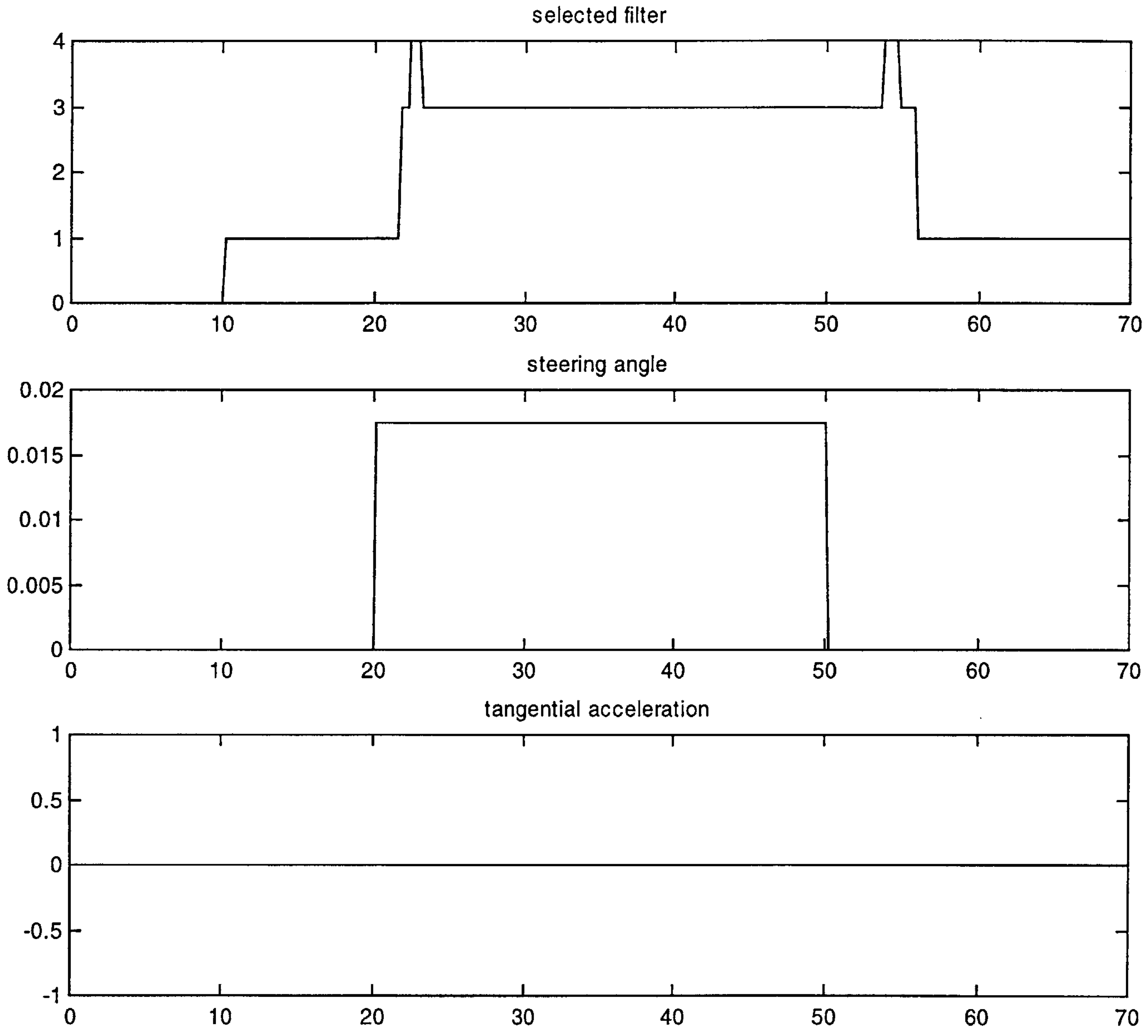


Figure 5.5: Selection Output and Driver Input for Constant Lateral Acceleration

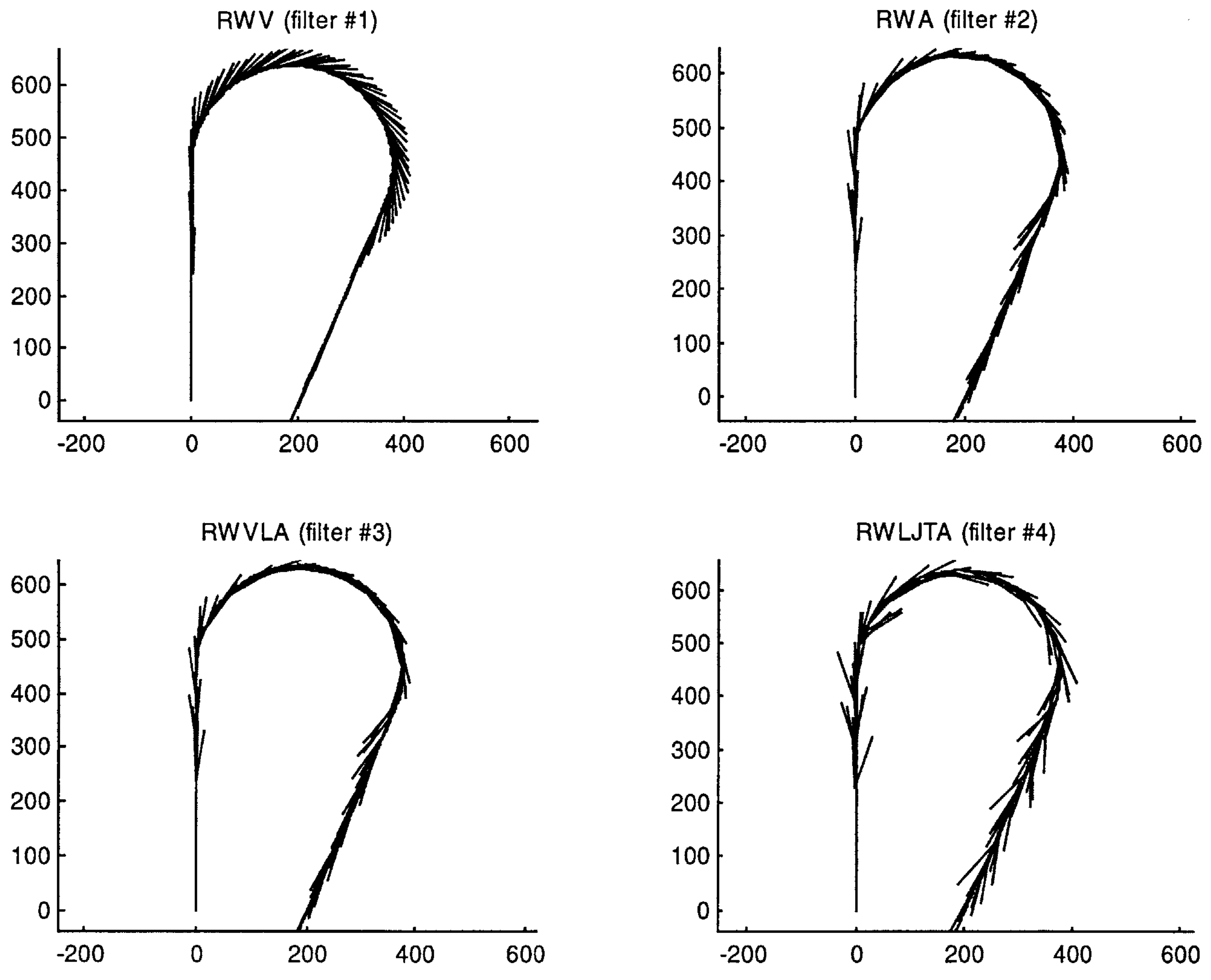


Figure 5.6: Prediction Plot for Constant Lateral Acceleration

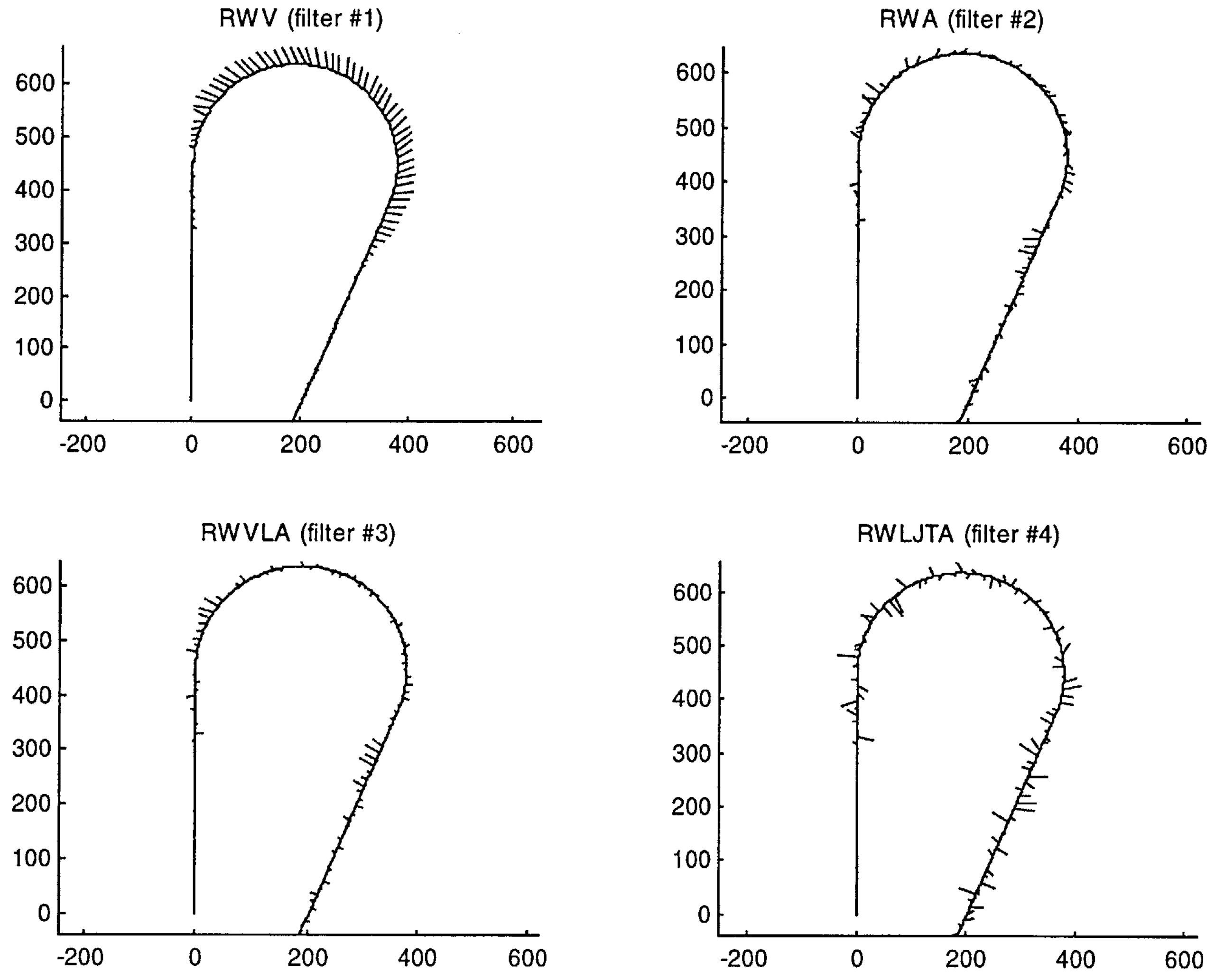


Figure 5.7: Prediction Error Plot for Constant Lateral Acceleration

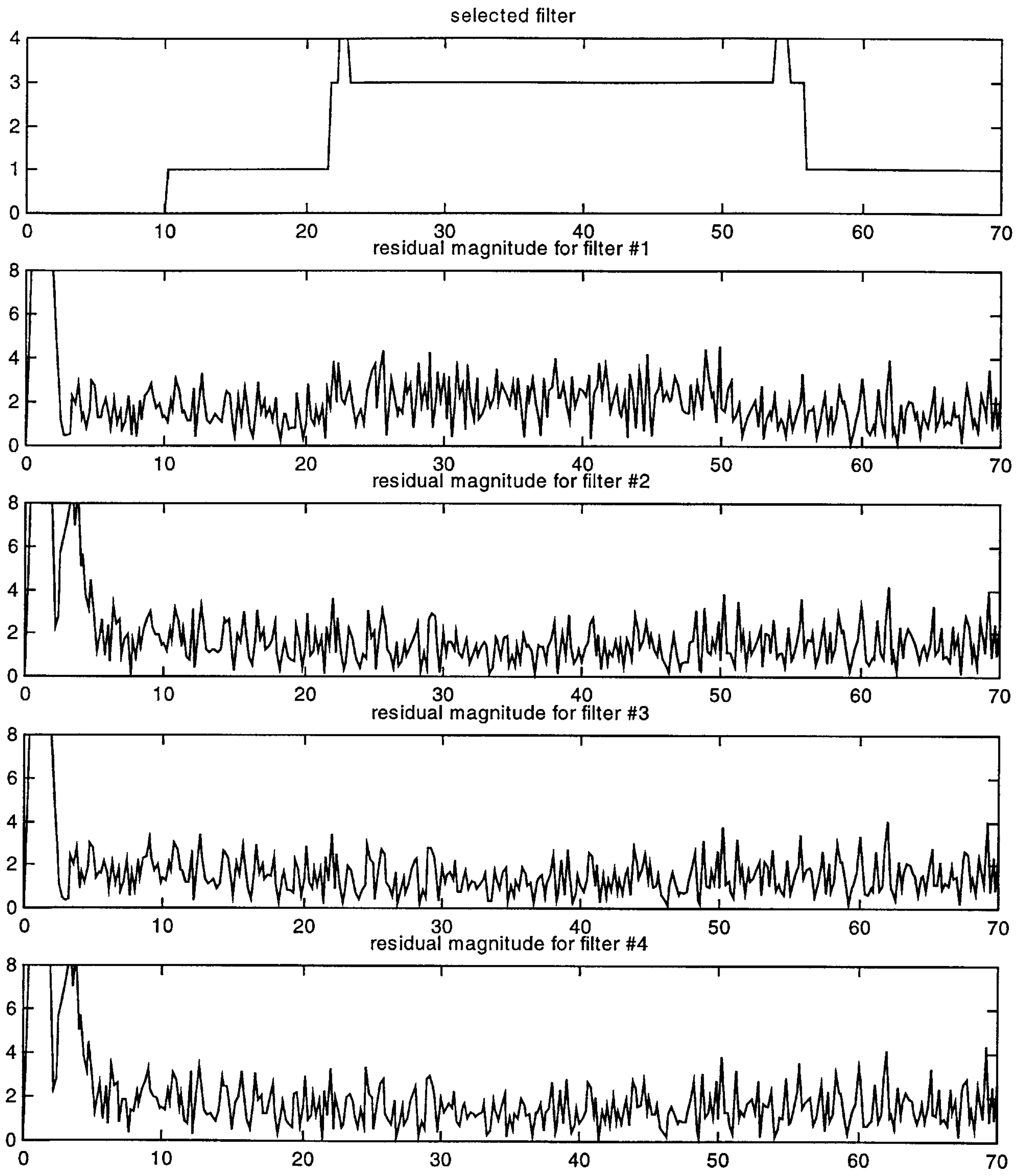


Figure 5.8: Selection and Residual Magnitude Time Histories for Constant Lateral Acceleration

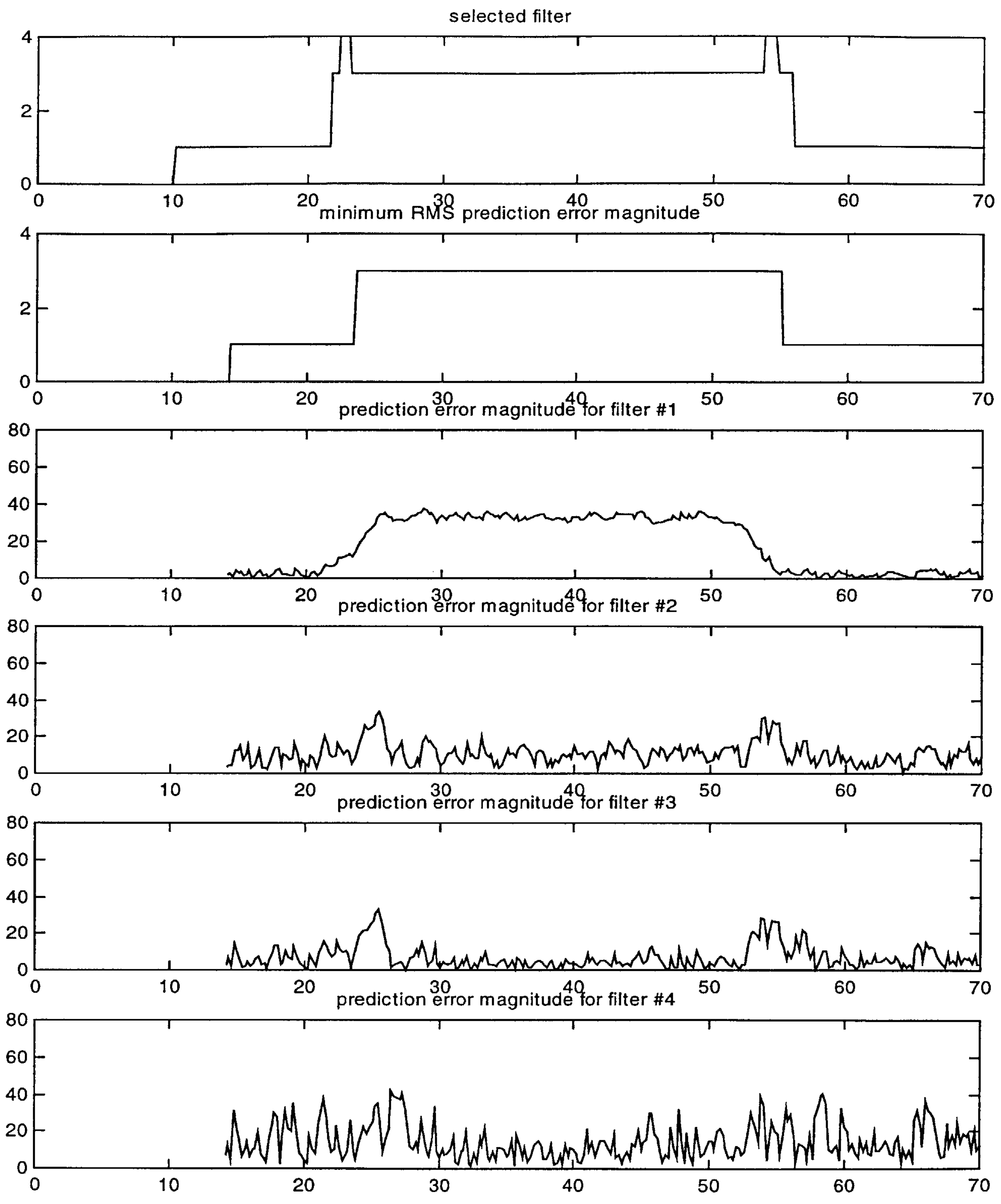


Figure 5.9: Prediction Error Plot for Constant Lateral Acceleration

The third maneuver demonstrates a period of constant lateral and tangential acceleration bracketed by straight-line travel. This maneuver is intended to highlight the RWLJTA model. Figure 5.10 shows the driver suddenly turning to a steering angle of 20° (.35 radians) and goes to a tangential acceleration of 5 mph/sec (2.24 m/s^2). Shortly after accelerating, the maximum acceleration for the tires is reached. At this