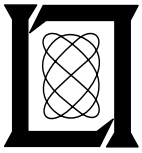
**Project Report**
**ATC-360**

# Model-Based Optimization of Airborne Collision Avoidance Logic

M.J. Kochenderfer
J.P. Chryssanthacopoulos
L.P. Kaelbling
T. Lozano-Perez

26 January 2010

**Lincoln Laboratory**
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
*LEXINGTON, MASSACHUSETTS*



Prepared for the Federal Aviation Administration,
Washington, D.C. 20591

This document is available to the public through
the National Technical Information Service,
Springfield, Virginia 22161

TECHNICAL REPORT STANDARD TITLE PAGE

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| ATC-360 | | |

| 4. Title and Subtitle | 5. Report Date |
|---|---|
| Model-Based Optimization of Airborne Collision Avoidance Logic | 26 January 2010 |
| | 6. Performing Organization Code |

| 7. Author(s) | 8. Performing Organization Report No. |
|---|---|
| M.J. Kochenderfer, J.P. Chryssanthacopoulos, L.P. Kaelbling, T. Lozano-Perez | ATC-360 |

| 9. Performing Organization Name and Address | 10. Work Unit No. (TRAIS) |
|---|---|
| MIT Lincoln Laboratory<br>244 Wood Street<br>Lexington, MA 02420-9108 | 11. Contract or Grant No.<br>FA8721-05-C-0002 |

| 12. Sponsoring Agency Name and Address | 13. Type of Report and Period Covered |
|---|---|
| Department of Transportation<br>Federal Aviation Administration<br>800 Independence Ave., S.W.<br>Washington, DC 20591 | Project Report |
| | 14. Sponsoring Agency Code |

15. Supplementary Notes

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology, under Air Force Contract FA8721-05-C-0002.

16. Abstract

The Traffic Alert and Collision Avoidance System (TCAS) is designed to reduce the risk of mid-air collisions by providing resolution advisories to pilots. The current version of the collision avoidance logic was hand-crafted over the course of many years and contains many parameters that have been tuned to varying extents and heuristic rules whose justification has been lost. Further development of the TCAS system is required to make the system compatible with next generation air traffic control procedures and surveillance systems that will reduce separation between aircraft. This report presents a decision-theoretic approach to optimizing the TCAS logic using probabilistic models of aircraft behavior and a cost metric that balances the cost of alerting with the cost of collision. Such an approach has the potential for meeting or exceeding the current safety level while lowering the false alert rate and simplifying the process of re-optimizing the logic in response to changes in the airspace and sensor capabilities.

| 17. Key Words | 18. Distribution Statement |
|---|---|
| | This document is available to the public through the National Technical Information Service, Springfield, VA 22161. |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of Pages | 22. Price |
|---|---|---|---|
| Unclassified | Unclassified | 126 | |

This page intentionally left blank.

# EXECUTIVE SUMMARY

The Traffic Alert and Collision Avoidance System (TCAS) is a widely-deployed safety system for reducing the risk of mid-air collision between aircraft. TCAS II provides advisories to pilots on how to resolve potential conflicts. The threat resolution logic of TCAS II is the product of several decades of research by many organizations. The logic was tested in simulation on a large collection of encounter scenarios generated by models derived from operational data. The system was evaluated using various performance metrics, including collision risk and false alert rate. The development cycle involved analyzing problematic encounters and adapting the logic manually, which was then re-evaluated in simulation.

As the airspace evolves with the introduction of new air traffic control procedures and surveillance systems, it is likely that the TCAS II threat detection and resolution logic will require modification to meet safety and operational requirements. Due to the complexity of the logic, modifying the logic may require significant engineering effort. This report suggests a new approach to TCAS logic development where the engineering effort is focused on developing models, allowing computers to optimize the logic according to agreed-upon performance metrics. Because models of sensor characteristics, pilot response behavior, and aircraft dynamics can be constructed from operational data, they should be straightforward to justify and vet within the safety community. The optimization of the logic according to these models would be done using principled techniques that are well established in theory and practice over the past 50 years.

The objective of this report is not to develop a particular conflict resolution algorithm, but to connect this concept of TCAS logic optimization to the existing literature on model-based optimization. Problems involving sequential decision making in a dynamic environment are typically modeled by a Markov decision process, where the state at the next decision point depends probabilistically on the current state and the chosen action. Assuming some objective measure of cost, the best action from the current state is the one that minimizes the expected future cost. Dynamic programming can be used to solve for the optimal action from all possible states.

To illustrate some of the key concepts of how dynamic programming might be applied to TCAS logic optimization, this report uses a simple encounter model and evaluates the resulting logic in simulation using various performance metrics. This report identifies some of the issues with applying a dynamic programming approach. One issue is the scalability of existing solution methods to higher dimensions. Adding additional dimensions results in an exponential increase in memory and computational requirements, but several techniques suggested in this report address this issue.

This report discusses alternatives to dynamic programming. One approach that has been explored by others involves using conflict probability estimates to decide when to issue resolution advisories. Although this approach will not result in the optimal solution, it may approximate the optimal logic well. This report also discusses the strengths and weaknesses of other approaches, such as rapidly-expanding random trees, potential field methods, policy search, and geometric optimization. One of the primary strengths of the dynamic programming approach over these other methods is that it directly leverages models of sensor error and aircraft behavior to find the optimal logic.

Although this report focuses primarily on the computational aspect of optimizing collision avoidance logic, there are other issues that require further study. In particular, since this is a new approach to TCAS logic development, the certifiability of the resulting logic is of particular concern. If this new approach is to be used simply as an aid to engineers who are developing or revising collision avoidance pseudocode, then there would be little impact on the certification process. However, if the logic produced by dynamic programming or some other automated process is to be used directly in a future version of TCAS, then the certification process may be somewhat different. The core of the certification process will be the same, involving rigorous simulation studies and flight tests to prove safety and demonstrate operational acceptability. However, the vetting of the logic itself will involve more than just studying the logic that will be deployed on the system. Depending on the representation of the logic, it may not be directly comprehensible by an engineer. Therefore, confidence would need to be established in the safety community that the methods used to generate the logic are sound. This report represents a first step in justifying an automated approach for generating optimized TCAS logic.

## ACKNOWLEDGMENTS

This report is the result of research and development sponsored by the TCAS Program Office at the FAA. The authors appreciate the assistance provided by the TCAS Program Manager, Neal Suchy, who has been supportive of pursuing new ideas and innovation for advancing aviation safety.

This work has greatly benefited from discussions with Thomas B. Billingsley, Barbara J. Chludzinski, Ann C. Drumm, Matthew W. M. Edwards, Leo P. Espindle, J. Daniel Griffith, William H. Harman, James K. Kuchar, Wesley A. Olson, David A. Spencer, Selim Temizer, Panayiotis C. Tzanos, Roland E. Weibel, Richard E. Williams, and M. Loren Wood.

This page intentionally left blank.

## TABLE OF CONTENTS


| | | Page |
|---|---|---|
| | Executive Summary | iii |
| | Acknowledgments | v |
| | List of Illustrations | xi |
| | List of Tables | xiii |
| 1. | INTRODUCTION | 1 |
| | 1.1 Objectives | 1 |
| | 1.2 TCAS Logic | 3 |
| | 1.3 Hypothetical Collision Avoidance Problem | 7 |
| | 1.4 Overview | 8 |
| 2. | PROBLEM FORMULATION | 11 |
| | 2.1 Performance Metric | 11 |
| | 2.2 State Uncertainty | 12 |
| | 2.3 Dynamic Uncertainty | 13 |
| | 2.4 Optimal Decision Making | 13 |
| | 2.5 Discussion | 15 |
| 3. | DYNAMIC PROGRAMMING APPROACH | 17 |
| | 3.1 Fitted Value Iteration | 17 |
| | 3.2 Reachable State Space | 18 |
| | 3.3 Branch and Bound | 19 |
| | 3.4 Policy Plots | 21 |
| | 3.5 Discussion | 23 |
| 4. | CONFLICT PROBABILITY ESTIMATION APPROACH | 25 |
| | 4.1 Analytical Approximation | 25 |
| | 4.2 Dynamic Programming | 26 |
| | 4.3 Monte Carlo | 26 |

4.4 Estimation Comparison 28
4.5 Policy Generation 29
4.6 Policy Plots 32
4.7 Discussion 38

5. EVALUATION 41
5.1 Alerting System Outcomes 41
5.2 Metrics 42
5.3 System Operating Characteristic Curves 44
5.4 Preliminary Safety Evaluation 49
5.5 Example Encounters 56
5.6 Discussion 58

6. ALTERNATIVE APPROACHES 61
6.1 Potential Field Methods 61
6.2 Rapidly Expanding Random Trees 61
6.3 Geometric Optimization 62
6.4 Policy Search 62
6.5 Nonstationary Approximate Policy Iteration 63
6.6 Discussion 64

7. FURTHER RESEARCH 65
7.1 Three-Dimensional Dynamics 65
7.2 Nondeterministic Pilot Response 66
7.3 Strengthening and Reversal Advisories 67
7.4 Equipped Intruders 68
7.5 Model Robustness Analysis 69
7.6 Non-Gaussian Dynamics 69
7.7 Additional Resolution Advisories 69
7.8 Leveraging Intent Information 70
7.9 Multiple Intruders 70

7.10 Logic Representation 70

8. CONCLUSIONS 73

A. HYPOTHETICAL COLLISION AVOIDANCE DYNAMICS 75

B. ANALYTIC APPROXIMATION 77

C. INTERPOLATION METHODS 79
C.1 Nearest-Neighbor Interpolation 79
C.2 Multilinear Interpolation 79
C.3 Simplex-based Interpolation 80
C.4 Local Lagrange Interpolation 81

D. SAMPLING METHODS 83
D.1 Direct Monte Carlo 83
D.2 Importance Sampling 83
D.3 Sigma-Point Sampling 84

E. PROPOSAL DISTRIBUTIONS 85
E.1 Constant Acceleration Proposal 86
E.2 Maximum-Likelihood Acceleration Proposal 88
E.3 Analytic Proposal 89
E.4 Dynamic Programming Proposal 91

F. PROOF OF PARETO OPTIMALITY 93

G. COMPUTING THE TIME-TO-CONFLICT DISTRIBUTION 95

H. TRACKER 97

I. MINI TCAS 99
I.1 Threat Detection 100
I.2 Sense Selection 101
I.3 Strength Selection 102

References 103

Notation 111

## LIST OF ILLUSTRATIONS


| Figure No. | | Page |
|---|---|---|
| 1 | Initial RA sense selection process. | 6 |
| 2 | Hypothetical collision avoidance problem. | 7 |
| 3 | Data flow between the tracker and logic. | 15 |
| 4 | Notional diagram showing reachable states under the optimal policy and reachable states under any policy. | 19 |
| 5 | Effect of branch-and-bound pruning. | 20 |
| 6 | Collision avoidance policies generated using dynamic programming. | 22 |
| 7 | Evolution of the Gaussian distribution of $h$ until CPA. | 26 |
| 8 | Comparison of state propagation using direct sampling versus importance sampling. | 27 |
| 9 | Convergence plots from several different states for the no alert action. | 30 |
| 10 | Convergence plots from different states for the descend action. | 31 |
| 11 | Decision trees for alerting policies. | 33 |
| 12 | A notional diagram illustrating the difference between the three alerting strategies. | 34 |
| 13 | Collision avoidance policy generated by conservative strategy. | 35 |
| 14 | Collision avoidance policy generated by delay strategy. | 36 |
| 15 | Collision avoidance policy generated by conservative delay strategy. | 37 |
| 16 | An example of a situation where dynamic programming dominates a conflict probability approach. | 39 |
| 17 | Alerting outcome categories. | 42 |
| 18 | Example SOC curve. | 44 |
| 19 | Simple simulation framework. | 45 |
| 20 | SOC curves for the DP logic. | 46 |
| 21 | Effect of different sampling methods on overall system performance. | 48 |
| 22 | SOC curves for threshold alerting policies. | 50 |
| 23 | CASSATT simulation framework. | 51 |

| | | |
|---|---|---|
| 24 | Risk ratio convergence. | 55 |
| 25 | Comparison of VMD using TCAS v. 7.1 with VMD using the DP logic. | 56 |
| 26 | Simulated encounter in which the DP logic prevents an NMAC that TCAS fails to prevent. | 57 |
| 27 | Simulated encounter in which TCAS prevents an NMAC that the DP logic fails to prevent. | 58 |
| 28 | Estimation of time to horizontal conflict. | 66 |
| 29 | Transition model for $s_{\mathrm{RA}}$ where the response delay is given by a geometric distribution. | 67 |
| A-1 | The variable $s_{\mathrm{RA}}$ tracks which RA has been issued and the delay. | 75 |
| C-1 | Multilinear interpolation. | 80 |
| C-2 | Comparision of interpolation methods in two dimensions. | 82 |
| E-1 | Comparison of distributions $p$ and $q$. | 87 |

## LIST OF TABLES


| Table No. | | Page |
|---|---|---|
| 1 | List of initial RAs. | 4 |
| 2 | Distribution of aircraft start states in the hypothetical collision avoidance problem. | 8 |
| 3 | Grid edges for the hypothetical collision avoidance problem. | 18 |
| 4 | Outcome categories. | 41 |
| 5 | Own altitude, sensitivity level, and altitude layer of TCAS operating points. | 46 |
| 6 | Probability of each alerting system outcome. | 53 |
| 7 | Risk ratios with TCAS v. 7.1 and the DP logic. | 54 |
| I-1 | Constants used in mini TCAS. | 99 |

This page intentionally left blank.

# 1. INTRODUCTION

The Traffic Alert and Collision Avoidance System (TCAS) is designed to reduce the rate of mid-air collisions between aircraft. TCAS I, intended primarily for general aviation aircraft, provides traffic advisories (TAs) to pilots. In addition to TAs, TCAS II provides resolution advisories (RAs) that instruct pilots on how to resolve potentially hazardous situations. The threat resolution logic of TCAS II has been shown to significantly reduce the risk of collision when other safety layers, such as air traffic control services, have failed to maintain safe separation between aircraft [1–4]. TCAS II is currently mandated worldwide on board all large transport aircraft.

The threat declaration and resolution logic in the current version of TCAS is the product of several decades of research by different organizations. During the development of TCAS, the logic was tested in simulation on a large collection of encounter scenarios. The encounters were generated randomly from models derived from operational data. The performance of TCAS was evaluated using various performance metrics, including collision risk and false alarm rate. The development cycle involved analyzing problematic encounters and adapting the logic manually, which was then re-evaluated in simulation.

Due to the complexity of TCAS, modifying the logic to correct issues identified in simulation can be difficult. For example, recent efforts to correct an issue with the critical interval portion of the logic were complicated due to unanticipated ripples in other parts of the logic. Next-generation air traffic control procedures and new sensor systems like Automatic Dependent Surveillance-Broadcast (ADS-B) will likely require re-engineering much of the system and tuning many parameters embedded in the logic.

This report investigates a decision-theoretic approach to developing collision avoidance logic that directly leverages encounter models to optimize threat-resolution behavior. In this framework, the optimal resolution advisory is the one that provides the best expected outcome, balancing the competing objectives of minimizing unnecessary alerts and collisions. The engineering effort is focused on building models instead of designing complex logic.

Since the development of TCAS, there have been significant technological and algorithmic advances that may make an automated logic optimization approach practical. The theory of optimal decision making under uncertainty has been applied to a wide variety of problems, from robotic control to medical diagnosis. Leveraging these advances in theory and practice has the potential of shortening the TCAS development cycle, reducing unnecessary alerts, and making the system more robust to unexpected events. As the airspace and sensor capabilities evolve, the system can be re-optimized based on updated models with relatively little engineering effort.

## 1.1 OBJECTIVES

The objectives of this report are as follows:

1. **Provide motivation:** A model-based optimization approach to collision avoidance is a significant departure from how the TCAS logic has been developed in the past. This report

will identify the strengths and challenges associated with such an approach and discuss why consideration of a new approach may be worthwhile. (Section 2)

2. **Connect problem to existing literature:** A large body of literature exists on optimal decision making under uncertainty, and there has been substantial progress made in both theory and practice in recent years that has not been fully leveraged by the TCAS development community. This report will connect the problem of collision avoidance to the key concepts, models, and solution methods developed in the field. (Section 3)

3. **Demonstrate concepts on simple model:** Because of the complexity of a realistic model of the airborne collision avoidance problem, this report focuses on a simplified model (described later in this section) that has many of the important attributes of the real problem but can be discussed more easily. This simplified model will be used to demonstrate the key concepts of the approach. (Section 3)

4. **Discuss performance metrics:** Many of the performance metrics used to evaluate previous versions of TCAS can be applied to measure the performance of a system developed using the proposed approach. This report will compare the performance of the new approach against the existing logic using several different metrics. (Section 5)

5. **Compare with alternative approaches:** A wide variety of other approaches to airborne collision avoidance have been suggested in the literature. This report will briefly survey some of the most significant methods and relate them to the proposed decision-theoretic approach. (Sections 4 and 6)

6. **Identify issues for further research:** Further research will be required to scale the simple model and solution methods to a working prototype system. This report will focus largely on the computational issues but will also discuss other issues related to certification. (Section 7)

The scope of this report includes only the threat-resolution behavior of TCAS II. This report will only discuss the conditions for issuing resolution advisories, not the conditions for issuing traffic advisories. The approach suggested in this report can accommodate different surveillance systems, including Mode S and ADS-B, but the discussion in most of this report does not depend upon the specifics of the surveillance system.

This report does not advocate any specific collision avoidance logic—only an approach that can lead to the development of new logic or suggestions for improving existing logic. The development of a new logic will require the participation and consensus of many different organizations, and the certification process will involve rigorous simulation studies and flight tests as done historically with previous versions of TCAS.

The remainder of this section provides a short introduction to the existing TCAS logic and describes the hypothetical collision avoidance problem used to demonstrate the approach suggested in this report. Throughout this report, the aircraft equipped with TCAS is called *own aircraft*, while a potentially hazardous nearby aircraft, which is not necessarily equipped with TCAS, is called the *intruder*.

## 1.2 TCAS LOGIC

TCAS decides to issue RAs based upon range and altitude separation criteria. If it is projected that range separation with an intruder might be lost within a threshold time and that there might not be at least a minimum altitude separation during the time interval when range separation might be lost, an RA will be triggered. The timing of an RA depends on the values of various parameters used in the resolution logic. The values of these parameters, in turn, depend on the altitude layer of own aircraft and the so-called sensitivity level. Higher sensitivity levels generally result in earlier RAs. When the sensitivity level is set to two, its lowest value, the generation of RAs is inhibited. During normal operation, the sensitivity level is most often determined automatically based on the altitude of own aircraft.

TCAS issues either upward or downward sense RAs. The Climb and Descend RAs advise the pilot to climb or descend at least 1500 ft/min, respectively. These RAs are called positive RAs because they require positive action on the part of the pilot in order to resolve the conflict. They generally require the most aggressive maneuvers from the pilot. Other positive RAs include Crossing Climb, when own aircraft is projected to cross altitudes with the intruder while climbing, and Maintain Climb, when own aircraft is already climbing at least 1500 ft/min. In addition to these, TCAS also issues RAs that correspond to vertical speed limits (VSLs), such as Do Not Climb and Do Not Climb $>$ 500 ft/min. The least restrictive, or weakest, VSL is Do Not Climb $>$ 2000 ft/min, while the most restrictive, or strongest, VSL is Do Not Climb (and their upward sense counterparts). The VSLs are also called negative RAs because they only require that the pilot not do something to avoid conflict. Table 1 is a list of all possible RAs that TCAS can initially issue to the pilot. Depending on the development of an encounter, TCAS can modify the strength of the RA or issue sense reversals and increase rate RAs if it predicts that the vertical separation, perhaps due to neglect of the issued RA, will be insufficient in the future.

The TCAS logic consists of several components: threat detection, initial sense selection, initial strength selection, and encounter monitoring and RA modification. The following sections outline the behavior of these components; further detail can be found elsewhere [5, 6].

### 1.2.1 Threat Detection

The threat detection component of the logic determines whether any nearby altitude-reporting aircraft are potential collision threats. TCAS decides to issue RAs based on the projected time until the closest point of approach (CPA) in range and in altitude. These quantities are called the range tau and the vertical tau, respectively. The range tau is equal to the slant range to the intruder divided by the closure rate. It is the amount of time required to reach zero separation assuming constant closure rate for the remainder of the encounter. The vertical tau, similarly, is equal to the altitude separation divided by the relative vertical rate between the aircraft.

To account for the possibility that the intruder might accelerate toward own aircraft at some time in the future, thereby shortening the time to CPA, a second quantity is defined called modified tau, which is always less than tau. Modified tau assumes a particular model for future range acceleration by the intruder toward own aircraft. It is approximately equal to the time until the

**TABLE 1**

**List of initial RAs, adapted from [5].**

| | Upward Sense | | Downward Sense | |
|---|---|---|---|---|
| RA Type | RA | Target Rate (ft/min) | RA | Target Rate (ft/min) |
| Positive | Climb | 1500 to 2000 | Descend | $-1500$ to $-2000$ |
| Positive | Crossing Climb | 1500 to 2000 | Crossing Descend | $-1500$ to $-2000$ |
| Positive | Maintain Climb | 1500 to 4400 | Maintain Descend | $-1500$ to $-4400$ |
| Negative | Do Not Descend | $> 0$ | Do Not Climb | $< 0$ |
| Negative | Do Not Descend $> 500$ | $> -500$ | Do Not Climb $> 500$ | $< 500$ |
| Negative | Do Not Descend $> 1000$ | $> -1000$ | Do Not Climb $> 1000$ | $< 1000$ |
| Negative | Do Not Descend $> 2000$ | $> -2000$ | Do Not Climb $> 2000$ | $< 2000$ |

intruder aircraft is projected to penetrate a safety buffer around own aircraft, defined as a sphere of radius DMOD, and is zero when the intruder range is less than or equal to DMOD. Modified tau also addresses another problem with using tau alone to define the time until CPA and to trigger RAs: at very slow closure rates, the intruder can get very close to own aircraft before tau becomes small enough to trigger an RA. The interval between modified tau and tau is called the critical interval and is taken to be the interval of time during which horizontal separation could be lost.

Additional constraints are placed on tau and modified tau to avoid problems that arise if the intruder is not on a direct collision course (which TCAS cannot always determine based on range and range rate measurements) or if the intruder is closing very slowly. In the former case, tau and modified tau will reach a minimum and begin to increase prior to CPA, giving a false estimate of the remaining time until CPA. In the latter case, their values can become unreasonably large. To prevent this, the critical interval is limited to a maximum time interval of interest.

If the intruder is diverging in altitude, vertical tau is not meaningful. In that case, the decision as to whether to issue an RA is based on whether the projected vertical separation during the critical interval exceeds a minimum threshold.

Before being declared a threat, the intruder aircraft must pass both range and altitude tests. The range test checks whether modified tau is less than a certain time threshold. The time threshold, called TRTHR, is typically larger at higher altitudes. As part of the range test, TCAS also performs several nuisance alarm tests to prevent intruders that have large horizontal miss distances from being declared threats. It also may delay threat detection if it appears likely that by doing so it can avoid issuing an RA that forces the aircraft to cross altitudes.

The altitude test depends on whether the vertical separation is currently less than a threshold. This threshold, called ZTHR, is a design parameter of the logic and is a function of the altitude layer. If the intruder's vertical separation is less than ZTHR, the intruder passes the altitude test if the projected vertical separation during the critical interval is also less than ZTHR. If the current vertical separation is greater than ZTHR, the altitude test is passed if the intruder is converging in altitude and vertical tau is less than a threshold called TVTHR. TVTHR is typically larger at higher

altitudes and is typically the same as the range test threshold TRTHR. During threat detection, future aircraft altitudes are predicted using linear extrapolation.

### 1.2.2 Initial Sense Selection

If TCAS declares an intruder a threat, it proceeds to select the sense of the initial RA to issue the pilot. The sense can either be up or down, depending on the maneuver performed as a result of executing the RA. The up sense indicates that own aircraft is expected to pass above the intruder as a result of RA execution, the down sense that own aircraft is expected to pass below. If the sense is anticipated to cause the aircraft to cross in altitude, the sense is termed an altitude-crossing sense. TCAS is strongly biased against selecting altitude-crossing RAs.

Often the sense selection is determined by the intruder, if it is TCAS-equipped and happens to declare own aircraft a threat and performs sense selection first. In that case it will send a sense-coordination message to own aircraft. If the sense of the intruder RA is not altitude-crossing, own TCAS will select the complementary sense for its RA. However, if the sense transmitted by the intruder is an altitude-crossing sense, TCAS may independently select its sense, provided certain conditions are met—e.g., the Mode S address of own aircraft must be smaller than that of the intruder, among others. If it selects the sense that is not complementary to that selected by the intruder, its sense-coordination message to the intruder will cause the intruder to reverse its sense selection.

In the event that TCAS determines its own RA sense, it models the response to both Climb and Descend RAs and calculates the projected vertical separation during the critical interval. As in the threat detection component, TCAS models the intruder's trajectory as a straight line with a constant vertical rate. As for own aircraft, TCAS implements a pilot delay model in which the pilot requires five seconds to respond to an initial RA. After the pilot-response delay, the own aircraft is modeled as accelerating at 0.25 g until reaching the target vertical rate, after which it maintains that rate for the remainder of the encounter. The target vertical rate is 1500 ft/min for a Climb RA and −1500 ft/min for a Descend RA. If own vertical rate is greater than 1500 ft/min in the sense direction, the target vertical rate is the current vertical rate of own aircraft in that direction up to a maximum modeled target vertical rate of 4400 ft/min.

TCAS issues RAs to ensure that the aircraft maintain at least the desired minimum vertical separation during the critical interval. The desired minimum vertical separation, called ALIM, is a design parameter of the logic. It is a function of the altitude layer and is less than the threshold ZTHR.

TCAS will select the non-altitude-crossing sense, provided the aircraft are not within 100 ft of each other vertically, if the projected vertical separation is at least equal to ALIM. Additionally, an altitude crossing must not be projected to be inevitable. If these conditions are not met, TCAS selects the sense that provides the greater separation. In the case that both senses provide the same separation, TCAS always selects the down sense. Note that if the non-altitude-crossing sense provides at least the minimum separation, TCAS selects this sense over the altitude-crossing sense even if the latter provides more separation.

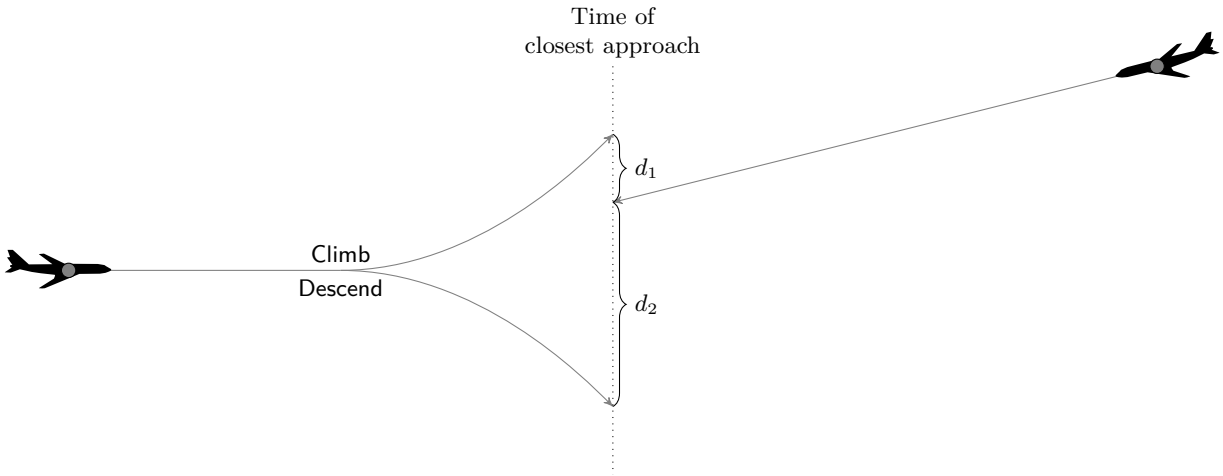



*Figure 1. Initial RA sense selection process.*

Figure 1 is an illustration of how RA sense selection is performed. Because the up-sense RA is an altitude-crossing RA, TCAS will select the down-sense RA if $d_2 \geq$ ALIM and the other conditions of the previous paragraph are met. Otherwise, TCAS will select the sense that provides the greater separation, which again happens to be the down sense because $d_2 > d_1$.

### 1.2.3 Initial Strength Selection

The initial sense selection component determines whether a Climb or Descend RA, depending on the sense selected, is projected to provide at least the minimum separation, or at least more separation than the opposite sense. However, there may exist a vertical speed limit of the same sense that provides sufficient separation and is less disruptive to the own aircraft trajectory. When no prior RA has been issued against a particular intruder, the strength selection component of the logic determines the weakest VSL that achieves at least the desired separation during the critical interval. If none of the VSLs provides sufficient separation, the logic will select the positive Climb or Descend RA as appropriate for the selected sense.

### 1.2.4 Encounter Monitoring and RA modifications

TCAS continues to monitor the development of an encounter and, if necessary, modifies the initial RA that it issued. TCAS re-evaluates the effect of the current RA strength during every update cycle after initial RA selection. If the selected RA is a VSL, it will be strengthened to a stronger VSL or to a positive Climb or Descend if necessary to ensure adequate separation during the critical interval. The strength of an RA is never reduced except when a Climb or Descend RA can be weakened to a Do Not Descend or Do Not Climb VSL when own aircraft can safely level off.

If a Climb or Descend RA has been issued against a certain intruder but subsequent update cycles indicate insufficient vertical separation, the logic performs tests to determine the feasibility of reversing the sense of the RA. If the reversal tests do not permit sense reversal, the Climb or Descend

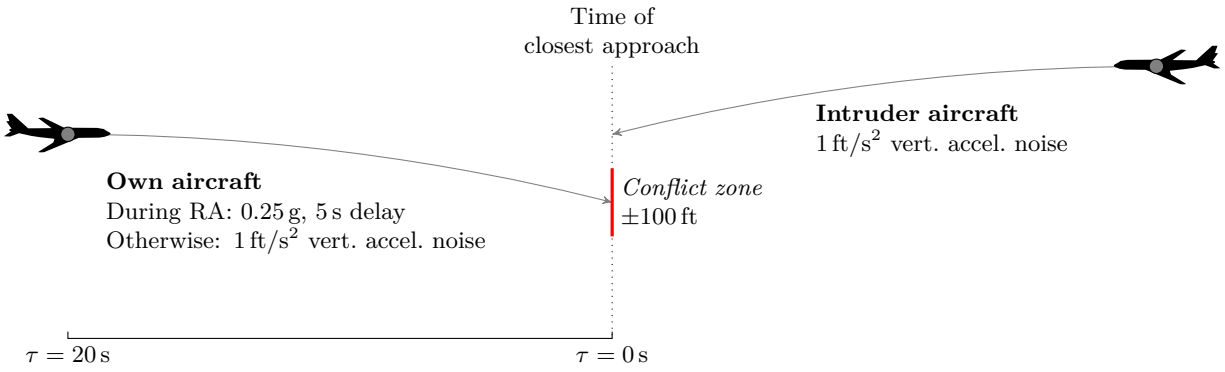



*Figure 2. Hypothetical collision avoidance problem.*

RAs may be strengthened to Increase Climb or Increase Descend RAs, respectively, under certain conditions. These increase rate RAs advise the pilot to climb or descend at least 2500 ft/min.

The pilot-response delay assumed by TCAS in projecting the results of all RA modifications is 2.5 seconds. For sense reversals and increase rate RAs, own aircraft is assumed to accelerate at 0.35 g until reaching the target vertical rate.

### 1.3 HYPOTHETICAL COLLISION AVOIDANCE PROBLEM

This report uses a relatively simple, hypothetical collision avoidance problem to demonstrate the concept of model-based logic optimization. There is a single unequipped intruder approaching with a constant closure rate. The collision avoidance system on the own aircraft may issue one of two RAs that correspond to climb and descend vertical maneuvers. The Climb RA advises the pilot to climb at least 1500 ft/min. The Descend RA advises the pilot to descend at least 1500 ft/min. When an RA is issued, it takes the pilot five seconds to respond, after which the pilot applies a 0.25 g acceleration to meet the minimum desired vertical rate. If the pilot is already complying with the RA (e.g., descending 2500 ft/min when a descend RA is issued), no acceleration is applied. The intruder vertical acceleration follows a white noise model with a standard deviation of 1 ft/s$^2$. When the own aircraft is not responding to an RA, its vertical acceleration follows the same noise model as the intruder. Figure 2 summarizes the properties of the problem.

The state variables are as follows:

- $h$, the altitude of intruder relative to own,
- $\tau$, the time to closest horizontal approach,
- $\dot{h}_1$, own vertical rate,
- $\dot{h}_2$, intruder vertical rate, and
- $s_{\text{RA}}$, RA state variable.

**TABLE 2**

**Distribution of aircraft start states in the hypothetical collision avoidance problem.**

| Variable | Range | Distribution |
|---|---|---|
| $h$ (ft) | $[-500, 500]$ | uniform |
| $\tau$ (s) | 20 | — |
| $\dot{h}_1$ (ft/min) | $[-1000, 1000]$ | uniform |
| $\dot{h}_2$ (ft/min) | $[-1000, 1000]$ | uniform |
| $s_{\mathrm{RA}}$ | clear of conflict | — |

In this problem, a conflict occurs when $\tau = 0$ and $|h| < 100\,\mathrm{ft}$. Appendix A provides a mathematical specification for the probabilistic dynamics of these variables.

The initial state is selected randomly according to the distribution specified in Table 2. All encounters start 20 seconds prior to closest horizontal approach. The values for $h$, $\dot{h}_1$, and $\dot{h}_2$ are chosen uniformly within the specified ranges. No RA was previously issued prior to the start of the encounter, hence the clear of conflict RA state. The distribution was chosen so that the system on own aircraft alerts at a relatively frequent rate.

Although the model contains five state variables, it only adequately represents motion in a head-on encounter in two spatial dimensions. In three spatial dimensions, aircraft may be adequately separated horizontally at the time of closest approach ($\tau = 0$). It is possible to extend this model into three spatial dimensions by changing $\tau$ to mean the time to horizontal conflict and modeling the influence of horizontal motion on $\tau$ (Section 7.1). Recently, several probabilistic models have been developed based on radar data [7–10]. These can be integrated into the current framework.

## 1.4 OVERVIEW

This section outlined the objectives of this report, briefly outlined the TCAS logic, and described a simple encounter model to be used as a running example to illustrate the decision-theoretic approach introduced in this report. The remainder of this report proceeds as follows.

Section 2 formulates collision avoidance as an optimization problem. It begins by formally specifying the performance metric to be optimized in terms of conflict and alerting probabilities. Because sensor information is imperfect and the future behavior of the aircraft involved in the encounter cannot be known perfectly, the optimal logic will need to leverage models of uncertainty in observation and state dynamics. This section connects the requirements of an optimal logic to the body of work on Markov decision processes that can be solved using dynamic programming.

Section 3 discusses how dynamic programming can be used to find the optimal logic. It presents a dynamic programming algorithm called fitted value iteration that involves discretizing the state space and using a local averaging scheme to interpolate between the discrete states. Because

the number of discrete states can become prohibitively large, methods to reduce the planning complexity are introduced.

Section 4 presents an alternative method for developing collision avoidance logic that relies on online sample-based estimates of the probability of future conflict between aircraft. Various methods for estimating the probability of conflict are introduced. A variance-reduction technique known as importance sampling is shown to provide more accurate estimates using fewer samples than the naive approach. Several ways of using the probability of conflict in the development of collision avoidance logic are discussed.

Section 5 discusses performance metrics and summarizes the results of the preliminary evaluation of the logic developed in Sections 3 and 4. Performance tradeoffs inherent in the development of collision avoidance systems are analyzed through the use of system operating characteristic curves. This section also focuses on the instrumental role these curves play in the effective placement of system parameters. The optimal logic is evaluated in a safety assessment tool developed at Lincoln Laboratory and used in prior TCAS safety studies. The results of this preliminary evaluation are presented.

Section 6 describes a collection of alternative approaches suggested in the literature for conflict avoidance, including potential field methods, rapidly exploring random trees, geometric optimization, and policy search. Advantages and disadvantages of each approach are discussed.

