

DOT/FAA/CT-97/4

User Request Evaluation Tool (URET) Algorithm Assessment Report

February 1997



**Federal Aviation Administration
William J. Hughes Technical Center
ATC Engineering and Test Division, ACT-200
Traffic Flow Management Branch, ACT-250
Atlantic City International Airport, NJ 08405**

Executive Summary

This report presents the results of an independent assessment of the User Request Evaluation Tool (URET) core algorithms conducted by the Traffic Flow Management Branch (ACT-250) at the FAA William J. Hughes Technical Center. This document contains a description of major functions comprising each algorithm set, including variable definitions and mathematical equations, and provides an assessment of the assumptions/approximations and their impact on the accuracy of the algorithm. The cooperative support provided by the URET developer, MITRE/CAASD, facilitated the accomplishment of this effort.

URET is an automated conflict detection (ACD) tool intended for use as a decision support aid for the en route air traffic controller. URET detects aircraft-to-aircraft and aircraft-to-airspace conflicts for Instrument Flight Rules (IFR) aircraft tracked by the Host Computer System (HCS), and provides alert information to the controller when such conflicts are detected. The technical performance and accuracy of the URET algorithms are critical issues to be assessed in preparation for a Joint Resources Council (JRC) investment decision for an ACD tool.

The scope of ACT-250's effort was limited to the core algorithms implemented in URET Delivery 1.1 (D1.1). This includes the Trajectory Modeler (TJM), Track Management (TKM), and Automated Problem Detection (APD) algorithms. The ACT-250 independent assessment effort was based on determining the validity of the URET algorithms and verifying the engineering principles upon which the algorithms were established. ACT-250's approach was to review MITRE/CAASD's algorithmic documentation and the applicable source code. By taking this approach, ACT-250 became very knowledgeable about the algorithms' details and the approximations and assumptions that were made during the URET development.

A URET testbed was established in the Terminal Air Traffic Control Automation (TATCA)/Automated En Route Air Traffic Control (AERA) laboratory at the Technical Center. Various simulated exercises were designed to "push the envelope" on the constraints established by the identified assumptions and approximations. The completion of the algorithm assessments and their validation via structured simulations using this laboratory, as originally planned, was curtailed in late 1996 when ACT-250 was redirected by the Air Traffic Management (ATM) Prototype Product Team (AUA-540) to focus on another effort. Consequently, while the majority of the key functions have been assessed, this report does not contain an assessment of every URET function.

The source code of the algorithms that were assessed was found to be based on sound engineering principles. The assumptions and approximations made by MITRE/CAASD are reasonable for the current prototype software requirements. Since the scope of this assessment was scaled back to an analytic assessment of the algorithmic source code modules, there is no empirical data derived from simulations or live data to validate the algorithms. Therefore, the assumptions and approximations should still be independently validated with a stringent set of simulations and live data tests to ensure the robustness and accuracy of the algorithms.

The information provided by this report is valuable to both the developer of the URET prototype and a production contractor. It bridges the documentation gap between the source code and existing software design and algorithmic definition documents, outlines URET D1.1 limitations and assumptions, and identifies suggested improvements to specific source code modules.

This report should be used as a source for any future independent assessments of the URET algorithms; particularly for sensitivity or algorithmic accuracy assessments. The assessment of the design and implementation of the algorithms should be completed for the current version of the URET prototype and this version of the software should be rigorously tested by an independent assessment group.

Acknowledgments

This report consolidates the assessment efforts of the ACT-250 AERA/URET prototype assessment team. Each member of the team focused on one of the core URET algorithms, but the overall activity was a combined team effort. Each member analyzed and assessed numerous functions within their assigned algorithm area, assisted team members in other areas when necessary, and wrote the necessary material for the corresponding sections of this document. The cooperative support provided by MITRE/CAASD facilitated the accomplishment of this effort.

Steve Kazunas, CTA Incorporated, focused on the assessment of the Trajectory Modeler and the Track Management, and associated utility functions. Mike Paglione, Signal Corporation, focused on the Automated Problem Detection and associated utility functions. Dr. Hollis Ryan, Signal Corporation, focused on the coordinate conversion functions and provided support in all three algorithm areas. Bernie Day, Signal Corporation, provided invaluable support towards developing the simulations originally planned as part of this assessment effort. Mary Lee Cale, FAA/ACT-250, led the task and coordinated team activities and MITRE/CAASD support, and also focused on the Track Management algorithm.

Table of Contents

1. Introduction	1
1.1 Purpose	1
1.2 Background	1
1.3 Scope	1
1.4 Document Organization	1
2. Assessment Overview	2
2.1 Algorithm Analysis	2
2.2 Simulation	2
3. Algorithm Descriptions	4
3.1 Automated Problem Detection	5
3.1.1 Function: CFP_COARSE_HORIZ (C)	5
3.1.2 Function: CFP_FINE (C)	18
3.1.3 Function: CFP_INTERSECT_TIME (C)	25
3.1.4 Function: CFP_MIDDLE_HORIZ (C)	27
3.1.5 Function: CFP_MIDDLE_VERT (C)	33
3.1.6 Function: CFP_RELVEC (C)	40
3.1.7 Function: CFP_TRIM (C)	43
3.1.8 Function: CFP_V_INT (C)	43
3.1.9 Function: CFP_POSIT (C)	48
3.1.10 Function: ECP (PL/I)	52
3.2 Trajectory Modeler	52
3.2.1 Function: ARDXY (PL/I)	54
3.2.2 Function: EGRAD (PL/I)	55
3.2.3 Function: INTMDL (PL/I)	57
3.2.4 Function: HRB (PL/I)	58
3.3 Track Management	60
3.3.1 Function: CNV_GRD_TO_TAS (PL/I)	61
3.3.2 Function: GM_PTSEG (PL/I)	62
3.3.3 Function: LEASTSQ (C)	62
3.3.4 Function: TKM_CATEGORY_CHANGE (C)	63
3.3.5 Function: TKM_CHECK_AIRSPACE (C)	64
3.3.6 Function: TKM_COMPUTE_RECONFORMANCE (C)	64
3.3.7 Function: TKM_CONFORMANCE_MONITOR (C)	65
3.3.8 Function: TKM_CONTROL (PL/I)	66
3.3.9 Function: TKM_CORE (C)	67
3.3.10 Function: TKM_FOR_AERA (PL/I)	68
3.3.11 Function: TKM_GET_RTE_ORIS (PL/I)	68
3.3.12 Function: TKM_GM_REGN (C)	68
3.3.13 Function: TKM_GM_TSTPNT (C)	72
3.3.14 Function: TKM_MATCH_ID (PL/I)	76
3.3.15 Function: TKM_TK_HDG (C)	76
3.3.16 Function: TKM_VERIFY_DATA (C)	78
3.3.17 Function: UTL_XY_ARD_BY_RTE (PL/I)	78
3.4 General Purpose Utilities	78
3.4.1 Function: CNV_CNVSPD (PL/I)	79
3.4.2 Function: CNV_GNOMONIC_STEREO (PL/I)	79
3.4.3 Function: CNV_GRDSPD (PL/I)	83
3.4.4 Function: CNV_LLXY (PL/I)	85
3.4.5 Function: CNV_RADDMS (PL/I)	92
3.4.6 Function: CNV_SPEED (C)	94
3.4.7 Function: CNV_STD_ATMOS (PL/I)	100
3.4.8 Function: CNV_STEREO_GNOMONIC (PL/I)	104