point the steering angle decays as the velocity increases keeping the vehicle at a constant lateral acceleration as it accelerates tangentially. This happens between the twenty to thirty second time points.

Figure 5.11 shows the predictions made by each filter. Small line segments connect the predicted point and the estimated point from which the prediction was extrapolated. Filter one and appear to have a bias. When the trajectory changes to straight-line travel, filter two still predicts an acceleration component that causes the prediction fan to cross to the other side of the true trajectory path. Filter three predicts an ever expanding arc biased to the inside of the expanding spiral trajectory. Filter four appears noisy as in the other maneuver with the exception of a small section of tightly packed relatively accurate estimates. Figure 5.12 shows the filter prediction errors more clearly. Filter four appears to make a very short interval of small error predictions near the transition point to straight-line travel.

Figure 5.13 shows the time history of the combined residual magnitudes for each filter. Filter one has an obvious hump during the maneuver time when filter four is selected. What is not obvious is the difference in the RMS values for the remaining three filters that was picked up by the scoring algorithm. The correlation between the residual trends and prediction error is shown in Figure 5.14. The correlation for filter one is very strong. The hump evident in the residual plot is evident in the prediction error plot and the good performance during the straight-line travel portions is clear. Filter two has clearly large prediction errors during the maneuver as well. The correlation between the residuals and prediction errors in filters three and four is not clear. Filter three seems consistently high. Filter four, however, seems to peak higher but also drop to a lower error in the middle of the maneuver. This corresponds to the low error spot in Figure 5.12 discussed earlier.