Section 7 outlines several areas of further research. The simple collision avoidance model introduced earlier needs to be extended in several different ways. For example, the model will need to be adapted to support advisory strengthening and reversal, equipped intruders, and nondeterministic pilot response. Further investigation is also required into policy representation and issues of model robustness.

Section 8 concludes the report.

There are nine appendices that further elaborate on concepts introduced in the body of this report. Notation is local to the section or appendix in which it is introduced, except for the notation used for the state variables (Section 1.3) and for dynamic programming (Section 3), which are summarized in a table following the references.

This page intentionally left blank.

# 2. PROBLEM FORMULATION

Before attempting to find an *optimal* collision avoidance system logic, it is important to define the metric by which collision avoidance systems are to be measured and compared. Section 2.1 describes a performance metric that may be used. Logic that optimizes performance must take into account state uncertainty and dynamic uncertainty, both of which are discussed in Sections 2.2 and 2.3. Section 2.4 discusses how to use models of state and dynamic uncertainty to make optimal decisions. Further discussion of some of the points introduced in these earlier sections is reserved for Section 2.5.

## 2.1 PERFORMANCE METRIC

The performance metric needs to take into account the competing objectives of preventing conflict and minimizing alert rate (where "alert" refers to an RA, not a TA, for the purposes of this report). Other objectives may also be taken into consideration, such as flight-plan deviation. Prior TCAS studies defined a conflict as a loss of separation 100 ft vertically and 500 ft horizontally [4]. Such conflicts have been called near mid-air collisions (NMACs). Although this report focuses on conflicts with other aircraft, in general conflicts could involve other forms of failure, such as collision with terrain.

One way to balance conflict prevention with alert minimization is to define a cost metric that is a function of whether a conflict or an alert occurred. Let $C(\omega)$ be one if a conflict occurred during encounter $\omega$ and zero otherwise, and let $A(\omega)$ be one if the encounter involved an alert and zero otherwise. The cost associated with a particular encounter $\omega$ may be denoted

$$c(\omega) = C(\omega) + \lambda A(\omega). \tag{1}$$

The scalar $\lambda$ is a parameter of the cost function that controls the cost of alerting relative to the cost of conflict. This cost function amounts to assigning unit cost to conflict and a cost of $\lambda$ to an alert. The value of $\lambda$ depends on the preference of the evaluators of the system. It may be difficult for human designers to think in terms of relative costs, so instead of choosing $\lambda$ directly, it may be desirable to choose the $\lambda$ that translates to the desired safety threshold as discussed in Section 5. The linear form of Equation 1 is often used for multiple-attribute decision problems [11], but the cost function need not be linear in general. Winder and Kuchar, for example, assign the same cost to any encounter that involves a collision, regardless of whether an alert was issued [12].

One way to compare two different collision avoidance logics is to compare their *expected cost* over the space of encounter scenarios. The expected cost is given by

$$\sum_{\omega} \Pr(\omega) c(\omega), \tag{2}$$

where $\Pr(\omega)$ is the probability of encounter $\omega$ when using the system. Prior TCAS studies involved developing encounter models from radar data to represent this probability mass function [1–3, 7, 9, 13–16], although they focused primarily on computing probability of conflict metrics.

A performance metric based on expected cost is certainly a sensible way to compare logics, but alternative metrics exist [17].

Before discussing how to find a logic that provides the lowest expected cost, it should be emphasized that, prior to this work, a multi-objective metric has not been directly used to evaluate TCAS. Various metrics related to successful alerts and unnecessary alerts were treated independently [18]. Even when using a multi-objective metric to compare logic, there is still value in analyzing performance relative to individual objectives (Section 5).

## 2.2 STATE UNCERTAINTY

Determining whether to alert and which RA to issue requires knowledge of the state of the world. Several attributes of the state are important in collision avoidance, such as own and intruder positions and velocities. Because TCAS can only make imperfect measurements due to the limitations of its sensors, it is never completely certain about the actual state of the world. This uncertainty can be modeled as a probability distribution over the space of possible states. In this report, $b(s)$ indicates the probability assigned to state $s$. In the literature, the mapping $b$ is called a *belief state* [19].

The current version of TCAS uses an altimeter to measure own altitude and beacon surveillance to measure the range, altitude, and bearing of intruders. This report refers to the aggregate of all of these sensor measurements at a particular time as an observation. Updating the belief state based on an observation requires knowledge of sensor performance—certainly, it would be difficult to make good decisions if there was no knowledge of how sensor measurements relate to the state of the world. In particular, it is useful to know as well as possible the probability distribution over observations given that the current state is $s$, which may be written $\Pr(\cdot \mid s)$. The model used by the system to represent this probability is called the *observation model* (or, sometimes, sensor or noise model). It is also useful to have a *dynamic model* (or, sometimes, state-transition or process model), which represents $\Pr(\cdot \mid s, a)$, a distribution over future states given that action $a$ was taken from state $s$. The dynamic model represents how the state evolves in response to the actions taken by the collision avoidance system.

Given the current belief state $b$, action $a$, and observation $o$, it follows from Bayes' rule that the updated belief state $b'$ is given by

$$\begin{aligned} b'(s') &= \Pr(s' \mid o, a, b) \\ &\propto \Pr(o \mid s', a, b) \Pr(s' \mid a, b) \\ &= \Pr(o \mid s') \sum_s \Pr(s' \mid s, a) b(s). \end{aligned} \tag{3}$$

The process of updating the belief state is known as belief updating, filtering, or state estimation, and there is a wealth of literature on the subject [20, 21]. A Kalman filter is one way to efficiently update the belief state for certain classes of dynamic and observation models [22–24]. The current version of TCAS uses an $\alpha$-$\beta$ tracker for altitude and an $\alpha$-$\beta$-$\gamma$ tracker for intruder range. These trackers do not directly take into account explicit dynamic and observation models, but they have

been tuned to work well for the TCAS operating environment. They also do not explicitly represent a distribution over states; they simply output a single state estimate that is used by the threat logic.

## 2.3 DYNAMIC UNCERTAINTY

The behavior of the aircraft involved in an encounter and the response of pilots to RAs is nondeterministic. If the current state is $s$ and action $a$ is taken by the collision avoidance system, the probability distribution over the next state (say one second later) is given by $\Pr(\cdot \mid s, a)$, which is specified by the dynamic model. Because the current state is not known exactly, as discussed in Section 2.2, the belief state $b$ must be used to predict the state at the next time step:

$$\Pr(s' \mid b, a) = \sum_s b(s) \Pr(s' \mid s, a). \tag{4}$$

The dynamic model may be repeatedly applied to predict the future state any number of steps into the future. Assuming that the state distribution is known $t$ steps into the future, the following formula may be used to estimate the state distribution $t+1$ steps into the future:

$$\Pr(s_{t+1} \mid b_0, a_0, \ldots, a_t) = \sum_{s_t} \Pr(s_{t+1} \mid s_t, a_t) \Pr(s_t \mid b_0, a_0, \ldots, a_{t-1}), \tag{5}$$

where $a_0, \ldots, a_t$ is the sequence of actions taken by the collision avoidance system. Equation 5 assumes that the dynamics are *Markovian*, meaning the distribution over states at the next time step depends only on the current state and action executed. So long as sufficient information is encoded in the state, this assumption is reasonable for a large class of dynamic systems [25].

As discussed by Yang, it is important that the dynamic model be as accurate as possible [26]. An inaccurate model can increase the collision or alert rate because it cannot adequately distinguish safe from hazardous flight trajectories. The current version of TCAS, as well as many alerting systems suggested in the literature [27–29], use deterministic (noiseless) dynamic models. These systems often use straight-line projection of aircraft trajectories to predict whether a conflict will occur. In order to compensate for the assumption that aircraft will not deviate from their nominal trajectories, these systems artificially enlarge their conflict boundaries. Although such systems can provide low probability of conflict, it is at the expense of a higher alert rate. Other systems use worst-case trajectory models that determine whether a conflict is possible [30–32]. Although this type of approach does not require artificially enlarging the conflict boundaries, it typically has an unnecessarily high false alert rate because it does not distinguish threats according to their likelihood. Kuchar and Yang survey a variety of different approaches that use probabilistic, deterministic, and worst-case state projection [33].

## 2.4 OPTIMAL DECISION MAKING

Figure 3 shows the relationship between the tracker (belief updating process) and the logic that decides upon the action to take at the current instant in time. The belief state $b$, as computed

by the tracker in the current version of TCAS, is simply a single point estimate that assigns all probability to a single state. In general, this need not be the case, and, in fact, it may be possible to make better decisions if the uncertainty in the underlying state is distributed more realistically.

Although this report discusses tracking issues to some extent, the primary focus of this report is on the logic. The logic is a function that takes as input a belief state $b$ and outputs an action $a$ to be executed. In the optimal control literature, this function is often called a *policy* or decision rule [25, 34, 35]. The objective is to use the dynamic and sensor models to find an optimal policy $\pi^*$ that provides the lowest expected cost according to some metric.

Assuming that the state is known exactly, the expected cost when following a policy (not necessarily the optimal one) satisfies the recursion

$$J^\pi(s) = \underbrace{c(s, \pi(s))}_{\text{current cost}} + \underbrace{\sum_{s'} \Pr(s' \mid s, \pi(s)) J^\pi(s')}_{\text{expected future cost}}, \tag{6}$$

where $c(s, \pi(s))$ is the immediate cost associated with being in state $s$ and executing the action specified by the policy $\pi$ for that state. If Equation 1 is used as the encounter cost function, the immediate cost is one if a conflict occurred for the first time, $\lambda$ if the system alerted for the first time, and zero otherwise. The function $J^\pi$ is called the *cost-to-go* function.

In order to calculate $J^\pi$, it is useful to define the mapping $B_\pi$:

$$B_\pi J(s) \equiv c(s, \pi(s)) + \sum_{s'} \Pr(s' \mid s, \pi(s)) J(s'). \tag{7}$$

If $J_0$ is the cost-to-go function that assigns zero to all states, then $B_\pi J_0$ is the cost-to-go function after one time step according to policy $\pi$. The cost-to-go function when following $\pi$ for $k$ decisions is given by $B_\pi^k J_0$. Repeated application of $B_\pi$ leads to convergence to $J^\pi$ [25].

The optimal cost-to-go function obtained by following the optimal policy satisfies the recursion

$$J^*(s) = \min_a \left[ c(s, a) + \sum_{s'} \Pr(s' \mid s, a) J^*(s') \right]. \tag{8}$$

The optimal cost-to-go function may be computed in a way similar to $J^\pi$ using the mapping $B$, known as the *Bellman update operator*,

$$BJ(s) \equiv \min_a \left[ c(s, a) + \sum_{s'} \Pr(s' \mid s, a) J(s') \right]. \tag{9}$$

Repeated application of $B$ to $J_0$ leads to convergence to $J^*$. This process is called *value iteration*. Once the optimal cost-to-go function is computed, the optimal policy $\pi^*$ may be extracted as follows:

$$\pi^*(s) = \arg\min_a \left[ c(s, a) + \sum_{s'} \Pr(s' \mid s, a) J^*(s') \right]. \tag{10}$$

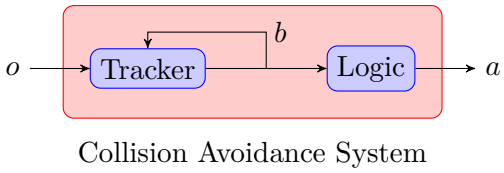



*Figure 3. Data flow between the tracker and logic.*

Generalizing Equation 8 to handle belief states instead of states is not difficult and is shown elsewhere [19]. However, solving for the optimal cost-to-go function, and hence the policy, is very difficult in general. A variety of approximation methods have been proposed in the literature [36,37] and have been applied to airborne collision avoidance [38]. These methods plan over a sampling of the space of belief states and appear to scale well to problems of small to moderate size.

The approach explored in this report avoids planning over the space of belief states by making the following approximation:

$$\pi^*(b) = \arg\min_a \sum_s b(s) \left[ c(s,a) + \sum_{s'} \Pr(s' \mid s,a) J^*(s') \right]. \tag{11}$$

Equation 11 is similar to Equation 10, except that the immediate cost and expected future cost are weighted by the distribution encoded by the belief state. This approximation is known as the *QMDP value method* [39, 40]. It amounts to assuming that all state uncertainty will vanish at the next step. This approximation appears to work well for many applications but tends to fail in problems where taking particular actions results in a significant reduction in state uncertainty. For the TCAS sensor—as well as GPS-based sensors such as ADS-B—the particular action taken by the collision avoidance system will have a negligible impact on state uncertainty. Nevertheless, it would be interesting to compare the performance of QMDP against policies found by algorithms that plan over the belief space, such as HSVI [36] and SARSOP [37].

## 2.5 DISCUSSION

This section has focused on logic optimization where the performance metric is a function of conflict and alert rates, but other objectives may also be taken into consideration. For example, Wolf explored using flight-plan deviation as a performance metric for unmanned aircraft collision avoidance [41]. This report does not focus on explicitly minimizing flight-plan deviation for two reasons. First, minimizing false alerts will result in fewer flight-plan deviations. Second, flight-plan information is not an input into the current version of TCAS.

The problem formulation presented in this section is known as a Markov decision process (MDP). Such problems have been well studied since the work by Bellman in the 1950s [42], and several books treat the subject in depth [34, 35, 43–45]. When the state is not known exactly, the formulation is called a partially observable Markov decision process (POMDP) [19]. MDPs and POMDPs have been applied to a variety of different problems, including robotic motion planning [46], agricultural management [47], medical diagnosis [48], and spoken dialog systems [49]. There

have also been several airborne collision avoidance applications for both manned and unmanned aircraft [12, 38, 41, 50–52].

Depending on the problem, it may be more natural to speak in terms of *reward* and *value functions*, which are the opposites of cost and cost-to-go functions, respectively. Many of the references provided in this report take the perspective of reward instead of cost. Although defining the problem of collision avoidance in terms of reward is entirely valid, this report uses cost to avoid having to work with negative rewards.

This report assumes that the dynamic and observation models are known. The dynamic model can be inferred from recorded operational data and the observation model can be based on sensor performance specifications. There is a body of work, however, on problems where the dynamic model is unknown and performance is optimized through interaction with the world. This active area of research is known as reinforcement learning [53, 54]. Some reinforcement learning approaches involve learning an explicit dynamic model, but others do not. Because reinforcement learning leads to changes in the behavior of the system over time in response to experience in the world, this kind of approach may not be appropriate for a safety-critical system that needs to be standardized and certified. However, some of the techniques can be applied to finding approximately optimal policies offline that can then be "frozen" and evaluated just like those found using dynamic programming.

The equations in this section assume that the state space (and hence the encounter space) is discrete, allowing the summation over probability masses as in Equation 2, repeated here:

$$\sum_{\omega} \Pr(\omega) c(\omega).$$

If the space is continuous, the summation would need to be changed to an integral and the probability mass would need to be changed to a density:

$$\int p(\omega) c(\omega)\, d\omega. \tag{12}$$

This report assumes discrete state and encounter spaces to simplify notation. A continuous space may be approximated arbitrarily well by a sufficiently fine discretization scheme. The next section shows how to discretize the state space and apply dynamic programming.

# 3. DYNAMIC PROGRAMMING APPROACH

The previous section described how to formulate the problem of collision avoidance using a decision-theoretic framework. This section shows how dynamic programming can be used to find the optimal logic. There are a variety of techniques for applying dynamic programming to a problem with continuous variables. This report focuses on a grid-based method that involves discretizing the state space, as will be discussed in Section 3.1. The number of discrete states grows exponentially with the dimensionality of the state space. Depending on the size of the discrete state space, it might not be practical to compute the optimal action for every possible state. Section 3.2 shows how the planning complexity can be reduced by eliminating states not reachable from the current state. Section 3.3 shows how to use branch-and-bound pruning to further eliminate states that are determined to be unreachable under the optimal policy. Section 3.4 shows plots of the optimal policy evaluated on different cross sections of the state space. Section 3.5 provides further discussion of some of the concepts introduced in this section.

## 3.1 FITTED VALUE ITERATION

Applying the value iteration algorithm (Section 2.4) is straightforward when the state space is finite. The initial cost-to-go function $J_0$ can be represented as an array in memory, where each element in the array corresponds to the cost-to-go for a particular state. The array is updated with each application of the Bellman operator until convergence.

If the state space is continuous, it is no longer possible to represent the cost-to-go function directly as a finite array. There are many strategies for representing the cost-to-go function, including decision trees [55], neural networks [44], and self-organizing maps [56]. This report focuses on a method called *fitted value iteration* that uses local averaging from a finite set of states $S = \{s_1, \ldots, s_n\}$ selected from the continuous state space. This method was shown by Gordon to provide a stable approximation of the optimal cost-to-go function [57, 58].

A variety of different sampling schemes can be used to choose $S$. One common method is to define $S$ as the vertices of a multidimensional grid spanning the state space. Table 3 shows the grid edges used for the five-dimensional hypothetical collision avoidance problem (Section 1.3). This discretization scheme results in $|S| = 2.14$ million states.

The cost-to-go function $J$ is represented as an array of $|S|$ elements, where the $i$th element corresponds to $J(s_i)$. To compute $J(\mathbf{x})$ at an arbitrary state $\mathbf{x}$, a variety of different interpolation schemes can be used. Appendix C discusses several interpolation methods that were evaluated on the hypothetical collision avoidance problem. One method that was found to work particularly well is *multilinear interpolation*, which computes $J(\mathbf{x})$ by taking the weighted average of the cost-to-go function at the vertices of the grid cell (hyper-rectangle) that encloses $\mathbf{x}$. The weight assigned to a particular vertex is related to its distance from $\mathbf{x}$.

Fitted value iteration begins by initializing all elements in the array representing $J$ to zero. The algorithm applies the Bellman update operator $B$ (Section 2.4) to $J$ at the states in $S$. The

**TABLE 3**

**Grid edges for the hypothetical collision avoidance problem.**

| Variable | Symbol | Grid Edges |
|---|---|---|
| Relative altitude | $h$ | $-1000, -900, \ldots, 1000$ |
| Time to closest horizontal approach | $\tau$ | $0, 1, \ldots, 20$ |
| Own vertical rate | $\dot{h}_1$ | $-2500, -2250, \ldots, 2500$ |
| Intruder vertical rate | $\dot{h}_2$ | $-2500, -2250, \ldots, 2500$ |
| RA state | $s_{\text{RA}}$ | clear of conflict, climb in 4 s, descend in 4 s, ... |

Bellman update operator generalized for a continuous state space is as follows:

$$BJ(s) \equiv \min_a \left[ c(s,a) + \int p(\mathbf{x}' \mid s,a) J(\mathbf{x}')\, d\mathbf{x}' \right]. \tag{13}$$

The integral on the right-hand side is the expectation of $J$ over states selected from $p(\cdot \mid s,a)$, which is the distribution over states at the next time step given that action $a$ is executed from the current state $s$. In general, an analytical solution to this integral does not exist, but it may be approximated using sampling methods as discussed in Appendix D. Sigma-point sampling, which relies on a small number of deterministically-chosen samples, works particularly well on the hypothetical problem as shown in Section 5.3 (Figure 21).

The update operator when sampling from the next state distribution (Appendix D) and interpolating the cost-to-go function (Appendix C) reduces to

$$BJ(s) \equiv \min_a \left[ c(s,a) + \sum_{s'} T(s' \mid s,a) J(s') \right], \tag{14}$$

where $T(s' \mid s,a)$ is a discrete transition probability function defined over states in $S$. Repeated application of this update operator results in the (approximately) optimal cost-go-go function $J^*$, which may be used to extract an (approximately) optimal policy $\pi^*$. The quality of the approximation depends on the level of discretization, interpolation method, and sampling scheme.

To determine the optimal policy from an arbitrary state $\mathbf{x}$, simply apply the continuous state-space generalization of Equation 10:

$$\pi^*(\mathbf{x}) = \arg\min_a \left[ c(\mathbf{x},a) + \int p(\mathbf{x}' \mid \mathbf{x},a) J^*(\mathbf{x}')\, d\mathbf{x}' \right]. \tag{15}$$

Again, the integral on the right-hand side can be evaluated using sigma-point sampling or some other sampling scheme.

### 3.2 REACHABLE STATE SPACE

The relatively coarse discretization used for the hypothetical collision avoidance problem (Table 3) results in $|S| = 2.14$ million states, which is manageable—value iteration can be done within seconds

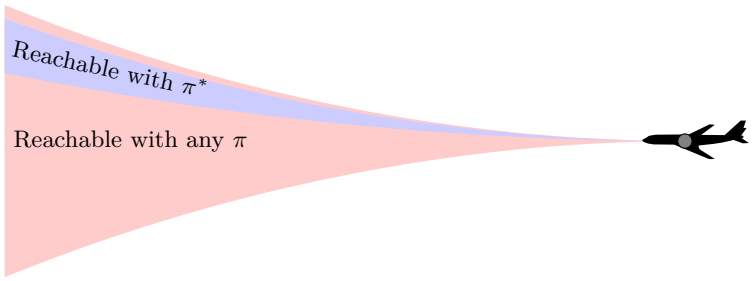



*Figure 4. Notional diagram showing reachable states under the optimal policy $\pi^*$ and reachable states under any policy $\pi$.*

using a standard desktop computer. However, increasing the dimensionality of the state space to include multiple intruders or other features such as intruder bearing can result in exponential growth of the number of states. One approach to reduce the number of discrete states used in value iteration is to only consider states reachable from the current state. The number of states reachable from a given state is typically a small fraction of the total state space.

The procedure for determining the optimal action from the current state $\mathbf{x}$ begins by incrementally constructing a set of discrete states. The set is initialized with the discrete states reachable after a single step following any action. Then, the discrete states that are reachable from the discrete states already in the set are added to the set. The process repeats recursively until there are no longer any new states to be added to the set. Once a complete set of reachable discrete states is obtained, value iteration can be used to find $J^*$ at these states and the optimal action from $\mathbf{x}$ can be determined using Equation 15.

Figure 5 shows the distribution over the number of reachable states from 10,000 randomly-sampled initial states. The mean number of reachable states is approximately 838,606 states, which is 39% of the entire state space. For more complex problems, the savings from only planning over the reachable states can be even more significant.

### 3.3 BRANCH AND BOUND

The number of reachable states under any policy is generally small compared to the full state space, but the number of reachable states under the optimal policy can be even smaller as illustrated in Figure 4. Although finding the set of states reachable under the optimal policy requires knowing the optimal policy, a branch-and-bound method can come close to restricting the planning effort to states reachable under the optimal policy. The branch-and-bound method computes the optimal policy with a $k$-step lookahead. If the time to horizontal closest approach is 20 seconds in the simple two-dimensional problem, then $k$ should be 20.

The branch-and-bound algorithm involves computing

$$J^*(s,a) = c(s,a) + \sum_{s'} T(s' \mid s,a) J^*(s'), \tag{16}$$

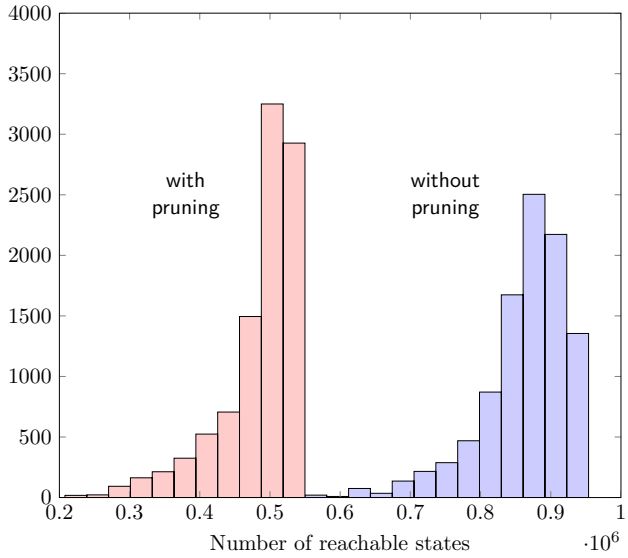



*Figure 5. Effect of branch-and-bound pruning.*

which is the cost-to-go from state $s$ when executing $a$ from the current state and then following the optimal policy from that point on. In order to prune suboptimal actions, the algorithm requires knowledge of a lower bound on $J^*(s, a)$. The lower bound does not necessarily have to be tight, although tighter bounds can lead to more pruning. In the hypothetical collision avoidance problem, a lower bound on $J^*(s, a)$ is $\underline{J}^*(s, a) = C(s) + \lambda A(a)$, where $C(s) = 1$ if $s$ is in conflict (zero otherwise) and $A(a) = 1$ if $a$ is an alert (zero otherwise).

Given some state $s$, the branch-and-bound algorithm recursively computes $\pi^*(s)$ and $J^*(s)$, where $\pi^*$ is the optimal policy and $J^*$ is the optimal cost-to-go function. If the lookahead is zero, then $\pi^*(s) = \arg\min_a c(s, a)$ and $J^*(s) = \min_a c(s, a)$. If the lookahead is $k$ and the available actions are $a_1, \ldots, a_n$ (sorted in ascending $\underline{J}^*(s, a_i)$), it first computes $J^*(s, a_1)$ using Equation 16. The $J^*(s')$ on the right-hand side of Equation 16 is recursively computed using the branch-and-bound algorithm with a lookahead of $k - 1$. If $J^*(s, a_1) \leq \underline{J}^*(s, a_2)$, then the algorithm will prune $a_2, \ldots, a_n$ from consideration since it knows that choosing any of those actions will not lower the expected cost. If, on the other hand, $J^*(s, a_1) > \underline{J}^*(s, a_2)$, then it is worth computing $J^*(s, a_2)$ recursively. The process continues until the remaining actions have been explored or pruned from consideration. The optimal action $\pi^*(s)$ is the action that provides the lowest value for $J^*(s, a)$.

Figure 5 shows the reduction in the number of reachable states due to branch-and-bound pruning. The mean number of reachable states was reduced to 475,279 states, which represents 22% of the state space. The savings can be even more significant when there are more actions (i.e., more kinds of alerts) from which to choose (Section 7.7).

### 3.4 POLICY PLOTS

To obtain an approximation to the optimal policy of Equation 15, the state space was discretized and fitted value iteration was performed. The integral of Equation 15 was approximated as a sum over the next state distribution, evaluated using sigma-point sampling and multilinear interpolation. The cost of alerting $\lambda$ was arbitrarily chosen to be 0.1.

Figure 6 shows the policy for different $h$-$\tau$ cross sections of the state space. The cross sections are defined by the tuple $(\dot{h}_1, \dot{h}_2, s_{\mathrm{RA}})$ and are supplied in the subfigure captions. Overlaying the plots are three trajectories starting from a variety of states. The trajectories correspond to the projected noiseless motion of the aircraft while executing each of the three actions. After the five-second pilot-response delay, the trajectory moves downward (to smaller values of $h$) while executing the Climb action and upward (to larger values of $h$) while executing the Descend action. The noiseless motion of the aircraft without alerting, called the nominal trajectory, follows a straight line. A trajectory is colored red if it terminates in conflict.

The policies largely agree with intuition. For instance, when both aircraft are flying level, as in Figure 6(a), the policy is symmetrical, indicating the best action for the own aircraft is to descend if the intruder is close enough above and to climb if the intruder is close enough below. This is because, although the aircraft are flying level, the noisiness in the aircraft vertical rates does have the potential of causing a conflict when the aircraft are near each other. Effective placement of the alerting boundaries, however, is not intuitive. Through the use of dynamic programming, the placement of the alerting boundaries was optimized to minimize the expected cost. Because the policy generation process does not rely on random sampling, the alerting boundaries are smooth and well defined.

The other plots, which represent the policy for slices in which the aircraft are either climbing or descending, are not symmetrical. In Figure 6(b), for example, because the own aircraft is flying at 1000 ft/min, issuing a descend RA has a greater effect on the own vertical rate than issuing a climb RA, as the simulated trajectories serve to show. Therefore, in cases when an alert is necessary, the policy favors descending. However, in some cases, it is still necessary to climb. For example, in some regions where the intruder is close enough above the own aircraft, the policy suggests climbing. This is similar to an altitude-crossing RA. As expected, the policy rarely advises a maneuver when the own aircraft is above the intruder.

A striking feature of the policies is the conspicuous absence of any alert when there is little time until horizontal closest approach and the aircraft are closely separated in altitude. Intuition would seem to suggest that this is a crucial time at which an alerting system should alert to reduce the threat of conflict. However, because the hypothetical collision avoidance problem assumes that the pilot takes five seconds to respond to the RA, the aircraft are very likely to come into conflict with or without an RA. In this case, there is no incentive to alert because the expected cost without alerting is actually lower than the expected cost with alerting. As the pilot-response delay is decreased, the alert boundary moves to the left. If the pilot-response delay is made probabilistic with some probability of immediate response, the alerting region may extend fully to the left (Section 7.2).

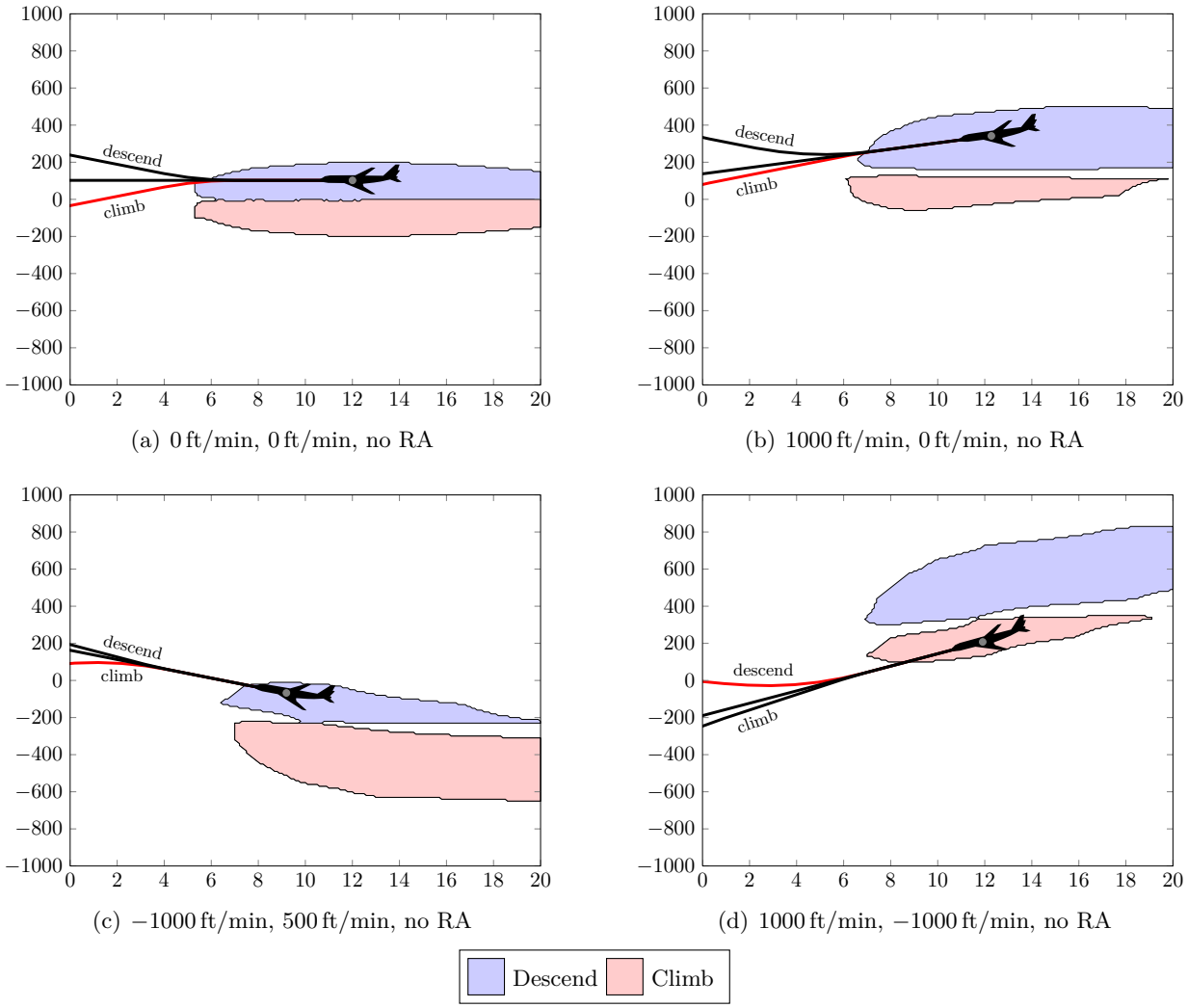



*Figure 6. Collision avoidance policies generated using dynamic programming. The caption of each subfigure indicates the cross section $(\dot{h}_1, \dot{h}_2, s_{RA})$ for which the policy is evaluated. The horizontal axis represents $\tau$ in seconds, and the vertical axis represents $h$ in feet.*

In each of the plots, although the nominal trajectory does not result in conflict, the policy recommends executing an action. Due to the noise in the system, the relative path of the intruder will deviate, however slightly, from the projected straight-line path. The introduction of this noise causes the expected probability of conflict in the future to be non-zero. The expected cost while following the RA trajectory, in turn, is less than the expected cost while following the nominal trajectory.

### 3.5 DISCUSSION

This section discussed how to compute the optimal policy *offline* (Section 3.1) and how to compute the optimal action from the current state *online* (Sections 3.2 and 3.3). Offline solution methods involve computing a representation of the optimal policy from all possible states. The policy representation is then used online with minimal computational effort. Online solution methods involve computing the best action from the current state during execution. In general, online solution methods require much more computational effort while making decisions and can only plan for contingencies within a finite time horizon.

There are many other online solution methods besides the ones discussed here [59], including methods like LAO* that use heuristics to reduce the planning space [60]. It is important to note that online solution methods can be used offline to determine the optimal action from a sampling of states. The optimal policy can be approximated by a classifier trained on the sampled states [61,62]. The classifier can be compactly represented, for example, as a support vector machine, neural network, decision tree, or human-readable logic. For TCAS, an online solution method can be used to aid human designers in tuning the parameters of existing logic to match the optimal policy at a selection of states.

Many offline methods, such as fitted value iteration, involve modeling the cost-to-go function over the entire state space using local approximation. This section discussed one way to model the cost-to-go function by interpolating values at the vertices of a multidimensional grid spanning the state space. The experiments discussed in this report use grids with regularly-spaced edges, but for some problems it may be desirable to increase the resolution in some regions of the state space where it is needed to better approximate the cost-to-go function, while leaving the resolution coarse in other regions to reduce memory and computational requirements. Munos and Moore show how to dynamically infer a suitable multiresolution representation [63].

An alternative to local approximation of the cost-to-go function is global approximation using a parametric representation, such as a neural network. Such an approach has been studied extensively in the reinforcement learning community and has been successfully applied to a variety of problems [44]. If development of the grid-based local approximation method pursued in this report is found to have difficulty scaling to higher dimensions, a global parametric representation of the cost-to-go function may be a way forward.

Approximate dynamic programming is an active area of research and there have been many important advances in recent years allowing them to scale to increasingly more complex problems [64]. This report explores only a handful of methods that were relatively simple to implement and

appeared to have the most promise. Increasing the complexity of the model to include multiple intruders, for example, may require adopting or extending other solution methods.

# 4. CONFLICT PROBABILITY ESTIMATION APPROACH

Although dynamic programming is a powerful optimization method, it can be difficult to scale to problems with many state variables. An alternative method that may scale more gracefully involves estimating the probability of conflict from the current state when issuing different alerts. This differs from TCAS, which uses temporal and spatial criteria to issue alerts (Section 1.2). A conflict probability estimation approach has been applied to a prototype alerting system for free flight [65] and an alerting logic for closely spaced parallel approaches [66]. In the late 1980s, researchers at Lincoln Laboratory investigated a conflict probability estimation approach for TCAS III [67].

The ability to accurately estimate the probability of conflict upon which these alerting systems are predicated is crucial if they are to properly detect conflicts. For example, a system that overestimates the conflict probability is likely to have a high false alarm rate, while one that underestimates the conflict probability is likely to have a high rate of missed or late detection [26]. Much research, therefore, has focused on conflict probability estimation based on analytic [68, 69], numerical approximation [70, 71], and Monte Carlo methods [72].