3.4.9 Function: CNV_XYLL (PL/I).....	108
3.4.10 Function: DB_AIR_AT_POINT (PL/I)	119
3.4.11 Function: DB_CDMERG (PL/I)	119
3.4.12 Function: DB_FIND_AUD_PTR (PL/I)	120
3.4.13 Function: GM_BRNG (PL/I)	120
3.4.14 Function: GM_CONVEX (C)	120
3.4.15 Function: GM_INSEC (C)	125
3.4.16 Function: GM_PTLNE (PL/I).....	131
3.4.17 Function: GM_REGN (PL/I).....	133
3.4.18 Function: GM_TSTPNT (PL/I).....	137
3.4.19 Function: GM_TURN (PL/I).....	140
3.4.20 Function: LO_FIND (PL/I)	141
3.4.21 Function: ST_ARD_SSGDATA (PL/I)	141
3.4.22 Function: ST_CHK_VP (PL/I).....	144
3.4.23 Function: ST_CLIMB_DIST (PL/I).....	148
3.4.24 Function: ST_CLIMB_GRADIENT (PL/I)	150
3.4.25 Function: ST_DESCENT_DIST (PL/I)	150
3.4.26 Function: ST_DESCENT_GRADIENT (PL/I).....	153
3.4.27 Function: ST_FINDARD (PL/I).....	154
3.4.28 Function: ST_IASALT (PL/I).....	156
3.4.29 Function: ST_MACHALT (PL/I).....	165
3.4.30 Function: ST_MAXTAS (PL/I)	172
3.4.31 Function: ST_MINTAS (PL/I)	172
3.4.32 Function: ST_TIME_SSGDATA (PL/I)	173
3.4.33 Function: ST_TRANSLATE_ARD (PL/I).....	176
3.4.34 Function: ST_XYTOTIME (PL/I)	178
4. Assessment Findings and Observations	181
4.1 URET D1.1 Limitations	181
4.2 URET D1.1 Assumptions.....	181
4.3 Summary of Algorithmic Assessment Tables	183
4.4 Suggested Improvements.....	188
4.4.1 CFP_COARSE_HORIZ.....	188
4.4.2 CFP_MIDDLE_HORIZ	188
4.4.3 CFP_V_INT	188
4.4.4 CNV_LLXY	189
4.4.5 CNV_SPEED	189
4.4.6 CNV_XYLL.....	189
4.4.7 GM_CONVEX.....	189
4.4.8 GM_INSEC.....	189
4.4.9 GM_PTLNE.....	190
4.4.10 ST_CHK_VP.....	190
4.4.11 ST_FIND_ARD.....	190
4.4.12 ST_IASALT	190
4.4.13 ST_TIME_SSGDATA	191
4.4.14 TKM_GM_REGN.....	192
4.4.15 TKM_GM_TSTPNT	192
4.4.16 TKM_TK_HDG	193
4.4.17 TKM_GET_RTE_ORIS	193
4.4.18 UTL_XY_ARD_BY_RTE.....	193
4.5 Conclusions.....	193
Appendix A: Simulation Experimentation	195
Appendix B: LIST OF ACRONYMS	204
REFERENCES.....	205

Table of Figures

Figure 3.1.1-1: Example of Strips for Horizontal Coarse Filter.....	6
Figure 3.1.1-2: Example of Case 1.....	8
Figure 3.1.1-3: Example of Case 2 (distance calculated for one aircraft holding).....	11
Figure 3.1.1-4: Coarse Horizontal Function Overall Logic.....	16
Figure 3.1.2-1: Diagram of examples of each case where relative velocity is non-zero.....	20
Figure 3.1.2-2: Initial Steps in CFP_FINE.....	21
Figure 3.1.2-3: Flowchart where P is inside octagon only (continued from Figure 3.1.2-2).....	22
Figure 3.1.2-4: Flowchart where Q is inside octagon only (continued from Figure 3.1.2-2).....	23
Figure 3.1.2-5: Flowchart where both P and Q are outside octagon (continued from Figure 3.1.2-2).....	24
Figure 3.1.4-1: Relative Geometry with Circular Conformance Boundary.....	27
Figure 3.1.4-2: Rotating P vector to axis frame of subject aircraft.....	29
Figure 3.1.4-3: Relative velocity.....	30
Figure 3.1.5-1: Cases of End Point Intersections.....	35
Figure 3.1.5-2: Situations for calling CFP_V_INT.....	36
Figure 3.1.5-3: Equation 3.1.5-2 > 0 for both situations.....	36
Figure 3.1.5-4: Crossing Trajectories for Case 6.....	37
Figure 3.1.5-5: Middle Vertical Filter Logic Flowchart.....	38
Figure 3.1.6-1: Rotating P vector to axis frame of subject aircraft.....	41
Figure 3.1.6-2: Relative velocity.....	42
Figure 3.1.8-1: Example of lower bound trajectory and tbend.....	45
Figure 3.1.8-2: Examples of mesh point combinations for each case.....	46
Figure 3.1.8-3: Mesh point intersection (altitude vs. time).....	47
Figure 3.2.4-1: Example of an HRB Return-to-Route.....	60
Figure 3.3.12-1: Logic Flow of TKM_GM_REGN.....	71
Figure 3.3.13-1: Diagram of test point to line distance.....	73
Figure 3.4.2-1: Gnomonic to Stereographic Projection.....	82
Figure 3.4.4-1: Mapping Geometry - Geodetic to Conformal.....	90
Figure 3.4.4-2: Stereographic Projection.....	91
Figure 3.4.7-1: Temperature Distribution in the Standard Atmosphere.....	100
Figure 3.4.8-1: Stereographic to Gnomonic Projection.....	107
Figure 3.4.9-1: Stereographic Projection Details.....	116
Figure 3.4.9-2: Mapping Geometry - Conformal to Geodetic.....	117
Figure 3.4.9-3: Spherical Triangle on the Conformal Sphere.....	118
Figure 3.4.14-1: Side of octagon and test point.....	122
Figure 3.4.14-2: GM_CONVEX Main Function Loop.....	124
Figure 3.4.15-1: Example of area diagram for δ 's.....	126
Figure 3.4.15-2: Ratio of Distance B to A.....	129
Figure 3.4.17-1: Logic of GM_REGN.....	136
Figure 3.4.18-1: Diagram of test point to line distance.....	137
Figure 3.4.27-1: Diagram of closest approach point (x_i, y_i) which is on the flight segment.....	155

Table of Tables

Table 3.1.1-1: Boolean Variable Table	14
Table 3.1.5-1: Logical Vertical Intersection Cases	34
Table 3.1.5-2: Beginning and End Conflict Variables	35
Table 3.1.8-1: Mesh point combinations	46
Table 3.3.12-1: Iteration key TKM_GM_REGN	70
Table 3.4.17-1: Iteration key GM_REGN	135

1. Introduction

1.1 Purpose

This report presents the results of an independent assessment of the logic and mathematics within the User Request Evaluation Tool (URET) core algorithms. This assessment was conducted by the Traffic Flow Management Branch (ACT-250) at the Federal Aviation Administration (FAA) William J. Hughes Technical Center.

1.2 Background

URET is an automated conflict detection (ACD) tool intended for use as a decision support aid for the en route air traffic controller. URET detects aircraft-to-aircraft and aircraft-to-airspace conflicts for IFR aircraft tracked by the Host Computer System (HCS), and provides alert information to the controller when such conflicts are detected.

The technical performance and accuracy of the URET algorithms are critical issues to be assessed in preparation for a Joint Resources Council (JRC) investment decision for an ACD tool. MITRE/CAASD has been evaluating the accuracy and performance of the URET algorithms throughout the URET development effort as well as during the field evaluations conducted at the Indianapolis Air Route Traffic Control Center (ARTCC - ZID) in FY96/97. ACT-250 was tasked by the Air Traffic Management (ATM) Prototype Product Team (AUA-540) to provide an independent assessment of the technical accuracy of the core URET algorithms. A URET system was installed in the Terminal Air Traffic Control Automation (TATCA)/Automated En Route Air Traffic Control (AERA) laboratory at the FAA William J. Hughes Technical Center in early 1996. ACT-250 utilized this system and the capabilities of other Technical Center laboratories in accomplishing this assessment. The cooperative support provided by MITRE/CAASD facilitated the accomplishment of this effort.

1.3 Scope

The scope of the FY96/97 effort, as discussed in this report, was limited to the core algorithms implemented in URET Delivery 1.1 (D1.1) (installed in ZID in May 1996). This includes the Trajectory Modeler (TJM), Track Management (TKM), and Automated Problem Detection (APD) algorithms.