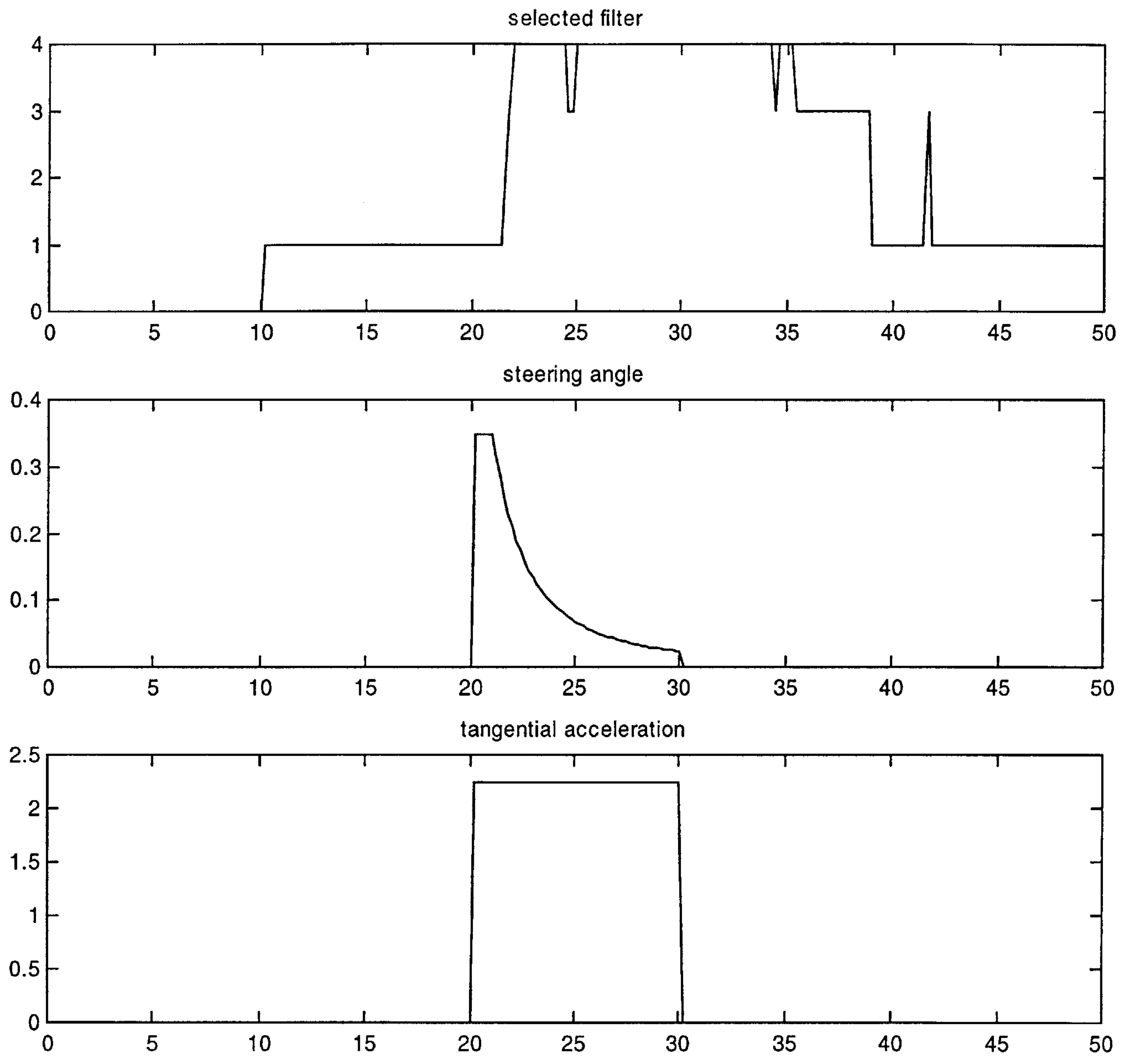


Figure 5.10: Selection Output and Driver Input for Constant Lateral and Tangential Acceleration