Conflict probability estimates based on analytic or numerical methods require strong assumptions about the form of the dynamics, typically that the dynamics are adequately described by linear-Gaussian equations. Monte Carlo simulation allows much greater flexibility in modeling, although the computational demands can become prohibitively high. Estimating the probability of a rare event such as aircraft conflict with a high level of confidence requires a large number of samples. A variance-reduction technique known as importance sampling can deliver an estimator that uses significantly fewer samples than direct sampling while providing the same accuracy. Dynamic programming can also be used to estimate conflict probability, but such an approach requires discretization of the state space.

This section discusses and compares the performance of conflict probability estimation methods based on analytic approximation, dynamic programming, and Monte Carlo sampling. There are conceivably many different alerting schemes that can be constructed based on the probability of conflict. This section discusses three of them and presents plots of the resulting policies for slices through the state space.

## 4.1 ANALYTICAL APPROXIMATION

The dynamics of the aircraft in the hypothetical collision avoidance problem can be approximated by a linear-Gaussian system (Appendix B) that has two discrete modes: no-RA execution mode and RA execution mode. By definition, a switch to RA execution mode occurs when own aircraft, at any time, begins to apply a 0.25-g acceleration to reach the RA target vertical rate. During RA execution, the motion of own aircraft becomes deterministic. The motion of the intruder remains probabilistic.

Starting from an arbitrary initial state, the distribution representing the state uncertainty can be propagated forward in time using the equations in Appendix B until $\tau = 0$. The probability

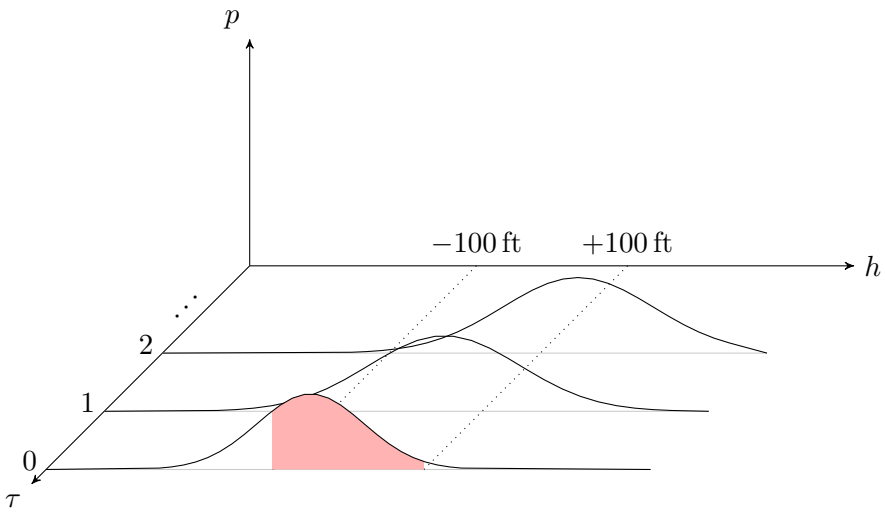



*Figure 7. Evolution of the Gaussian distribution of h until CPA.*

of conflict can be estimated by integrating the Gaussian distribution over $h$ at $\tau = 0$ from $-100\,\text{ft}$ to $+100\,\text{ft}$. Figure 7 illustrates the evolution of the distribution of $h$ until the closest point of approach, where the area under the curve is the estimate of the probability of conflict.

Another method of propagating the Gaussian distribution of the aircraft state relies on sigma-point sampling (Appendix D) in which, at each time step, the distribution is approximated by deterministically chosen sample points that are propagated through the true nonlinear system dynamics. The resulting points approximate the distribution at the next time step.

### 4.2 DYNAMIC PROGRAMMING

Fitted value iteration (Section 3) can be used to estimate the probability of conflict for not alerting, issuing a Climb RA, and issuing a Descend RA. The state space can be discretized using the same scheme used earlier (Table 3). The cost function $c(s)$ is one if $s$ is a conflict state ($\tau = 0$ and $|h| < 100\,\text{ft}$) and zero otherwise. The Bellman update operator for computing the probability of conflict for action $a$ at a discrete state $s$ is

$$BJ(s,a) \equiv c(s) + \int p(\mathbf{x}' \mid s,a) J(\mathbf{x}')\, d\mathbf{x}', \tag{17}$$

which, as suggested in Section 3, can be approximated using sigma-point sampling and mulitilinear interpolation. The repeated application of the Bellman update operator leads to an estimate of the cost-to-go function that is equivalent to the probability of conflict from $s$ following $a$.

### 4.3 MONTE CARLO

An alternative method for estimating the probability of conflict relies on Monte Carlo sampling. For example, a number of trajectories can be sampled from the probabilistic dynamic model (Ap-

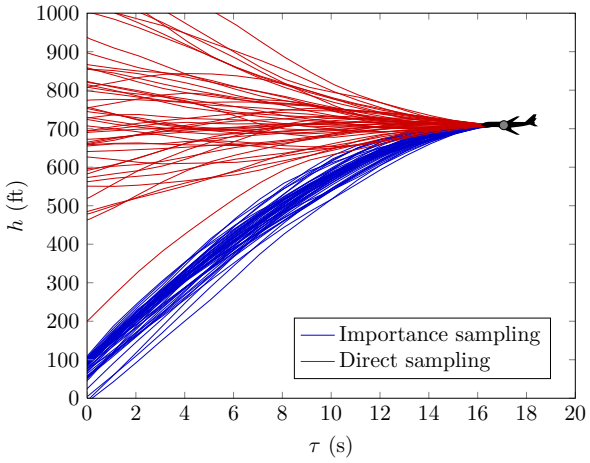



*Figure 8. Comparison of state propagation using direct sampling versus importance sampling.*

pendix A). The probability of conflict is the fraction of trajectories that result in conflict. This is known as the direct Monte Carlo method, or *direct sampling*. The accuracy of the estimate increases with the number of sample trajectories.

However, because the event of a conflict is typically rare, performing direct Monte Carlo sampling in this fashion is inefficient. Sampling the trajectories from a different distribution, called the *proposal distribution*, such that sample trajectories result in conflict with high probability would lead to increased efficiency in estimating the probability of conflict. This is known as *importance sampling*. To obtain an unbiased estimate of the probability of conflict, the sample trajectories must be weighted according to the likelihood that they would have been produced by the original distribution. Figure 8 highlights the difference between direct sampling and importance sampling. The fact that few trajectories terminate in conflict using the direct sampling method makes it a poor estimator of conflict probability [73]. Nearly all the trajectories produced by importance sampling, however, result in conflict and therefore contribute to the conflict probability estimate.

The following sections outline several proposal distributions that were explored. Appendix E provides a more detailed, mathematical discussion of the proposal distributions.

### 4.3.1 Constant Acceleration Proposal

The aircraft in the model experience random vertical accelerations in the form of zero-mean Gaussian noise. The history of vertical accelerations, called the *control trajectory*, uniquely specifies a state trajectory. Typically, these accelerations do not produce trajectories that result in conflict, except when the start state is already near conflict. However, the zero-mean Gaussian distribution from which the accelerations are sampled can be adjusted to artificially induce conflict trajectories. At each time step along the state trajectory, one may calculate the accelerations that, if applied constantly until CPA, would force the aircraft into conflict but that disturb the flight path of the aircraft as little as possible. These constant accelerations can serve as the mean of the Gaussian

distribution from which to sample the vertical accelerations at each time step. If this process continues until CPA, it is likely that the resulting state trajectory will result in conflict.

### 4.3.2 Maximum-Likelihood Acceleration Proposal

A good proposal distribution should produce samples that result in conflict while still resembling, as much as possible, the distribution of the model [73]. Because the accelerations in the model are zero-mean, the mean of the proposal distribution should be as close to zero as possible while still producing trajectories that terminate in conflict. Finding the sequence of accelerations can be framed as an optimization problem where the objective is to minimize the square norm of the accelerations subject to the constraint that a conflict occurs at $\tau = 0$. After solving for the acceleration sequence, the first set of accelerations is used as the mean of the proposal distribution and the remaining accelerations are discarded. (In this way, it is similar to model predictive control used in online trajectory planning [74].) As before, this process continues for the remainder of the encounter.

### 4.3.3 Analytic Proposal

It can be shown that the optimal proposal distribution from which to sample the accelerations at a particular time is proportional to the product of the probability of conflict at the next time step with the distribution of the accelerations according to the model. Of course, it is not possible to actually calculate the optimal proposal distribution because it requires that the probability of conflict be known. However, by using an estimate of the probability of conflict in place of its true value, the optimal proposal distribution can be approximated. An effective proposal distribution, though still suboptimal, is a Gaussian distribution whose mean is approximately equal to the mean of the optimal proposal distribution, which can be obtained by using the probability of conflict estimate afforded by the analytic approximation described earlier.

### 4.3.4 Dynamic Programming Proposal

The dynamic programming proposal distribution is identical to that of the analytic proposal distribution except that the dynamic programming estimate of the probability of conflict is used to determine the mean of the distribution.

## 4.4 ESTIMATION COMPARISON

This section presents results demonstrating the performance of the various estimation methods described above. The following abbreviations refer to the estimation methods: Analytic, DP, Direct, IS-Constant, IS-ML, IS-Analytic, and IS-DP. The performance of an estimator is related to how quickly the estimator converges to a stationary estimate. The *standard error* (SE) of the estimator is $\sigma/\sqrt{N}$, where $\sigma$ is the sample standard deviation and $N$ is the number of samples.

Figures 9 and 10 show convergence plots for $\widehat{\Pr}(\mathrm{C} \mid \mathbf{x}, a)$ and $\mathrm{SE}[\widehat{\Pr}(\mathrm{C} \mid \mathbf{x}, a)]$, estimates of the probability of conflict and its standard error, from three different states while executing the no

alert and descend actions, respectively. The plots on the left show $\widehat{\Pr}(\mathrm{C} \mid \mathbf{x}, a)$, while the plots on the right show $\mathrm{SE}[\widehat{\Pr}(\mathrm{C} \mid \mathbf{x}, a)]$. The tuples on the left panel indicate the states from which the probability of conflict is estimated. All plots show convergence for up to 1000 sample trajectories except for the last two plots of both figures for which more trajectories are needed because the probability of conflict is low. In all cases, importance sampling converges fastest, and the analytic proposal distribution consistently provides the lowest standard error. When the probability of conflict is low, on the order of $10^{-3}$ for the last two plots, estimating the probability using direct sampling is inefficient, requiring many more samples than the importance sampling estimators to converge to a reasonable estimate. Observe that in some of the plots, certain estimates do not appear because they are far removed from the other estimates. Moreover, the standard error for the Analytic and DP estimates is zero, but these are not included in the standard error plots.

Note particularly that for the no alert action, the analytic approximation is quite accurate as all estimators tend to approach its estimate as the number of sample trajectories increases. The analytic approximation for the other actions, however, performs poorly due to its linear-Gaussian approximation. However, using importance sampling with a proposal distribution derived from the analytic approximation converges to values comparable to those of the other estimators with low standard error. This result suggests that rough approximations of conflict probability can still be exploited to generate proposal distributions that achieve lower variance than their direct counterpart.

The accuracy of the dynamic programming estimates can be quite poor compared to the other methods because of the coarseness of the discretization. Several experiments have shown that increasing the granularity of the discretization improves the accuracy of its conflict probability estimates at the expense of requiring many more states.

### 4.5 POLICY GENERATION

Several different strategies have been suggested for using conflict probability estimates to decide when to alert. This report focuses on the following three strategies for constructing policies:

1. **Conservative policy:** Carpenter and Kuchar [66], in the development of logic for closely spaced parallel approach, suggested alerting when the probability of conflict without alerting exceeds a set threshold. The alert that minimizes the probability of conflict is chosen. If the probability of conflict is minimal without alerting, no alert is issued. This policy is called conservative because it alerts no later (and typically much earlier) than the other policies. (Figure 11(a))

2. **Delay policy:** In the late 1980s, Lincoln Laboratory researchers developing TCAS III proposed waiting to alert until the conflict probabilities for all actions reach or exceed the threshold [67]. This strategy is referred to as the delay policy because it tries to delay alerting as much as possible. (Figure 11(b))

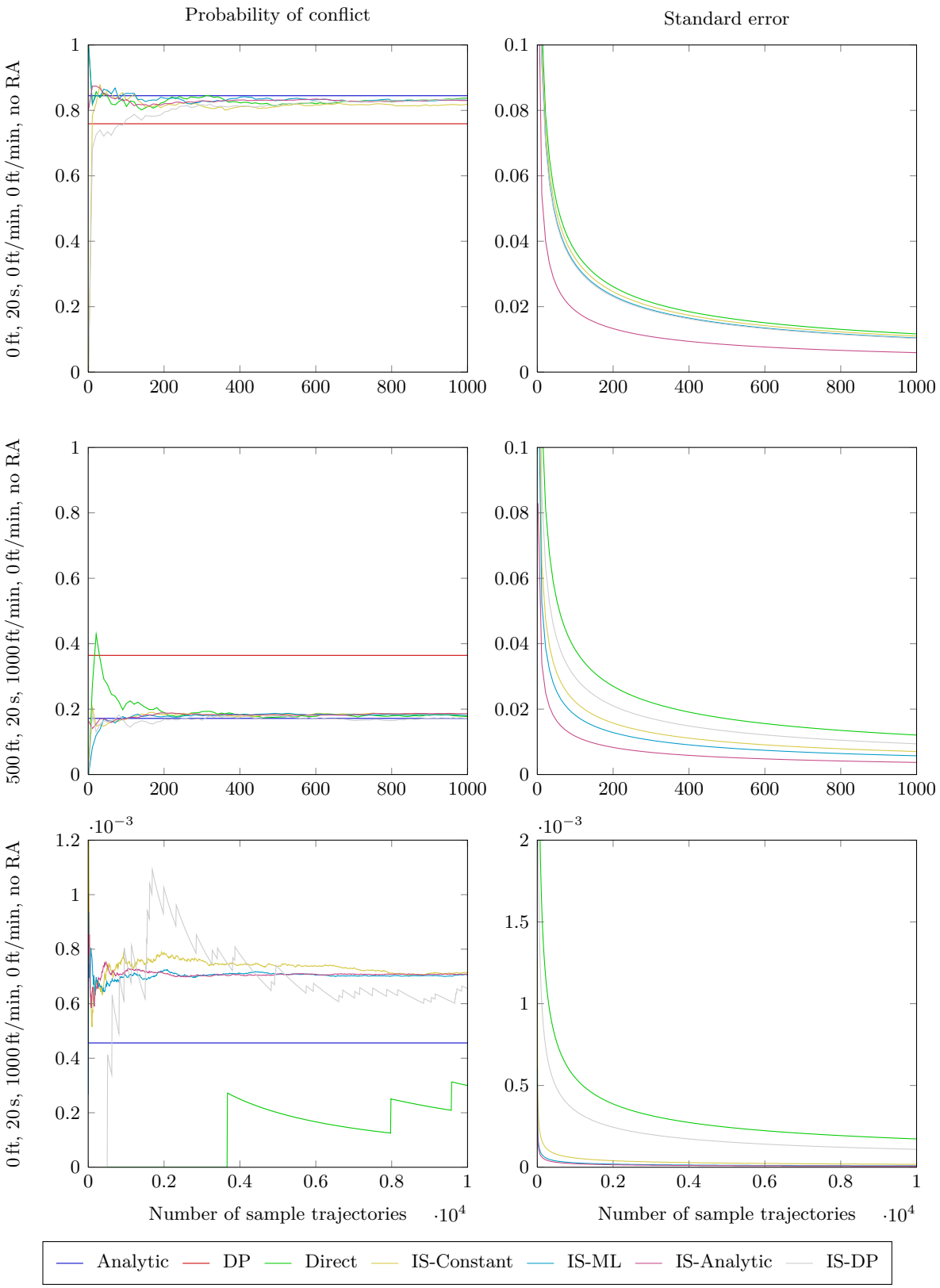



*Figure 9. Convergence plots from several different states for the* no alert *action.*

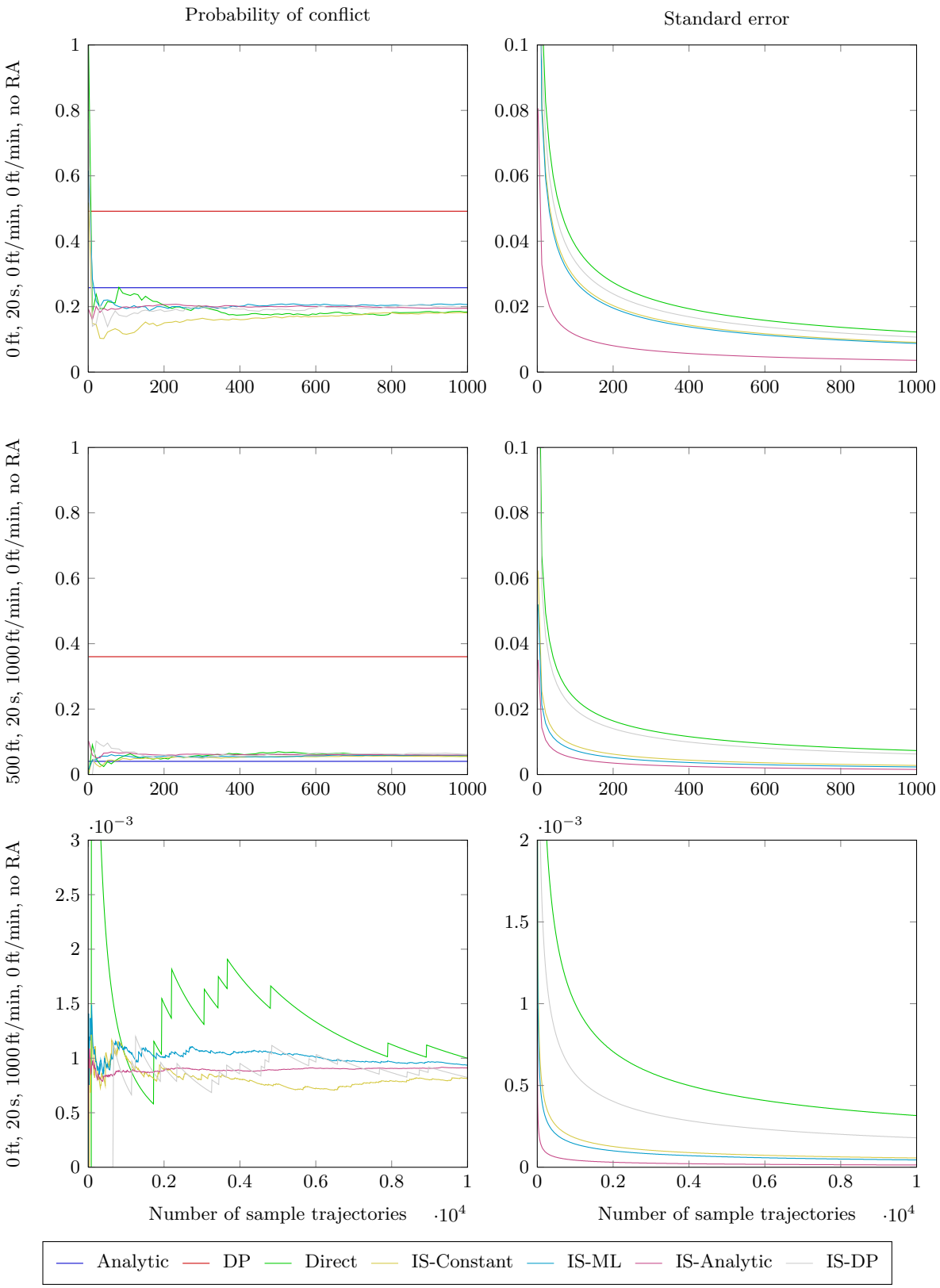



*Figure 10. Convergence plots from different states for the* descend *action.*

3. **Conservative delay policy:** This strategy waits until there is a unique alert that has a probability of conflict below the threshold. It typically alerts after the conservative policy but before the delay policy. (Figure 11(c))

The longer alerts are delayed, the less likely the alert will be unnecessary but the more likely it will come too late to prevent conflict. Figure 12 illustrates the difference in alert timing for the three strategies when the alert threshold is set to 0.1. The conservative strategy alerts as soon as the conflict probability when not alerting reaches the threshold. It issues a climb advisory because it provides the lowest probability of conflict at that point in time. The conservative delay strategy waits until there is a unique advisory (in this case, climb) that provides a conflict probability less than the threshold. The delay strategy waits until the moment all alerts meet or exceed the threshold before it issues a climb advisory.

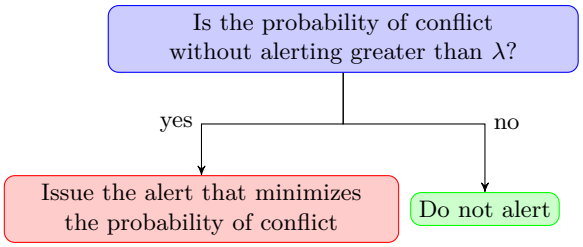
Yang and Kuchar developed a multistaged threshold alerting system for free flight that does not fit into one of the three categories above [65]. In their prototype system, low-probability threats produced passive alerts (e.g., changing the color of a traffic symbol) while high-probability threats produced aural zone transgression messages to indicate to the pilot that an evasive maneuver should be performed to resolve the conflict. There are four alert stages in total. During the first three stages, alerts are issued to aid the pilot in resolving the conflict before tactical maneuvering is required. During the fourth stage, air traffic control intervenes by issuing commands (e.g., heading, vertical rate, or speed changes) intended to increase the amount of separation. The alerting logic delays the issuance of an alert if a sufficient number of maneuvers is still available to the pilot in order to resolve a conflict. They defined a maneuver as available if the probability of conflict would be reduced to less than 0.05 by performing the maneuver.

## 4.6 POLICY PLOTS

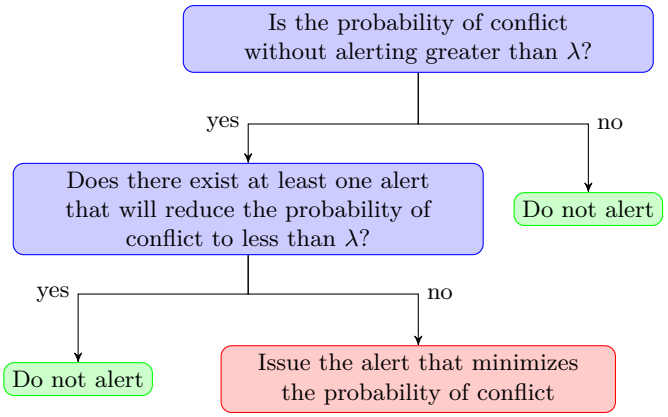
This section presents visual representations of the policies of Section 4.5. The policy plots are similar to those presented in Section 3.4. All plots show the policy for $h$-$\tau$ cross sections of the state space. Figures 13, 14, and 15 show the policies generated by the three alerting strategies for four different cross sections defined by the tuple $(\dot{h}_1, \dot{h}_2, s_{\mathrm{RA}})$, as supplied in the subfigure captions. The probability of conflict was estimated using importance sampling with the maximum-likelihood proposal distribution (Appendix E.2) with 100 sample trajectories. The threshold on the probability of conflict was arbitrarily set to 0.01.

The policies largely agree with intuition. For instance, when both aircraft are level, as in subfigures (a), the best action for the own aircraft is to descend if the intruder is close enough above and to climb if the intruder is close enough below. This is because, although the aircraft are flying level, the noisiness in the aircraft vertical rates does have the potential of causing a conflict when the aircraft are near each other.

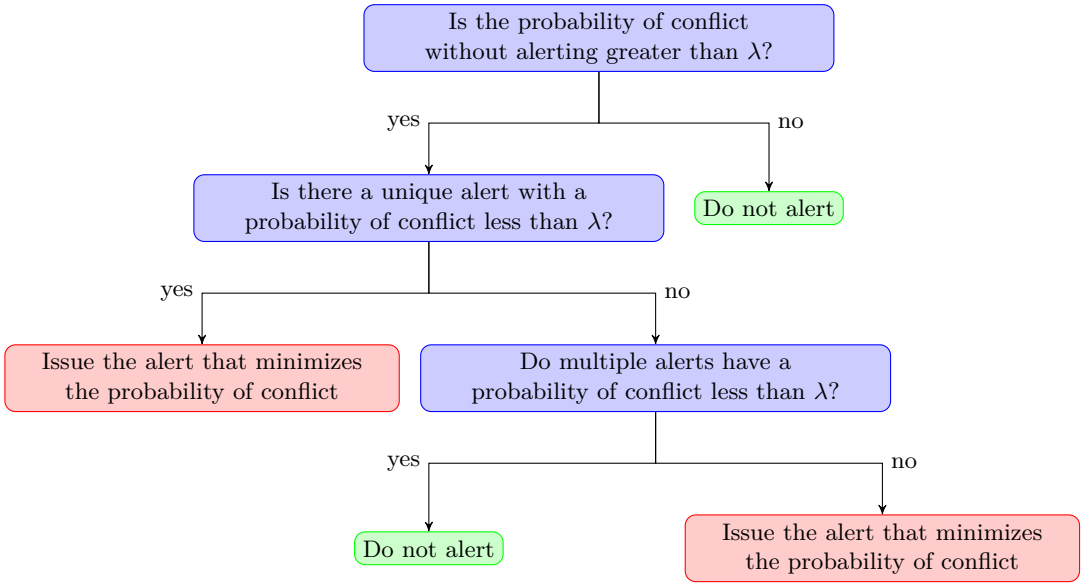
When the own aircraft is climbing at 1000 ft/min and the intruder is level, as in subfigures (b), the best action is to descend when the intruder is above even when the vertical separation is large. When the intruder is above and very close in altitude, i.e., less than 100 ft, the policy recommends climbing. This is similar to an altitude-crossing RA in the current TCAS logic (Section 1.2).




(a) Conservative policy.




(b) Delay policy.




(c) Conservative delay policy.

*Figure 11. Decision trees for alerting policies.*

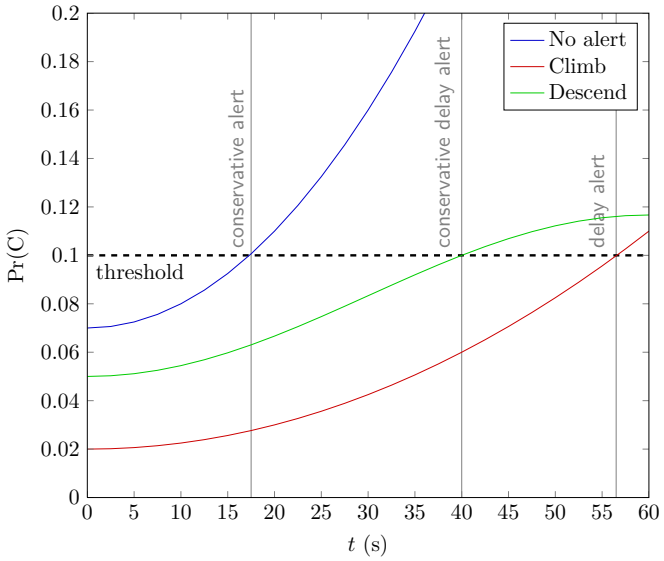



*Figure 12. A notional diagram illustrating the difference between the three alerting strategies. In this diagram, the alerting threshold is set to 0.1. The probability of conflict for the various actions is plotted. The times at which the different alerting strategies issue a climb advisory are indicated.*

This might appear counter-intuitive. However, this is justified because the own aircraft is already climbing. If the descend maneuver were issued, much of the RA execution would consist of reversing direction, thus resulting in a higher probability of conflict. An analogous argument holds for subfigures (c) and (d).

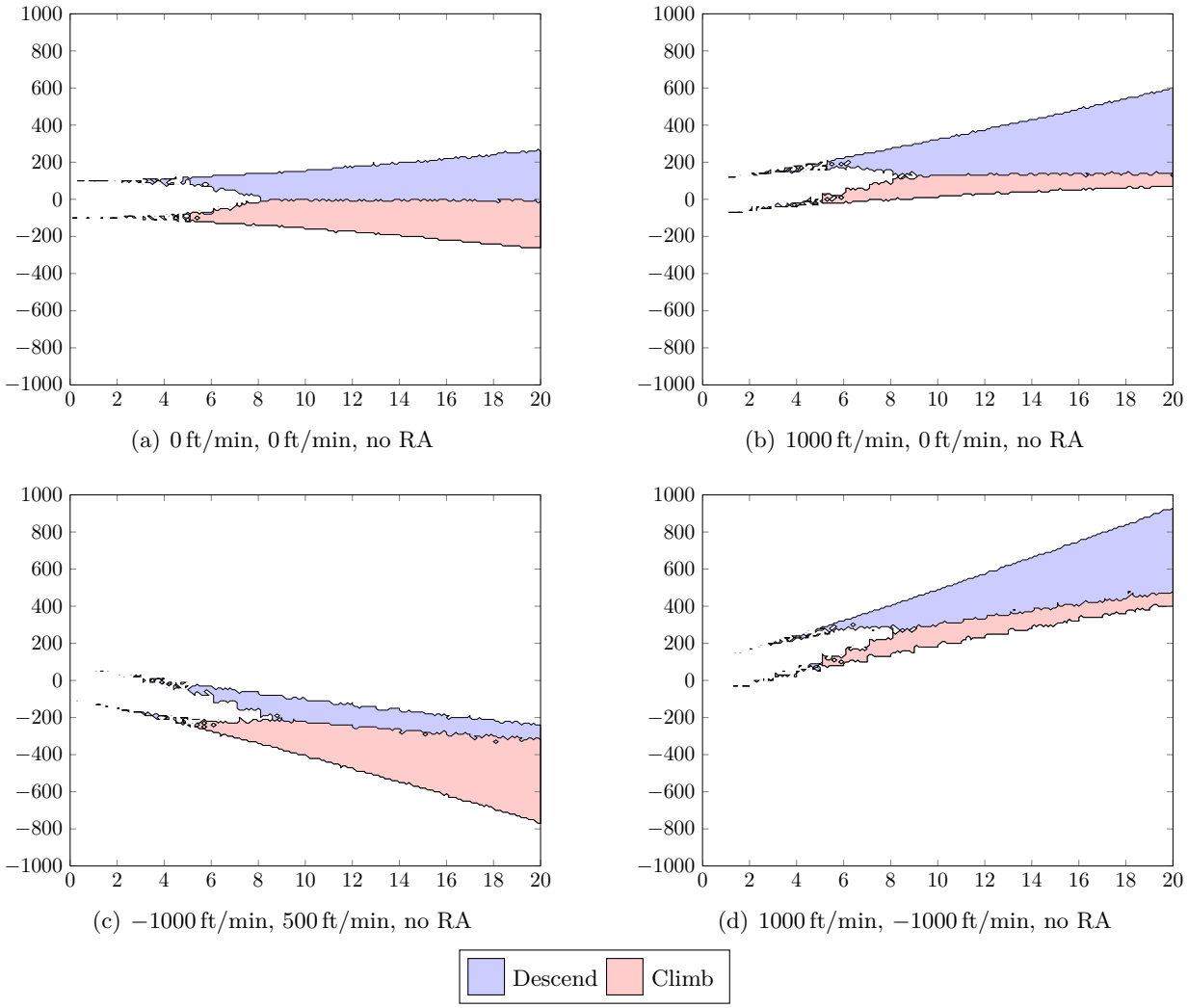
Other observations about the policies are also worthy of mention:

- When $\tau$ is small, there is usually no incentive in alerting because the five-second pilot delay makes any evasive maneuvers useless.

- The policy of Figure 14 rarely alerts due to the fact that, as Figure 11(b) indicates, there is almost always one action for which the probability of conflict falls below the threshold.

- The conservative policy and conservative delay policy are very similar except that the latter alerts less when $\tau$ is large and the vertical separation is relatively small. This is because in such circumstances there are multiple actions, i.e., both the climb and descend maneuvers, for which the probability of conflict falls below the threshold.

- The conservative policy is likely to have a high false alarm rate and low conflict rate compared to the delay policy. The conservative delay policy is likely to provide a balance between the two.