Although a testbed was established in the TATCA/AERA laboratory, the completion of the algorithm assessments and their validation via structured simulations using this testbed, as originally planned, was curtailed in late 1996 when ACT-250 efforts were redirected to focus on preparing for the comparison of Conflict Probe prototypes (e.g., URET and NASA's User Preferred Routing (UPR) system) currently planned for mid-1997. Consequently, while the majority of the key functions have been assessed, this report does not contain an assessment of every URET function. This report does provide a description of major functions comprising each algorithm set, including variable definitions and mathematical equations, along with an assessment of the assumptions/approximations and their impact on the accuracy of the algorithm, and identifies URET D1.1 assumptions and limitations and suggested improvements.

1.4 Document Organization

This report is organized into four sections. Section 2 provides an overview of ACT-250's independent assessment approach. Section 3 presents detailed descriptions of software modules comprising each of the core algorithm sets and associated utility functions. Section 4 presents the assessment findings and observations, including algorithmic assumptions and limitations, and suggested improvements. An example of a proposed simulation design and scenarios to further evaluate the algorithms is presented for information in Appendix A, and a list of acronyms is provided in Appendix B. A list of references used during this activity is provided at the end of the document.

2. Assessment Overview

The ACT-250 independent assessment effort was based on determining the validity of the URET algorithms and verifying the engineering principles upon which the algorithms were established. ACT-250's initial approach was to review MITRE/CAASD's algorithmic documentation and the applicable source code. By taking this approach, ACT-250 became very knowledgeable about the algorithms' details, and the approximations and assumptions that were made during the URET development. ACT-250 used this acquired knowledge and the URET system in the TATCA/AERA laboratory at the Technical Center to design various simulated exercises to "push the envelope" on the constraints established by the identified assumptions, approximations and parameter constraints (however, the conduct of these simulations was not completed as discussed in Section 1.3). A testbed has been established in the TATCA/AERA laboratory where independent evaluations of future URET prototype enhancements can be conducted. In addition, ACT-250 now has an in-depth understanding of many critical areas of the URET algorithms where future analysis should be focused.

2.1 Algorithm Analysis

ACT-250 conducted a detailed analysis of the algorithms, constraints, and assumptions comprising the URET Delivery 1.1 system. This analysis was based on:

1. a comprehensive study of the existing MITRE/CAASD algorithmic documentation, software design data and the URET source code,
2. deriving many of the mathematical constructs represented in the URET source code,
3. technology transfer meetings with the MITRE/CAASD developers, and
4. unit testing of specific algorithmic functions¹.

During this analysis period, ACT-250 documented both the algorithmic functions' derivation in generic mathematical terms, and the assumptions and approximations made by MITRE/CAASD during the development of the URET algorithms (see Section 3). The assumptions and approximations are summarized in matrix form in Section 4.3. These detailed algorithmic function descriptions should prove extremely useful to a production contractor.

ACT-250 concentrated on the lowest level details of the algorithms, thus, many of the higher level functions of the algorithmic subsets (which serve to control logic flow or manipulate data base elements) are not included. Many of these high level functions are adequately described in the MITRE/CAASD documentation. In addition, a majority of the algorithmic calculations are actually performed by a library of utility functions; therefore, much of this report's analysis for TJM and TKM is actually found in the assessment of the appropriate utility functions (see Section 3.4).

As a side-product of this effort, the adequacy of the technical documentation available with the URET prototype development was assessed. During the time period that the algorithm analysis was being conducted, MITRE/CAASD delivered updated algorithmic documentation for review. ACT-250 thoroughly reviewed this documentation and provided detailed comments to MITRE/CAASD and to AUA-540, including an assessment of the adequacy (i.e., accuracy, clarity, consistency and completeness) of the documentation for use by a production contractor.

2.2 Simulation

The original plan for this independent assessment effort called for an evaluation of the accuracy of the algorithms implemented in the URET D1.1 software to be conducted in the TATCA/AERA laboratory.

¹ Limited Unit Testing was performed on the following functions: CNV_LLXY, CNV_XYLL, ST_MACHALT, ST_IASALT, CFP_POSIT, CFP_V_INT, CNV_SPEED, GM_REGN, GM_TSTPNT, ST_FINDARD, GM_CONVEX, GM_INSEC.

The intent was to demonstrate URET parameter constraints, and validate assumptions, via structured simulation scenarios designed to exercise URET algorithms under various conditions. While the testbed was established in the laboratory, the conduct of the simulations was curtailed because of AUA-540 redirection of ACT-250 efforts to focus on the comparison of en route Conflict Probe prototypes (planned for mid-1997). Current planning for this effort calls for the URET system in the TATCA/AERA laboratory to be adapted to Cleveland ARTCC (ZOB) for use in conducting this comparison. The proposed simulation design for the original activity is provided for information in Appendix A.

3. Algorithm Descriptions

The following sections contain descriptions of many of the major functions comprising each of the core algorithm sets and the general purpose utility functions. Where appropriate*, each function is described in terms of the variable definitions and mathematical equations, along with ACT-250's assessment of the assumptions/approximations, and the impact of these factors on the accuracy of the algorithm. Each section's organization reflects the general directory structure of the URET source code; within each subsection, the individual functions are listed in alphabetical order. In addition, the programming language in which the function was coded (either **C** or **PL/I**) is identified in parentheses in each subsection heading.

An Assessment Table is provided at the end of each section, where appropriate* (these tables are summarized in Section 4.3). The table consists of the following elements:

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on (Algorithm)
R 3.1.1-1	Description of Approximation/Assumption	Description of ACT-250's assessment of the Approximation/Assumption	ACT-250's determination of the Impact (defined below) of this Approximation/ Assumption on the specified algorithm

Unique number identifying specific assessment item within section

Section number of functional assessment

IMPACT:

- **Critical** - if conditions existed which were determined to be unfavorable to the assumption or approximation, there would be a significant impact on the accuracy or stability of the algorithm. Some of the criteria the analyst considered in classifying an assumption/approximation as **critical** were:
 - the module is called many times from many different areas of the code
 - the module calculates a core value which is used as a basis for many subsequent calculations
 - the assumption/approximation is over-simplified and would be inadequate during many normal operating conditions
 - there are no other identifiable corrective measures elsewhere in the source code which would compensate for the inaccuracy of the assumption/approximation

- **Important** - if conditions existed which were determined to be unfavorable to the assumption or approximation, there could be an impact on the accuracy or stability of the algorithm. Some of the criteria the analyst considered in classifying an assumption/approximation as **important** were:
 - the module calculates a value which is used as a basis for subsequent calculations
 - the assumption/approximation is simplified and would be inadequate during some reasonable operating conditions
 - there may be corrective measures elsewhere in the source code which could compensate for some of the inaccuracy of the assumption/approximation, but they may not be sufficient

* High level functions that control logic flow or manipulate the data base may not be described at this level (many of these high level functions are adequately described in the MITRE/CAASD documentation which ACT-250 thoroughly reviewed for completeness and accuracy as part of this task).

- **Minor** - if conditions existed which were determined to be unfavorable to the assumption or approximation, there would be little or no impact on the accuracy or stability of the algorithms. Some of the criteria the analyst considered in classifying an assumption/approximation as **minor** were:
 - the assumption/approximation is based on classic, proven methods necessary for real time processing
 - there are corrective measures elsewhere in the source code which could compensate for some of the inaccuracy of the assumption/approximation

3.1 Automated Problem Detection

Automated Problem Detection (APD) detects aircraft-to-aircraft and aircraft-to-airspace conflicts within a parameter look-ahead time. It is initiated for a given aircraft when: the aircraft enters the ARTCC, a remodeling of the aircraft's trajectory occurs, or a controller invokes the trial planning function.

To balance efficiency with accuracy, a series of filters are activated which serve to narrow the range of aircraft undergoing each problem detection check. This range is narrowed further, since only three of the six URET categories of aircraft are probed. For each pair of candidate aircraft trajectories, the separation standards and regions of uncertainty built around the aircraft, called conformance bounds, are utilized to predict each conflict situation.

Those functions which are associated with aircraft-to-aircraft conflict detection comprise the conflict probe (CFP) directory. Those functions which are associated with aircraft-to-airspace conflict detection comprise the environmental conflict probe (ECP) directory. Low level functions which perform some of the algorithmic calculations are described in the library of utility functions in Section 3.4 (GM_INSEC, GM_CONVEX, ST_CHK_VP, GM_REGN, GM_TSTPNT).