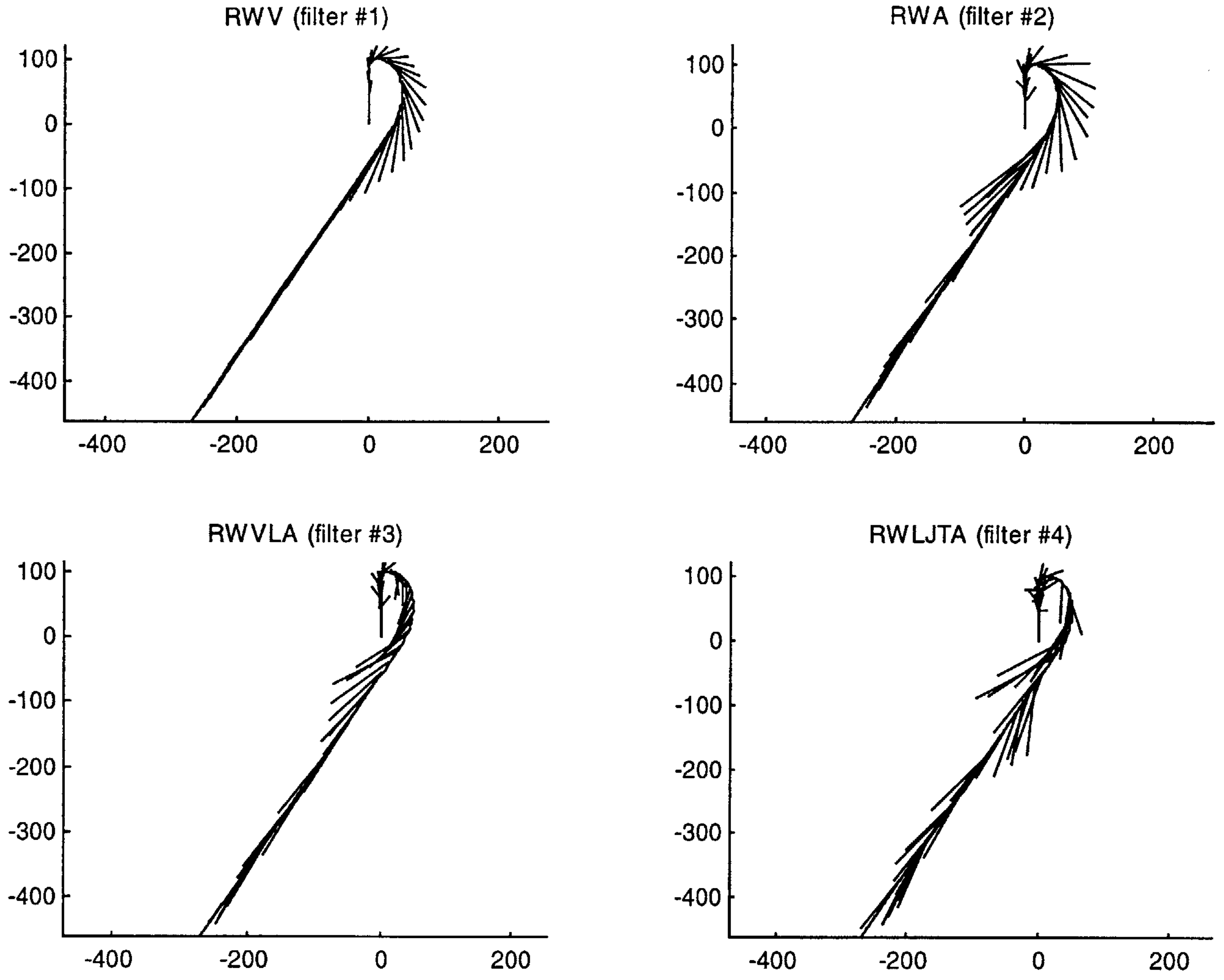


Figure 5.11: Prediction Plot for Constant Lateral and Tangential Acceleration

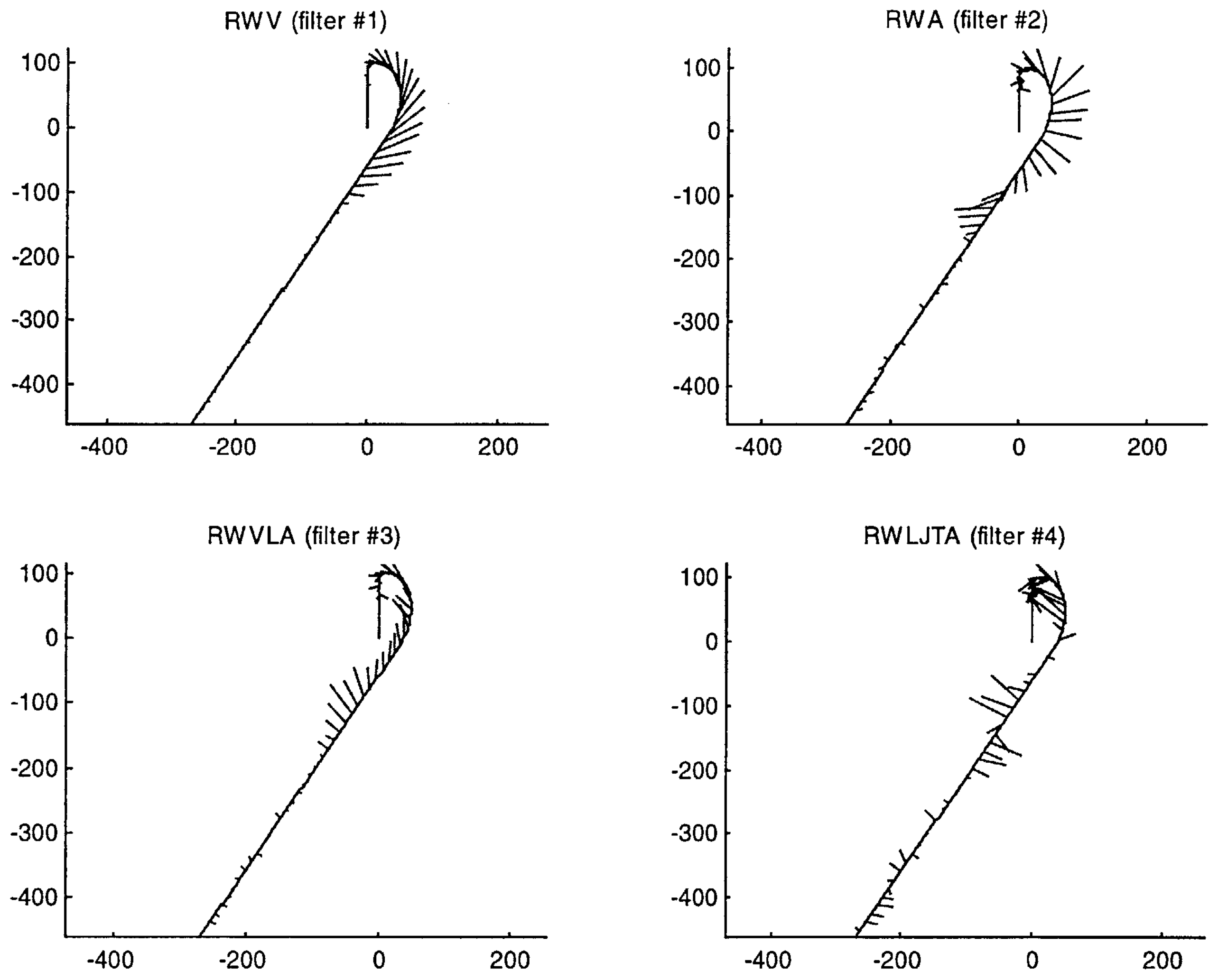


Figure 5.12: Prediction Error Plot for Constant Lateral and Tangential Acceleration