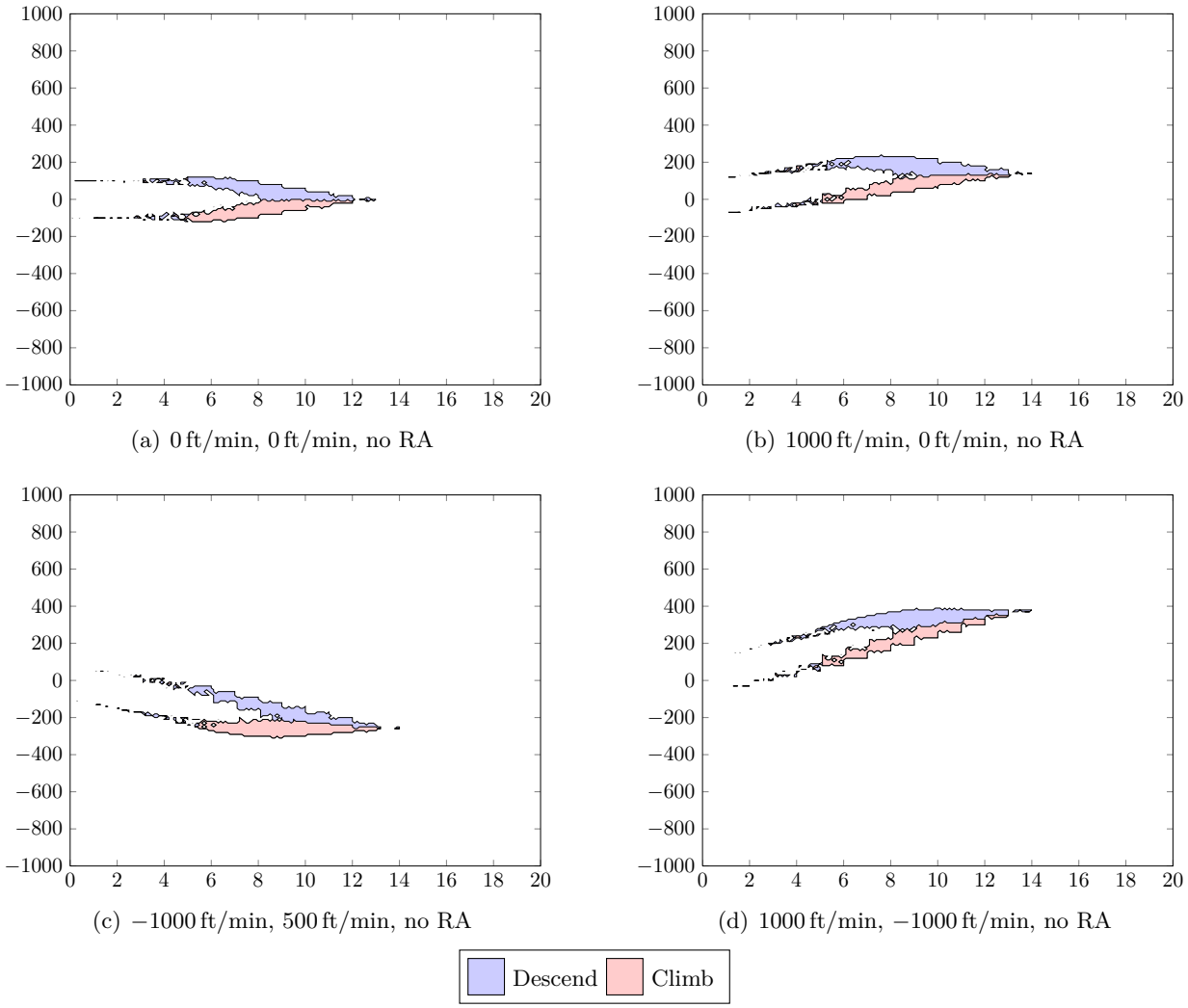

(a) 0 ft/min, 0 ft/min, no RA

(b) 1000 ft/min, 0 ft/min, no RA

(c) −1000 ft/min, 500 ft/min, no RA

(d) 1000 ft/min, −1000 ft/min, no RA

*Figure 13. Collision avoidance policy generated by the conservative strategy. The caption of each subfigure indicates the cross section $(\dot{h}_1, \dot{h}_2, s_{RA})$ for which the policy is evaluated. The horizontal axis represents $\tau$ in seconds, and the vertical axis represents $h$ in feet.*



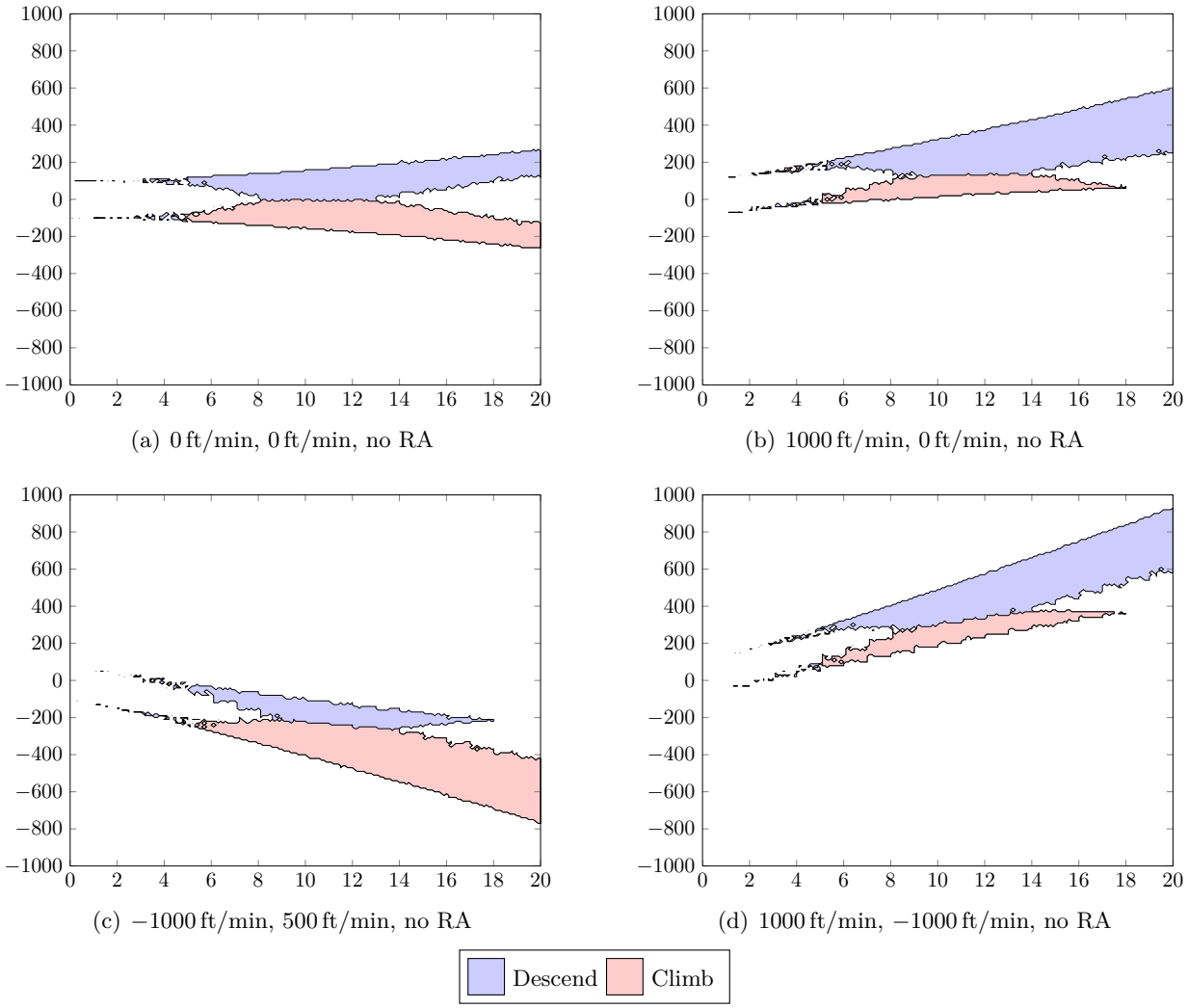

(a) 0 ft/min, 0 ft/min, no RA

(b) 1000 ft/min, 0 ft/min, no RA

(c) −1000 ft/min, 500 ft/min, no RA

(d) 1000 ft/min, −1000 ft/min, no RA

*Figure 14. Collision avoidance policy generated by the delay strategy. The caption of each subfigure indicates the cross section $(\dot{h}_1, \dot{h}_2, s_{RA})$ for which the policy is evaluated. The horizontal axis represents $\tau$ in seconds, and the vertical axis represents $h$ in feet.*




(a) 0 ft/min, 0 ft/min, no RA

(b) 1000 ft/min, 0 ft/min, no RA

(c) −1000 ft/min, 500 ft/min, no RA

(d) 1000 ft/min, −1000 ft/min, no RA

*Figure 15. Collision avoidance policy generated by the conservative delay strategy. The caption of each subfigure indicates the cross section $(\dot{h}_1, \dot{h}_2, s_{RA})$ for which the policy is evaluated. The horizontal axis represents $\tau$ in seconds, and the vertical axis represents $h$ in feet.*

## 4.7 DISCUSSION

This section compared analytic, dynamic programming, and Monte Carlo methods for estimating the probability of conflict in the hypothetical collision avoidance problem. Although the encounter model is relatively simple, the analytic method had difficulty providing accurate conflict probability estimates because the dynamics are not exactly linear-Gaussian. Approximating the behavior of more complex encounter scenarios with linear-Gaussian dynamics and analytically solving for the conflict probability will likely result in poor estimates. The accuracy of the estimates provided by dynamic programming was limited due to the coarseness of the state space discretization. Although the dynamic programming estimates could be improved by increasing the level of discretization, this can require an infeasible number of states, especially if the dimensionality of the state space is large.

Of the methods discussed in this section, conflict probability estimation using Monte Carlo sampling seems to be the most promising. The quality of Monte Carlo estimates can be significantly improved if a suitable proposal distribution is used and the samples are appropriately weighted. Several different proposal distributions were considered, including proposal distributions that leverage information from analytic approximations and dynamic programming solutions. Although the analytic or dynamic programming estimates may in themselves be inaccurate, it was shown that the heuristic information they provide can be used to improve the efficiency of Monte Carlo estimates through importance sampling.

Incorporating higher-fidelity encounter dynamics in three spatial dimensions would likely degenerate the accuracy of analytic methods even further. Moreover, discretizing the state space sufficiently fine for accurate dynamic programming estimates may be challenging due to the higher dimensionality of the models. In three-dimensional space, conflicts are much rarer because aircraft can miss each other laterally in addition to vertically. Consequently, the advantage of using importance sampling over direct sampling will be even more significant.

The performance of the policies of Section 4.5 is dependent on a number of factors. Using a more accurate method for calculating the probability of conflict or increasing the number of sample trajectories generally enhances performance. In the generation of the policies of Figures 13, 14, and 15, the threshold was arbitrarily chosen. Effective threshold placement can be aided through the use of system operating characteristic curves to be discussed in Section 5.

Although policies based on conflict probability estimates may perform well, they will not be optimal in general. A simple example of this is shown in Figure 16. The current state is represented by the root node. From this state, the system may either alert or not alert (for simplicity in this example, climb and descend are not distinguished). If an alert is issued, then the probability of conflict is some $\gamma$ between zero and one. If an alert is not issued from the current state, the system is given another opportunity to alert at the next state. From this next state, if an alert is issued, a conflict is guaranteed not to occur, but if an alert is not issued a conflict is guaranteed to occur. Dynamic programming, in this situation, will advise waiting to alert until the next state and will prevent conflict. Any alerting strategy based solely on current conflict probability estimates will alert because the conflict probability from the current state is one.

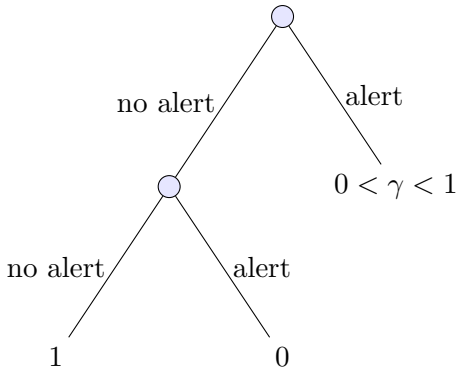



*Figure 16. An example of a situation where dynamic programming dominates a conflict probability approach regardless of cost function or alerting threshold.*

The example in Figure 16 was constructed to serve as a very simple situation where a dynamic programming policy dominates a threshold policy, regardless of cost function or alerting threshold. In realistic encounters, dynamic programming will provide better policies than a threshold approach, assuming that the discretization used for dynamic programming is sufficiently fine. In practice, it may be difficult to provide the level of discretization needed by dynamic programming to provide a close to optimal policy. Further experimentation will reveal how well threshold alerting strategies approximate the optimal policy in practice.

This page intentionally left blank.

# 5. EVALUATION

Due to the safety-critical nature of collision avoidance systems, extensive research over the years has focused on developing a methodology for evaluating such systems. This section enumerates a set of mutually exclusive and collectively exhaustive alerting system outcomes and discusses performance metrics used in prior safety assessments of TCAS and other alerting systems. It describes how system operating characteristic curves can provide a visual representation of performance tradeoffs when varying design parameters. Finally, this section presents the results from a preliminary evaluation of the dynamic programming (DP) logic using a safety assessment tool used for evaluating previous versions of the TCAS logic.

## 5.1 ALERTING SYSTEM OUTCOMES

The outcomes of an alerting system scenario can be divided into six different categories depending on three criteria: (1) whether an alert was necessary, (2) whether an alert was issued, and (3) whether a conflict occurred. Before enumerating these categories, it is important to first define what is meant by a necessary alert. Determining whether an alert is necessary in a particular situation is not exactly straightforward because the dynamics are nondeterministic. The standard approach is to run simulations both with and without the alerting system with the same random seed. Because the seeds are the same, the trajectories with and without the system will be exactly the same until after an alert is issued. Once the pilot responds to the alert, the trajectory with the alerting system will diverge from the trajectory without the alerting system, called the *nominal trajectory*. An alert is defined to be necessary if the nominal trajectory results in conflict.

Table 4 enumerates the possible alerting system outcomes as described by Winder and Kuchar [75]. Figure 17 illustrates the outcomes using example trajectories. Note that, according to the definition of necessary alert, the following two outcomes are not possible: (1) an alert is necessary, an alert is not issued, and a conflict does not occur; and (2) an alert is not necessary, an alert is not issued, and a conflict occurs.

**TABLE 4**

**Outcome categories.**

| Outcome category | Abbreviation | Alert Necessary? | Alert Issued? | Conflict Occurred? |
|---|---|---|---|---|
| Correct Rejection | CR | | | |
| Correct Detection | CD | ✓ | ✓ | |
| False Alarm | FA | | ✓ | |
| Missed Detection | MD | ✓ | | ✓ |
| Induced Conflict | IC | | ✓ | ✓ |
| Late Alert | LA | ✓ | ✓ | ✓ |

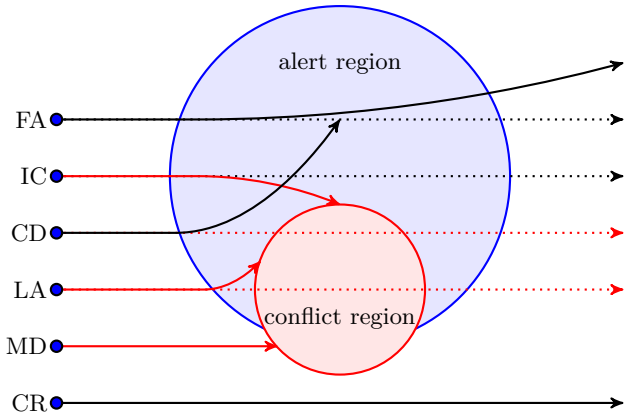



*Figure 17. Alerting outcome categories. Solid lines indicate trajectories with the system. Dotted lines indicate trajectories without the system. Red lines indicate that a conflict occurred with the system (solid) or that a conflict would have occurred without the system (dotted).*

### 5.2 METRICS

The performance of alerting systems is usually assessed using a few relevant quantifiable performance metrics. Many of these metrics can be calculated from counts of the six outcome categories of Section 5.1 generated in simulation over a wide range of possible aircraft encounters. Here, CR, CD, FA, MD, IC, and LA denote the counts of the alerting system outcomes.

1. **Probability of conflict:** The probability that a conflict occurs with the alerting system can be estimated from the outcome counts, assuming each encounter is equally likely, as follows:

$$\Pr(\mathrm{C}) = \frac{\mathrm{MD} + \mathrm{IC} + \mathrm{LA}}{\mathrm{CR} + \mathrm{CD} + \mathrm{FA} + \mathrm{MD} + \mathrm{IC} + \mathrm{LA}}. \tag{18}$$

   If the encounters are not equally likely, as will be the case in Section 5.4, the relative likelihood assigned to each encounter must be taken into account.

2. **Probability of alert:** The probability that the system alerts can be approximated by the frequency

$$\Pr(\mathrm{A}) = \frac{\mathrm{CD} + \mathrm{FA} + \mathrm{IC} + \mathrm{LA}}{\mathrm{CR} + \mathrm{CD} + \mathrm{FA} + \mathrm{MD} + \mathrm{IC} + \mathrm{LA}}. \tag{19}$$

   Together with the previous metric, an alerting system is deemed effective if Pr(A) is large enough to achieve a suitably small Pr(C), but not any larger.

3. **Probability of unnecessary alert:** An alert is considered to be unnecessary if it is not necessary to prevent conflict (i.e., the nominal trajectory does not result in conflict). The probability of unnecessary alert, Pr(UA), is the probability that, when an alert is issued, it is unnecessary. Pr(UA) can be approximated by

$$\Pr(\mathrm{UA}) = \frac{\mathrm{FA} + \mathrm{IC}}{\mathrm{FA} + \mathrm{IC} + \mathrm{CD} + \mathrm{LA} + \mathrm{MD}}. \tag{20}$$

4. **Probability of successful alert:** An alert is considered to be successful if it is issued and a conflict does not occur. The probability of successful alert, Pr(SA), is the probability that, when an alert is issued, it is successful. An effective alerting system has a low rate of unnecessary alert and a high rate of successful alert. Note that, according to the preceding definitions of unnecessary and successful alert, an alert can be simultaneously classified as unnecessary and successful. Pr(SA) can be approximated by

$$\Pr(\text{SA}) = \frac{\text{CD} + \text{FA}}{\text{FA} + \text{IC} + \text{CD} + \text{LA} + \text{MD}}. \tag{21}$$

5. **Risk ratio:** The risk ratio is a measure of the change in the probability of conflict due to the equipage of an alerting system. It is a concise metric for assessing the safety benefit of equipping an alerting system. It is defined as the ratio between the probability that a conflict will occur with the alerting system and the probability that a conflict will occur without the alerting system:

$$\text{RR} \equiv \frac{\Pr(\text{C} \mid \text{alerting system})}{\Pr(\text{C} \mid \text{no alerting system})} = \frac{\text{MD} + \text{IC} + \text{LA}}{\text{CD} + \text{MD} + \text{LA}}, \tag{22}$$

   where Pr(C | alerting system) is the probability of conflict with the alerting system and Pr(C | no alerting system) is the probability of conflict without the alerting system. A risk ratio of zero indicates that the system resolves all conflicts, while a risk ratio of one indicates that the system provides no additional benefit in reducing the number of conflicts. Risk ratios above one indicate poorly-designed alerting systems that increase the probability of conflict. Risk ratio has been used in prior TCAS safety studies [1–3, 14, 76–78].

6. **Unresolved risk ratio component:** This is the component of the risk ratio that is due to unresolved conflict risk. A conflict is *unresolved* if it occurs both with and without the alerting system. It is given by

$$\text{RR}_{\text{unresolved}} = \frac{\text{MD} + \text{LA}}{\text{CD} + \text{MD} + \text{LA}}. \tag{23}$$

   This is equivalent to the conditional probability of an unresolved conflict given that an alert is necessary.

7. **Induced risk ratio component:** This is the component of the risk ratio that is due to induced conflict risk. A conflict is induced if it occurs with the alerting system but not without the alerting system. It is given by

$$\text{RR}_{\text{induced}} = \frac{\text{IC}}{\text{CD} + \text{MD} + \text{LA}}. \tag{24}$$

8. **Vertical miss distance (VMD):** This report defines VMD as the (nonnegative) vertical separation between the aircraft at the point in the encounter when the horizontal miss distance is minimal. Good alerting systems should increase VMD without excessively disturbing the nominal flight path.

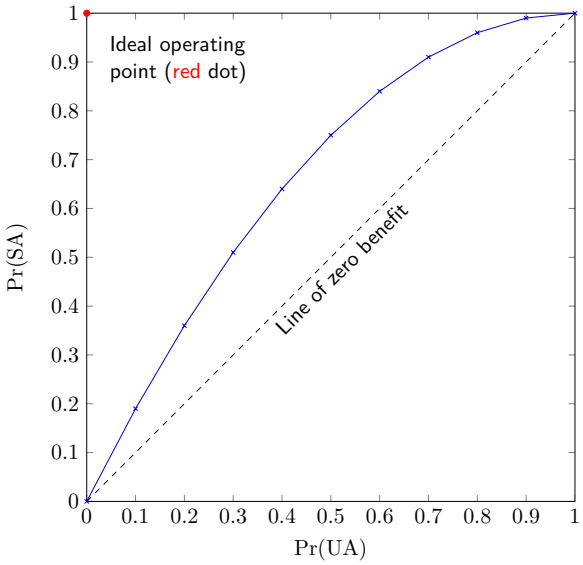



*Figure 18. Example SOC curve.*

With additional information about encounter rates, several other metrics may be derived such as mean time between conflict, which may be more intuitive as a safety indicator. The metrics above depend on the rates of specific discrete events (e.g., whether or not a conflict or an alert occurred). However, other metrics can be derived that summarize various continuous physical properties of the simulations. For example, it may be insightful to study the effect of different systems on the average flight-plan deviation [41] and the average vertical speed and acceleration [38]. Once the logic supports RA changes and VSLs, a collection of other metrics can be defined (e.g., the rate of RA reversals) to predict operational acceptability. Prior TCAS studies have used "triplet" outcome metrics to compare performance across no alerting system and two other alerting systems [79]. These metrics will be discussed in detail in a future safety study after further development of the DP logic.

### 5.3 SYSTEM OPERATING CHARACTERISTIC CURVES

The performance of an alerting system depends on many different parameters, such as dynamic model parameters, sensor model parameters, and alerting thresholds. The evaluation of alerting-system performance, therefore, can be quite complex. Kuchar [18,80] developed a unified methodology for the evaluation of alerting-system performance in which performance tradeoffs are analyzed through the use of system operating characteristic (SOC) curves. SOC curves, analogs of receiver operating characteristic (ROC) curves in signal detection theory [81], were originally introduced as plots of Pr(CD) versus Pr(FA). This report considers a modified version in which Pr(SA) is plotted against Pr(UA). A notional example of an SOC curve is shown in Figure 18.

Each point on the SOC curve is called an *operating point*. The shape of the curve is traced out as the alerting threshold, or any system parameter, is adjusted. Each alerting threshold maps to an

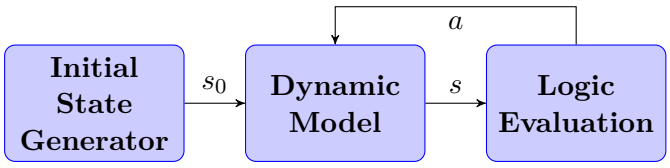



*Figure 19. Simple simulation framework.*

operating point. The ideal operating point is at the top-left corner of the plot, where Pr(UA) is zero and Pr(SA) is one. Due to uncertainty in the state and measurements, the ideal operating point is usually not obtainable. The dashed line represents the SOC curve for a system that provides no additional benefit; an alert is equally likely to be successful as it is unnecessary. Curves to the left of this line indicate systems that increase the probability of successful alert for the same level of unnecessary alerts. The closer the operating points are to the upper-left corner, the better the system performs. Through the use of an SOC curve, the effect of a system parameter change directly maps to a difference in system performance, thus highlighting the performance tradeoffs that must be brought to bear in the placement of effective system parameters.

Variants of the SOC curve are possible, so long as the horizontal axis represents the undesirable outcome (to be minimized) and the vertical axis the desirable one (to be maximized). One variant of the SOC curve involves plotting $1 - \Pr(\text{C})$ against $\Pr(\text{A})$. The advantage of this SOC curve is that the changes in the overall level of safety, $1 - \Pr(\text{C})$, which is sometimes more tangible, can be analyzed directly.

This section discusses SOC curves generated in a simulation framework that uses the hypothetical collision avoidance dynamics of Appendix A. Figure 19 outlines the simulation framework used to compute the performance metrics of Section 5.2. The next section discusses evaluation with an aircraft dynamic model that is significantly more realistic and has been used in prior TCAS safety studies.

Figure 20(a) presents the SOC curves of Pr(SA) versus Pr(UA) produced by varying the cost of alerting $\lambda$ from zero to one in discrete steps. Each operating point was obtained from running 10,000 encounters in simulation. The upper-right regions of the curves correspond to costs of alerting near zero and the lower-left regions correspond to costs near one. Each of the three SOC curves was produced by varying the level of uncertainty in the state as encoded in the amount of noise in the aircraft vertical rates. The amount of noise is controlled by the standard deviation of the vertical acceleration of the aircraft, $\ddot{h}$. Figure 20(b) is a plot of $1 - \Pr(\text{C})$ versus $\Pr(\text{A})$, again for the three levels of system noise.

To gauge the performance of the DP logic against the existing TCAS logic, a simplified version of the TCAS logic, called mini TCAS, was implemented. Appendix I outlines the principal assumptions of mini TCAS. The real TCAS algorithm periodically receives noisy measurements of the range, bearing, and altitude of nearby intruders and initializes and maintains intruder tracks based upon these surveillance data. Additionally, TCAS knows information regarding the own aircraft state with some level of uncertainty. Mini TCAS, on the other hand, receives perfect information about the own aircraft and intruder states and issues alerts based solely on this information.

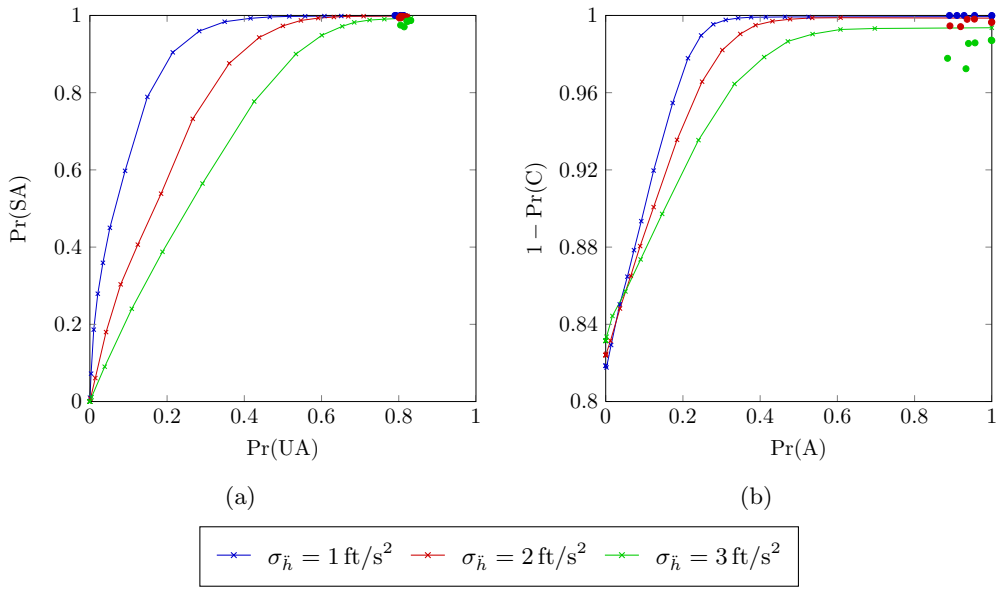



*Figure 20. SOC curves for the DP logic at different noise levels. Operating points for the TCAS logic at different sensitivity levels are shown as dots.*

This amounts to perfect tracking of own and intruder aircraft. Figure 20 shows several operating points of the mini TCAS system (represented as colored dots). Each operating point corresponds to a different altitude for own aircraft. Each altitude maps directly to a particular value for the altitude layer and sensitivity level parameters of the TCAS logic. All the TCAS thresholds vary as functions of either the altitude layer or the sensitivity level. Therefore, adjusting the altitude of own aircraft causes mini TCAS to alert differently. Table 5 is a list of the own altitude, sensitivity level, and altitude layer for each of the TCAS operating points of Figure 20.

If the cost of alerting is high, Pr(SA) and Pr(UA) are small because it is not advantageous to alert. When the cost of alerting is decreased, alerts are issued more frequently, increasing both Pr(SA) and Pr(UA). In general, the system performance degrades as the system uncertainty

**TABLE 5**

**Own altitude, sensitivity level, and altitude layer of TCAS operating points.**

| Own Altitude (ft) | Sensitivity Level | Altitude Layer |
|---|---|---|
| 1000 | 3 | 1 |
| 3000 | 4 | 2 |
| 7000 | 5 | 3 |
| 15,000 | 6 | 4 |
| 30,000 | 7 | 5 |
| 43,000 | 7 | 6 |

increases. The alerts produced by the TCAS logic are mostly successful but, as is often the case, mostly unnecessary. This is understandable because the objective of TCAS is to provide a minimum level of vertical separation (ALIM) that is always greater than the vertical separation that defines a conflict. In other words, TCAS alerts conservatively.

If, from an operational standpoint, the requirements on the system dictate that the rate of unnecessary alert (or false alarm) cannot exceed a certain level, SOC curves can be used to find the cost of alerting for which the probability of successful alert (or correct detection) is the highest.

Similar trends can be observed in Figure 20(b). When the cost of alerting is low, the overall level of safety is as high as 99% but at the expense of over-alerting. In general, any alerting system that issues alerts at a high rate is capable of ensuring a high level of safety. Even when the cost of alerting is high, causing Pr(A) to be effectively zero, approximately 80% of encounters do not result in conflict. This is equivalent to the percentage of conflict encounters without the alerting system.

Appendix F proves that an optimal policy generated by dynamic programming satisfies the following properties:

1. There is no other policy with the same alert rate and a lower conflict rate.

2. There is no other policy with the same conflict rate and a lower alert rate.

It follows from this result that optimal policies generated by dynamic programming will trace out the best possible SOC curve, assuming that the discretization is sufficiently fine.

Figure 20 shows SOC curves for the logic obtained by approximating the integral of Equation 15 using sigma-point sampling and multilinear interpolation. This is just one of several methods for estimating the expected cost-to-go at the next time step from each state. For example, instead of drawing a fixed number of deterministic samples, as the sigma-point sampling method does, direct Monte Carlo samples from $p(\mathbf{x}' \mid \mathbf{x}, a)$ can be drawn. Figure 21 shows the effect that using direct Monte Carlo sampling has on the overall system performance. Unsurprisingly, performance improves as more samples are used to estimate the expected cost-to-go. The use of the sigma-point sampling method, which uses only five deterministically chosen samples that capture the mean and covariance of $p$, results in the best performance while relieving much of the computational burden involved in sampling the next states. In a similar manner, the effect of using different interpolation methods and changing the resolution of the discretization can also be explored, although it is not done so in this report.

The next set of SOC curves evaluates the conflict probability estimation approach for different threshold settings. Figure 22 shows the SOC curves for the three policies of Section 4: conservative, delay, and conservative delay. As before, 10,000 encounters were used to calculate each operating point. The curves were traced out as the threshold on the probability of conflict was varied. The action that was executed at each time step was determined by calculating the probability of conflict online and applying the collision avoidance logic of Section 4. The probability of conflict was estimated using 100 trajectories drawn from the maximum-likelihood acceleration proposal

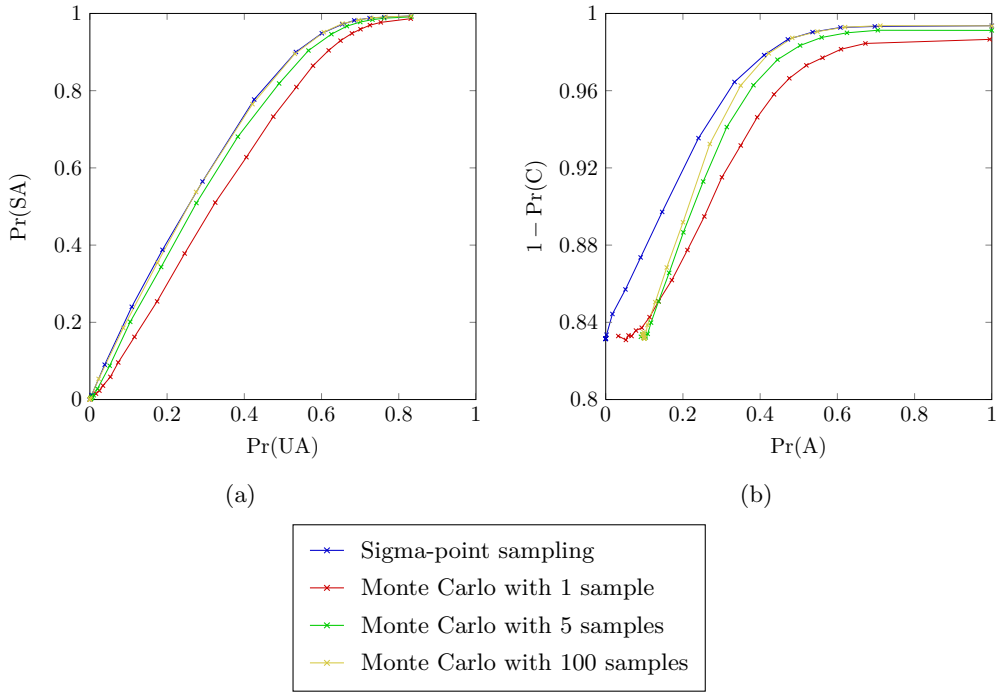



*Figure 21. Effect of different sampling methods on overall system performance.*

distribution. Alternatively, one could reduce computation time during the evaluation process, at the expense of accuracy, through the use of a lookup table generated offline.

The range over which the threshold was varied to produce SOC curves that span the metric space well depends on the policy. In the case of the conservative policy, the SOC curves were plotted by varying the probability of conflict threshold from zero to one in uniform increments. When the threshold is close to one, the system rarely alerts. When the threshold is near zero, the system almost always alerts. Presumably, if the threshold is one, the probability of alerting, Pr(A), should be zero. The curves of the conservative policy never extend to $\Pr(\mathrm{A}) = 0$, however. This is because the estimate of the probability of conflict using importance sampling is not confined to the interval $[0, 1]$, as a true probability should. Therefore there is some non-zero probability that the probability of conflict without alerting is greater than one and the system alerts. In the limit as the number of sample trajectories approaches infinity, however, the estimate should lie within the interval $[0, 1]$. The estimate can be forced to lie within the interval $[0, 1]$ by normalizing the weights. However, this introduces a bias that goes to zero as the number of samples approaches infinity [82].

The SOC curves for the delay policy and the conservative delay policy were obtained by varying the threshold from $10^{-30}$ to one and from $10^{-8}$ to one in a logarithmic scale, respectively. The operating point corresponding to a threshold of zero was also computed. A logarithmic scale was necessary to capture the variability of the curve in the regions where Pr(SA) and Pr(UA) are high.

The results suggest that all three policies perform similarly. The conservative policy outperforms the other two, as shown by its SOC curves that almost always lie above and to the left of the curves formed by the other policies. In other words, the conservative policy nearly always achieves a higher rate of successful alert given some rate of unnecessary alert and a lower rate of conflict given some alert rate. Moreover, the conservative policy tends to issue fewer alerts while guaranteeing the same level of safety. In the limit as the alert rate goes to one, the rate of successful alert and the rate that no conflict occurs for all policies tend to unity.

Appendix F implies that the curve of $1-\Pr(\text{C})$ versus $\Pr(\text{A})$ for the optimal policy never lies to the right of or below that of any other policy. However, because the policy calculated in this report is only approximately optimal, due primarily to the discretization process, the results do indeed indicate that there are some regions in which the other policies outperform the (approximately) optimal policy. For example, when $\sigma_{\dot{h}} = 1\,\text{ft/s}^2$, the lowest conflict rate that the optimal policy achieves is 0.006 while the other policies achieve a minimum conflict rate of zero, given a sufficient alert rate.

## 5.4 PRELIMINARY SAFETY EVALUATION

This section presents the preliminary safety evaluation of the dynamic programming logic using the Collision Avoidance System Safety Assessment Tool (CASSATT), developed at Lincoln Laboratory. CASSATT performs fast-time Monte Carlo analysis of close encounters between two or more aircraft over a period on the order of one minute near the closest point of approach. CASSATT has been used for prior TCAS safety analysis [76] and sense-and-avoid development for unmanned aircraft [77].

The CASSATT framework was built in Matlab and Simulink and has been compiled into native code using Real-Time Workshop. The framework was designed to be modular to allow different collision avoidance systems and sensor models to be easily incorporated. CASSATT was extended to allow communication with an arbitrary collision avoidance system over a TCP/IP socket connection [38]. The collision avoidance system runs as a server to which CASSATT connects as a client.

Figure 23 provides an overview of the simulation framework. An encounter model generates the initial conditions and scripted maneuvers for both aircraft involved in the encounter. For the purposes of this study, the encounter model developed by Lincoln Laboratory for cooperative aircraft [7] was used. This encounter model was developed from nine months of national radar data from over 120 sensors maintained by the Federal Aviation Administration and the Department of Defense. A dynamic Bayesian network representing the behavior of the aircraft was learned from actual encounters extracted from the data. Generating new encounters for use in Monte Carlo analysis involves sampling from the dynamic Bayesian network.

The initial conditions define the starting positions of the aircraft. The scripted maneuvers control aircraft velocities and accelerations at each time step. The dynamic model takes as input the state and scripted maneuvers at the previous time step and returns the state of the aircraft at

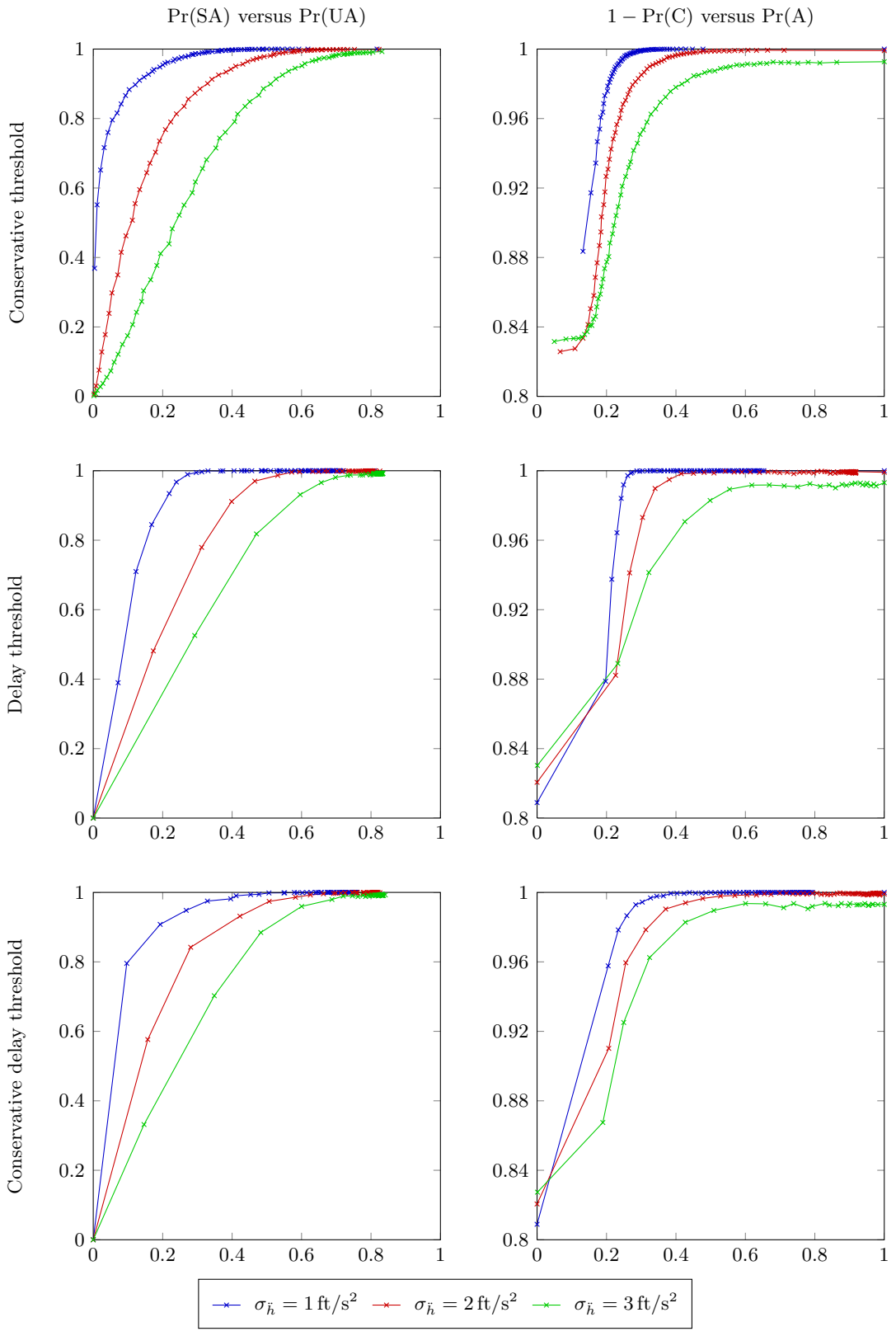



*Figure 22. SOC curves for threshold alerting policies.*

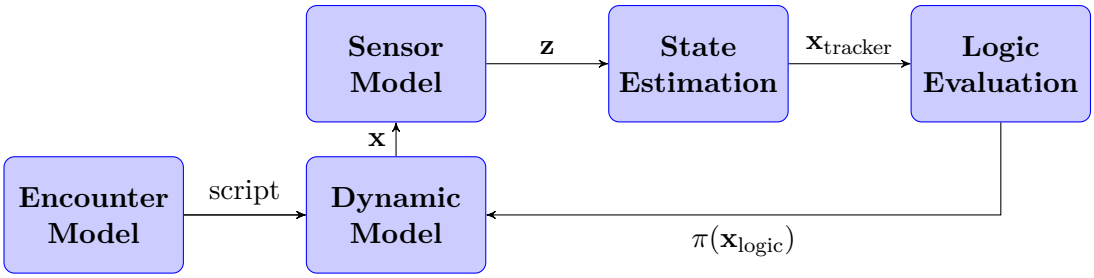



*Figure 23. CASSATT simulation framework.*

the current time step. The state of one aircraft is represented by a seven-dimensional vector

$$(v, N, E, h, \psi, \theta, \phi), \tag{25}$$

where $N$, $E$, and $h$ are the north, east, and altitude components of the position in a local flat-earth coordinate system, $\psi$, $\theta$, and $\phi$ are the yaw, pitch, and roll angles, and $v$ is the true airspeed. In a two-aircraft system, the state is

$$\mathbf{x} = (v_1, N_1, E_1, h_1, \psi_1, \theta_1, \phi_1, v_2, N_2, E_2, h_2, \psi_2, \theta_2, \phi_2), \tag{26}$$

where subscript 1 refers to the own aircraft and subscript 2 refers to the intruder. The motion of the aircraft is driven by controls in the turn rate, vertical rate, and airspeed acceleration, as supplied by the encounter model or, optionally, as defined by the user. These control values may change every tenth of a second. Aircraft transient response characteristics and performance limits such as maximum pitch rate or bank angle are also included in the dynamic model.