Conflict Probe (CFP)

The functions comprising CFP detect aircraft-to-aircraft conflicts and are applied from the current aircraft position to a parameter lookahead time into the future (D1.1: 20 minutes).

Environmental Conflict Probe (ECP)

The Environmental Conflict Probe (ECP) is the function subset of APD that determines if an aircraft is in conflict with Special Use Airspace (SUA). The function is applied from the current aircraft's position to the end of the trajectory calculated by TJM.

3.1.1 Function: CFP_COARSE_HORIZ (C)

This function eliminates from consideration those aircraft that have trajectory segments so far apart in the (x, y) plane that a potential conflict can be ruled out.

3.1.1.1 Description:

With the two input line segments in the (x, y) plane, the function is a simple filter that determines if the segments get within a fixed distance M of each other. The M value is equal to the sum of both conformance bounds plus the separation standard. Referring to Figure 3.1.1-1, the approach is to create infinite strips of width 2*M. The strips are centered around the line segments. The function checks for an intersection of a segment's strip with the opposite line segment. For a segment not to intersect the strip it is necessary and sufficient that both endpoints of the segment lie on the same side of the centerline of the strip and not be within distance M of that centerline.

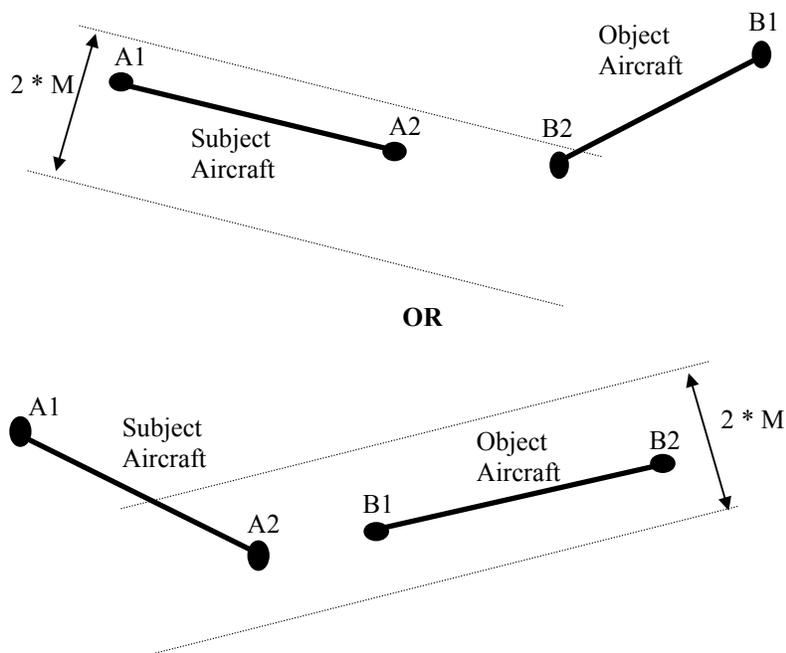


Figure 3.1.1-1: Example of Strips for Horizontal Coarse Filter
(here both segments are in conflict with the other's strip)

The algorithm checks the four following potential cases:

1. Neither aircraft are in a hold pattern.
2. The subject aircraft (A) is in the holding pattern and the object aircraft (B) is not.
3. The object aircraft (B) is in the holding pattern and the subject aircraft (A) is not.
4. Both subject and object aircraft are in hold.

Each case is evaluated to determine the distance from the opposite line segment's center line. If the distance is greater than the separation standard plus both conformance bound distances, no conflict can take place. If the distance is less, the aircraft pair may or may not be in conflict.

Table of Variable Definitions

Function Variable	Description	Math Symbol
m	critical distance including conformance radius of subject aircraft and object aircraft plus separation ²	M
m2	M^2 or $M * M$	M^2
g1	numerator of signed distance from beginning of subject segment to line through object segment; = $x_{s1}(y_{o2} - y_{o1}) + x_{o2}(y_{o1} - y_{s1}) + x_{o1}(y_{s1} - y_{o2})$	G_1
g2	numerator of signed distance from end of subject segment to line through object segment; = $x_{s2}(y_{o2} - y_{o1}) + x_{o2}(y_{o1} - y_{s2}) + x_{o1}(y_{s2} - y_{o2})$	G_2
h1	numerator of signed distance from beginning of object segment to line through subject segment; = $x_{o1}(y_{s2} - y_{s1}) + x_{s2}(y_{s1} - y_{o1}) + x_{s1}(y_{o1} - y_{s2})$	H_1
h2	numerator of signed distance from end of object segment to line through subject segment; = $x_{o2}(y_{s2} - y_{s1}) + x_{s2}(y_{s1} - y_{o2}) + x_{s1}(y_{o2} - y_{s2})$	H_2
u1	signed perpendicular distance from A1 to the line through B1 and B2; $U_1 = G_1 / \sqrt{d_o^2}$	U_1
u2	signed perpendicular distance from A2 to the line through B1 and B2; $U_2 = G_2 / \sqrt{d_o^2}$	U_2
z1	signed perpendicular distance from B1 to the line through A1 and A2; $Z_1 = H_1 / \sqrt{d_s^2}$	Z_1
z2	signed perpendicular distance from B2 to the line through A1 and A2; $Z_2 = H_2 / \sqrt{d_s^2}$	Z_2
g3	alternate numerator for h1, used when subject is in hold; = $(y_{o1} - y_{s1})(y_{o1} - y_{o2}) + (x_{o1} - x_{s1})(x_{o1} - x_{o2})$	G_3
g4	alternate numerator for h2, used when subject is in hold; = $(y_{o2} - y_{s1})(y_{o1} - y_{o2}) + (x_{o2} - x_{s1})(x_{o1} - x_{o2})$	G_4
h3	alternate numerator for h1, used when subject is in hold; = $(y_{s1} - y_{o1})(y_{s1} - y_{s2}) + (x_{s1} - x_{o1})(x_{s1} - x_{s2})$	H_3
h4	alternate numerator for h1, used when subject is in hold; = $(y_{s2} - y_{o1})(y_{s1} - y_{s2}) + (x_{s2} - x_{o1})(x_{s1} - x_{s2})$	H_4
denomo	object segment distance ² = $(x_{o2} - x_{o1})^2 + (y_{o2} - y_{o1})^2$; denominator	d_o^2
denoms	subject segment distance ² = $(x_{s2} - x_{s1})^2 + (y_{s2} - y_{s1})^2$; denominator	d_s^2

² The conformance radius of an aircraft is the hypotenuse of the right triangle formed by half the longitudinal and lateral conformance bounds (i.e. $2.92nm = \sqrt{1.5nm^2 + 2.5nm^2}$).

3.1.1.2 Mathematics

To compare the aircraft's potential separation against the distance M, the function calculates the distance formed by a perpendicular line drawn from the given aircraft's endpoint to the line formed by the opposite aircraft's endpoints. The following discussion will provide derivations on each potential case of endpoint to line perpendicular distances.

Case 1

For a Case 1 example, neither aircraft are in a holding pattern, so they both have defined line segments. Figure 3.1.1-2 illustrates an example of a Case 1 scenario. For this scenario, the function determines the distance of the perpendicular line drawn from the subject aircraft's cusp A1 to the object aircraft's line.