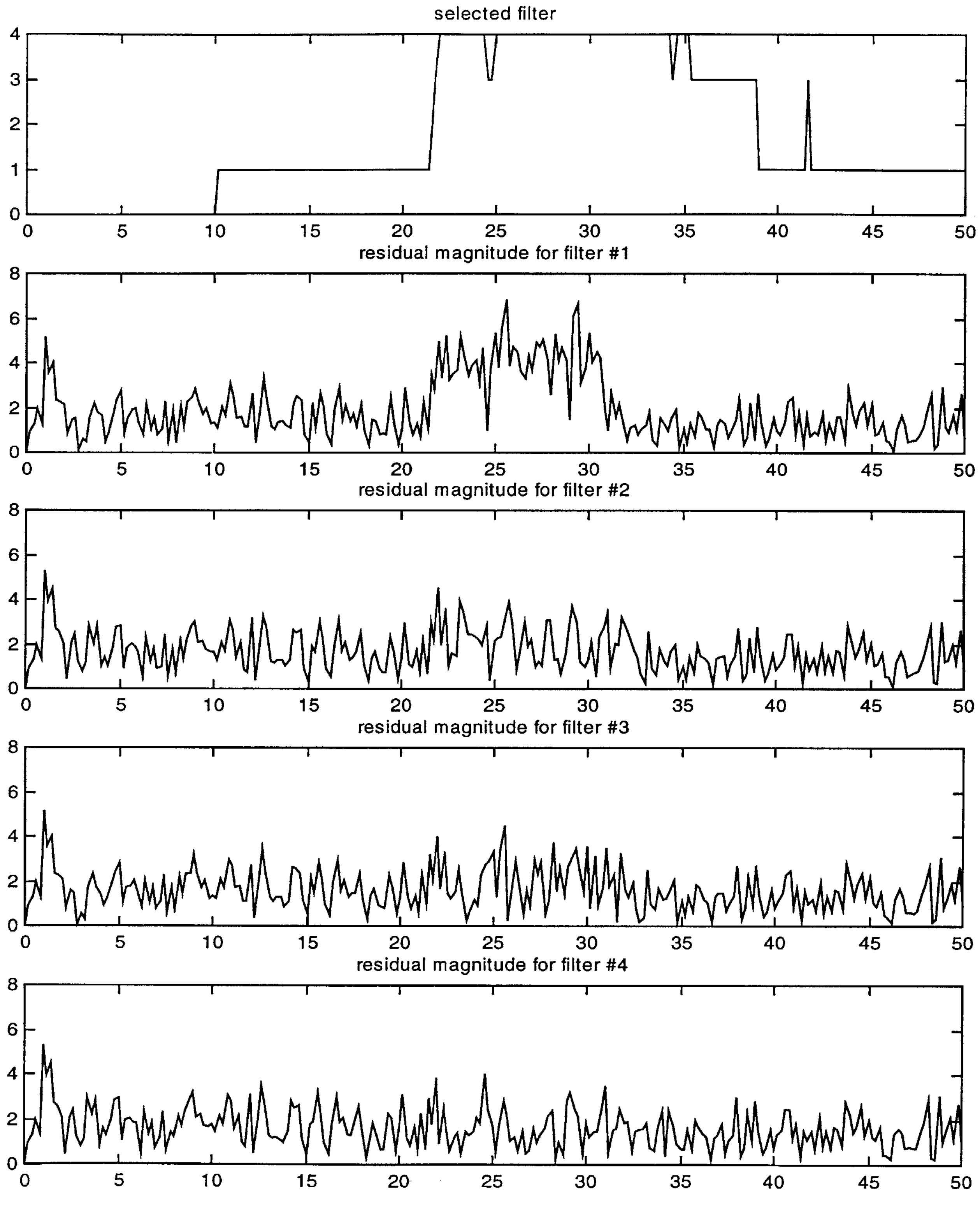


Figure 5.13: Selection and Residual Magnitude Time Histories

for Constant Lateral and Tangential Acceleration

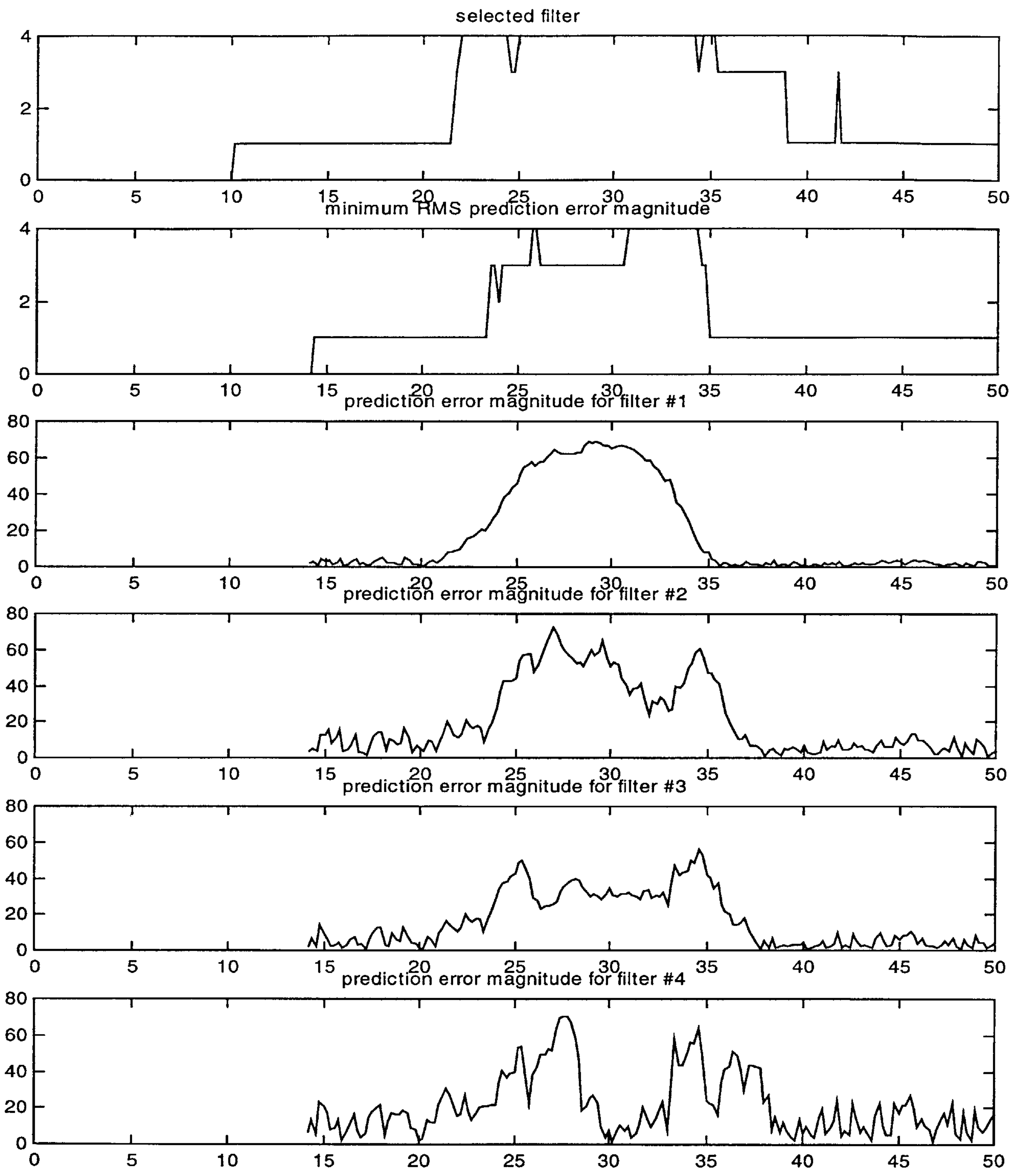


Figure 5.14: Prediction Error Plot

for Constant Lateral and Tangential Acceleration

6 Conclusions and Future Work

Section 6.1 summarizes the significant developments and results connected with the development and demonstration of the ADTR system. Section 6.2 describes recommendations for future work in development of this approach to real-time model identification.

6.1 Summary

This thesis describes research work pertaining to real-time model identification for ground vehicle trajectory estimation by means of the Automatic Dynamic Trajectory Recognition System. Theoretical development of the ADTR system and supporting simulation results give insight into problems and solutions surrounding the objective.

This thesis develops a theoretical approach to model identification for the ADTR system using Kalman filter residual analysis. This approach selects the best trajectory model from an array of candidates by comparing residual vectors generated by a bank of Kalman filters built upon the candidate models. The filter with the lowest RMS residual magnitude value is identified as containing the best trajectory model.

Significant theoretical developments include the formulation of two nonlinear target models and a new discrete version of the extended Kalman filter algorithm used in processing them. The new nonlinear models represent a refinement to the random walk velocity (RWV) and random walk acceleration (RWA) models found in the literature. The RWV model implies a constant velocity trajectory and the RWA model implies a constant acceleration or parabolic trajectory. The two new nonlinear models imply circular and spiral trajectory associated with turning maneuvers. Circular trajectories are implied by the random walk velocity and lateral acceleration model (RWVLA). Spiral trajectories are implied by the random walk lateral jerk and tangential acceleration model (RWLJTA).

This thesis develops a unique discrete formulation for the extended Kalman filter algorithm. This discrete formulation is applicable to a special class of nonlinear models to which the RWVLA and RWLJTA belong. The discrete formulation makes the nonlinear Kalman filter and the standard Kalman filter nearly identical algorithmically.

The results from the ADTR system simulation support the model identification approach. There is a reasonable correlation between the RMS value of the filter residual magnitude and the RMS magnitude of the position error of predicting four seconds into the future. In the case of the RWLJTA model, the correlation is not as good. Perhaps this is a result of the high order of the model.