The sensor model takes as input the current state from the dynamic model and produces an observation, or sensor measurement. The sensor produces measurements of the own aircraft altitude, $h_{\text{own}}$, the intruder altitude, $h_{\text{int}}$, the bearing to the intruder, $\chi$, and the range to the intruder, $r$. The measurement $\mathbf{z}$ is given by

$$\mathbf{z} = (\tilde{h}_{\text{own}}, \tilde{h}_{\text{int}}, \tilde{\chi}, \tilde{r}), \tag{27}$$

where here the tilde is used to designate a measurement. The measured bearing is equal to the true bearing plus additive zero-mean Gaussian noise with a standard deviation of 10°. Similarly, the modeled range noise is zero-mean Gaussian with a standard deviation of 50 ft. The altitude measurements are quantized to 25-ft increments. The bias in the altimetry error is distributed according to a Laplacian distribution whose parameters are dependent upon the altitude layer. All sensor error parameters match those specified in the International Civil Aviation Organization (ICAO) model [14].

Based on the most recent observation, the state estimation process updates the estimates of the positions, $h_{\text{own}}$, $h_{\text{int}}$, and $r$, of the rates, $\dot{h}_{\text{own}}$, $\dot{h}_{\text{int}}$, and $\dot{r}$, and of the range acceleration, $\ddot{r}$. The state estimate of the tracker $\mathbf{x}_{\text{tracker}}$ is

$$\mathbf{x}_{\text{tracker}} = (\widehat{h}_{\text{own}}, \widehat{h}_{\text{int}}, \widehat{r}, \widehat{\dot{h}}_{\text{own}}, \widehat{\dot{h}}_{\text{int}}, \widehat{\dot{r}}, \widehat{\ddot{r}}), \tag{28}$$

where a hat is used to designate an estimate. The state of the tracker (Equation 28) is not the same as the full 14-dimensional state of the aircraft (Equation 26) as used in the dynamic model (which is not completely observable). An $\alpha$-$\beta$ filter was used to track altitude and altitude rate, and an $\alpha$-$\beta$-$\gamma$ filter was used to track intruder range, range rate, and range acceleration. The tracker is a simplified version of the tracker implemented in the current version of TCAS for intruders broadcasting altitude with 25-ft quantization. Appendix H discusses the tracker in further detail.

The DP logic is evaluated on the updated state estimate. Before evaluating the logic, the output of the tracker, $\mathbf{x}_{\text{tracker}}$, must be mapped to an approximate input to logic evaluation, $\mathbf{x}_{\text{logic}}$, defined by

$$\mathbf{x}_{\text{logic}} = (h, \tau, \dot{h}_1, \dot{h}_2, s_{\text{RA}}), \tag{29}$$

as in Table 3. If an RA has not been issued and the aircraft are estimated to be closing (i.e, $\widehat{\dot{r}} < 0$), the mapping from $\mathbf{x}_{\text{tracker}}$ to $\mathbf{x}_{\text{logic}}$ is described by the following equations:

$$\begin{cases} h &= \widehat{h}_{\text{int}} - \widehat{h}_{\text{own}}, \\ \tau &= |\widehat{r}/\widehat{\dot{r}}|, \\ \dot{h}_1 &= \widehat{\dot{h}}_{\text{own}}, \\ \dot{h}_2 &= \widehat{\dot{h}}_{\text{int}}, \\ s_{\text{RA}} &= \text{clear of conflict}. \end{cases} \tag{30}$$

The optimal action to take is $\pi^*(\mathbf{x}_{\text{logic}})$. Once an RA has been issued, the RA is maintained for the remainder of the encounter. If the aircraft are diverging in range, no RA is issued. Receiving the appropriate action from the logic evaluation step, the dynamic model updates the state $\mathbf{x}$, and the process continues until the end of the encounter.

For the preliminary safety evaluation of the DP logic discussed in this report, 500,000 trajectories were generated using the encounter model. The aircraft trajectories were rotated and translated to create the geometry at CPA as given by the model. The impact of using the DP logic was evaluated in the CASSATT simulation framework (Figure 23). As a baseline, the performance of TCAS II version 7.1 was also evaluated for comparison. For the purposes of this study, the intruder was equipped with a Mode S transponder but not equipped with a collision avoidance system. Both own aircraft and the intruder reported altitude in 25-ft increments. The Mode S address of the intruder was higher than that of own aircraft. For this preliminary evaluation, the DP logic was calculated using a cost of alerting of 0.1. Analysis of the performance of the DP logic in the CASSATT framework when using a different cost of alerting is beyond the scope of this initial study.

In the calculation of the DP logic, conflict between the aircraft was defined as less than 100 ft vertical separation at the point when horizontal separation has been lost. It is rare, however, for the aircraft to lose all horizontal separation in the simulation framework of Figure 23. CASSATT defines a conflict more generally as a near mid-air collision (NMAC), a loss of separation less than 500 ft horizontally and 100 ft vertically [4].

Instead of sampling the encounters directly from the distribution given by the encounter model, encounters are sampled from an alternative distribution, called the proposal distribution,

to increase the probability that an encounter results in an NMAC. This procedure is known as importance sampling, as discussed in a different context earlier in Section 4. In addition to reducing the computational time, importance sampling produces lower variance estimates of performance metrics. The encounters generated using importance sampling are not equally likely to occur. Instead, a weight $w^{(i)}$ is assigned to each encounter that indicates the likelihood with which it would have been sampled from the original distribution of the model. The probabilities of Section 5.2 are approximated using weighted encounters.

Table 6 lists the probabilities of each of the alerting system outcomes of Section 5.1 for both the DP logic and TCAS. These estimates ignore the effect of altimetry bias. The remainder of this section compares several metrics of Section 5.2 between the DP logic and TCAS.

**TABLE 6**

**Probability of each alerting system outcome.**

| Outcome category | DP | TCAS |
|---|---|---|
| Correct Rejection | $8.69 \cdot 10^{-1}$ | $4.89 \cdot 10^{-1}$ |
| Correct Detection | $2.88 \cdot 10^{-3}$ | $2.93 \cdot 10^{-3}$ |
| False Alarm | $1.28 \cdot 10^{-1}$ | $5.08 \cdot 10^{-1}$ |
| Missed Detection | $6.25 \cdot 10^{-5}$ | $0.00 \cdot 10^{0}$ |
| Induced Conflict | $8.45 \cdot 10^{-5}$ | $3.56 \cdot 10^{-5}$ |
| Late Alert | $4.50 \cdot 10^{-5}$ | $6.08 \cdot 10^{-5}$ |

### 5.4.1 Probability of alert

The probability of alert can be approximated using importance sampling as follows:

$$\Pr(\mathrm{A}) \approx \frac{1}{N} \sum_{i=1}^{N} w^{(i)} f(\omega^{(i)}), \tag{31}$$

where $N$ is the number of encounters, $\omega^{(i)}$ represents the $i$th encounter, and $f(\omega)$ is a function that takes as input an encounter and returns one if the alerting system alerted during the encounter and zero otherwise. A high probability of alert is generally undesirable and may indicate that the system may be alerting more often than necessary. The probability of alert can also be estimated by summing the probabilities of correct detection, false alarm, induced conflict, and late alert as given in Table 6.

The results of the simulation using 500,000 encounters show that the probability of alert for TCAS is 51.1%, whereas the probability of alert for the DP logic is 13.1%. This situation admits several possible explanations. As previously discussed, TCAS alerts frequently due to the minimum vertical separation requirements that it tries to meet. Additionally, TCAS has a wider spectrum of alerts that it can issue (Section 1.2). In addition to climb and descend RAs, it can issue vertical speed limits, increase rate RAs, and sense reversals. Notwithstanding, TCAS does

**TABLE 7**

**Risk ratios with TCAS v. 7.1 and the DP logic.**

| Logic | Risk Ratio | Unresolved Risk Ratio | Induced Risk Ratio |
|---|---|---|---|
| TCAS | $5.54 \cdot 10^{-2}$ | $2.89 \cdot 10^{-2}$ | $2.64 \cdot 10^{-2}$ |
| DP | $1.36 \cdot 10^{-1}$ | $9.19 \cdot 10^{-2}$ | $4.43 \cdot 10^{-2}$ |

not issue RAs against intruders for which a large horizontal miss distance is projected. The DP logic does not account for such circumstances and therefore, although it alerts less than TCAS, still has the tendency to over-alert. By adding a miss distance filter similar to that which is included in TCAS, the alert rate could be reduced further.

### 5.4.2 Probability of NMAC

Using importance sampling, the probability of an NMAC, given that the aircraft are involved in an encounter, can be estimated by

$$\Pr(\text{NMAC}) \approx \frac{1}{N} \sum_{i=1}^{N} w^{(i)} \Pr(\text{NMAC} \mid \omega^{(i)}), \tag{32}$$

where $\Pr(\text{NMAC} \mid \omega^{(i)})$ is the probability that the $i$th encounter resulted in an NMAC. The probability that the $i$th encounter resulted in an NMAC is calculated by integrating the combined altimeter error distribution of the aircraft over the range of errors that would cause the true vertical separation to be less than 100 ft [79, 83].

Without any collision avoidance system, $\Pr(\text{NMAC}) = 0.00311$. This means that the simulation indicates that one in approximately 321 close encounters results in an NMAC when no collision avoidance system is equipped. With TCAS, $\Pr(\text{NMAC})$ is reduced to 0.000172. With the DP logic, $\Pr(\text{NMAC}) = 0.000424$. In either case, the risk of an NMAC is significantly reduced. The probability of NMAC with the DP logic is approximately twice as large as the probability of NMAC with the existing TCAS.

It is encouraging to observe that the probability of an NMAC using the DP logic is comparable to that using TCAS. Decreasing the cost of alerting will decrease the probability of an NMAC with the DP logic even further. Apart from this, there are still several fundamental ways in which the DP logic can be substantially improved, including extending the motion model used to compute the logic, accounting for the effects of sensor uncertainty, and expanding the number of alerts that the DP logic can issue. These are discussed in Section 5.6. While the results summarized in this report are indeed promising, they are still preliminary in nature, and further refinement of the DP logic is expected to result in significant performance gains.

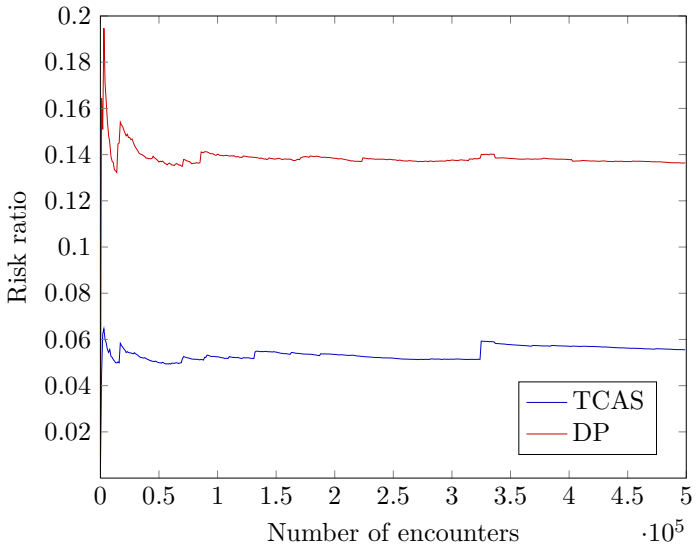



*Figure 24. Risk ratio convergence.*

### 5.4.3 Risk Ratios

Section 5.2 defined the risk ratio as the ratio between the probability that a conflict would occur with the alerting system and the probability that a conflict would occur without the alerting system. A risk ratio of zero is indicative of an effective collision avoidance system, while a risk ratio above one is indicative of an adverse system. If a conflict is defined as an NMAC, the risk ratio for TCAS is 0.0554, while the risk ratio for the DP logic is 0.136. Table 7 lists the risk ratios for TCAS and the DP logic as well as the unresolved and induced risk ratio components.

These estimates of risk ratio are sensitive to the number of samples (encounters) used to calculate them. As more samples are used, the accuracy of the estimates increases. Figure 24 shows the convergence of the risk ratio estimates as the number of encounters increases. Empirically, it appears that the estimates have converged. This also serves to show that, with a high degree of confidence, the risk ratio of TCAS is lower than that of the DP logic. Further research will reveal the extent to which the DP logic can be improved.

### 5.4.4 Vertical Miss Distance

Recall that VMD is defined as the (nonnegative) vertical separation between the aircraft at the point in the encounter when the horizontal miss distance is minimal. Figure 25 compares the VMD using TCAS and VMD using the DP logic for all 500,000 weighted simulated encounters. The figure consists of several regions. Points on the diagonal line represent encounters for which VMD using TCAS and VMD using the DP logic are the same. Points that lie below the line correspond to encounters for which the DP logic provides greater vertical separation than TCAS. Points that lie above the line correspond to encounters for which TCAS provides greater vertical separation than the DP logic. As the figure shows, most of the time DP and TCAS produce the same vertical separation, which is because neither issues an alert. When at least one system alerts, TCAS is

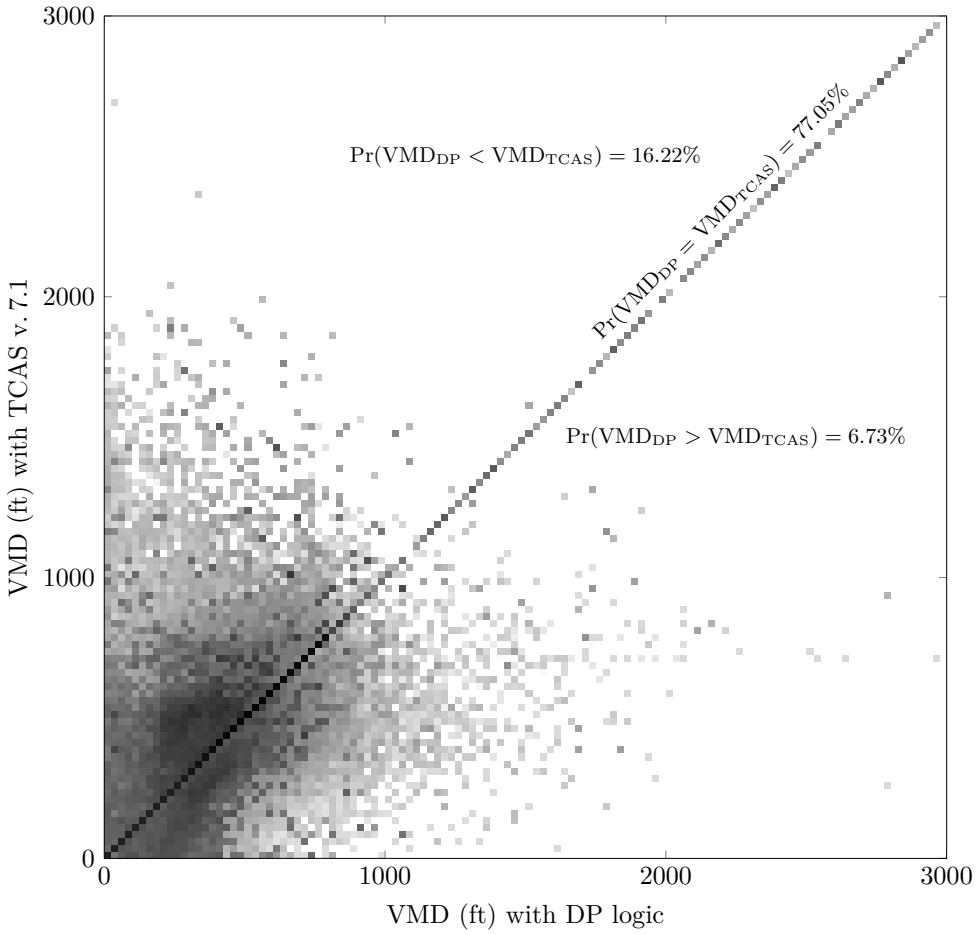


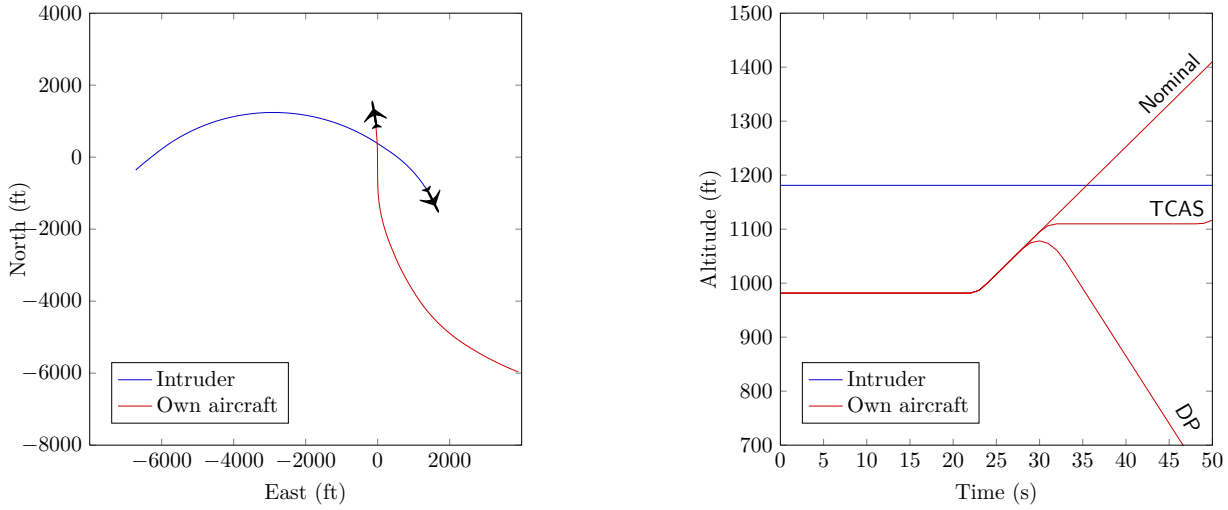

*Figure 25. Comparison of VMD using TCAS v. 7.1 with VMD using the DP logic. The relative frequency of the various cells is indicated using a logarithmic gray scale. Altimetry bias is zero.*

more than twice as likely to provide greater vertical separation. Decreasing the cost of alerting will result in a DP logic that will, on average, increase vertical separation.

## 5.5 EXAMPLE ENCOUNTERS

The previous sections assessed the overall performance of the logic in terms of a few comprehensive metrics. It can be enlightening, however, to closely examine the logic on a small set of simulated encounters. This section analyzes two simulated encounters: one in which the DP logic prevents an NMAC that TCAS fails to prevent and, conversely, one in which TCAS prevents an NMAC that the DP logic fails to prevent. Approximately 0.008% of encounters fall under the former category, while approximately 0.011% fall under the latter category.

Figure 26 shows the horizontal and vertical profiles of a simulated encounter in which the DP logic prevents an NMAC that the TCAS fails to prevent. Both the own aircraft and the intruder




(a) Horizontal profile.

(b) Vertical profile.

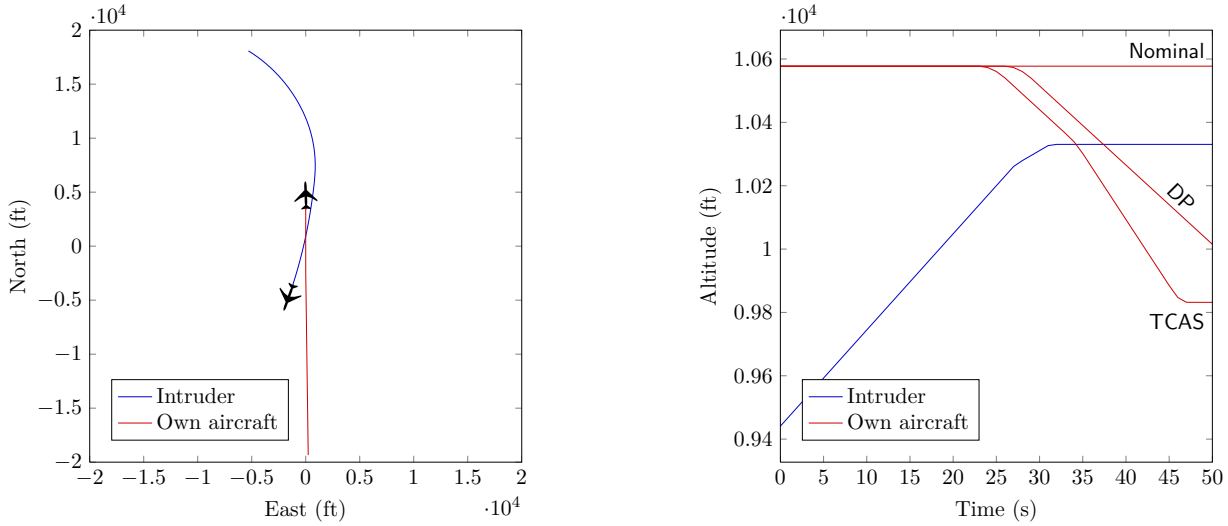
*Figure 26. Simulated encounter in which the DP logic prevents an NMAC that TCAS fails to prevent.*

are initially flying level. After 22 seconds, own aircraft begins to climb, which causes TCAS to detect a threat and issue a Do Not Climb VSL three seconds later. However, even though the own aircraft is compliant with the RA and the intruder remains level, an NMAC occurs. The DP logic, on the other hand, prevents an NMAC by issuing a descend advisory 23 seconds in, which remains in effect for the remainder of the encounter.

Analysis of other encounters of this type suggests other reasons why the DP logic performs better than TCAS in select encounters:

- TCAS issues corrective RAs later than is necessary in preventing an NMAC, while the DP logic alerts earlier;
- TCAS, being strongly biased against altitude-crossing RAs, issues non-altitude-crossing RAs that result in NMACs, while the DP logic makes no such distinction; and
- TCAS either fails to alert or reverses the sense of an RA but, nonetheless, the encounter results in an NMAC.

Figure 27 shows another simulated encounter where TCAS prevents an NMAC but the DP logic fails. The intruder climbs at approximately 1800 ft/min for 27 seconds. Twenty-one seconds into the encounter, while the intruder is still climbing, the DP logic advises own aircraft to descend 1500 ft/min. This is because the expected cost of climbing or remaining level is greater than the expected cost of descending. After the five-second pilot delay, own aircraft begins descending and continues to descend for the remainder of the encounter. An NMAC occurs, however, because the intruder begins to level off as own aircraft is descending.




(a) Horizontal profile.

(b) Vertical profile.

*Figure 27. Simulated encounter in which TCAS prevents an NMAC that the DP logic fails to prevent.*

This encounter evolved into an NMAC because, although the alert to descend 1500 ft/min seemed reasonable when the intruder was climbing, this alert was inappropriate when the intruder began leveling off. Similar to the DP logic, TCAS issues an altitude-crossing descend RA, which is then changed to a non-altitude-crossing descend RA. However, as the encounter progresses, the RA is strengthened to an increase descent RA, allowing the aircraft to cross safely. Currently no provisions have been made in the DP logic to change the RA when the evolution of the encounter requires it. Maintaining the vertical rate that the DP logic indicates for the remainder of the encounter could lead to unsafe encounters. Extending the DP logic to include strengthening and reversals is discussed in Section 7.3.

## 5.6 DISCUSSION

This section showed that the DP logic performed relatively well compared to the existing TCAS logic. However, the current implementation of the DP logic was at a disadvantage against the TCAS logic in the three-dimensional simulations for several reasons:

1. **Motion model:** The DP logic is calculated using a model that only captures motion in the vertical plane. The motion in the horizontal plane is important to model because it affects the time to potential conflict. Expanding the vertical motion model to capture horizontal dynamics is discussed in Section 7.1.

2. **Sensor uncertainty:** The DP logic made decisions assuming that the tracked sensor measurements were correct. However, due to noise in the sensor measurements, making decisions based on this assumption can lead to unnecessary or unsuccessful alerts. The sensor error

model should be used to estimate the state uncertainty, which can be used by the QMDP value method discussed in Section 2.4.

3. **Action space:** TCAS allows RAs to be strengthened and reversed, whereas the DP logic must commit to a particular RA once it is issued for the remainder of the encounter. Therefore it is not surprising that TCAS resolves more NMACs than the DP logic. Section 7.3 discusses how additional RAs might be included in the DP logic.

All three of these issues will be addressed in further work and should result in a significant improvement in performance. Further work will also investigate other operating points in the CASSATT simulation for the DP logic.

This page intentionally left blank.

# 6. ALTERNATIVE APPROACHES

Kuchar and Yang provide a thorough survey of conflict detection and resolution methods [33]. This section discusses some of the key approaches they identified as well as other approaches introduced since their survey. A variety of other algorithms have been suggested in the literature, but the majority are variations of one of the general approaches discussed in this section—or are based on the dynamic programming or conflict probability estimation approaches discussed earlier.

## 6.1 POTENTIAL FIELD METHODS

Potential fields have been a very popular method for collision avoidance in robotics since they were introduced over thirty years ago [84–86] and have been applied to collision avoidance for aircraft [87, 88]. The typical potential field works by exerting virtual forces on the aircraft, usually an attractive one from the goal or waypoint and repelling ones from nearby traffic. The approach is very simple to describe and easy to implement, but it has some fundamental problems [89].

One problem with potential field methods is that it is a greedy strategy that is subject to local minima, which can lead to the aircraft getting "stuck." For example, the sum of the repulsive forces from the traffic can cancel the attractive force from the goal, resulting in zero net potential. The problems with local minima have led to the development of heuristics on top of the potential fields to escape such traps and eventually to the construction of navigation functions, which are potential fields with unique minima. These navigation functions are global and are equivalent to value functions [90].

Another issue with potential fields is that they do not take into account the probabilistic dynamics of the intruders. The potential fields are typically defined in two or three spatial dimensions without taking into consideration the aircraft dynamics. If the aircraft are moving at different speeds, large virtual forces are needed to repel the traffic. If the forces are too small, a collision can result with a fast intruder. If the forces are too large, slower aircraft will deviate unnecessarily from their intended course. These dynamic problems are often overlooked for slow-moving mobile robots, but they are more significant in aircraft.

Perhaps the most significant problem with potential fields in the context of last-minute collision avoidance is that they do not account for uncertainty in control or observation. Because of the short time frame and the catastrophic nature of collision, decisions must be robust to noise in the sensor measurements and unexpected deviation from the projected flight course.

## 6.2 RAPIDLY EXPANDING RANDOM TREES

Rapidly expanding random trees (RRTs) were originally developed in the context of robot navigation [91]. RRTs explore the configuration space of the robot by generating random samples and connecting them to one or more trees rooted at the origin and at the destination. RRTs exhibit what is known as Voronoi-bias, which allows them to sample preferentially in previously unexplored

areas, and so they tend to cover the space more quickly than a random walk. Typically, RRTs find paths from the start to the goal long before the space is densely sampled.

RRTs have been remarkably successful in practice. They have become the solution of choice for high-dimensional robot motion planning problems and for kino-dynamic problems that involve not just positions but velocities [92]. The MIT autonomous ground vehicle that completed the DARPA Urban Challenge used RRTs to successfully drive in lanes, execute three-point turns, park, and maneuver with other traffic and obstacles [93]. RRTs have also been used for obstacle avoidance for micro air vehicles [94].

To balance their many desirable properties, it is important to note that RRTs make no guarantee of optimality of the path they find. In practice, the paths found by RRTs can be quite poor due the fact that they are constructed by joining together random samples. RRT-based motion planners typically go through a second phase of smoothing the resulting path to get one that is acceptable, but generally far from optimal [95]. The other limitation of RRTs is that they do not consider uncertainty, which is critical to airborne collision avoidance. There is some current research aimed at extending RRTs for solving MDPs and situations where the environment model is uncertain, but the results are preliminary [96].

### 6.3 GEOMETRIC OPTIMIZATION

Bilimoria introduced a lateral conflict resolution algorithm based on geometric optimization [28]. Given position and velocity information, the algorithm uses straight-line projection to determine whether the own aircraft will penetrate a circular protected zone of an intruder. If a conflict is predicted to occur, the algorithm will compute the minimal change in velocity required to avoid the protected area of an intruder. The assumption of constant velocity allows the algorithm to quickly compute an analytical solution. KB3D is a generalization of this algorithm to three-dimensional space where cylinders define the protected zones of intruders [29]. The algorithm outputs the minimal change in airspeed, vertical rate, or heading required to prevent the aircraft from entering a protected zone.

These geometric optimization approaches typically lend themselves to fast, analytical solutions. Because they recommend the minimal velocity change, the projected flight path (assuming constant velocity) will always barely miss the edge of the protected zone. If this approach is to be used for collision avoidance with real aircraft, the protected zone would need to be enlarged to make the system robust to flight path variability. Future work can investigate the SOC curve traced when varying the size of the protected zone. It would be interesting to compare how close a geometric optimization curve approaches those in Section 5.

### 6.4 POLICY SEARCH

One approach is to choose a policy representation whose behavior is defined by a vector of parameters $\boldsymbol{\theta}$. The choice of parameters requires insight into the structure of the problem and engineering judgment. The choice of policy representation is flexible; for example, one could use pseudocode

resembling the current version of TCAS or a neural network. Local or global search methods can search the space of parameters for an optimal $\boldsymbol{\theta}$ that minimizes $J(\boldsymbol{\theta})$, which is the expected cost of following the policy determined by $\boldsymbol{\theta}$ assuming some distribution over initial states. One advantage of policy search is that the state space does not require discretization, therefore allowing the approach to better scale to higher dimensional problems.

A local search method known as *policy gradient* starts with some initial parameter vector $\boldsymbol{\theta}_0$ and estimates the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0)$ using sampling. The next parameter vector $\boldsymbol{\theta}_1$ is chosen by taking a small step in the direction of the gradient, and the process continues until reaching a local minimum [97]. Policy gradient has been applied to aircraft collision avoidance [52, 73] and autonomous helicopter flight [98], among other domains.

The gradient descent approach will not necessarily find the globally optimal parameter vector due to local minima. Various methods have been suggested to make local search more robust to local minima, such as tabu search [99] and simulated annealing [100]. Another approach is to use genetic algorithms or some other evolutionary search method to find a policy that minimizes $J(\boldsymbol{\theta})$ [101–103]. Durand, Alliot, and Médioni demonstrated how genetic algorithms can be used to find a neural network controller to resolve lateral conflicts between aircraft [104].

The computation involved in policy search can be done offline by powerful computers, as opposed to online by the collision avoidance system during flight. The "fitness evaluation" stage of evolutionary algorithms can be parallelized, and local search can be conducted from a large collection of initial points in the parameter space in parallel. However, adequately estimating $J(\boldsymbol{\theta})$ or its gradient requires many samples, even when using importance sampling, which can make the pace of the search very slow. Finding optimal solutions may be faster using a dynamic programming formulation (Section 3) because it explicitly leverages models of aircraft dynamics and sensor error.

### 6.5 NONSTATIONARY APPROXIMATE POLICY ITERATION

Another approach is to search a policy in stages, working backwards from the terminal states (i.e., when there is a collision or there is negligible probability of conflict occurring). The learning problem at each stage is treated as a supervised multiclass classification problem, which can leverage any classifier suited for the domain. This kind of approach, known as *nonstationary approximate policy iteration* [105] or *policy search by dynamic programming* [106], has been applied by Kaelbing and Lozano-Perez to a hypothetical collision avoidance scenario similar to the one explored in this report [52]. The policy generated using this approach is called *nonstationary* because it is time dependent. Instead of being defined by a simple mapping $\pi$ from states to actions, the nonstationary policies are defined by a sequence of mappings $(\pi_1, \ldots, \pi_k)$ from states to actions, where $\pi_i$ dictates the action to execute $i$ steps before termination.

The algorithm suggested by Kaelbling and Lozano-Perez begins by simulating a large collection of trajectories through the state space until they reach a terminal state. All of the states from the trajectories are partitioned into a collection of sets, $D_0, \ldots, D_k$, where $D_i$ contains all the states $i$ steps from termination.

The following procedure is used to define $\pi_1$. For each state in $D_1$, the expected cost for executing each action is estimated using sampling (Appendix D). Each state is paired with the action that resulted in the lowest expected cost. This set of state-action pairs serves as a training set for a multiclass classification algorithm [61, 62], resulting in $\pi_1$ that maps states to actions.

Once $\pi_i$ is known, $\pi_{i+1}$ may be computed as follows. For each state in $D_{i+1}$, the expected cost-to-go for executing each action is estimated by sampling the next state and then following the nonstationary policy defined by $(\pi_1, \ldots, \pi_i)$. Each state is paired with the action that resulted in the lowest expected cost-to-go. This set of state-action pairs serves as a training set for a multiclass classification algorithm, resulting in $\pi_{i+1}$ that maps states to actions.

As with policy search, this approach does not attempt to represent the cost-to-go function, which may allow it to scale to more complex problems. The policy can be compactly represented if the series of classifiers $\pi_1, \ldots, \pi_k$ are defined by a series of relatively short parameter vectors $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k$. The success of this algorithm depends upon the choice of classifier, sampling scheme, and the number of training examples.

## 6.6 DISCUSSION

Perhaps the primary strength of approaches based on dynamic programming and probabilistic conflict estimates is that they directly account for uncertainty in the future path of the aircraft. Of the approaches discussed in this section, only the policy search and nonstationary approximate policy iteration methods leverage models of uncertainty. For last-minute collision avoidance, accounting for sensor error and dynamic uncertainty is critically important. Many of the algorithms suggested in the literature that are based on the other approaches are intended to assist in air traffic control separation, where accounting for these uncertainties is less important because of the larger time scale involved.