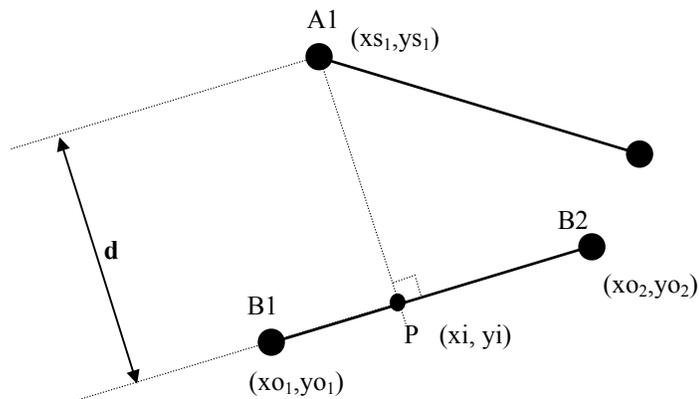


Figure 3.1.1-2: Example of Case 1

Derivation of the distance d in the Figure 3.1.1-2 above:

First it is necessary to define the equations for the lines of both object and subject aircraft. The object aircraft's line equation can be expressed by the general line equation:

$$L: Ax + By + C = 0 \quad \text{Equation 3.1.1-1}$$

The slope of this line L is $-A/B$, which is the slope of the line through the point P in Figure 3.1.1-2. If the line is neither vertical or horizontal, the perpendicular line through the object's line equation, L, is $+B/A$. The equation of the perpendicular line using the point slope formula is:

$$(y - ys_1) = \left(\frac{B}{A}\right)(x - xs_1) \Leftrightarrow Bx - Ay + Ays_1 - Bxs_1 = 0 \quad \text{Equation 3.1.1-2}$$

To find the intersection point of the line L and the perpendicular line through A1, we solve the following equations simultaneously:

$$Ax + By + C = 0 \text{ and } Bx - Ay + Ays_1 - Bxs_1 = 0 \quad \text{Equation 3.1.1-3}$$

In Equation 3.1.1-3 above by multiplying the first equation by A and the second by B, and then solving for x by adding them together, yields:

$$xi = \frac{B^2 xs_1 - ABys_1 - AC}{A^2 + B^2}, \quad \text{Equation 3.1.1-4}$$

By multiplying the first equation by B and the second by A and then subtracting, yields:

$$yi = \frac{A^2 ys_1 - ABxs_1 - BC}{A^2 + B^2} \quad \text{Equation 3.1.1-5}$$

The distance, d , between the point at A1 and the point at (xi, yi) according to the distance formula.

$$d = \sqrt{(xs_1 - xi)^2 + (ys_1 - yi)^2} \quad \text{Equation 3.1.1-6}$$

Substitution of Equation 3.1.1-4 and Equation 3.1.1-5, yields:

$$d^2 = \left[xs_1 - \frac{B^2 xs_1 - ABys_1 - AC}{A^2 + B^2} \right]^2 + \left[ys_1 - \frac{A^2 ys_1 - ABxs_1 - BC}{A^2 + B^2} \right]^2 \quad \text{Equation 3.1.1-7}$$

Now, combine the terms to yield:

$$d^2 = \left[\frac{A^2 (Axs_1 + Bys_1 + C)^2}{(A^2 + B^2)^2} \right] + \left[\frac{B^2 (Axs_1 + Bys_1 + C)^2}{(A^2 + B^2)^2} \right] \quad \text{Equation 3.1.1-8}$$

Finally, by combining the terms further and taking the square root we get the distance formula:

$$d = \frac{Axs_1 + Bys_1 + C}{\sqrt{A^2 + B^2}}$$

$$d^2 = \frac{(Axs_1 + Bys_1 + C)^2}{A^2 + B^2} \quad \text{Equation 3.1.1-9}$$

Note: The function actually does not calculate the square root for efficiency purposes, but uses the square of the distance (d^2) in the actual algorithm, as in Equation 3.1.1-9.

As defined in the function, the square root of the numerator in Equation 3.1.1-9 can be expressed using a 3x3 matrix determinant and expanded as follows:

$$G_1 = \det \begin{vmatrix} xs_1 & xo_2 & xo_1 \\ ys_1 & yo_2 & yo_1 \\ 1 & 1 & 1 \end{vmatrix}$$

$$G_1 = xs_1 \begin{vmatrix} yo_2 & yo_1 \\ 1 & 1 \end{vmatrix} - xo_2 \begin{vmatrix} ys_1 & yo_1 \\ 1 & 1 \end{vmatrix} + xo_1 \begin{vmatrix} ys_1 & yo_2 \\ 1 & 1 \end{vmatrix}$$

$$G_1 = xs_1(yo_2 - yo_1) + xo_2(yo_1 - ys_1) + xo_1(ys_1 - yo_2) \quad \text{Equation 3.1.1-10}$$

To express the numerator as G_1 as in Equation 3.1.1-10, return to the point-slope equation of the line of the object aircraft.

$$y = m(x) - m(xo_1) + yo_1^*$$

$$y = \left(\frac{yo_2 - yo_1}{xo_2 - xo_1} \right) x - \left(\frac{yo_2 - yo_1}{xo_2 - xo_1} \right) xo_1 + yo_1 \quad \text{Equation 3.1.1-11}$$

By rearranging the terms in Equation 3.1.1-11, the equation can provide the A, B, and C terms of the general equation.

$$y = \left(\frac{(yo_2 - yo_1)}{-(xo_2 - xo_1)} \right) (-x) - \left(\frac{-(yo_2 - yo_1)xo_1 + yo_1(xo_2 - xo_1)}{-(xo_2 - xo_1)} \right)$$

The equation above can be expressed as:

$$y = -\frac{A}{B}x - \frac{C}{B}, \text{ where } A = (yo_2 - yo_1), B = -(xo_2 - xo_1), \text{ and}$$

$$C = -(yo_2 - yo_1)xo_1 + yo_1(xo_2 - xo_1)$$

Therefore, using the terms above, the square root of the numerator in Equation 3.1.1-9 can be shown to be equivalent to Equation 3.1.1-10.

* The slope, m, for the object aircraft line is $\left(\frac{yo_2 - yo_1}{xo_2 - xo_1} \right)$.

$$Axs_1 + Bys_1 + C$$

$$(yo_2 - yo_1)xs_1 + - (xo_2 - xo_1)ys_1 + -(yo_2 - yo_1)xo_1 + yo_1(xo_2 - xo_1)$$

$$(yo_2 - yo_1)xs_1 + -ys_1xo_2 + ys_1xo_1 + -xo_1yo_2 + xo_1yo_1 + yo_1xo_2 + -yo_1xo_1$$

$$xs_1(yo_2 - yo_1) + xo_2(yo_1 - ys_1) + xo_1(ys_1 - yo_2)$$

For this case, Equation 3.1.1-12 expresses the denominator in Equation 3.1.1-9 from the formula $(A^2 + B^2)$. It represents the distance of the object aircraft's segment from cusp A1 to A2.

$$d_o^2 = (xo_2 - xo_1)^2 + (yo_2 - yo_1)^2 \quad \text{Equation 3.1.1-12}$$

The derivation is similar for the other endpoints and from the object aircraft to the subject aircraft.

Cases 2 and 3

For Cases 2 and 3, *one of the two aircraft is in a holding pattern*, making their line segment a point. Since there are an infinite number of perpendicular lines from the given aircraft to the holding aircraft, the function uses the perpendicular line drawn from the given point and the point forming the intersection. This intersection point is the endpoint of the perpendicular line drawn from the holding point to the opposite line.

For example, for Case 2 the subject aircraft is on hold, so the signed perpendicular distance from the object aircraft's first cusp to the subject aircraft is indeterminate. As illustrated in Figure 3.1.1-3 below, the distance calculated by the function is the signed perpendicular distance from the object aircraft's first endpoint to the intersection point, P. The point, P, is formed by the intersection of the line from the subject aircraft's holding point to the object aircraft's line.

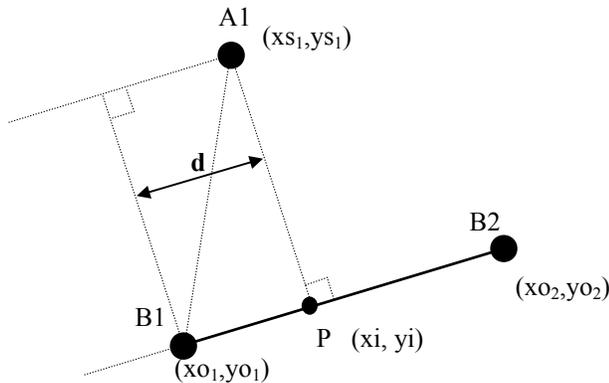


Figure 3.1.1-3: Example of Case 2 (distance calculated for one aircraft holding)

Derivation of the distance, d, in Figure 3.1.1-3:

The triangle formed by the points B1, A1, and P can be used to determine the distance, d , in Figure 3.1.1-3. Using the Pythagorean Theorem and the formulas listed in the function, the relationship of the triangle's side distances can be expressed as:

$$c^2 = a^2 + b^2$$

$$R^2 = (G_1^2 + G_3^2) / d_o^2 \quad \text{Equation 3.1.1-13}$$

Where, R = distance of line from B1 to A1, diagonal of triangle,
 G_1/d_o = distance from A1 to P as defined in previously,
 G_3/d_o = distance from B1 to P as defined in the function code

G_3 is the numerator in the distance equation above and is defined by the function as:

$$G_3 = (y_{o1} - y_{s1})(y_{o1} - y_{o2}) + (x_{o1} - x_{s1})(x_{o1} - x_{o2}) \quad \text{Equation 3.1.1-14}$$

The distance from B1 to P (or d), as defined in the function, is the numerator G_3 divided by the squared root of the denominator, d_o^2 . Using Equation 3.1.1-13 we will derive Equation 3.1.1-14 or G_3 and the distance d , but first a few more terms must be defined.