6.2 Future work

There are many areas in which future work could build upon the material presented in this thesis. Much work is needed in testing the performance of the system in a less ideal environment. A higher fidelity simulation could be built to test the ADTR system. A more realistic vehicle simulation might include terrain, roads and changes of elevation. A more realistic sensor simulation might include a UAV dynamics simulation flying on a spherical earth with sensor noise specified in terms of GPS satellite signal transmission noise and LADAR gimbal noise and range noise. A simulation for the UAV LADAR and GUI are in development for the TOPART program.

Future work is also needed in developing useful models for ground vehicle prediction. One such model is described in the following section.

6.2.1 Random Walk Sinusoidal Lateral Acceleration Model

One idea for future work is to create a model of a ground vehicle performing a serpentine maneuver [10]. This maneuver is an evasive tactic described as weaving back and forth while around a mean direction of travel. A vehicle with a sinusoidal lateral acceleration and a constant tangential acceleration describes a similar trajectory. A model for a sinusoidal lateral acceleration was developed for this thesis called the random walk sinusoidal lateral acceleration model (RWSLA) as described below.

The model for a sinusoid was taken from the RWVLA model described in Section 3.3.3. For the RWVLA model, a single velocity component describes a sinusoid if the lateral acceleration is constant. This is applied to the RWSLA model by analogy. The difference being that two orthogonal sinusoidal velocity components exist in the RWVLA model but only one sinusoidal acceleration component exists in the RWSLA model. Thus we invent an imaginary sinusoidal acceleration component A_{iL} that is orthogonal to the real component of the lateral acceleration

A_L as represented in a complex plane. Further, we must also create a variable analogous to lateral acceleration, which we will call Ω. This variable, Ω, is the product of the amplitude $\|A\|$ and frequency ω of the sinusoid.

$$\Omega = \|A\|\omega \tag{6.1}$$

The amplitude of the sinusoid $\|A\|$ is equal to the magnitude of the complex vector defined by the lateral acceleration A_L and its imaginary orthogonal component A_{iL}. It is also the maximum value of A_L. The continuous time expression of the RWSLA model is:

$$\dot{X} = \begin{pmatrix} \dot{P}_x \\ \dot{P}_y \\ \dot{V}_x \\ \dot{V}_y \\ \dot{A}_L \\ \dot{A}_{LO} \\ \dot{\Omega} \end{pmatrix} = \begin{pmatrix} V_x \\ V_y \\ -\frac{V_y}{\|V\|}A_L \\ \frac{V_x}{\|V\|}A_L \\ -\frac{A_{LO}}{\|A\|}\Omega \\ \frac{A_L}{\|A\|}\Omega \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ w \end{pmatrix} \qquad X = \begin{pmatrix} P_x \\ P_y \\ V_x \\ V_y \\ A_L \\ A_{LO} \\ \Omega \end{pmatrix} \tag{6.2}$$

Again, this model is similar to the random walk lateral acceleration model. The partial derivative matrix must be calculated before a discrete approximation can be made. The partial derivative matrix is as follows:

$$F(t)=\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{V_x V_y A_L}{\|V\|^3} & \frac{-V_x^2 A_L}{\|V\|^3} & \frac{-V_y}{\|V\|} & 0 & 0 \\ 0 & 0 & \frac{V_y^2 A_L}{\|V\|^3} & \frac{-V_x V_y A_L}{\|V\|^3} & \frac{V_x}{\|V\|} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{A_L A_{LO}\Omega}{\|A\|^3} & \frac{-A_L^2\Omega}{\|A\|^3} & \frac{-A_{LO}}{\|A\|} \\ 0 & 0 & 0 & 0 & \frac{A_{LO}^2\Omega}{\|A\|^3} & \frac{-A_L A_{LO}\Omega}{\|A\|^3} & \frac{A_L}{\|A\|} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.3}$$

As stated for the previous linearized models, the partial derivative matrix multiplied times the state vector at the evaluation point is equivalent to the original nonlinear equation. This allows us to make an approximate discretization of the $F(t)$ matrix and apply it to all points along a small forward time step. This discretization is the approximate state transition matrix Φ_{k-1} corresponding to the partial derivative matrix at evaluation point F_{k-1}.

This particular model was abandoned before being refined as the previous models have been. For this reason Φ_{k-1} was approximated at a crude level:

$$\Phi_{k-1} = e^{F_{k-1}\Delta t} \approx I + \Delta t F_{k-1} \tag{6.4}$$

$$\Phi_{k-1} \approx \left.\begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1+\frac{V_x V_y}{\|V\|^3}\Delta t A_L & \frac{-V_x^2}{\|V\|^3}\Delta t A_L & \frac{-V_y}{\|V\|}\Delta t & 0 & 0 \\ 0 & 0 & \frac{V_y^2}{\|V\|^3}\Delta t A_L & 1-\frac{V_x V_y}{\|V\|^3}\Delta t A_L & \frac{V_x}{\|V\|}\Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1+\frac{A_L A_{LO}}{\|A\|^3}\Delta t\Omega & \frac{-A_L^2}{\|A\|^3}\Delta t\Omega & -\frac{A_{LO}}{\|A\|}\Delta t \\ 0 & 0 & 0 & 0 & \frac{A_{LO}^2}{\|A\|^3}\Delta t\Omega & 1-\frac{A_L A_{LO}}{\|A\|^3}\Delta t\Omega & \frac{A_L}{\|A\|}\Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}\right|_{\hat{x}_k}$$