# 7. FURTHER RESEARCH

There are several ways in which the simple collision avoidance model needs to be extended to handle realistic encounter scenarios. This section outlines how the existing model can be extended and discusses potential computational issues that may arise and how to address them. The main priorities for further research in the short term include extending the model to three spatial dimensions and equipped intruders, investigating model robustness, and exploring different policy representations and how they may impact certification.

## 7.1 THREE-DIMENSIONAL DYNAMICS

One way to extend the hypothetical collision avoidance model (Section 1.3) into three dimensions is to dispense of $\tau$ (time to horizontal conflict) and add the following variables:

- horizontal range to intruder $r$,
- bearing of intruder $\chi$,
- own ground speed $v_1$,
- intruder ground speed $v_2$,
- relative heading of intruder $\beta$,
- own turn rate $\dot{\psi}_1$, and
- intruder turn rate $\dot{\psi}_2$.

A point in the state space would then be an 11-dimensional tuple

$$(h, r, \chi, \dot{h}_1, \dot{h}_2, v_1, v_2, \beta, \dot{\psi}_1, \dot{\psi}_2, s_{\mathrm{RA}}),$$

which uniquely specifies (in a convenient coordinate system) the velocity vectors of both aircraft, the relative position of the intruder, and the RA response state. A probabilistic model can be used to specify the dynamics. Unfortunately, applying the dynamic programming approach suggested in Section 3 might be impractical because of the dimensionality of the state space. A grid with 21 edges per dimension would result in $3.5 \cdot 10^{14}$ discrete states.

One way to reduce the dimensionality of the state space is to decompose the problem into motion along the horizontal plane and motion along the vertical plane. The dynamics in the horizontal plane can be modeled as an uncontrolled Markov process over the variables $r$, $\chi$, $v_1$, $v_2$, $\beta$, $\dot{\psi}_1$, and $\dot{\psi}_2$. The dynamics in the vertical plane would be modeled as discussed in Section 1.3 over the variables $h$, $\tau$, $\dot{h}_1$, $\dot{h}_2$, and $s_{\mathrm{RA}}$. Transforming the 11-dimensional problem into these two problems of smaller dimension requires that the vertical rates and vertical separation be independent of the behavior in the horizontal plane. Prior TCAS studies have made this assumption [15, 76].

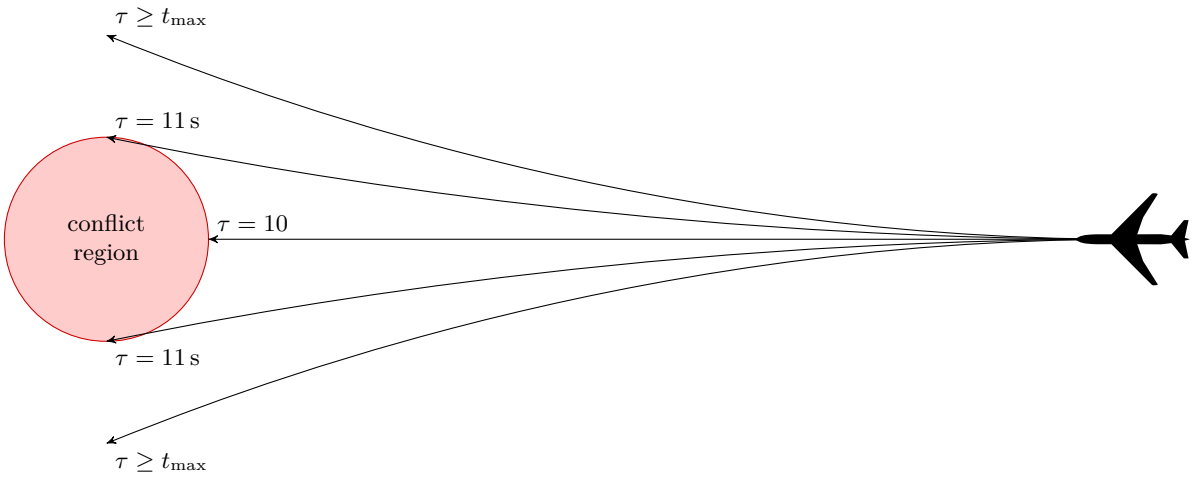



*Figure 28. Estimation of time to horizontal conflict.*

A distribution over $\tau$ can be inferred from the uncontrolled Markov process over $\mathbf{x}_{\text{horiz}} = (r, \chi, v_1, v_2, \beta, \dot{\psi}_1, \dot{\psi}_2)$. One way to estimate this distribution is to sample a large collection of trajectories starting from the current state (Figure 28). The variable $\tau$ may be discretized to one-second intervals up to the time horizon $t_{\max}$: $[0, 1), [1, 2), \ldots, [t_{\max}, \infty)$. A histogram over these bins can be inferred from the samples. As with conflict probability estimation (Section 4), importance sampling techniques can result in improved estimates of the distribution $\Pr(\tau \mid \mathbf{x}_{\text{horiz}})$.

Another way to estimate $\Pr(\tau \mid \mathbf{x}_{\text{horiz}})$ is to use dynamic programming, which may be done offline for the entire discrete state space. Appendix G provides details of an algorithm. During execution, computing $\Pr(\tau \mid \mathbf{x}_{\text{horiz}})$ for an arbitrary $\mathbf{x}_{\text{horiz}}$ would involve interpolating (Appendix C) distributions at nearby states. If the dimensionality of $\mathbf{x}_{\text{horiz}}$ proves to be computationally burdensome, it may be worth assuming constant ground speed and turn rate to bring the number of dimensions down to three from seven. Of course, dynamic programming would then need to be run online.

Once $\Pr(\tau \mid \mathbf{x}_{\text{horiz}})$ is known, the QMDP value method (Section 2.4) can be used to choose the best action

$$\arg\min_a \sum_\tau \Pr(\tau \mid \mathbf{x}_{\text{horiz}}) J^*((h, \tau, \dot{h}_1, \dot{h}_2, s_{\text{RA}}), a), \tag{33}$$

where

$$J^*(s, a) = c(s, a) + \sum_{s'} \Pr(s' \mid s, a) J^*(s'). \tag{34}$$

When $\tau \geq t_{\max}$, $J^*(s)$ is set to zero.

## 7.2 NONDETERMINISTIC PILOT RESPONSE

Kuchar and Drumm analyzed TCAS monitoring data in the Boston area from June 2005 to January 2006, and they observed the following:

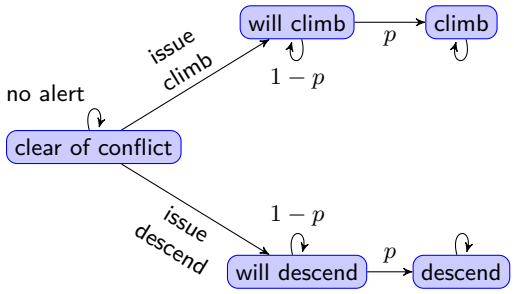



*Figure 29. Transition model for $s_{RA}$ where the response delay is given by a geometric distribution.*

> Examination of the data . . . indicates that only 13% of pilot responses met the assumption used by TCAS: pilot responses within 5 seconds and achieving a 1500 ft/min vertical rate. In 63% of the cases, the pilots maneuvered in the proper direction but were not as aggressive or prompt as TCAS assumed. Pilots maneuvered in the opposite direction to the RA in 24% of the cases. Some of these opposite responses are believed to be due to visual acquisition of the intruder aircraft and the pilot's decision that following the RA was not necessary. [4]

In the hypothetical collision avoidance problem, the pilot of the own aircraft always responds deterministically to RAs, which reduces the amount of uncertainty the collision avoidance system must account for when deciding when to issue an RA. If the model allows for late or opposite responses, then the optimal logic will likely issue alerts earlier.

The transition probabilities for the $s_{\text{RA}}$ variable would need to be altered to allow for nondeterministic responses. For example, instead of transitioning deterministically from clear of conflict to climb in 4 s when issuing a climb RA, the model might allow a transition to climb in 1 s or climb in 10 s or even never climb with some probability, for example.

Alternatively, the transition probabilities for the $s_{\text{RA}}$variable can be as shown in Figure 29. Once a climb or descend advisory is issued, $s_{\text{RA}}$ will go into a "holding state" (either will climb or will descend) from which it will transition to climbing or descending with some probability $p$ at each time step. The distribution over the time spent in the holding state is given by a geometric distribution with mean $1/p$. The mean total delay, which includes the step from clear of conflict to one of the holding states, is $1 + 1/p$. The probability $p$ can be chosen to provide the desired average delay. For example, $p = 0.25$ provides a 5 s delay on average. One helpful byproduct of this approach is that it reduces the size of the state space since explicit states are no longer required for delay in 4 s, . . . , delay in 1 s (as in Appendix A).

### 7.3 STRENGTHENING AND REVERSAL ADVISORIES

The hypothetical collision avoidance problem in this report does not allow the system to change the RA once it has been issued. However, it is critical that TCAS be able to either strengthen or

reverse RAs as the encounter develops because pilots often do not follow their RAs exactly or even at all. According to Kuchar and Drumm [4], the mid-air collision of a Russian Tu-154 and a DHL B-757 over Überlingen in 2002 may have been averted if TCAS had properly reversed the RA it had issued to the DHL aircraft.

The current version of TCAS incorporates reversal logic. According to TCAS monitoring data obtained from seven sensors since 2008, approximately 1% of RA enounters involve reversals. The logic in TCAS responsible for RA changes is rather complex in order to address issues identified in simulation and during operational use. Modeling collision avoidance as an optimization problem may result in logic that is more robust to unexpected events without requiring substantial engineering effort and the development of complex logic.

The MDP framework can be extended to allow strengthening and reversal. The behavior of the $s_{\text{RA}}$ variable (Figure A-1) would require adjustment and the number of actions available from non-clear-of-conflict states would need to be expanded. The cost function may also need adjustment, depending on whether it is desirable to penalize strengthening or reversing RAs.

The approach that involves making decisions based on the probability of conflict (Section 4) can also be extended to allow strengthening and reversal. If it is determined that a different RA provides a lower probability of conflict than the current RA, then the system would change the RA. Experiments would be able to reveal how well this approach compares to the MDP framework.

### 7.4 EQUIPPED INTRUDERS

This report has only considered intruders that are not equipped with a collision avoidance system. However, a future version of TCAS will have to interact with intruders equipped with the future and legacy versions of TCAS. Separate sublogics for each equipage category can be optimized according to different dynamic models.

Applying dynamic programming to problems with an intruder with the current version of TCAS may involve adding additional information to the state representation and modifying the transition model. However, for an intruder with this future collision avoidance system, the logic needs to be jointly optimized on both aircraft.

One way to jointly optimize the logic on both aircraft is to define the state variables relative to the aircraft with the lower Mode S address. (The current version of TCAS uses Mode S address to break ties in coordinated encounters.) The action space would be the set of possible pairs of resolution advisories for both aircraft: $\{(\mathsf{no\ alert}, \mathsf{no\ alert}), (\mathsf{climb}, \mathsf{descend}), \ldots\}$. The dynamic model would take into account the RAs of both aircraft. The cost function would be similar to before, except that the cost imposed for alerting may depend on whether one or both aircraft alert. It may be desirable to enforce the constraint that both aircraft alert simultaneously. The optimal joint policy can be found using dynamic programming.

When an aircraft equipped with this future system discovers an intruder with the same system, it will estimate the state relative to the aircraft with the lower Mode S address. It will then apply the policy computed using dynamic programming to determine the best pair $(a_1, a_2)$. If the own

aircraft has the lower Mode S address, it will execute action $a_1$; otherwise, it will execute action $a_2$. Ideally, both aircraft will have estimated the same exact state, agreed upon the same action pair, and executed their respective actions. However, it is possible that sensor noise results in the two aircraft believing they are in different states, which may result in disagreement over the action pairing. One way to mitigate this problem is to have the aircraft with the lower Mode S address dictate the action of both aircraft over the data link. The current version of TCAS uses a similar coordination scheme, but further research would be required to assess potential issues and vulnerabilities of such an approach.

The current version of TCAS provides a much lower risk ratio when both aircraft are equipped with TCAS [78]. It would be interesting to determine through Monte Carlo simulation how much better a jointly optimal logic performs when both aircraft follow their RA. It would also be interesting to determine how performance degrades when intruders do not follow their RA.

## 7.5 MODEL ROBUSTNESS ANALYSIS

The methods discussed in this report assume that the internal model of the system dynamics and sensor performance is correct. Before deploying a collision avoidance system that is optimized for a particular model, it is important to assess the risk of an inaccurate model. Monte Carlo analysis across a spectrum of gradually different models can reveal how inaccurate the internal model must be before performance is significantly degraded.

## 7.6 NON-GAUSSIAN DYNAMICS

The dynamic model used in this report assumes that accelerations are chosen according to a Gaussian distribution. However, the logic is evaluated using an encounter model with much more sophisticated dynamics represented by a dynamic Bayesian network derived from radar data. It may be worth experimenting with how much the policy improves if a more sophisticated dynamic model is used for planning. If there is little to gain, then it may be wise to continue using the simpler Gaussian model. However, if there is much to gain, then it could be worth adopting a more complex model.

## 7.7 ADDITIONAL RESOLUTION ADVISORIES

TCAS III, which is no longer under development and has not been deployed, was intended as a more capable version of TCAS II that incorporated horizontal RAs such as "turn left." Experiments in the late 1980s indicated that horizontal RAs were expected to provide a modest gain in safety while significantly lowering the alert rate [107]. A more recent study showed that horizontal RAs can make TCAS more efficient in some situations, such as crossing situations that can result in reversal RAs [108]. Pilots generally view the concept of horizontal RAs favorably, although the views of controllers tend to be less favorable [108].

Introducing horizontal RAs to the MDP formulation would require expanding the model into three spatial dimensions. An MDP would be required to model the horizontal dynamics instead of the uncontrolled Markov process discussed in Section 7.1. The action space would need to be expanded to include horizontal RAs. The computational impact may be significant, but simulation using an encounter model could quantify the anticipated benefit of horizontal RAs.

## 7.8 LEVERAGING INTENT INFORMATION

Automatic Dependent Surveillance-Broadcast (ADS-B) enables aircraft to broadcast their position and other relevant information to the ground and nearby traffic. It is expected that a future version of TCAS will leverage this technology (at least to some degree) to improve collision avoidance. Because ADS-B can broadcast intent information, it may be possible to reduce the uncertainty in the future state of the aircraft, thereby lowering the false alert rate [109]. Since pilots do not always follow their intended flight plan, the dynamic model used for planning must capture intent deviation [110]. Experiments can confirm whether leveraging intent information significantly lowers the rate of unnecessary alerts.

## 7.9 MULTIPLE INTRUDERS

As to be expected, the multi-threat logic is perhaps one of the most complex components of TCAS. This part of the logic has not been the focus of the same rigorous testing as the single-threat logic, due in part to the rarity of multi-threat events. It was not until recently that an encounter model based on operational data was developed for multi-threat situations [111]. Preliminary results indicate that the current version of TCAS performs reasonably well against multiple intruders.

Expanding the MDP model to capture the behavior of multiple intruders would significantly increase the dimensionality of the state space. The fitted value iteration approach with grid-based interpolation of the value function (Section 3) is unlikely to scale well to multiple intruders unless some approximations are made. Methods that use conflict probability estimates (Section 4) or other online methods will likely scale better in multi-threat situations. Although multi-threat situations contribute little to the overall risk associated with TCAS because of their rarity, it is important to ensure that any future logic be robust to encounters with multiple intruders.

## 7.10 LOGIC REPRESENTATION

This report has focused on how to compute the optimal policy, but it has not dealt with the issue of how the optimal policy will be used or represented. One option is to not use the optimal policy directly as the future TCAS logic, but instead use the optimal policy as a tool to aid engineers designing the new logic. The optimal policy can help justify parameter settings, such as DMOD and ALIM. Human-engineered logic and the optimal logic can be compared in simulation. If the human-engineered logic performs significantly worse in some situations compared to the optimal logic, then those situations can be inspected and the logic revised to better match the optimal logic.

The output of the effort would be human-engineered pseudocode that can be certified like previous versions of TCAS and then implemented by manufacturers.

One advantage of using human-designed pseudocode to represent the logic is that there is already experience and familiarity with certifying pseudocode with prior versions of TCAS. Although the pseudocode of the current version of TCAS is remarkably complex, an engineer could read through the pseudocode and gain some understanding of how the logic works. Disadvantages of using the optimal policy merely as a design aid include sacrificing performance and extending the development process.

An alternative is to use the optimal policy directly. For example, the optimal cost-to-go function could be represented as a table of values, as discussed in Section 3, and the logic would involve performing the necessary calculations on these values to select the action with the lowest expected cost. The certification process would require certifying the table and the limited amount of pseudocode defining the required calculations. Instead of having the logic specified entirely in pseudocode, this approach would shift most of the complexity of the logic into a large numerical table. The certification of the table and accompanying pseudocode would be similar to the certification process for prior versions of TCAS. The logic would need to be rigorously evaluated on millions of encounters in simulation to ensure the logic is safe and meets operational requirements.

There are some advantages of a tabular representation over the pseudocode representation of prior TCAS logics. For example, the table can be easily updated as the airspace changes (which is anticipated over the next 20 years) without having to revise implementations of pseudocode. A tabular representation would also reduce the amount of effort required by manufacturers to implement the logic and validate that their implementations meet the logic specification.

There are several alternatives to a tabular representation of the cost-to-go function. One alternative is to use regression to approximate the cost-to-go function using a parametric function. This can be done using either an online or offline dynamic programming solution method. Another alternative is to dispense with representing the cost-to-go function and switch to representing the policy directly using a classifier trained on a large collection of samples [61, 62]. A decision tree would be one way to represent the classifier [112].

Depending on the representation, the decision-making process of the logic may not be entirely transparent. Although a decision tree can be easily inspected and perhaps understood, a table of values would not provide much insight into how the system made its decision. However, the specification of how the representation was created would provide an understanding of the rationale of the logic. Additionally, studying the policy plots (Section 3.4) can provide some insight into the behavior of the logic. Because it is important that the behavior of the logic be understood as well as possible due to its safety-critical nature, future research will need to further investigate ways of visualizing and understanding the behavior of the system.

This page intentionally left blank.

# 8. CONCLUSIONS

As the airspace evolves with the introduction of new air traffic control procedures and surveillance systems, it is likely that the TCAS II threat detection and resolution logic will require modification to meet safety and operational requirements. Due to the complexity of the logic, modifying the logic may require significant engineering effort.

This report suggests a new approach to TCAS logic development where the engineering effort is focused on developing models, allowing computers to optimize the logic according to agreed-upon performance metrics. Because models of sensor characteristics, pilot response behavior, and aircraft dynamics can be constructed from operational data, they should be straightforward to justify and vet within the safety community. The optimization of the logic according to these models would be done using principled techniques that are well established in theory and practice over the past 50 years. The performance metrics are based on quantities that have long been used by the aviation safety community.

The objective of this report was to connect this concept of TCAS logic optimization to the existing literature on model-based optimization, not to develop a particular conflict resolution algorithm. Problems involving sequential decision making in a dynamic environments are typically modeled by Markov decision processes, where the state at the next decision point depends probabilistically on the current state and the chosen action. Assuming some objective measure of cost, the best action from the current state is the one that minimizes the expected future cost. Computing the optimal action from all possible states requires a process known as dynamic programming, which has been used for a wide variety of problems in computer science.

To illustrate some of the key concepts of how dynamic programming might be applied to TCAS logic optimization, this report used a simple encounter model and evaluated it in simulation. Because the model is defined by continuous variables, it is necessary to use some approximation methods. This report explored interpolation-based methods and evaluated the resulting logic using various performance metrics.

This report identified some of the issues with applying a dynamic programming approach. One issue is the scalability of existing solution methods to higher dimensions. Adding additional dimensions to the grid-based representation used in this report results in an exponential increase in memory and computational requirements. As discussed in this report, there are several methods worth exploring that may address these issues.

One approach that may scale well to higher dimensions involves using conflict probability estimates to decide when to issue resolution advisories, as has been suggested by others. Although this approach will not result in the optimal solution, it may approximate the optimal logic well. One of the challenges of this approach is estimating the probability of a rare event, but this report discussed a variety of techniques for estimating probability of conflict that are more accurate than direct Monte Carlo sampling.

Besides methods based on dynamic programming and conflict probability estimates, several other approaches have been suggested in the literature. This report discussed some of these meth-

ods, highlighting their strengths and weaknesses. One of the primary strengths of the dynamic programming approach over the other methods is that it directly leverages models of sensor error and aircraft behavior to find the optimal logic. Future simulation studies can determine how well these alternative methods perform compared to the optimal policy found through dynamic programming.

Experiments using the simple encounter model indicate that dynamic programming is a promising approach. However, further work will be required to extend the approach to three spatial dimensions. This report outlined methods for handling three spatial dimensions while keeping the memory and computational requirements tractable. Although computational requirements could limit the success of applying dynamic programming to TCAS logic development, this report has suggested ways to help manage the requirements for memory and computation.

This work has focused primarily on the computational aspect of optimizing collision avoidance logic, but there are other issues, such as certification, that require further study. If this new approach to developing logic is simply used as an aid to engineers who are developing or revising collision avoidance pseudocode, then the use of these methods would have little impact on the certification process. However, if the logic produced by dynamic programming or some other automated process is to be used directly in a future version of TCAS, then the certification process may be somewhat different. The core of the certification process will be the same, involving rigorous simulation studies and flight tests to prove safety and demonstrate operational acceptability. However, the vetting of the logic itself will involve more than just studying the logic that will be deployed on the system. Depending on the representation of the logic, it may not be directly comprehensible by an engineer. Therefore, confidence would have to be established in the safety community that the methods used to generate the logic are sound. This report represents a first step in justifying an automated approach for generating optimized TCAS logic.

# APPENDIX A
# HYPOTHETICAL COLLISION AVOIDANCE DYNAMICS

Section 1.3 introduced a hypothetical collision avoidance problem that was used as an example to demonstrate the logic discussed in this report. This appendix provides a mathematical specification of the underlying model. The model uses a state representation to encode all the properties of the system that are of interest at any given time. The state at time $t$ is a vector

$$\mathbf{x}(t) = \begin{bmatrix} h(t) \\ \tau(t) \\ \dot{h}_1(t) \\ \dot{h}_2(t) \\ s_{\mathrm{RA}}(t) \end{bmatrix}, \tag{A-1}$$

where $h$ is the altitude of the intruder relative to own, $\tau$ is the time to closest approach, $\dot{h}_1$ is own vertical rate, and $\dot{h}_2$ is the intruder vertical rate. The magnitudes of $\dot{h}_1$ and $\dot{h}_2$ are limited to $L = 2500\,\mathrm{ft/min}$. The $s_{\mathrm{RA}}$ variable keeps track of the RA that has been issued so that the proper 0.25-g acceleration can be applied after a five-second delay. As Figure A-1 illustrates, 11 discrete states are required.

The system dynamics are governed by the following hybrid, discrete-time Markov model:

$$\mathbf{x}(t + \Delta t) = \mathbf{f}(\mathbf{x}(t), \mathbf{w}(t), a(t)), \tag{A-2}$$

where

$$\mathbf{w}(t) = \begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix} \tag{A-3}$$

is a random variable representing the noise in the vertical rates of the aircraft, and $a(t)$ is the action performed at time $t$. The action can take on three possible values: no alert, issue descend, and issue climb. Once an action other than no alert is taken, subsequent values of $a$ have no effect upon the evolution of the system. Hence, the model allows for at most one resolution advisory which, if issued, remains in effect for the remainder of the scenario.

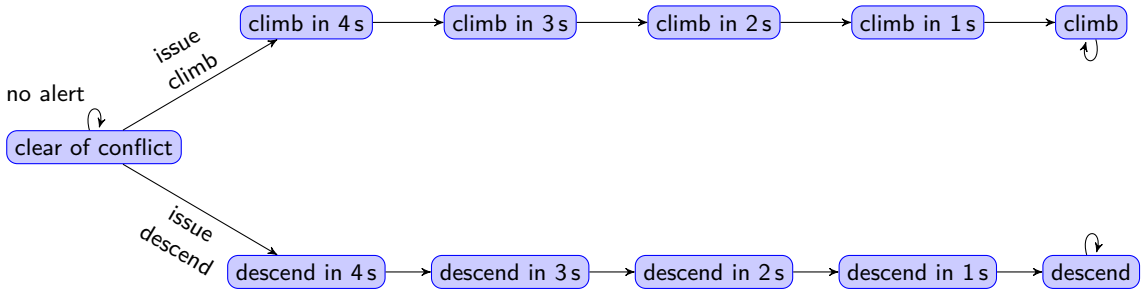



*Figure A-1. The variable $s_{RA}$ tracks which RA has been issued and the delay.*

The noise is a zero-mean white Gaussian process with time-invariant covariance $R$, that is, $\mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}, R)$. It is assumed that the noise in the vertical rates of the own aircraft and intruder, $w_1(t)$ and $w_2(t)$, respectively, are uncorrelated and that they both have a standard deviation of $1\,\mathrm{ft/s^2}$. The process noise enables the model to capture the stochastic nature of aircraft behavior.

The equations of motion can be written as

$$h(t+\Delta t) = h(t) + (\dot{h}_2(t) - \dot{h}_1(t))\Delta t + \frac{1}{2}(\ddot{h}_2(t) - \ddot{h}_1(t))\Delta t^2, \tag{A-4}$$

$$\tau(t+\Delta t) = \tau(t) - \Delta t, \tag{A-5}$$

$$\dot{h}_1(t+\Delta t) = \phi_L\Big(\dot{h}_1(t) + \ddot{h}_1(t)\Delta t\Big), \tag{A-6}$$

$$\dot{h}_2(t+\Delta t) = \phi_L\Big(\dot{h}_2(t) + \ddot{h}_2(t)\Delta t\Big), \tag{A-7}$$

where the saturation function $\phi_L(y) = \max(-L, \min(L, y))$ ensures that the magnitude of either vertical rate never exceeds the model parameter $L$. The time step $\Delta t$ is set to one second. The acceleration noise values $w_1$ and $w_2$ are sampled from $\mathcal{N}(\mathbf{0}, R)$ every second. The applied vertical accelerations are given by

$$\ddot{h}_1(t) = \begin{cases} -0.25\ \mathrm{g} & \text{if } s_{\mathrm{RA}}(t) = \textsf{descend} \text{ and } \dot{h}_1(t) > -1500\,\mathrm{ft/min}, \\ +0.25\ \mathrm{g} & \text{if } s_{\mathrm{RA}}(t) = \textsf{climb} \text{ and } \dot{h}_1(t) < +1500\,\mathrm{ft/min}, \\ w_1(t) & \text{otherwise}, \end{cases} \tag{A-8}$$

$$\ddot{h}_2(t) = w_2(t). \tag{A-9}$$

# APPENDIX B
# ANALYTIC APPROXIMATION

Section 4.1 discusses an analytic approximation to the probability of conflict for the model of Appendix A. This appendix is an overview of the linear-Gaussian approximation to the dynamics that makes this analytic approximation possible.

The system dynamics are governed by the hybrid, discrete-time Markov model of Equation A-2, repeated here for convenience:

$$\mathbf{x}(t+\Delta t) = \mathbf{f}(\mathbf{x}(t), \mathbf{w}(t), a(t)), \tag{B-1}$$

where

$$\mathbf{w}(t) = \begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix} \tag{B-2}$$

is a random variable representing the noise in the vertical rates of the two aircraft, and $a(t)$ is the action performed at time $t$. The dynamics can be approximated by the Gaussian system

$$\mathbf{x}(t+\Delta t) = A(\Delta t)\mathbf{x}(t) + G(\Delta t, \mathbf{x}(t))\mathbf{w}(t) + B(\Delta t)\mathbf{u}(t, \mathbf{x}(t)), \tag{B-3}$$

where

$$\mathbf{u}(t, \mathbf{x}(t)) = \begin{bmatrix} \ddot{h}_{1_{\text{cmd}}}(t) \\ -\Delta t \end{bmatrix} \tag{B-4}$$

is the control vector representing the control in the own vertical rate, $\ddot{h}_{1_{\text{cmd}}}$, and the rate at which the time to closest approach $\tau$ decreases each time step, $-\Delta t$. When the pilot is responding to an RA (after a five-second delay), $\ddot{h}_{1_{\text{cmd}}}$ is a 0.25-g acceleration in the RA direction; otherwise, $\ddot{h}_{1_{\text{cmd}}}$ is zero. Hence, $\ddot{h}_{1_{\text{cmd}}}$ is a deterministic function of $\mathbf{x}(t)$.

Outside of an RA response, the system dynamics (excluding $s_{\text{RA}}$) can be written

$$\begin{aligned} \begin{bmatrix} h(t+\Delta t) \\ \tau(t+\Delta t) \\ \dot{h}_1(t+\Delta t) \\ \dot{h}_2(t+\Delta t) \end{bmatrix} = &\begin{bmatrix} 1 & 0 & -\Delta t & \Delta t \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h(t) \\ \tau(t) \\ \dot{h}_1(t) \\ \dot{h}_2(t) \end{bmatrix} + \begin{bmatrix} -\frac{1}{2}\Delta t^2 & \frac{1}{2}\Delta t^2 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix} \\ &+ \begin{bmatrix} -\frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 \\ \Delta t & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{h}_{1_{\text{cmd}}}(t) \\ -\Delta t \end{bmatrix}. \end{aligned} \tag{B-5}$$

During the RA response, there is no contribution from the noise $w_1(t)$. Therefore,

$$G(\Delta t, \mathbf{x}(t)) = \begin{bmatrix} 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 \\ 0 & 0 \\ 0 & \Delta t \end{bmatrix} \tag{B-6}$$

during RA response. Regardless of whether the pilot is responding to an RA, the mean of $\mathbf{x}(t+\Delta t)$ is

$$\mathrm{E}[\mathbf{x}(t+\Delta t)] = A(\Delta t)\mathrm{E}[\mathbf{x}(t)] + B(\Delta t)\mathrm{E}[\mathbf{u}(t,\mathbf{x}(t))], \tag{B-7}$$

and the covariance of $\mathbf{x}(t+\Delta t)$ is

$$\mathrm{cov}[\mathbf{x}(t+\Delta t)] = A(\Delta t)\mathrm{cov}[\mathbf{x}(t)]A(\Delta t)^T + G(\Delta t,\mathbf{x}(t))RG(\Delta t,\mathbf{x}(t))^T. \tag{B-8}$$

Both of these recursions depend upon the value of the random variable $\mathbf{x}(t)$. To obtain analytic approximations $\widehat{\mu}$ and $\widehat{P}$, the approximate mean and covariance of $\mathbf{x}(t)$, the approximate mean $\widehat{\mu}(t)$ is substituted for the true random variable $\mathbf{x}(t)$, yielding recursions

$$\widehat{\mu}(t+\Delta t) = A(\Delta t)\widehat{\mu}(t) + B(\Delta t)\mathbf{u}(t,\widehat{\mu}(t)), \tag{B-9}$$

$$\widehat{P}(t+\Delta t) = A(\Delta t)\widehat{P}(t)A(\Delta t)^T + G(\Delta t,\widehat{\mu}(t))RG(\Delta t,\widehat{\mu}(t))^T. \tag{B-10}$$

Hence, the system described by Equations B-9 and B-10 is a linear-Gaussian system that switches between two modes, no-RA execution mode and RA execution mode. The only difference between the two is a change in the matrix $G$.

This analytic solution is only an approximation. The actual dynamics are not linear-Gaussian for two reasons. First, the vertical rate saturates at $\pm 2500\,\mathrm{ft/min}$, causing the random accelerations to affect the state in a generally nonlinear fashion. Second, accelerations in response to RAs cause the own aircraft to transition deterministically, but this occurs only when the vertical rate is outside the target range of the RA, a nonlinear dependence on the current state. Representing the distribution of the state as a multivariate normal fails to capture exactly how the state evolves differently in different regions of the state space.

The method described here is only one of several approaches for approximately propagating Gaussian distributions through nonlinear system dynamics. In particular, the sigma-point sampling technique, described in Appendix D, offers an alternative means of transforming means and covariances that may provide better results [113].

# APPENDIX C
# INTERPOLATION METHODS

Interpolation can be viewed as approximating an unknown function $f$ given only values of this function at a finite set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n$. This is of particular interest because fitted value iteration (Section 3.1) requires interpolating between estimates of the cost-to-go function at a set of discrete states. There are many different interpolation schemes [114], but this appendix focuses on the class of interpolation functions of the form

$$g(\mathbf{x}) = \sum_{k=1}^{n} \beta_k(\mathbf{x}) f(\mathbf{x}_k), \tag{C-1}$$

where $\beta_k$ is a weighting function, such that $\sum_{k=1}^{n} \beta_k(\mathbf{x}) = 1$. Generally, $\beta_k(\mathbf{x})$ should not decrease as the distance between $\mathbf{x}_k$ and $\mathbf{x}$ increases. There are several ways to define the weighting function $\beta$, as discussed in this appendix. Figure C-2 compares the various interpolation methods on a two-dimensional problem.

## C.1 NEAREST-NEIGHBOR INTERPOLATION

The simplest approach is to assign all weight to the closest discrete state, resulting in a piecewise constant function $g$. Another approach, which can result in a smoother $g$, involves finding the $k$-nearest discrete states to $\mathbf{x}$ and assigning weight $1/k$ to each and zero to all other states. Figure C-2(a) is an illustration of nearest-neighbor interpolation in two dimensions. Each black cross represents a data point, and each color represents a different function value. Observe that the estimates obtained by nearest-neighbor interpolation are particularly coarse, the interpolating function being piecewise defined.

## C.2 MULTILINEAR INTERPOLATION

An alternative approach is to use multilinear interpolation over the box (also called a hyperrectangle or orthotope) in the grid that encloses the point $\mathbf{x}$ as shown by the notional diagram in Figure C-1. The weights of the states at the vertices of the box are related to how close they are to $\mathbf{x}$. The formula for calculating the weights in one dimension is

$$g(x) = \underbrace{(1 - \frac{x - x_1}{x_2 - x_1})}_{\beta_1(x)} f(x_1) + \underbrace{(1 - \frac{x_2 - x}{x_2 - x_1})}_{\beta_2(x)} f(x_2), \tag{C-2}$$

where $x_1$ is the vertex to the left of $x$ and $x_2$ is the vertex to the right of $x$. Multilinear interpolation is a multidimensional generalization of the linear interpolation of Equation C-2. The two-dimensional example in Figure C-1 results in the following estimate for $g(\mathbf{x})$:

$$g(\mathbf{x}) = (1/8)f(\mathbf{x}_1) + (1/8)f(\mathbf{x}_2) + (3/8)f(\mathbf{x}_3) + (3/8)f(\mathbf{x}_4). \tag{C-3}$$

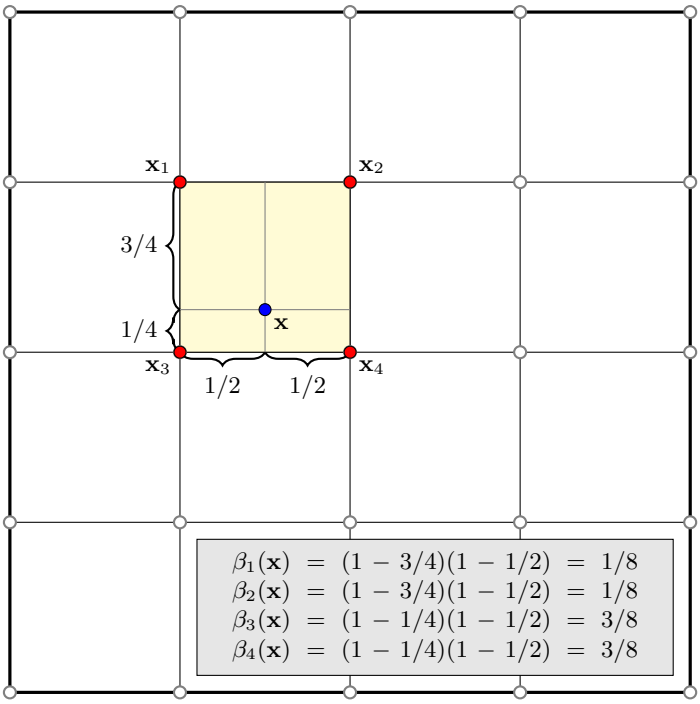



*Figure C-1. Multilinear interpolation.*

Figure C-2(b) is an illustration of multilinear interpolation in two dimensions. Although the function is not technically smooth, there is a greater amount of gradation than nearest-neighbor interpolation.