R is the distance of the line segment from B1 to A1 and can be expressed by the general Euclidean distance equation as:

$$R = \sqrt{(x_{o1} - x_{s1})^2 + (y_{o1} - y_{s1})^2} \quad \text{Equation 3.1.1-15}$$

The numerator and denominator of the distance from A1 to P, as derived previously, is restated here as:

$$G_1 = x_{s1}(y_{o2} - y_{o1}) + x_{o2}(y_{o1} - y_{s1}) + x_{o1}(y_{s1} - y_{o2}) \quad \text{Equation 3.1.1-16}$$

$$d_o^2 = (x_{o2} - x_{o1})^2 + (y_{o2} - y_{o1})^2 \quad \text{Equation 3.1.1-17}$$

By substitution, the Equation 3.1.1-13 can be expressed as:

$$\begin{aligned} d_o^2 R^2 &= G_1^2 + G_3^2 \\ &= \left[(x_{o1} - x_{s1})^2 + (y_{o1} - y_{s1})^2 \right] \left[(x_{o2} - x_{o1})^2 + (y_{o2} - y_{o1})^2 \right] = \\ &= \left[x_{o1}(y_{s1} - y_{o2}) - y_{o1}(x_{s1} - x_{o2}) + (x_{s1}y_{o2} - x_{o2}y_{s1}) \right]^2 \\ &= \left[(y_{o1} - y_{s1})(y_{o1} - y_{o2}) + (x_{o1} - x_{s1})(x_{o1} - x_{o2}) \right]^2 \end{aligned} \quad \text{Equation 3.1.1-18}$$

To simplify the algebraic manipulation of Equation 3.1.1-18, assign A, B, M, N to the following values:

$$A = (y_{o2} - y_{o1}); B = -(x_{o2} - x_{o1}); M = (x_{o1} - x_{s1}); N = (y_{o1} - y_{s1})$$

Now, substitute these terms into Equation 3.1.1-18, to get:

$$d_o^2 R^2 = G_1^2 + G_3^2$$

$$(A^2 + B^2)(N^2 + M^2) = (NM - MA)^2 + (-NA - MB)^2 \quad \text{Equation 3.1.1-19}$$

It can easily be shown that in Equation 3.1.1-19 G_3 is represented as $(-NA - MB)$, but it must be shown that the terms in Equation 3.1.1-19 can be expanded to confirm $G_3^2 = d_o^2 R^2 - G_1^2$. In other words, we will solve for G_3 in Equation 3.1.1-13 represented in the terms of Equation 3.1.1-19 for simplification. If the value of G_3 can be expressed in terms of $d_o^2 R^2 - G_1^2$, then G_3 has been effectively derived by Equation 3.1.1-13. Therefore, by moving G_1 to the other side and squaring the terms, Equation 3.1.1-19 reduces to the G_3 equivalent to the function's definition (Equation 3.1.1-14).

$$(A^2 + B^2)(N^2 + M^2) - (NM - MA)^2 = (-NA - MB)^2$$

$$(N^2 A^2 + M^2 A^2 + N^2 B^2 + N^2 M^2 - N^2 B^2 + 2MNAB - M^2 A^2) = (-NA - MB)^2$$

$$(N^2 A^2 + 2MNAB + M^2 A^2) = (N^2 A^2 + 2MNAB + M^2 A^2)$$

Like Case 1, the same derivation can be applied to the other endpoints and from the object aircraft to the subject aircraft.

Case 4:

For the case where both subject and object aircraft are in a holding pattern, the function calculates the Euclidean plane distance between the two points. The general distance formula is used.

$$\Omega = \sqrt{(x_{o1} - x_{s1})^2 + (y_{o1} - y_{s1})^2} \quad \text{Equation 3.1.1-20}$$

Note: Once again the square root is not calculated in the function due to efficiency, but the Ω^2 is compared against M^2 .

3.1.1.3 Boolean Logic

The function uses five Boolean variables to decide whether the filter passes the aircraft pair (conflict could exist) or rejects the aircraft pair (no conflict could exist). These Boolean variables are defined as follows:

Variable	Description	Formula
L1	subject segment does not cross line containing object segment	$U1*U2>0$
L2	object segment does not cross line containing subject segment	$Z1*Z2>0$
L3	ends of subject segment close to line containing object segment	$U1^2<M^2$ OR $U2^2<M^2$
L4	ends of object segment close to line containing subject segment	$Z1^2<M^2$ OR $Z2^2<M^2$
L5	throw out the segment pair	(L1 and not L3) OR (L2 and not L4)

Table 3.1.1-1: Boolean Variable Table

The overall function's logic and how the Boolean variables are used is illustrated in Figure 3.1.1-4. First, the function determines if both of the aircraft are in hold. If both are in hold, the calculation is relatively simple. Both aircraft are at points, so the squared distance between them is calculated and compared against the strip width M^2 . This result is stored in Boolean variable L5. If L5 is true, the holding aircraft are separated by a distance greater than M resulting in the rejection of the aircraft as a viable conflict. If L5 is false, the aircraft are passed on to the next filter as a potential conflict.

If one of the aircraft were not in hold, the function calculates the numerator formulas (i.e. $g1$, $g2$, $h1$, etc.). It first checks for the object aircraft being in hold and if true uses the $h3$ and $h4$ as the numerators in the $U1$ and $U2$. If the object is not in hold, it uses the $g1$ and $g2$ numerators. The numerators are explained in detail in the case descriptions in Section 3.1.1.2, but the values being calculated here are L1 and L3 which use both $U1$ and $U2$ distances (refer to the Table 3.1.1-1 above). Also if L1 is true (which means the subject segment does not cross the object line) and L3 is false (which means the end cusps of the subject aircraft is beyond the M distance from the object segment), there would be no potential for conflict and L5 would be true.

A similar check is evaluated for the subject aircraft where it is checked if in hold. If true, the $g3$ and $g4$ numerators are calculated and used for L2 and L4. If the subject aircraft is not in hold, the numerators $h1$ and $h2$ are calculated. Just like the previous for the object in hold, the subject in hold will determine if the object segment crosses the subject line. Also the object end cusp's distance from the subject segment is determined. Referring to the Table 3.1.1-1, if the L5 is true, the L2 must be true meaning the object aircraft does not cross the subject line, and the L4 is false meaning the object cusps are less than a distance M from the subject line.

For example, take a pair of aircraft that are both not in hold, so all the numerators are calculated. First, L1 and L3 are calculated (using the $g1$ and $g2$ numerators) that check if the subject segment does not cross the object line and the cusps of the subject aircraft are greater than the M distance to the object segment. Therefore, if the lines do not cross and they are greater than M, the aircraft are considered not potentially in conflict making the L5 variable true. Next, the similar variables are calculated for the object aircraft against the subject using the L2 and L4 Boolean variables. For this check (using the $h1$ and $h2$ numerators), the object aircraft checks for a non-crossing of the subject line and if the object segment cusps are greater than M distance to the subject segment.