7 Appendix

Filter 4 Discretization

for phi = I + dt*F + 1/2!*dt^2*F^2 + 1/3!*dt^3*F^3

```
> phi[1,1];
                                         1
> phi[1,2];
                                         0
> phi[1,3];
                 2                                  3
               dt  Vy (Vx Al + Vy At)             dt  Vy (Vx Al + Vy At) At
        dt + 1/2 ---------------------- + 1/6 -------------------------
                       2     2 3/2                        2     2 2
                   (Vx  + Vy )                        (Vx  + Vy )

> phi[1,4];
                 2                                  3
               dt  Vx (Vx Al + Vy At)             dt  (Vx Al + Vy At) Vx At
          - 1/2 ---------------------- - 1/6 -------------------------
                       2     2 3/2                        2     2 2
                   (Vx  + Vy )                        (Vx  + Vy )

> phi[1,5];
                              2                 3
                            dt  Vy            dt  (Vx Al + Vy At)
                 - 1/2 -------------- - 1/6 -------------------
                            2     2 1/2              2     2
                        (Vx  + Vy )                 Vx  + Vy

> phi[1,6];
                                            3
                                          dt  Vy
                                 - 1/6 --------------
                                           2     2 1/2
                                       (Vx  + Vy )

> phi[1,7];
                                           2
                                         dt  Vx
                                  1/2 --------------
                                          2     2 1/2
                                      (Vx  + Vy )

************************************************************************
> phi[2,1];
                                         0
> phi[2,2];
                                         1
> phi[2,3];
                2                                   3
              dt  Vy (-Vy Al + Vx At)             dt  (-Vy Al + Vx At) Vy At
        - 1/2 ----------------------- - 1/6 --------------------------
                      2     2 3/2                         2     2 2
                  (Vx  + Vy )                         (Vx  + Vy )

> phi[2,4];
                2                                   3
              dt  Vx (-Vy Al + Vx At)             dt  Vx (-Vy Al + Vx At) At
      dt + 1/2 ----------------------- + 1/6 --------------------------
                      2     2 3/2                          2     2 2
                  (Vx  + Vy )                          (Vx  + Vy )
```

```
> phi[2,5];
                           2                3
                         dt  Vx          dt   (-Vy Al + Vx At)
                1/2 -------------- + 1/6 --------------------
                       2      2 1/2              2      2
                    (Vx  + Vy )               Vx  + Vy

> phi[2,6];
                                         3
                                      dt  Vx
                              1/6 --------------
                                     2      2 1/2
                                  (Vx  + Vy )

> phi[2,7];
                                         2
                                      dt  Vy
                              1/2 --------------
                                     2      2 1/2
                                  (Vx  + Vy )
************************************************************************
> phi[3,1];
                                            0

> phi[3,2];
                                            0

> phi[3,3];
                                   2                            3      2
    dt Vy (Vx Al + Vy At)       dt   Vy (Vx Al + Vy At) At    dt   Vy At   (Vx Al + Vy At)
1 + --------------------- + 1/2 ------------------------- + 1/6 --------------------------
        2      2 3/2                  2      2 2                      2      2 5/2
     (Vx  + Vy )                   (Vx  + Vy )                     (Vx  + Vy )
> phi[3,4];
                                  2                             3      2
   dt Vx (Vx Al + Vy At)        dt   (Vx Al + Vy At) Vx At    dt   Vx At   (Vx Al + Vy At)
 - --------------------- - 1/2 ------------------------- - 1/6 --------------------------
       2      2 3/2                  2      2 2                      2      2 5/2
    (Vx  + Vy )                   (Vx  + Vy )                     (Vx  + Vy )
> phi[3,5];
                                    2                        3
              dt Vy              dt   (Vx Al + Vy At)     dt   At  (Vx Al + Vy At)
          - -------------- - 1/2 ------------------- - 1/6 ----------------------
               2      2 1/2            2      2              2      2 3/2
            (Vx  + Vy )              Vx  + Vy             (Vx  + Vy )

> phi[3,6];
                                  2                 3
                               dt   Vy           dt   (Vx Al + Vy At)
                    - 1/2 -------------- - 1/6 -------------------
                             2      2 1/2             2      2
                          (Vx  + Vy )               Vx  + Vy

> phi[3,7];
                                       dt Vx
                                   --------------
                                      2      2 1/2
                                   (Vx  + Vy )

************************************************************************
> phi[4,1];
                                            0

> phi[4,2];
                                            0

> phi[4,3];
                                     2                               3      2
      dt Vy (-Vy Al + Vx At)       dt   (-Vy Al + Vx At) Vy At     dt   Vy At   (-Vy Al + Vx At)
    - ---------------------- - 1/2 -------------------------- - 1/6 ---------------------------
          2      2 3/2                  2      2 2                       2      2 5/2
       (Vx  + Vy )                   (Vx  + Vy )                      (Vx  + Vy )
```

```
> phi[4,4];
                                        2                                        3   2
       dt Vx (-Vy Al + Vx At)         dt  Vx (-Vy Al + Vx At) At         dt  At  Vx (-Vy Al + Vx At)
   1 + ---------------------- + 1/2 -------------------------- + 1/6 ----------------------------
             2      2 3/2                  2      2 2                       2      2 5/2
          (Vx  + Vy )                   (Vx  + Vy )                      (Vx  + Vy )

> phi[4,5];
                                        2                            3
               dt Vx                  dt  (-Vy Al + Vx At)         dt  At (-Vy Al + Vx At)
          -------------- + 1/2 -------------------- + 1/6 -----------------------
             2      2 1/2               2      2                   2      2 3/2
          (Vx  + Vy )                 Vx  + Vy                  (Vx  + Vy )

> phi[4,6];
                                        2                3
                                      dt  Vx           dt  (-Vy Al + Vx At)
                              1/2 -------------- + 1/6 --------------------
                                     2      2 1/2              2      2
                                  (Vx  + Vy )                Vx  + Vy

> phi[4,7];
                                               dt Vy
                                          --------------
                                             2      2 1/2
                                          (Vx  + Vy )

************************************************************************
> phi[5,1];
                                                 0

> phi[5,2];
                                                 0

> phi[5,3];
                                                 0

> phi[5,4];
                                                 0

> phi[5,5];
                                                 1

> phi[5,6];
                                                dt

> phi[5,7];
                                                 0
************************************************************************
> phi[6,1];
                                                 0

> phi[6,2];
                                                 0

> phi[6,3];
                                                 0

> phi[6,4];
                                                 0

> phi[6,5];
                                                 0

> phi[6,6];
                                                 1

> phi[6,7];
                                                 0
************************************************************************
> phi[7,1];
                                                 0

> phi[7,2];
                                                 0

> phi[7,3];
                                                 0

> phi[7,4];
                                                 0

> phi[7,5];
                                                 0

> phi[7,6];
                                                 0

> phi[7,7];
                                                 1
```

8 References

[1] Andrisani, D., Tao, X., Kuhl, F. P., "A Comparison of Aircraft Trajectory Predictors for Target Tracking Using Simulated Data," *Remote Sensing Reviews*, Vol. 6, Gordon and Breach Science Publishers, 1992, pp. 81-94.

[2] Bird, J.S., *Some Application of Kalman Filtering in Advanced Land Fire Control Systems*, Defence Research Establishment Ottawa, Report No. 1172, 1993.

[3] Bogler, P. L., *Radar Principles with Application to Tracking Systems*, Wiley & Sons, New York, 1990.

[4] Chang, C. B. and J. A. Tabaczynski, "Application of State Estimation to Target Tracking," IEEE Transactions on Automatic Control, Vol. AC-29, No. 2, 1984.

[5] Cutaia, N. J. and O'Sullivan, J. A., *Identification of Maneuvering Aircraft Using Class Dependent Kinematic Models*, Electronic Signals and Systems Research Laboratory Monograph ESSRL-95-13, St. Louis, MO, 1995.

[6] Gelb, A., editor, Applied Optimal Estimation. The MIT Press, Cambridge, MA, 1974.

[7] Maybeck, P. S., *Stochastic Models, Estimation, and Control: Volume 1*. Vol. 141-1 of Mathematics in Science and Engineering. Academic Press, Boston, 1979.

[8] Maybeck, P. S., *Stochastic Models, Estimation, and Control: Volume 2*. Vol. 141-2 of Mathematics in Science and Engineering. Academic Press, Boston, 1982.

[9] McConley, M.W., *Nonlinear Estimation for Gyroscope Calibration for the Inertial Pseudo Star Reference Unit*, Master's Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1994.

[10] Perkins, T. R., "Sub-optimal State Estimation as Related to Predictive Fire Control System Design," *Proceedings of the First Meeting of the Special Interest Group on Control Theory*, Aberdeen Proving Ground, MD, 1980, pp. 9-19.

[11] Stevens, B. L., Lewis, F. L., *Aircraft Control and Simulation*, John Wiley & Sons, New York, 1992.