## C.3 SIMPLEX-BASED INTERPOLATION

One potential issue with multilinear interpolation is that the number of vertices used for interpolation grows exponentially with the dimensionality of the state space. If $d$ is the number of dimensions, multilinear interpolation can use up to $2^d$ vertices to estimate the cost-to-go function for a single point. As suggested by Davies [115], an alternative is to use simplex-based interpolation. In the simplex method, the boxes are broken into $d!$ multidimensional triangles, called simplexes, according to the Coxeter-Freudenthal-Kuhn triangulation [116]. Instead of interpolating over a $d$-dimensional box with up to $2^d$ vertices, the simplex-based method interpolates over a simplex defined by up to $d + 1$ vertices. Hence, the simplex method scales linearly instead of exponentially with the dimensionality of the state space. However, multilinear interpolation can provide higher quality estimates that can lead to better policies for the same grid resolution.

Figure C-2(c) is an illustration of simplex interpolation in two dimensions. Simplex interpolation provides higher quality estimates than nearest-neighbor interpolation while using less data points than multilinear interpolation.

## C.4 LOCAL LAGRANGE INTERPOLATION

Lagrange interpolation is often used to interpolate functions of one-dimensional variables. Namely, if $(x_k, f(x_k)), k = 1, 2, \ldots, n$ are the training data, the Lagrange base polynomials are defined as

$$L_k(x) = \prod_{i=1, i\neq k}^{n} \frac{x - x_i}{x_k - x_i}, \tag{C-4}$$

from which the Lagrange polynomial that interpolates the data can be constructed as

$$g(x) = \sum_{k=1}^{n} L_k(x) f(x_k). \tag{C-5}$$

This is a one-dimensional interpolation scheme that is often applied globally, i.e., all data points are used to construct the polynomial C-5. This is of limited value for interpolation of the cost-to-go function of Section 3 because the state space is very large. Luo [117] proposed a local multivariate Lagrange interpolation scheme. Let $\mathbf{P} = (\mathbf{x}_k, f(\mathbf{x}_k)), k = 1, 2, \ldots, n$ represent the training data set, and let $\mathbf{p} = (\mathbf{x}_k, f(\mathbf{x}_k)), k = 1, 2, \ldots, N$ represent a subset of these data, where $\mathbf{p} \in \mathbf{P}$, $N \leq n$. Then the Lagrange base polynomials are

$$L_k(\mathbf{x}) = \prod_{i=1, i\neq k}^{N} \frac{(\mathbf{x} - \mathbf{x}_i)^T (\mathbf{x}_k - \mathbf{x}_i)}{(\mathbf{x}_k - \mathbf{x}_i)^T (\mathbf{x}_k - \mathbf{x}_i)}. \tag{C-6}$$

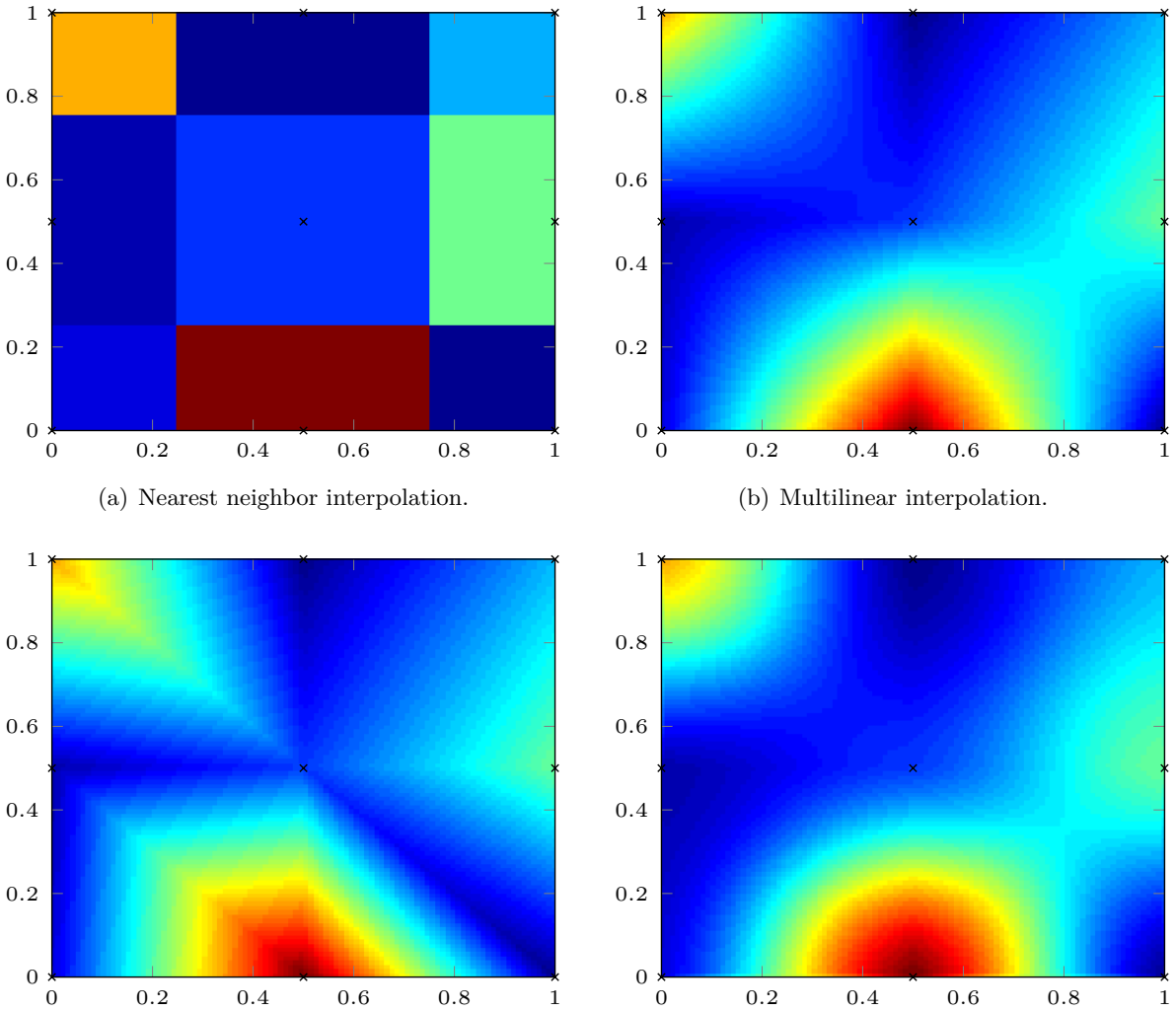
The Lagrange interpolating polynomial is

$$g(\mathbf{x}) = \sum_{k=1}^{N} \phi_k(\mathbf{x}) f(\mathbf{x}_k), \tag{C-7}$$

where the normalized Lagrange base polynomials are

$$\phi_k(\mathbf{x}) = \frac{L_k(\mathbf{x})}{\sum_{i=1}^{N} L_i(\mathbf{x})}. \tag{C-8}$$

Figure C-2(d) is an illustration of local Lagrange interpolation in two dimensions. The function value at each point was interpolated using the data at the corners of the smallest square in which the point is contained, thus requiring the same number of data points as multilinear interpolation. However, the performance is noticeably different.




(a) Nearest neighbor interpolation.

(b) Multilinear interpolation.

(c) Simplex interpolation.

(d) Local Lagrange interpolation.

*Figure C-2. Comparision of interpolation methods in two dimensions. The data points are indicated with black crosses.*

# APPENDIX D
# SAMPLING METHODS

If $\mathbf{X}$ is a (possibly multidimensional) random variable with density $p$ and $f$ is a function of $\mathbf{X}$, then the expected value of the random variable $f(\mathbf{X})$ is

$$\mathrm{E}_p[f(\mathbf{X})] = \int p(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x}. \tag{D-1}$$

Calculating the expected value for a function of a random variable arises in several contexts:

- **Applying the Bellman operator:** The random variable $\mathbf{X}$ represents the state at the next decision stage, and $f$ is the cost-to-go function. (Section 3.1)
- **Estimating the probability of conflict from the current state:** The random variable $\mathbf{X}$ represents the future trajectories of the aircraft, and $f$ indicates whether an encounter occurs. (Section 4.3)
- **Evaluating the performance of a system:** The random variable $\mathbf{X}$ represents an encounter, and $f$ is the performance metric used for evaluation (e.g., conflict probability). (Section 5.4)

In general, it is not possible to evaluate the integral in Equation D-1 analytically. This section outlines sampling methods for estimating $\mathrm{E}_p[f(\mathbf{X})]$.

## D.1 DIRECT MONTE CARLO

The direct Monte Carlo estimate of $\mathrm{E}_p[f(\mathbf{X})]$ is given by

$$\frac{1}{N} \sum_{k=1}^{N} f(\mathbf{x}^{(k)}), \tag{D-2}$$

where $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$ are sampled directly from $p$. Although this estimate is unbiased, direct Monte Carlo may require a large number of samples to provide adequate accuracy [118].

## D.2 IMPORTANCE SAMPLING

Importance sampling is a technique for improving the accuracy of estimates [73,119]. In importance sampling, one draws samples from a new distribution $q$, called the proposal distribution, that favors

important samples, i.e., those that contribute most to $p(\mathbf{x})f(\mathbf{x})$. Equation D-1 can be written as

$$\begin{aligned}\mathrm{E}_p[f(\mathbf{X})] &= \int p(\mathbf{x})f(\mathbf{x})\,d\mathbf{x} \\ &= \int q(\mathbf{x})\frac{p(\mathbf{x})}{q(\mathbf{x})}f(\mathbf{x})\,d\mathbf{x} \\ &= \mathrm{E}_q\left[\frac{p(\mathbf{X})}{q(\mathbf{X})}f(\mathbf{X})\right], \end{aligned} \tag{D-3}$$

where $p(\mathbf{x})/q(\mathbf{x})$ is called the likelihood ratio. The unbiased importance sampling estimator of $f$ is

$$\frac{1}{N}\sum_{k=1}^{N} f(\mathbf{x}^{(k)})\frac{p(\mathbf{x}^{(k)})}{q(\mathbf{x}^{(k)})}, \tag{D-4}$$

where $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$ are samples from $q$. In general, the proposal distribution must satisfy $q(\mathbf{X}) = 0 \Rightarrow f(\mathbf{X})\,p(\mathbf{X}) = 0$ to be admissible. The optimal proposal distribution $q^*$ is proportional to $|f(\mathbf{x})|p(\mathbf{x})$ [120].

### D.3 SIGMA-POINT SAMPLING

Sigma-point sampling involves generating deterministically-chosen samples that capture statistics of the distribution [113]. If $\mathbf{X}$ is $m$ dimensional, there are $2m+1$ sample points $\mathbf{x}^{(k)}$ with associated weights $w^{(k)}$ defined by

$$\begin{array}{lll} \mathbf{x}^{(k)} = \boldsymbol{\mu} & w^{(k)} = \frac{\kappa}{m+\kappa} & k = 0, \\ \mathbf{x}^{(k)} = \boldsymbol{\mu} + (\sqrt{(m+\kappa)\boldsymbol{\Sigma}})_k & w^{(k)} = \frac{1}{2(m+\kappa)} & k = 1, \ldots, m, \\ \mathbf{x}^{(k)} = \boldsymbol{\mu} - (\sqrt{(m+\kappa)\boldsymbol{\Sigma}})_{k-m} & w^{(k)} = \frac{1}{2(m+\kappa)} & k = m+1, \ldots, 2m, \end{array} \tag{D-5}$$

where $\kappa$ is a scaling parameter and $(\sqrt{(m+\kappa)\boldsymbol{\Sigma}})_k$ is the $k$th row of the matrix square root of $(m+\kappa)\boldsymbol{\Sigma}$. The mean and covariance of $\mathbf{X}$ are given by $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, respectively. These sample points are called sigma points. These weighted sigma points capture the true mean and covariance of $\mathbf{X}$. The sigma-point estimate is given by

$$\sum_{k=1}^{N} f(\mathbf{x}^{(k)})w^{(k)}, \tag{D-6}$$

where $N = 2m + 1$ [113, 121].

# APPENDIX E
# PROPOSAL DISTRIBUTIONS

This appendix presents the importance sampling proposal distributions discussed in Section 4. An effective proposal distribution favors sample trajectories that result in conflict but that are still likely to occur according to the system dynamics.

In the following discussion, let $\Pr(\mathrm{C} \mid \mathbf{x}, a)$ represent the probability that a conflict will occur from state $\mathbf{x}$ assuming action $a$ is continuously executed, and let $\widehat{\Pr}(C \mid \mathbf{x}, a)$ represent an estimate of that probability. Let trajectory $T = (\mathbf{x}(t_0), \ldots, \mathbf{x}(t_K))$ be a random variable representing a trajectory that is produced by starting at state $\mathbf{x}(t_0)$ and executing action $a$ until CPA. Let $W = (\mathbf{w}(t_0), \ldots, \mathbf{w}(t_{K-1}))$ be a random variable representing the noise at each time step along the trajectory. The noise $W$ maps uniquely to a trajectory $T$ according to Equation A-2. Let $p(T)$ and $p(W)$ represent the density of $T$ and of $W$, respectively. Define $C(T)$ as the conflict indicator function:

$$C(T) = \begin{cases} 1 & \text{if } T \text{ results in a conflict,} \\ 0 & \text{otherwise.} \end{cases} \tag{E-1}$$

The probability of conflict is

$$\Pr(\mathrm{C} \mid \mathbf{x}, a) = \mathrm{E}_p[C(T)] = \mathrm{E}_q\left[C(T)\,\frac{p(W)}{q(W)}\right], \tag{E-2}$$

for which the unbiased importance sampling estimator (Appendix D.2) is

$$\widehat{\Pr}(\mathrm{C} \mid \mathbf{x}, a) = \frac{1}{N_t}\sum_{i=1}^{N_t} C(T^{(i)})\,\frac{p(W^{(i)})}{q(W^{(i)})}, \tag{E-3}$$

where $W^{(1)}, \ldots, W^{(N_t)}$ are independent samples from $q$.

The density $p(W)$ can be written explicitly as

$$p(W) = p(\mathbf{w}(t_0))\,p(\mathbf{w}(t_1)) \cdots p(\mathbf{w}(t_{K-1})) = \prod_{k=0}^{K-1} \frac{1}{2\pi|R|^{1/2}} e^{-\frac{1}{2}\mathbf{w}(t_k)^T R^{-1}\mathbf{w}(t_k)}, \tag{E-4}$$

where $R$ is the covariance of the noise. Moreover, the proposal distribution $q(W)$ can be written as

$$q(W) = q(\mathbf{w}(t_0) \mid \mathbf{x}(t_0))\,q(\mathbf{w}(t_1) \mid \mathbf{x}(t_1)) \cdots q(\mathbf{w}(t_{K-1}) \mid \mathbf{x}(t_{K-1})). \tag{E-5}$$

The importance sampling estimator for $\Pr(\mathrm{C} \mid \mathbf{x}, a)$ becomes

$$\widehat{\Pr}(\mathrm{C} \mid \mathbf{x}, a) = \frac{1}{N_t}\sum_{i=1}^{N_t} C(T^{(i)}) \prod_{k=0}^{K-1} \frac{p(\mathbf{w}^{(i)}(t_k))}{q(\mathbf{w}^{(i)}(t_k) \mid \mathbf{x}^{(i)}(t_k))}, \tag{E-6}$$

where $\mathbf{w}^{(i)}(t_k)$ is the noise at time $t_k$ for sample trajectory $i$ and $\mathbf{x}^{(i)}(t_k)$ is the state at time $t_k$ for sample trajectory $i$. An effective proposal distribution $q(\mathbf{w}^{(i)}(t_k) \mid \mathbf{x}^{(i)}(t_k))$ from which to sample the noise at each time step is one in which the sample trajectory resulting from the noise will result

in a conflict with high probability. This process is known as sequential, or dynamic, importance sampling. The proposal distribution is state-dependent because the distribution from which to sample the noise to produce conflicts is a function of how far away the state is from conflict at each time.

The following sections describe several proposal distributions. In all of the descriptions, let $\mathbf{x}(t)$ represent the state at time $t$ in a sample trajectory for which a proposal distribution $q(\mathbf{w}(t) \mid \mathbf{x}(t))$ is desired.

## E.1 CONSTANT ACCELERATION PROPOSAL

If $\mathbf{x}(t)$ is not a state corresponding to an RA execution, both ownship and intruder accelerations can be applied to artificially force the next state closer to a conflict. This process continues until CPA. The resulting trajectory should result in a conflict with high probability. If, on the other hand, $\mathbf{x}(t)$ is a state corresponding to an RA execution, the ownship acceleration is dictated by the issued RA, and therefore only the intruder acceleration can be changed to cause a conflict.

A conflict can be induced by applying acceleration at each time step that reduces the projected vertical separation at CPA to zero. A more sophisticated way of inducing conflict is to force the trajectory to reach the vertical separation most likely to occur while executing the action but that still qualifies as a conflict. More explicitly, let $h_{\text{proj}}$ represent the projected vertical separation at CPA achieved by the noiseless trajectory starting from $\mathbf{x}(t)$ and executing action $a$. Then the targeted vertical separation $h_{\text{target}}$ should be 100 ft if $h_{\text{proj}} > 100\,\text{ft}$, $-100\,\text{ft}$ if $h_{\text{proj}} < -100\,\text{ft}$, and $h_{\text{proj}}$ otherwise. That is,

$$h_{\text{target}} = \begin{cases} +100\,\text{ft} & \text{if } h_{\text{proj}} > 100\,\text{ft}, \\ -100\,\text{ft} & \text{if } h_{\text{proj}} < -100\,\text{ft}, \\ h_{\text{proj}} & \text{otherwise.} \end{cases} \tag{E-7}$$

Forcing the trajectory to reach the closest point in the conflict region prevents the likelihood ratios from becoming too small. Small likelihood ratios indicate that the proposal distribution $q$ favors samples in low-probability regions of $p$. This generally is not preferable because, as the form of the optimal proposal distribution suggests, an effective proposal distribution should be as similar to $p$ as possible while still favoring trajectories that result in conflict.

Consider the case when no RA is currently being executed. If constant accelerations $\ddot{h}_1$ and $\ddot{h}_2$ are applied to the own and intruder aircraft, respectively, in order to achieve a relative altitude of $h_{\text{target}}$ at CPA, the accelerations must satisfy:

$$\frac{1}{2}(\ddot{h}_2 - \ddot{h}_1)\tau^2 + h_{\text{proj}} = h_{\text{target}} \tag{E-8}$$

$$\ddot{h}_2 - \ddot{h}_1 = \frac{2(h_{\text{target}} - h_{\text{proj}})}{\tau^2}. \tag{E-9}$$

Equation E-9 defines a set of infinitely-many combinations of $\ddot{h}_1$ and $\ddot{h}_2$ that could be applied to reach $h_{\text{target}}$. However, one would like to keep the accelerations close to the mean of the distribution

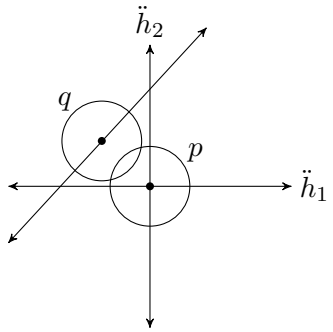



*Figure E-1. Comparison of distributions p and q. Shown are error ellipses for the two distributions. The distribution p is centered at the origin, while q is centered at the point on the line (Equation E-9) closest to the origin.*

$p(\mathbf{w}(t))$ to prevent the likelihood ratios from becoming too small. Because the mean of $p(\mathbf{w}(t))$ is zero, the task of finding the accelerations closest to the mean reduces to that of finding the point on the line $\ddot{h}_2 = \ddot{h}_1 + 2(h_{\text{target}} - h_{\text{proj}})/\tau^2$ closest to the origin. It is straightforward to show that the solution is

$$\ddot{h}_1 = \frac{h_{\text{proj}} - h_{\text{target}}}{\tau^2}, \tag{E-10}$$

$$\ddot{h}_2 = -\ddot{h}_1. \tag{E-11}$$

Sampling from a proposal distribution $q(\mathbf{w}(t) \mid \mathbf{x}(t))$ that is a Gaussian with mean as given by Equations E-10 and E-11 and covariance identical to $p(\mathbf{w}(t))$, namely $R$, should result in a lower variance estimate of the probability of conflict. Figure E-1 compares the distribution $p$ with the proposal distributon $q$. The form of the proposal distribution need not be Gaussian; any distribution that favors samples close to Equations E-10 and E-11 will work suitably. Moreover, it is crucial to note that although the mean of the proposal distribution are accelerations which, if applied constantly until CPA, would lead to a conflict, the accelerations are only applied for one time step, after which the next state is sampled and the calculations repeated.

Now consider the case when an RA is being executed. Only the own acceleration can be controlled to move closer to a conflict. Analogous to Equations E-8 and E-9, the requisite intruder acceleration is

$$\frac{1}{2}\ddot{h}_2\tau^2 + h_{\text{proj}} = h_{\text{target}} \tag{E-12}$$

$$\ddot{h}_2 = \frac{2(h_{\text{target}} - h_{\text{proj}})}{\tau^2}. \tag{E-13}$$

Intuitively, this is exactly twice that of the intruder acceleration when no RA is being executed. If the accelerations remain uncorrelated in the proposal distribution, the proposal distribution can be written as the product of the marginals

$$q(w_1(t) \mid \mathbf{x}(t))\, q(w_2(t) \mid \mathbf{x}(t)). \tag{E-14}$$

Since no change is made to the own acceleration, it is convenient to set $q(w_1(t) \mid \mathbf{x}(t)) = p(w_1(t))$ to simplify the likelihood ratio calculation. The marginal distribution $q(w_2(t) \mid \mathbf{x}(t))$ is a Gaussian with mean given by Equation E-13 and variance $R_{22}$.

## E.2 MAXIMUM-LIKELIHOOD ACCELERATION PROPOSAL

An alternative proposal distribution can be obtained as follows. Again, the RA execution and no-RA execution cases must be considered separately. The latter case is considered first. Previously, the requisite constant accelerations needed to arrive in the conflict region from an arbitrary $\mathbf{x}(t)$ were derived. Now let $(\ddot{h}_1(t_0), \ddot{h}_2(t_0), \ldots, \ddot{h}_1(t_{K-1}), \ddot{h}_2(t_{K-1}))$ denote an acceleration sequence of the aircraft starting from $\mathbf{x}(t)$. That is, $\ddot{h}_1(t_0)$ and $\ddot{h}_2(t_0)$ are applied for the first time step, $\ddot{h}_1(t_1)$ and $\ddot{h}_2(t_1)$ are applied for the second time step, and so on until CPA. If it is desired that the aircraft achieve a relative altitude of $h_{\text{target}}$ after $\tau$ seconds starting from state $\mathbf{x}(t)$, the control history must satisfy

$$\begin{aligned} h_{\text{target}} = h_{\text{proj}} &+ (\ddot{h}_2(t_0) - \ddot{h}_1(t_0))(\tau - 1) + \cdots \\ &+ (\ddot{h}_2(t_{K-2}) - \ddot{h}_1(t_{K-2})) + \frac{1}{2}(\ddot{h}_2(t_0) - \ddot{h}_1(t_0)) + \cdots \\ &+ \frac{1}{2}(\ddot{h}_2(t_{K-1}) - \ddot{h}_1(t_{K-1})), \end{aligned} \tag{E-15}$$

where $\Delta t = 1\,\text{s}$ was used. The objective is to find the accelerations that both satisfy Equation E-15 and that produce a trajectory that is most likely to occur by executing action $a$. This is equivalent to minimizing the difference between the trajectory produced by the accelerations $(\ddot{h}_1(t_0), \ddot{h}_2(t_0), \ldots, \ddot{h}_1(t_{K-1}), \ddot{h}_2(t_{K-1}))$ and the noiseless trajectory.

Begin by defining

$$u = \begin{bmatrix} \ddot{h}_1(t_0) \\ \ddot{h}_2(t_0) \\ \ddot{h}_1(t_1) \\ \ddot{h}_2(t_1) \\ \vdots \\ \ddot{h}_1(t_{K-1}) \\ \ddot{h}_2(t_{K-1}) \end{bmatrix}, \tag{E-16}$$

$$\alpha = \begin{bmatrix} \frac{1}{2} - \tau \\ \tau - \frac{1}{2} \\ \frac{3}{2} - \tau \\ \tau - \frac{3}{2} \\ \vdots \\ -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix}, \tag{E-17}$$

and $\Delta h = h_{\text{target}} - h_{\text{proj}}$. Then Equation E-15 can be expressed as

$$\alpha^T u = \Delta h. \tag{E-18}$$

Since the accelerations are nominally zero-mean Gaussian, the most likely trajectory that achieves the desired vertical separation is the one that minimizes the square norm of the acceleration vector $u$ subject to the constraint in Equation E-18. That is, the most likely accelerations are

$$u^* = \arg\min_u \frac{1}{2}\|u\|^2. \tag{E-19}$$

This is a quadratic programming problem that can be easily solved using the method of Lagrange multipliers [122]. If $f(u) = \frac{1}{2}\|u\|^2$ is the objective to be minimized and $g(u) = \alpha^T u - \Delta h$ is the linear constraint, then one must find $u$ that satisfies

$$\nabla_u f(u) - \lambda \nabla_u g(u) = 0, \tag{E-20}$$

where $\lambda$ is the Lagrange multiplier. This equation easily simplifies to $u - \lambda\alpha = 0$ or $u = \lambda\alpha$. Multiplying both sides by $\alpha^T$ yields $\alpha^T u = \Delta h = \lambda\|\alpha\|^2$. Thus $\lambda = \Delta h/\|\alpha\|^2$ and

$$u^* = \frac{\Delta h \alpha}{\|\alpha\|^2}. \tag{E-21}$$

The square norm of $\alpha$ is

$$\|\alpha\|^2 = \frac{1}{2} + 2\sum_{t=1/2}^{\tau-1/2} t^2. \tag{E-22}$$

It follows that the first accelerations $\ddot{h}_1$ and $\ddot{h}_2$ to apply to state $\mathbf{x}(t)$ are

$$\ddot{h}_1 = \frac{(1-2\tau)\Delta h}{2\|\alpha\|^2}, \tag{E-23}$$

$$\ddot{h}_2 = -\ddot{h}_1. \tag{E-24}$$

As before, the proposal distribution is a Gaussian with mean given by Equations E-23 and E-24. It is straightforward to show that the marginal distribution of $w_2(t)$ of the proposal distribution when executing an RA has a mean given by

$$\ddot{h}_2 = \frac{(2\tau-1)\Delta h}{\|\alpha\|^2}, \tag{E-25}$$

which is exactly double that of Equation E-24.

The optimization performed here happened to admit an analytic solution due to the simplicity of the problem. However, solving Equation E-19 typically requires approximation techniques such as gradient descent or a Newton-like method to minimize the objective [122].

## E.3 ANALYTIC PROPOSAL

Although there are analytic methods for computing the probability of conflict for some simple models [68], analytic solutions do not always exist. Even for the relatively simple encounter model of Appendix A, the expression for the probability of conflict quickly becomes intractable, as the

number of random accelerations that contribute to the state at CPA depends upon the vertical rate $\dot{h}_1$ at all preceding times. However, a reasonably accurate approximation to the analytic solution that is feasible to evaluate can be derived as follows.

In Appendix B, the following approximations to the mean and covariance updates of the aircraft state were derived:

$$\widehat{\mu}(t + \Delta t) = A(\Delta t)\widehat{\mu}(t) + B(\Delta t)\mathbf{u}(t, \widehat{\mu}(t)), \tag{E-26}$$

$$\widehat{P}(t + \Delta t) = A(\Delta t)\widehat{P}(t)A(\Delta t)^T + G(\Delta t, \widehat{\mu}(t))RG(\Delta t, \widehat{\mu}(t))^T. \tag{E-27}$$

The probability of a conflict, $\Pr(\mathrm{C} \mid \mathbf{x}, a)$, can then be estimated by initializing the mean $\widehat{\mu}(t_0) = \mathbf{x}$ and the covariance $\widehat{P}(t_0) = \mathbf{0}$ and applying the recursions in Equations E-26 and E-27 to obtain $\widehat{\mu}(t_K)$ and $\widehat{P}(t_K)$, the approximate mean and covariance of the state $\mathbf{x}(t_K)$ at CPA. The marginal distribution for $h$ at CPA is Gaussian with mean equal to the first element in $\widehat{\mu}(t_K)$ and variance equal to the first element in $\widehat{P}(t_K)$. To determine the probability of conflict, one simply integrates the density from $-100\,\mathrm{ft}$ to $+100\,\mathrm{ft}$.

It can be shown that the probability of a conflict from a state $\mathbf{x}$ not in the conflict region when executing action $a$ is $\Pr(\mathrm{C} \mid \mathbf{x}, a) = \mathrm{E}_{\Pr(\mathbf{x}'|\mathbf{x},a)}[\Pr(\mathrm{C} \mid \mathbf{x}', a)]$. This may be written

$$\mathrm{E}_{\Pr(\mathbf{x}'|\mathbf{x},a)}[\Pr(\mathrm{C} \mid \mathbf{x}', a)] = \int \Pr(\mathrm{C} \mid \mathbf{f}(\mathbf{x}, \mathbf{w}, a), a)\, p(\mathbf{w})\, d\mathbf{w}, \tag{E-28}$$

where $\mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{w}, a)$ is the successor state from $\mathbf{x}$ when taking action $a$ with noise $\mathbf{w}$. Suppose $\Pr(\mathrm{C} \mid \mathbf{x}', a)$ is known for all possible successor states $\mathbf{x}'$. In such a situation, one could estimate the probability of conflict $\Pr(\mathrm{C} \mid \mathbf{x})$ by producing samples of the immediate noise $\mathbf{w}$. It can be shown that the optimal proposal distribution from which to draw such samples is

$$q^*(\mathbf{w} \mid \mathbf{x}) \propto \Pr(\mathrm{C} \mid \mathbf{f}(\mathbf{x}, \mathbf{w}, a), a) \cdot p(\mathbf{w}). \tag{E-29}$$

The mean of this optimal proposal distribution can be approximated by drawing $N = 2n_{\mathbf{w}} + 1$ sigma-point samples $\chi^{(i)}$ from $p(\mathbf{w})$ with associated weights $w^{(i)}$. For each sample, define

$$P^{(i)} = \Pr(\mathrm{C} \mid \mathbf{f}(\mathbf{x}, \chi^{(i)}, a), a), \tag{E-30}$$

which can be estimated using the analytic approximation to $\Pr(\mathrm{C})$ described above. It follows that

$$\mathrm{E}_{q^*}[\mathbf{w}] = \int \mathbf{w} \cdot q^*(\mathbf{w} \mid \mathbf{x})\, d\mathbf{w} \approx \sum_{i=1}^{N} \widetilde{w}^{(i)} \chi^{(i)}, \tag{E-31}$$

where

$$\widetilde{w}^{(i)} = \frac{w^{(i)} P^{(i)}}{\sum_{k=1}^{N} w^{(k)} P^{(k)}} \tag{E-32}$$

are the normalized weights. Thus, this approximation of the mean of the optimal proposal distribution $q^*$ becomes the mean of a Gaussian proposal distribution. As with the preceding proposal distributions, the covariance was chosen arbitrarily to be the same as that of $p(\mathbf{w})$. This proposal distribution favors samples that have both a high probability of conflict and a high probability of occurring under the model. Thus, the analytic probability of conflict approximation is used as a heuristic to guide the sample trajectories toward conflict.

## E.4 DYNAMIC PROGRAMMING PROPOSAL

The dynamic programming proposal distribution is identical to that of the analytic proposal distribution except that the dynamic programming estimate of the probability of conflict (given in Section 4.2) is used to determine the mean of the distribution.

This page intentionally left blank.

# APPENDIX F
# PROOF OF PARETO OPTIMALITY

Suppose that the immediate cost function is of the form

$$c(s,a) = \sum_i \lambda_i f_i(s,a), \tag{F-1}$$

where $\lambda_i > 0$ and $f_i(s,a) \in \{0,1\}$. In addition, when following any policy, $f_i(s,a)$ may be 1 only once. The function $f_i(s,a)$ can be thought of as an indicator of whether event $i$ occurs at state $s$ when taking action $a$. In the hypothetical collision avoidance problem, there are two events: a conflict occurs for the first time and an alert is issued for the first time.

The cost-to-go function when using an immediate cost function of the form specified in Equation F-1 satisfies:

$$\begin{aligned} J^\pi(s) &= \mathrm{E}\left[\sum_t \sum_i \lambda_i f_i(s_t, \pi(s_t)) \mid S_0 = s, \pi\right] \\ &= \sum_i \lambda_i \,\mathrm{E}\left[\sum_t f_i(s_t, \pi(s_t)) \mid S_0 = s, \pi\right] \\ &= \sum_i \lambda_i p_i(\pi, s), \end{aligned} \tag{F-2}$$

where $s_t$ is the state at time $t$ and $S_0$ is a random variable specifying the initial state. The function $p_i(\pi, s)$ is the probability of event $i$ occurring when starting in state $s$ and following policy $\pi$.

Assuming some distribution $b$ over starting states, the probability that event $i$ occurs when following $\pi$ is given by

$$p_i(\pi) \equiv \sum_s b(s) p_i(\pi, s). \tag{F-3}$$

The expected cost of a policy $\pi$ is

$$J(\pi) \equiv \sum_s b(s) J^\pi(s). \tag{F-4}$$

It follows that

$$J(\pi) = \sum_i \lambda_i p_i(\pi). \tag{F-5}$$

Value iteration finds an optimal policy $\pi^*$ such that $J(\pi^*) \leq J(\pi)$ holds for all policies $\pi$. It can be shown that an optimal policy with respect to a cost function $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_n\}$ is Pareto optimal with respect to $p_1, \ldots, p_n$, as stated by the following proposition.

**Proposition 1.** *Suppose that an optimal policy $\pi^*$ is known for a given cost function. For any policy $\pi$ and any $j$, if $p_j(\pi) < p_j(\pi^*)$ then there exists some $i$ such that $p_i(\pi) > p_i(\pi^*)$.*

*Proof.* For a contradiction, assume that $p_j(\pi) < p_j(\pi^*)$ and $p_i(\pi) \leq p_i(\pi^*)$ for all $i$. It follows that, for all $\boldsymbol{\lambda}$,

$$\begin{aligned} J(\pi) &= \sum_i \lambda_i p_i(\pi) \\ &= \lambda_j p_j(\pi) + \sum_{i \neq j} \lambda_i p_i(\pi) \\ &< \lambda_j p_j(\pi^*) + \sum_{i \neq j} \lambda_i p_i(\pi^*) \\ &= J(\pi^*). \end{aligned} \tag{F-6}$$

Hence, $J(\pi) < J(\pi^*)$. If $\boldsymbol{\lambda}$ is chosen to be the cost function for which $\pi^*$ is optimal, then $J(\pi^*) \leq J(\pi)$, which is a contradiction. □

There are two important consequences of Pareto optimality.

1. Given an optimal policy, there is no other policy with the same alert rate and a lower conflict rate.
2. Given an optimal policy, there is no other policy with the same conflict rate and a lower alert rate.

Hence, the system operating characteristic curve (Section 5.3) traced by optimal policies with varying alert cost will never be to the right or below that of any other policy.

# APPENDIX G
# COMPUTING THE TIME-TO-CONFLICT DISTRIBUTION

Section 7.1 describes a method to efficiently handle dynamics in three spatial dimensions that involves estimating a distribution over $\tau$, the time to horizontal conflict. This appendix explains how to compute the time-to-conflict distribution for an arbitrary Markov process using dynamic programming. For simplicity, assume that time is discretized to one-second intervals and that the time horizon is $t_{\max}$. The probability of transitioning from $s$ to $s'$ is given by $\Pr(s' \mid s)$.

The algorithm uses $D_s(\tau)$ to represent the probability that the time-to-conflict falls within the interval $[\tau, \tau + 1)$ when starting at state $s$. If $\tau = t_{\max}$, then $D_s(\tau)$ represents the probability that the time-to-conflict falls within the interval $[t_{\max}, \infty)$. The algorithm uses arrays of length $t_{\max}$ to represent $D_s$ for each state $s$. The arrays are initialized as follows:

$$D_s(\tau) = \begin{cases} 1 & \text{if } s \in C \text{ and } \tau = 0, \\ 1 & \text{if } s \notin C \text{ and } \tau = t_{\max}, \\ 0 & \text{otherwise,} \end{cases} \tag{G-1}$$

where $s \in C$ means that $s$ is a conflict state.

Once $D_s$ is initialized, a copy is stored in $D'_s$. For each $s \notin C$, $D'_s$ is updated as follows:

$$D'_s(\tau) = \begin{cases} 0 & \text{if } \tau = 0, \\ \sum_{s'} \Pr(s' \mid s) D_{s'}(\tau - 1) & \text{if } 0 < \tau < t_{\max}, \\ \sum_{s'} \Pr(s' \mid s) \left( D_{s'}(\tau - 1) + D_{s'}(\tau) \right) & \text{otherwise.} \end{cases} \tag{G-2}$$

This update process assigns to $D'_s$ the weighted sum of the histograms represented by $D_{s'}$ shifted to the right one second, where the weights are determined by $\Pr(s' \mid s)$.

After $D'_s$ is computed for all states, it is copied to $D_s$. This process is repeated $t_{\max}$ times, after which $D_s$ will represent the true distribution over $\tau$, truncated at $t_{\max}$. If there are $n$ discrete states and there are a maximum of $k$ successor states, the time complexity of the algorithm is $O(t_{\max}^2 kn)$.

This page intentionally left blank.

# APPENDIX H
# TRACKER

For the purposes of evaluating the dynamic programming logic in simulation, a tracker was developed that emulates the behavior of the tracker implemented in TCAS for aircraft reporting altitude with 25-ft quantization [6]. The output of the tracker is not a probability distribution over states, a common example being a Gaussian distribution in the case of Kalman filtering, but rather a single point estimate of altitude, range, and range rate, among others. This appendix relates the details of the tracker.

The internal state of the tracker at time $t$ is

$$(\widehat{h}_{\text{own}}(t), \widehat{h}_{\text{int}}(t), \widehat{r}(t), \widehat{\dot{h}}_{\text{own}}(t), \widehat{\dot{h}}_{\text{int}}(t), \widehat{\dot{r}}(t), \widehat{\ddot{r}}(t)). \tag{H-1}$$

The measurement at time $t$ is

$$(\tilde{h}_{\text{own}}(t), \tilde{h}_{\text{int}}(t), \tilde{\chi}(t), \tilde{r}(t)). \tag{H-2}$$

Upon receiving the first measurement, the tracker initializes the altitudes and intruder range to their measured values and the intruder range rate to zero. After receiving the second measurement, the tracker estimates the intruder range rate and the altitude rates using finite differences, i.e., the difference between the current measurements and the previous ones divided by the time step between measurements $\Delta t$ (nominally one second). Upon receipt of the second measurement, moreover, the intruder range acceleration is initialized to zero.

The own altitude and own altitude rate are updated at each time using a two-step process. First, the own altitude at time $t + \Delta t$ is predicted using a constant velocity model:

$$h_{\text{own,pred}}(t + \Delta t) = \widehat{h}_{\text{own}}(t) + \widehat{\dot{h}}_{\text{own}} \Delta t. \tag{H-3}$$

Then the own altitude and own altitude rate at time $t + \Delta t$ are updated using the most recent measurement:

$$\widehat{h}_{\text{own}}(t + \Delta t) = h_{\text{own,pred}}(t + \Delta t) + \alpha(\tilde{h}_{\text{own}}(t + \Delta t) - h_{\text{own,pred}}(t + \Delta t)), \tag{H-4}$$

$$\widehat{\dot{h}}_{\text{own}}(t + \Delta t) = \widehat{\dot{h}}_{\text{own}}(t) + \frac{\beta}{\Delta t}(\tilde{h}_{\text{own}}(t + \Delta t) - h_{\text{own,pred}}(t + \Delta t)), \tag{H-5}$$

where $\alpha$ and $\beta$ control the level of correction in the own altitude and own altitude rate, respectively, due to the measurement. This is known as an $\alpha$-$\beta$ tracker. The intruder altitude and altitude rate are updated similarly. If the intruder altitude has not changed for over 6.5 seconds, the intruder altitude is reset to its measured altitude and the intruder altitude rate to zero. The $\alpha$ and $\beta$ values for the own aircraft updating are fixed at 0.6 and 0.257, respectively. However, the $\alpha$ and $\beta$ values for the intruder aircraft updating are dependent on the size of the prediction error and the magnitude of the intruder altitude rate.

The update for the range and range rate uses an $\alpha$-$\beta$-$\gamma$ tracker. The predicted range and range rate are

$$r_{\text{pred}}(t + \Delta t) = \widehat{r}(t) + \widehat{\dot{r}}(t)\Delta t, \tag{H-6}$$

$$\dot{r}_{\text{pred}}(t + \Delta t) = \widehat{\dot{r}}(t) + \widehat{\ddot{r}}(t)\Delta t. \tag{H-7}$$

The updated range and range rate at time $t + \Delta t$ are then

$$\widehat{r}(t + \Delta t) = r_{\text{pred}}(t + \Delta t) + \alpha(\tilde{r}(t + \Delta t) - r_{\text{pred}}(t + \Delta t)), \tag{H-8}$$

$$\widehat{\dot{r}}(t + \Delta t) = \dot{r}_{\text{pred}}(t + \Delta t) + \frac{\beta}{\Delta t}(\tilde{r}(t + \Delta t) - r_{\text{pred}}(t + \Delta t)). \tag{H-9}$$

When a sufficient amount of time has passed since the own aircraft first started a track on the intruder, i.e., if the intruder has sufficient "firmness," then the range acceleration is updated as

$$\widehat{\ddot{r}}(t + \Delta t) = \widehat{\ddot{r}}(t) + \frac{\gamma}{(\Delta t)^2}(\tilde{r}(t + \Delta t) - r_{\text{pred}}(t + \Delta t)). \tag{H-10}$$

The $\alpha$, $\beta$, and $\gamma$ coefficients for the range updating generally decrease as the firmness increases, indicating increased confidence in the prediction.

All values for $\alpha$, $\beta$, and $\gamma$ come from the minimum operational performance standards for TCAS II [6].

# APPENDIX I
# MINI TCAS

A simplified version of TCAS, called mini TCAS in this report, was implemented that issues RAs based only on one perfect aircraft state defined in terms of $h$, $\tau$, $\dot{h}_1$, and $\dot{h}_2$. The major assumptions of mini TCAS are:

1. The intruder is not TCAS-equipped but is reporting altitude and is under perfect surveillance.
2. The horizontal range rate is $-500\,\text{ft/s}$.
3. No tracking or encounter monitoring is performed. Hence, mini TCAS is a memory-less system.
4. Only initial RA sense and strength are selected. Thus, no strength increases or reversals are issued.
5. No minimum or maximum altitudes are enforced.
6. No intruder intent information, in the form of an RA coordination message, is received.
7. The tau-rising test and horizontal miss distance test are not performed.

The various constants that mini TCAS uses are listed in Table I-1. These constants will be discussed in the remainder of the appendix.

**TABLE I-1**

**Constants used in mini TCAS.**

| | |
|---|---|
| DMOD | Threshold that defines safety buffer around own aircraft used for threat detection |
| RDTHR | Threshold that defines converging intruders |
| TRTHR | Range threshold for converging intruders in the range test |
| H1 | Threshold for determining if diverging intruder passes the range test |
| ZTHR | Altitude threshold for threat detection |
| TVTHR | Time threshold for time until co-altitude in the altitude test |
| ALIM | Altitude threshold for RA selection |

Given $h$, $\tau$, $\dot{h}_1$, and $\dot{h}_2$, mini TCAS calculates the slant range, $r_s$, and the slant range rate, $\dot{r}_s$. It assumes that the intruder is closing horizontally at $\dot{r}_h = -500\,\text{ft/s}$. Using the fact that $\tau = -r_h/\dot{r}_h$, $r_h \geq 0$, the slant range is given by

$$r_s = \sqrt{r_h^2 + h^2} = \sqrt{(\dot{r}_h \tau)^2 + h^2}. \tag{I-1}$$

The slant range rate is

$$\dot{r}_s = \frac{r_h \dot{r}_h + h\dot{h}}{\sqrt{r_h^2 + h^2}}, \tag{I-2}$$

where $\dot{h} = \dot{h}_2 - \dot{h}_1$.

Because mini TCAS decides which RA, if any, to issue based solely on the current aircraft state, the critical interval was approximated by [modified_tau_uncapped, true_tau_uncapped]. If $r_s \leq$ DMOD, modified_tau_uncapped is zero, indicating the critical interval starts immediately because the own aircraft safety buffer defined by DMOD has already been violated. Otherwise,

$$\text{modified\_tau\_capped} = -\frac{r_s^2 - \text{DMOD}^2}{r_s \cdot \min(\dot{r}_s, -\text{RDTHR})}, \tag{I-3}$$

where RDTHR = 10 ft/s. The end of the critical interval, true_tau_uncapped, is always defined as

$$\text{true\_tau\_uncapped} = -\frac{r_s}{\min(\dot{r}_s, -\text{RDTHR})}. \tag{I-4}$$

If the intruder is diverging ($\dot{r}_s > 0$), the critical interval is [0,0], or undefined.

The following sections describe the threat detection, sense selection, and strength selection components of mini TCAS.

## I.1 THREAT DETECTION

The intruder is declared a threat if and only if it passes the range and altitude tests and fails the altitude separation test.

### I.1.1 Range Test

The first step in the threat detection process is the range test. The intruder passes the range test if either of the following is true.

1. (converging intruder) $\dot{r}_s \leq \text{RDTHR} \wedge r_s < \text{TRTHR} \wedge \text{modified\_tau\_uncapped} \leq \text{TRTHR}$;
2. (diverging intruder) $\dot{r}_s > \text{RDTHR} \wedge r_s < \text{DMOD} \wedge r_s \dot{r}_s \leq \text{H1}$,

where TRTHR and H1 are thresholds dependent on the encounter sensitivity level. Note that this is a simplification of the true range test in that the tau-rising test and the horizontal miss distance test are not performed.

### I.1.2 Altitude Test

If there is already insufficient vertical separation between the aircraft, i.e., if $|h| <$ ZTHR, the intruder passes the altitude test automatically if modified_tau_uncapped is zero. Otherwise, the

intruder passes the test if the projected vertical miss distance, $vmd$, during the critical interval is less than ZTHR. The projected relative altitudes at the beginning and end of the critical interval, respectively, are determined by linear extrapolation:

$$h_{\text{beg}} = h + (\dot{h}_2 - \dot{h}_1) \cdot \text{modified\_tau\_uncapped}, \tag{I-5}$$

$$h_{\text{end}} = h + (\dot{h}_2 - \dot{h}_1) \cdot \text{true\_tau\_uncapped}. \tag{I-6}$$

If the aircraft are projected to cross altitudes during the critical interval ($\text{sign}(h_{\text{beg}}) \neq \text{sign}(h_{\text{end}})$), then $vmd = 0$. Otherwise, $vmd = \min(|h_{\text{beg}}|, |h_{\text{end}}|)$. Consequently, if $vmd <$ ZTHR, the intruder passes the altitude test.

If, on the other hand, $|h| \geq$ ZTHR and there is some finite value for the vertical $\tau$, or time to co-altitude, the intruder passes the altitude test if $\tau \leq$ TVTHR. The vertical $\tau$ is finite if the rate of change of the vertical separation magnitude $A = |h|$ is less than $-1$ ft/s. That is,

$$\dot{A} = (\dot{h}_2 - \dot{h}_1)\text{sign}(h) < -1\,\text{ft/s}. \tag{I-7}$$

If $\dot{A} \geq -1$ ft/s, the intruder fails the altitude test. The threshold TVTHR is a function of the sensitivity level and dependent on whether or not the vertical closure rate is due primarily to the intruder. The vertical closure rate is attributed primarily to the intruder if either (1) the vertical rate of own aircraft is less than 600 ft/min or (2) both aircraft are climbing or both aircraft are descending and $|\dot{h}_2| > |\dot{h}_1|$. Otherwise, the vertical closure rate is not primarily due to the intruder.

Obviously, if $|h| \geq$ ZTHR and modified_tau_uncapped $= 0$, the intruder fails the altitude test.

### I.1.3 Altitude Separation Test

The intruder passes the altitude separation test if at least one of the following is true.

1. $|h| > 600\,\text{ft} \wedge (\dot{h}_1 = 0 \vee \dot{h}_2 = 0 \vee \text{sign}(\dot{h}_1) = \text{sign}(\dot{h}_2))$;
2. $|h| > 850\,\text{ft} \wedge \dot{h}_1 \neq 0 \wedge \dot{h}_2 \neq 0 \wedge \text{sign}(\dot{h}_1) = -\text{sign}(\dot{h}_2)$.

## I.2 SENSE SELECTION

Sense selection is performed if and only if the intruder is declared a threat. As discussed in Section 1.2, the response to both upward-sense (Climb) and downward-sense (Descend) RAs is modeled and the projected vertical separation at the beginning and end of the critical interval, $h_{\text{beg}}$ and $h_{\text{end}}$, respectively, is calculated. The intruder's trajectory is modeled as a straight line with a constant vertical rate. A pilot delay model is implemented for own aircraft in which the pilot requires five seconds to respond to the RA. After the pilot-response delay, the own aircraft trajectory is modeled as accelerating at 0.25 g (8 ft/s$^2$) until reaching the target vertical rate, after which it maintains that rate for the remainder of the encounter. The target vertical rate is 1500 ft/min for the Climb RA and $-1500$ ft/min for the Descend RA. If modified_tau_uncapped $= 0$, $h_{\text{beg}} = h$ and if true_tau_uncapped $= 0$, $h_{\text{end}} = h$. The vertical separation during the critical interval, $vmd$, is

$$vmd = \begin{cases} h_{\text{beg}} & \text{if } |h_{\text{end}}| > |h_{\text{beg}}|, \\ h_{\text{end}} & \text{otherwise.} \end{cases} \tag{I-8}$$

Let $vmd(\mathsf{Climb})$ denote $vmd$ for the Climb RA and $vmd(\mathsf{Descend})$ denote $vmd$ for the Descend RA.

If the aircraft are not considered to be co-altitude ($|h| > 100\,\text{ft}$), then the downward sense is selected if the upward-sense RA is an altitude-crossing RA, the downward-sense RA is not, and the downward-sense RA provides at least ALIM separation:

$$\text{sign}(h) \neq \text{sign}(vmd(\mathsf{Climb})) \wedge \text{sign}(h) = \text{sign}(vmd(\mathsf{Descend})) \wedge |vmd(\mathsf{Descend})| \geq \mathsf{ALIM}. \quad \text{(I-9)}$$

The threshold ALIM is a function of the altitude layer of own aircraft. Similarly, the upward sense is selected if

$$\text{sign}(h) \neq \text{sign}(vmd(\mathsf{Descend})) \wedge \text{sign}(h) = \text{sign}(vmd(\mathsf{Climb})) \wedge |vmd(\mathsf{Climb})| \geq \mathsf{ALIM}. \quad \text{(I-10)}$$

If the previous criteria do not hold, then the upward sense is selected if

$$|vmd(\mathsf{Climb})| > |vmd(\mathsf{Descend})| \quad \text{(I-11)}$$

and the downward sense is selected otherwise.

### I.3 STRENGTH SELECTION

The strength selection process proceeds by first calculating the vertical miss distance during the critical interval using linear extrapolation, as was done during the threat detection process (Equations I-5 and I-6). Vertical speed limits (VSLs) are not modeled if at least one of the following is true.

1. $|\dot{h}_2| < 1000\,\text{ft/min} \wedge |h| < \mathsf{ALIM} \wedge |vmd(\cdot)| < \mathsf{ALIM} \wedge |\dot{h}_1| \leq 600\,\text{ft/min}$;
2. $|\dot{h}_2| \geq 1000\,\text{ft/min} \wedge |vmd(\cdot)| < \mathsf{ALIM} \wedge |\dot{h}_1| \leq 600\,\text{ft/min}$,

where $vmd(\cdot)$ is the vertical separation during the critical interval, $vmd(\mathsf{Climb})$ if the upward sense is selected, $vmd(\mathsf{Descend})$ if the downward sense is selected. Otherwise, each VSL is modeled and the vertical separation during the critical interval is calculated exactly as in the sense selection process (Equation I-8).

The least restrictive VSL that provides at least the target separation is selected as the RA. The target separation for all VSLs except Do Not Climb and Do Not Descend is $\mathsf{ALIM} + 75\,\text{ft}$. The target separation for Do Not Climb and Do Not Descend is ALIM. If no VSL provides at least the target separation, the positive RA in the given sense is selected.

If a VSL is selected, but it is corrective, i.e., results in own aircraft changing its vertical rate, then Do Not Climb or Do Not Descend is selected instead, depending on the sense.

## REFERENCES


[1] J.E. Lebron, A.D. Zeitlin, N.A. Spencer, J.W. Andrews, and W.H. Harman, "System safety study of minimum TCAS II (Traffic Alert and Collision Avoidance System)," MITRE, Technical Rep. MTR-83W241 (1983).

[2] M.P. McLaughlin, "Safety study of the Traffic Alert and Collision Avoidance System (TCAS II)," MITRE Corporation, Technical Rep. MTR 97W32 (1997).

[3] T.A. Choyce and K.M. Ciaramella, "Test and evaluation of TCAS II logic version 7," Federal Aviation Administration (2000).

[4] J.K. Kuchar and A.C. Drumm, "The Traffic Alert and Collision Avoidance System," *Lincoln Laboratory Journal* 16(2), 277–296 (2007).

[5] Federal Aviation Administration, U.S. Dept. of Transportation, "Introduction to TCAS II version 7," Available from ARINC: `http://www.arinc.com/tcas` (2000).

[6] RTCA, "Minimum operational performance standards for Traffic Alert and Collision Avoidance System II (TCAS II), DO-185b," RTCA, Inc., Washington, D.C. (2008).

[7] M.J. Kochenderfer, L.P. Espindle, J.K. Kuchar, and J.D. Griffith, "A Bayesian approach to aircraft encounter modeling," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Honolulu, Hawaii (2008).

[8] M. Kochenderfer, J. Kuchar, L. Espindle, and J. Griffith, "Uncorrelated encounter model of the National Airspace System version 1.0," MIT Lincoln Laboratory, Project Report ATC-345 (2008).

[9] M. Kochenderfer, L. Espindle, J. Kuchar, and J. Griffith, "Correlated encounter model of the National Airspace System version 1.0," MIT Lincoln Laboratory, Project Report ATC-344 (2008).

[10] M. Edwards, M. Kochenderfer, J. Kuchar, and L. Espindle, "Encounter models for unconventional aircraft," MIT Lincoln Laboratory, Project Report ATC-348 (2009).

[11] K.P. Yoon and C.L. Hwang, *Multiple attribute decision making: An introduction*, Thousand Oaks, Calif.: Sage Publications (1995).

[12] L.F. Winder and J.K. Kuchar, "Hazard avoidance alerting with Markov decision processes," International Center for Air Transportation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Technical Rep. ICAT-2004-4 (2004).

[13] M.J. Kochenderfer, L.P. Espindle, J.K. Kuchar, and J.D. Griffith, "A comprehensive aircraft encounter model of the National Airspace System," *Lincoln Laboratory Journal* 17(2), 41–53 (2008).

[14] International Civil Aviation Organization, "Surveillance, radar and collision avoidance," in *International Standards and Recommended Practices*, vol. IV, annex 10, 4th ed. (2007).

[15] T. Miquel and K. Rigotti, "European encounter model," CENA/Sofréavia and QinetiQ, Technical Rep. ACASA/WP1/186/D (2002).

[16] S. Chabert, "Safety encounter model focused on issue SA01a," CENA/Sofréavia and QinetiQ, Technical Rep. SIRE/WP2/21/D (2005).

[17] P.J.H. Schoemaker, "The expected utility model: Its variants, purposes, evidence and limitations," *Journal of Economic Literature* 20(2), 529–563 (1982).

[18] J.K. Kuchar, "Methodology for alerting-system performance evaluation," *Journal of Guidance, Control, and Dynamics* 2(2), 438–444 (1996).

[19] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence* 101(1–2), 99–134 (1998).

[20] R.E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME—Journal of Basic Engineering* 82(Series D), 35–40 (1960).

[21] J.L. Crassidis and J.L. Junkins, *Optimal Estimation of Dynamic Systems*, CRC Press (2000).

[22] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, John Wiley & Sons, Inc. (2001).

[23] S.G. Mohinder and A.P. Andrews, *Kalman Filtering: Theory and Practice Using MATLAB*, John Wiley & Sons, Inc. (2001).

[24] G. Minkler and J. Minkler, *Theory and Applications of Kalman Filtering*, Magellan Book Company (1993).

[25] D.P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, Belmont, Mass.: Athena Scientific, 3rd ed. (2005).

[26] L.C. Yang, *Aircraft Conflict Analysis and Real-Time Conflict Probing Using Probabilistic Trajectory Modeling*, Ph.D. thesis, Massachusetts Institute of Technology (2000).

[27] E. Frazzoli, Z.H. Mao, J.H. Oh, and E. Feron, "Resolution of conflicts involving many aircraft via semidefinite programming," *Journal of Guidance, Control, and Dynamics* 24(1), 79–86 (2001).

[28] K.D. Bilimoria, "A geometric optimization approach to aircraft conflict resolution," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Denver, Colo. (2000).

[29] G. Dowek, A. Geser, and C. Muñoz, "Tactical conflict detection and resolution in a 3-D airspace," in *4th USA/Europe Air Traffic Management R&D Seminar*, Santa Fe, New Mexico (2001).

[30] R. Lachner, "Collision avoidance as a differential game: Real-time approximation of optimal strategies using higher derivatives of the value function," in *IEEE International Conference on Systems, Man, and Cybernetics* (1997), vol. 3, pp. 2308–2313.

[31] R.Y. Gazit and J.D. Powell, "Aircraft collision avoidance based on GPS position broadcasts," in *AIAA/IEEE Digital Avionics Systems Conference* (1996), pp. 393–399.

[32] R. Isaacs, *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*, New York: John Wiley and Sons (1965).

[33] J.K. Kuchar and L.C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems* 1(4), 179–189 (2000).

[34] M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley series in probability and mathematical statistics, New York: Wiley (1994).

[35] R.A. Howard, *Dynamic Programming and Markov Processes*, Cambridge, Mass.: MIT Press (1960).

[36] T. Smith and R.G. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Uncertainty in Artificial Intelligence* (2005).

[37] H. Kurniawati, D. Hsu, and W. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems* (2008).

[38] S. Temizer, M.J. Kochenderfer, L.P. Kaelbling, T. Lozano-Perez, and J.K. Kuchar, "Unmanned aircraft collision avoidance using partially-observable Markov decision processes," MIT Lincoln Laboratory, Project Report ATC-356 (2009).

[39] M.L. Littman, A.R. Cassandra, and L.P. Kaelbling., "Learning policies for partially observable environments: Scaling up," in *International Conference on Machine Learning* (1995).

[40] M. Hauskrecht, "Value-function approximations for partially observable Markov decision processes," *Journal of Artificial Intelligence Research* 13, 33–94 (2000).

[41] T.B. Wolf, *Aircraft Collision Avoidance Using Monte Carlo Real-Time Belief Space Search*, Master's thesis, Massachusetts Institute of Technology (2009).

[42] R. Bellman, *Dynamic Programming*, Princeton University Press (1957).

[43] D.P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*, Englewood Cliffs, N.J: Prentice Hall, 1st ed. (1987).

[44] D.P. Bertsekas and J.N. Tsitsiklis, *Neuro-Dynamic Programming*, Optimization and neural computation series, Belmont, Mass.: Athena Scientific (1996).

[45] W.H. Fleming and H.M. Soner, *Controlled Markov processes and viscosity solutions*, New York: Springer, 2nd ed. (2006).

[46] S. Chakravorty and J.L. Junkins, "Motion planning in uncertain environments with vision-like sensors," *Automatica* 43(12), 2104–2111 (2007).

[47] M.C.M. Mourits, R.B.M. Huirne, A.A. Dijkhuizen, A.R. Kristensen, and D.T. Galligan, "Economic optimization of dairy heifer management decisions," *Agricultural Systems* 61(1), 17–31 (1999).

[48] A.V. Gheorghe, H.N. Bali, W.J. Hill, and E.R. Carson, "Dynamic decision models for clinical diagnosis," *International Journal of Bio-Medical Computing* 7(2), 81–92 (1976).

[49] J.D. Williams and S. Young, "Partially observable Markov decision processes for spoken dialog systems," *Computer Speech & Language* 21(2), 393–422 (2007).

[50] R. Holdsworth, *Autonomous In-Flight Path Planning to Replace Pure Collision Avoidance for Free Flight Aircraft Using Automatic Dependent Surveillance Broadcast*, Ph.D. thesis, Department of Electrical Engineering, Swinburne University (2003).

[51] M.J. Kochenderfer, J.D. Griffith, and J.K. Kuchar, "Hazard alerting using line-of-sight rate," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Honolulu, Hawaii (2008).

[52] L.P. Kaelbling and T. Lozano-Perez, "Finding aircraft collision-avoidance strategies using policy search methods," Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Technical Rep. MIT-CSAIL-TR-2009-043 (2009).

[53] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, Adaptive computation and machine learning, Cambridge, Mass.: MIT Press (1998).

[54] L.P. Kaelbling, M.L. Littman, and A.W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research* 4, 237–285 (1996).

[55] W.T.B. Uther, *Tree Based Hierarchical Reinforcement Learning*, Ph.D. thesis, Computer Science Department, School of Computer Science, Carnegie Mellon University (2002).

[56] A.J. Smith, "Applications of the self-organising map to reinforcement learning," *Neural Networks* 15(8–9), 1107–1124 (2002).

[57] G. Gordon, "Stable function approximation in dynamic programming," Computer Science Department, Carnegie Mellon University, Technical Rep. CMU-CS-95-103 (1995).

[58] G.J. Gordon, *Approximate Solutions to Markov Decision Processes*, Ph.D. thesis, Computer Science Department, Carnegie Mellon University (1999).

[59] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online planning algorithms for POMDPs," *Journal of Artificial Intelligence Research* 32, 663–704 (2008).

[60] E.A. Hansen and S. Zilberstein, "LAO*: A heuristic search algorithm that finds solutions with loops," *Artificial Intelligence* 129(1–2), 35–62 (2001).

[61] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, New York: Wiley, 2nd ed. (2000).

[62] C.M. Bishop, *Pattern Recognition and Machine Intelligence*, New York: Springer (2007).

[63] R. Munos and A. Moore, "Variable resolution discretization in optimal control," *Machine Learning* 49(2–3), 291–323 (2002).

[64] W.B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Hoboken, N.J.: Wiley (2007).

[65] L.C. Yang and J.K. Kuchar, "Prototype conflict alerting system for free flight," *Journal of Guidance, Control, and Dynamics* 20(4), 768–773 (1997).

[66] B.D. Carpenter and J.K. Kuchar, "Probability-based collision alerting logic for closely-spaced parallel approach," in *AIAA 35th Aerospace Sciences Meeting*, Reno, NV (1997).

[67] W.H. Harman and M.L. Wood, Personal communication (2009).

[68] R.A. Paielli and H. Erzberger, "Conflict probability estimation for free flight," *Journal of Guidance, Control, and Dynamics* 20(3), 588–596 (1997).

[69] F.K. Chan, *Spacecraft Collision Probability*, El Segundo, Calif.: Aerospace Press (2008).

[70] T. Jones, "Tractable conflict risk accumulation in quadratic space for autonomous vehicles," *Journal of Guidance, Control, and Dynamics* 29(1), 39–48 (2006).

[71] C.E. van Daalen and T. Jones, "Fast conflict detection using probability flow," *Automatica* 45(8), 1903–1909 (2009).

[72] L. Yang, J.H. Yang, J. Kuchar, and E. Feron, "A real-time Monte Carlo implementation for computing probability of conflict," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Providence, RI (2004).

[73] R.E. Williams, *Importance Sampling Applied to Policy Gradient for Avoidance of Rare Events*, Master's thesis, Boston University (2009).

[74] T. Schouwenaars, *Safe Trajectory Planning of Autonomous Vehicles*, Ph.D. thesis, Massachusetts Institute of Technology (2006).

[75] L.F. Winder and J.K. Kuchar, "Evaluation of collision avoidance maneuvers for parallel approach," *Journal of Guidance, Control, and Dynamics* 22(6), 801–807 (1999).

[76] RTCA, "Safety analysis of proposed change to TCAS RA reversal logic, DO-298," RTCA, Inc., Washington, D.C. (2005).

[77] T. Billingsley, *Safety Analysis of TCAS on Global Hawk Using Airspace Encounter Models*, Master's thesis, Massachusetts Institute of Technology (2006).

[78] L.P. Espindle, J.D. Griffith, and J.K. Kuchar, "Safety analysis of upgrading to TCAS version 7.1 using the 2008 U.S. Correlated Encounter Model," MIT Lincoln Laboratory, Project Report ATC-349 (2009).

[79] RTCA, "TCAS II version 7 resolution advisory sense reversal logic evaluation criteria, performance metrics, and decision rationale," Special Committee SC-147 Doc 05-RWG-01 (2005).

[80] J.K. Kuchar, *A Unified Methodology for the Evaluation of Hazard Alerting Systems*, Ph.D. thesis, Massachusetts Institute of Technology (1995).

[81] T.D. Wickens, *Elementary Signal Detection Theory*, Oxford University Press (2002).

[82] R.Y. Rubinstein and D.P. Kroese, *Simulation and the Monte Carlo Method*, Hoboken, N.J: John Wiley and Sons, 2nd ed. (2008).

[83] G. Yates, "Investigation of the effects of modelling altimetry error by analytical and stochastic approaches," ACASA, Work Package 1, Working Paper 036 (1999).

[84] O. Khatib and J.F.L. Maitre, "Dynamic control of manipulators operating in a complex environment," in *Symposium on Theory and Practice of Robots and Manipulators*, Udine, Italy: Elsevier (1978), pp. 267–282.

[85] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *IEEE International Conference on Robotics and Automation* (1985), vol. 2, pp. 500–505.

[86] J.C. Latombe, *Robot motion planning*, Boston: Kluwer Academic Publishers (1991).

[87] V.N. Duong and K. Zeghal, "Conflict resolution advisory for autonomous airborne separation in low-density airspace," in *IEEE Conference on Decision and Control* (1997), vol. 3, pp. 2429–2434.

[88] M.S. Eby and W.E. Kelly, "Free flight separation assurance using distributed algorithms," in *IEEE Aerospace Conference* (1999), vol. 2, pp. 429–441.

[89] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *IEEE International Conference on Robotics and Automation* (1991), pp. 1398–1404.

[90] S.M. LaValle, *Planning Algorithms*, Cambridge University Press (2006).

[91] S.M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Department, Iowa State University, Technical Rep. 98-11 (1998).

[92] S.M. LaValle and J.J. Kuffner, "Randomized kinodynamic planning," in *IEEE International Conference on Robotics and Automation* (1999), vol. 1, pp. 473–479.

[93] Y. Kuwata, G.A. Fiore, J. Teo, E. Frazzoli, and J.P. How, "Motion planning for urban driving using RRT," in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2008), pp. 1681–1686.

[94] J. Saunders, R. Beard, and J. Byrne, "Vision-based reactive multiple obstacle avoidance for micro air vehicles," in *American Control Conference* (2009), pp. 5253–5258.

[95] J.J. Kuffner and S.M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation* (2000), vol. 2, pp. 995–1001.

[96] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and Gaussian processes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2008), pp. 1056–1062.

[97] A.Y. Ng and M.I. Jordan, "PEGASUS: A policy search method for large MDPs and POMDPs," in C. Boutilier and M. Goldszmidt (eds.), *Uncertainty in Artificial Intelligence*, Morgan Kaufmann (2000), pp. 405–415.

[98] A.Y. Ng, H.J. Kim, M.I. Jordan, and S. Sastry, "Autonomous helicopter flight via reinforcement learning," in S. Thrun, L.K. Saul, and B. Schölkopf (eds.), *Advances in Neural Information Processing Systems*, Cambridge, Mass.: MIT Press, vol. 16 (2004).

[99] F. Glover and M. Laguna, *Tabu Search*, Boston: Kluwer Academic Publishers (1997).

[100] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," *Science* 220(4598), 671–680 (1983).

[101] J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, Ann Arbor: University of Michigan Press (1975).

[102] J.R. Koza, *Genetic Programming: On The Programming of Computers by Means of Natural Selection*, Complex adaptive systems, Cambridge, Mass.: MIT Press (1992).

[103] W.E. Hart, N. Krasnogor, and J.E. Smith (eds.), *Recent Advances in Memetic Algorithms*, Berlin: Springer (2005).

[104] N. Durand, J.M. Alliot, and F. Médioni, "Neural nets trained by genetic algorithms for collision avoidance," *Applied Intelligence* 13(3), 205–213 (2004).

[105] S.M. Kakade, *On the Sample Complexity of Reinforcement Learning*, Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London (2003).

[106] J.A. Bagnell, S. Kakade, A. Ng, and J. Schneider, "Policy search by dynamic programming," in *Neural Information Processing Systems*, MIT Press (2003), vol. 16.

[107] W.D. Love, "TCAS III: Bringing operational compatibility to airborne collision avoidance," in *AIAA/IEEE Digital Avionics Systems Conference*, San Jose, Calif. (1988), pp. 877–881.

[108] G. Dean, "Final report on acceptability and efficiency of lateral collision avoidance manoeuvres," ACAS Programme, Work Package 8 (2002).

[109] L.C. Yang and J.K. Kuchar, "Using intent information in probabilistic conflict analysis," in *AIAA Guidance, Navigation, and Control Conference*, Boston, Mass. (1998), pp. 797–806.

[110] J.K. Kuchar and L.C. Yang, "Incorporation of uncertain intent information in conflict detection and resolution," in *IEEE Conference on Decision and Control* (1997), vol. 2, pp. 1810–1815.

[111] T.B. Billingsley, L.P. Espindle, and J.D. Griffith, "TCAS multiple threat encounter analysis," MIT Lincoln Laboratory, Project Report ATC-359 (2009).

[112] J.R. Quinlan, "Induction of decision trees," *Machine Learning* 1(1), 81–106 (1986).

[113] S.J. Julier and J.K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE* 92(3), 401–422 (2004).

[114] D. Kincaid and W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing*, Pacific Grove, Calif.: Brooks/Cole, 3rd ed. (2002).

[115] S. Davies, "Multidimensional triangulation and interpolation for reinforcement learning," in M.C. Mozer, M.I. Jordan, and T. Petsche (eds.), *Advances in Neural Information Processing Systems*, Cambridge, Mass.: MIT Press, vol. 9, pp. 1005–1011 (1997).

[116] D.W. Moore, *Simplical Mesh Generation with Applications*, Ph.D. thesis, Cornell University (1992).

[117] Y. Luo, "A local multivariate Lagrange interpolation method for constructing shape functions," *Communications in Numerical Methods in Engineering* (2008).

[118] S. Asmussen and P.W. Glynnn, *Stochastic Simulation: Algorithms and Analysis*, New York: Springer (2007).

[119] R. Srinivasan, *Importance Sampling: Applications in Communications and Detection*, Berlin: Springer (2002).

[120] G.C. Orsak and B. Aazhang, "Constrained solutions in importance via robust statistics," *IEEE Transactions on Information Theory* 37(2), 307–316 (1991).

[121] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Boston: Artech House (2004).

[122] D.P. Bertsekas, *Nonlinear Programming*, Athena Scientific (1999).

# NOTATION

| | |
|---|---|
| $a$ | action |
| $b$ | belief state |
| $B$ | Bellman update operator |
| $c(s, a)$ | cost function |
| $h$ | vertical separation |
| $\dot{h}_1$ | own vertical rate |
| $\dot{h}_2$ | intruder vertical rate |
| $J$ | cost-to-go function |
| $J^*$ | optimal cost-to-go function |
| $\Pr(\text{A})$ | probability of alert |
| $\Pr(\text{C})$ | probability of conflict |
| $\pi$ | policy |
| $\pi^*$ | optimal policy |
| $s$ | state |
| $s_{\text{RA}}$ | RA state |
| $t$ | time |
| $T(s' \mid s, a)$ | transition model |
| $\tau$ | time to horizontal conflict |
| $\mathbf{x}$ | state (interpreted as a vector) |

This page intentionally left blank.