If both conditions are true, a potential conflict cannot exist, so L5 variable is evaluated true. The routine ends by checking the L5 variable; if true, the segments are rejected as a potential conflict.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.1.1-1	For holding pattern ³ , aircraft are assumed to remain at a given point (<i>not a circular path, Equation 3.1.1-20</i>).	Currently, URET D1.1 does not model aircraft holds.	Important
R 3.1.1-2	For the holding aircraft, the perpendicular line drawn to holding point from opposite line is approximated by the perpendicular line from the opposite point to the intersection of the perpendicular drawn from the holding point to the opposite line (refer to Cases 2 & 3, i.e. G ₃ and G ₄ , Equation 3.1.1-14).	Reasonable, since two checks are always performed. In this case, the other check from the opposite line to holding point will determine the parallel distance to the strip M and this distance will determine the perpendicular distance to strip M.	Minor
R 3.1.1-3	Minimum input length used to determine if aircraft in holding pattern (found in code, i.e. <code>cfp_inp.min_seg_length</code>)	Reasonable, assuming the value is relatively small.	Minor
R 3.1.1-4	Misleading comments and documentation description of Case 2 and 3 perpendicular distance. The numerators: Z3, Z4, H3, and H4 are not equivalent to Case 1 perpendicular, but defined as the adjacent side of the right triangle (i.e. A1 to P to B1).	Need more descriptive comments and documentation for use of the adjacent side distance.	Minor

³ Aircraft flying holding patterns are not currently modeled by URET D1.1. However, in APD the code exists and is being assessed for sake of completeness.

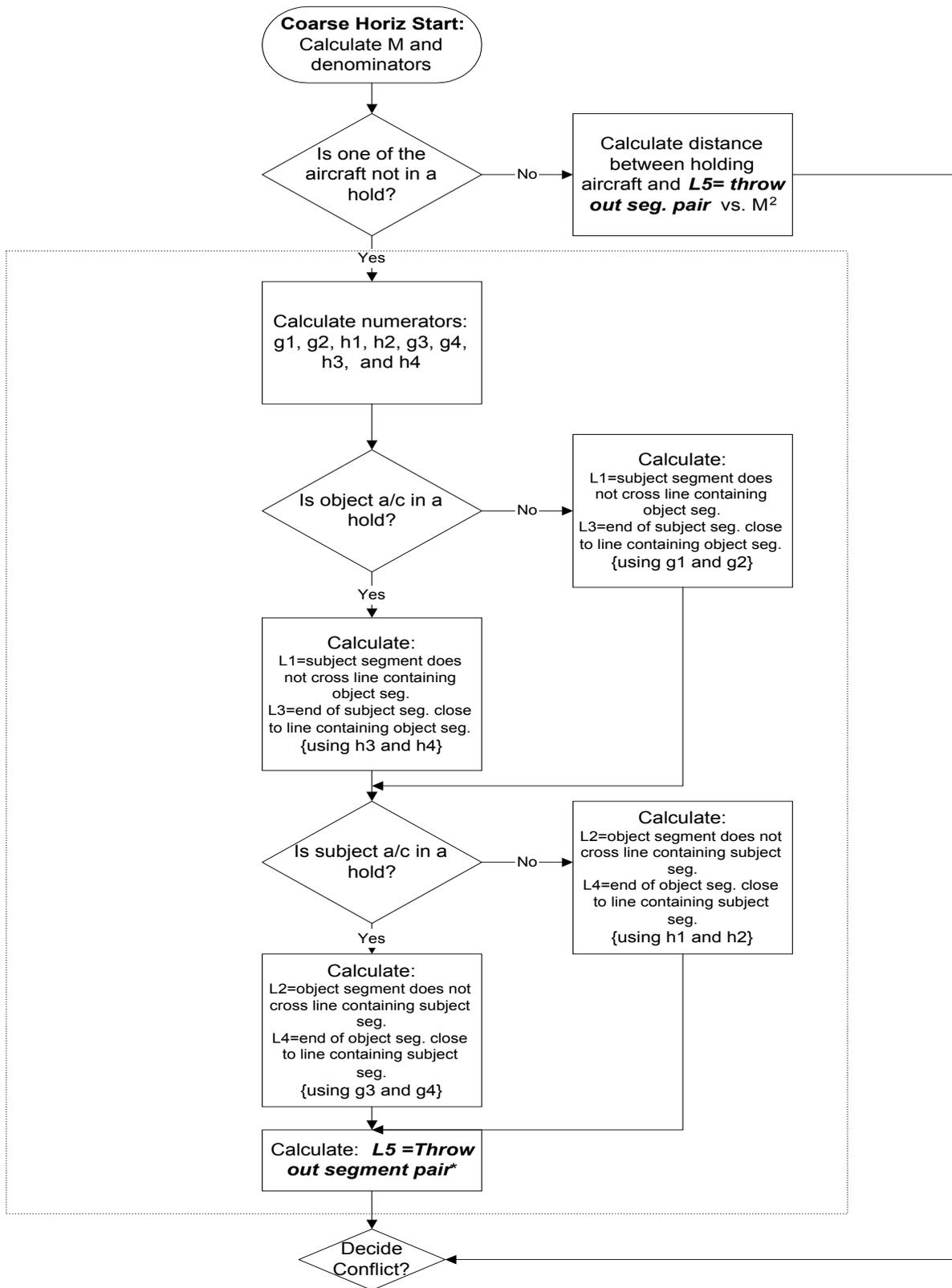
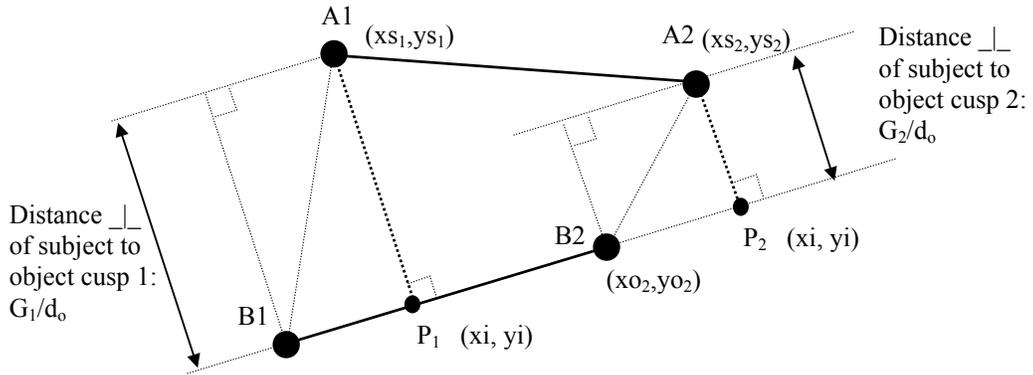


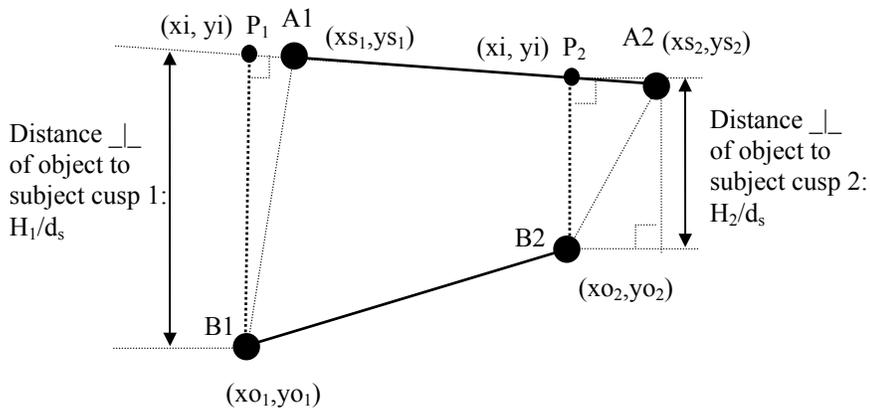
Figure 3.1.1-4: Coarse Horizontal Function Overall Logic

3.1.1.4 Additional Diagrams For Reference:

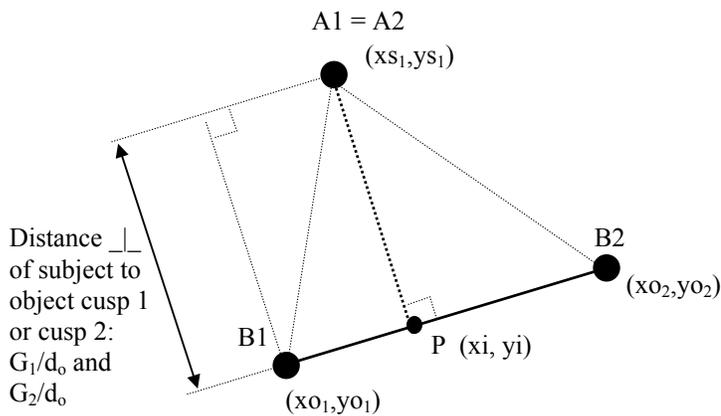
For Case1: Subject to Object Aircraft:



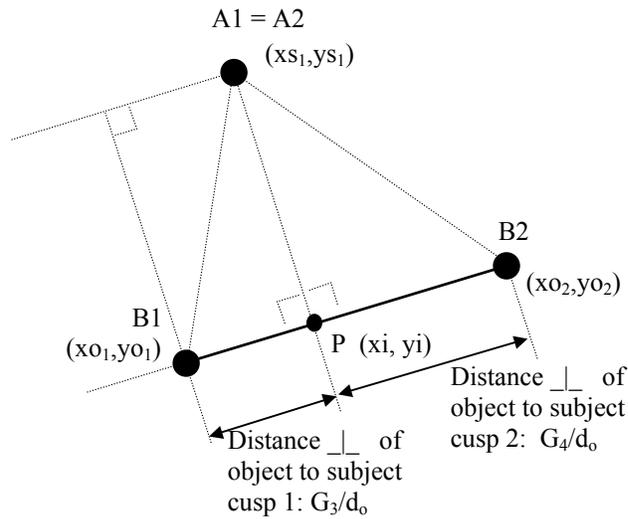
For Case1: Object to Subject Aircraft:



For Case 2: Subject to Object with Subject in hold:



For Case 2: Object to Subject with Subject in hold:



3.1.2 Function: CFP_FINE (C)

Using relative geometry, this function tests if a line segment and an octagon boundary intersect. If the conflict is present, it also computes the start and end of the intersection.

3.1.2.1 Description:

The vertices of the octagon boundary are calculated in a subfunction called CFP_OCTAGON. The octagon boundary is the finest relative boundary mesh and is formed by the two aircraft's rectangular conformance boundaries plus the separation distance. The Fine Filter calls GM_CONVEX to determine if the object aircraft's relative position is inside or outside the octagon. Depending on the specific case of which relative position is inside or outside the octagon, the GM_INSEC function is called to determine if and where the relative position vector intersects the octagon. The floating point computations relating the intersection points and relative positions of the aircraft may cause inaccurate results, so these are thoroughly checked for potential problems and corrected within the algorithm. Therefore, CFP_FINE filter acts as a manager of the algorithm, while other lower level functions actually calculate the intersections (i.e. GM_CONVEX, GM_INSEC, etc.).

Table of Variable Definitions

Function Variable	Description	Math Symbol
in_p	flag for p being inside (1) or outside (0) the octagon	<i>in_p</i>
in_q	flag for q being inside (1) or outside (0) the octagon	<i>in_q</i>
num_insecs	number of intersections	<i>num_insecs</i>
xi[5], yi[5]	x and y array coordinates for the intersection points	<i>xi, yi</i>
ratio1[5], ratio2	ratios of distance location on the intersection segment, returned by GM_INSEC function	<i>ratio1, ratio2</i>
max_ratio	maximum ratio generated from GM_INSEC	<i>max_ratio</i>
tp, tq	start and end times of adjusted vectors	<i>tp, tq</i>
ti[5]	array of intersection times	<i>ti</i>
poscd	return code for calls to CFP_POSIT	<i>poscd</i>
index	index for largest ratio1 in the array list	<i>index</i>
ver[1][8]	input matrix which contains vertices of the octagon boundary, where ver[0][1..8] = x coordinates and the ver[1][1..8] = y coordinates	<i>ver[1][8]</i>
cfpint	input pointer to internal data structure of the line segments of the object and subject aircraft	<i>cfpint</i>

3.1.2.2 Mathematics:

Since this function is essentially a low level manager of the calls to the subfunctions which determine the actual conflict segment, a flowchart is presented under this section and the mathematical descriptions are left to the subfunction assessments.

Figure 3.1.2-2 presents the initial part of the algorithm, including the steps taken if the relative velocity vector is smaller than epsilon. The process continues to the next three figures. Figure 3.1.2-3 presents the steps and calls for the case where the relative velocity is non-zero and the p position vector is inside the octagon while the q is not (refer to the following diagrams in Figure 3.1.2-1 for illustration of examples of the potential cases of the p and q). The Figure 3.1.2-4 presents the steps for a similar case as Figure 3.1.2-3, but for this case the q position vector is inside the octagon while p is not. The Figure 3.1.2-5 presents the case where both p and q vectors are on or outside the octagon. For both Figure 3.1.2-3 and Figure 3.1.2-4, there must be either 1 or 2 intersections found. There are four potential sub-cases to Figure 3.1.2-5: either 0, 2, 3, or 4 intersections were found.

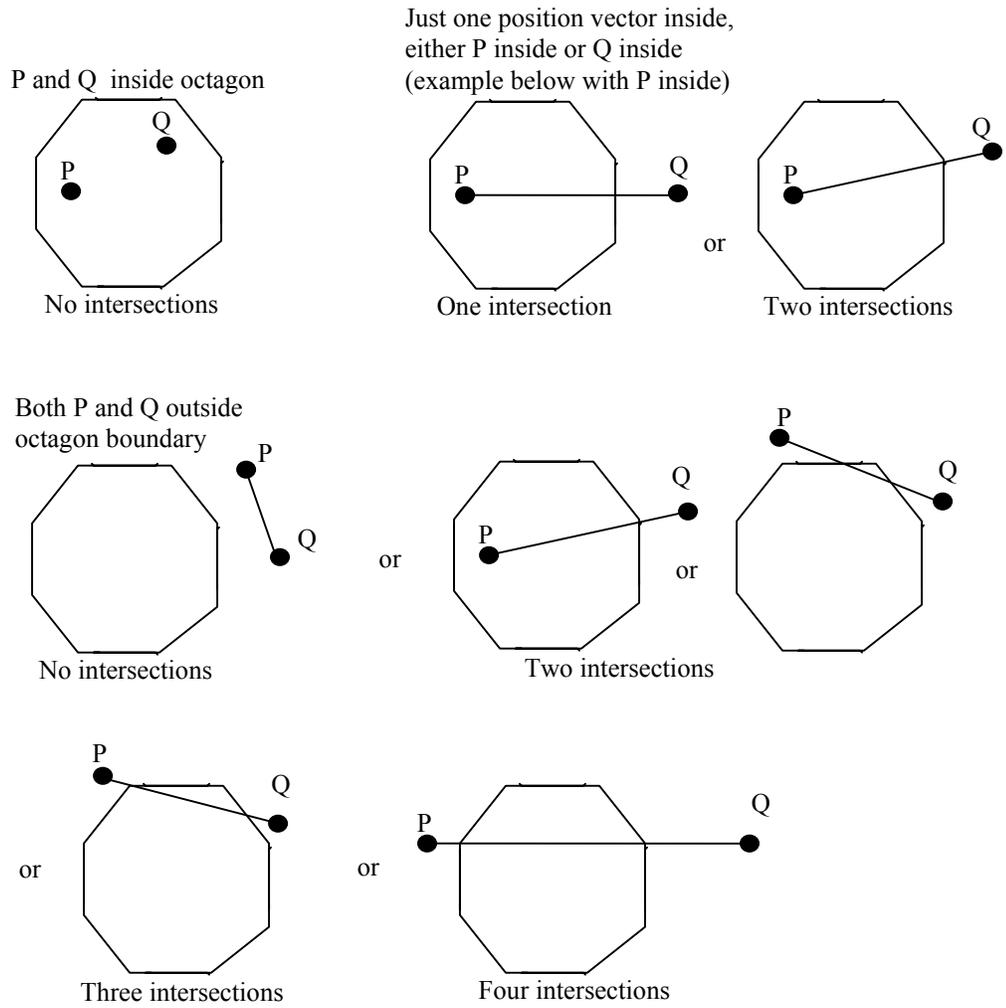


Figure 3.1.2-1: Diagram of examples of each case where relative velocity is non-zero

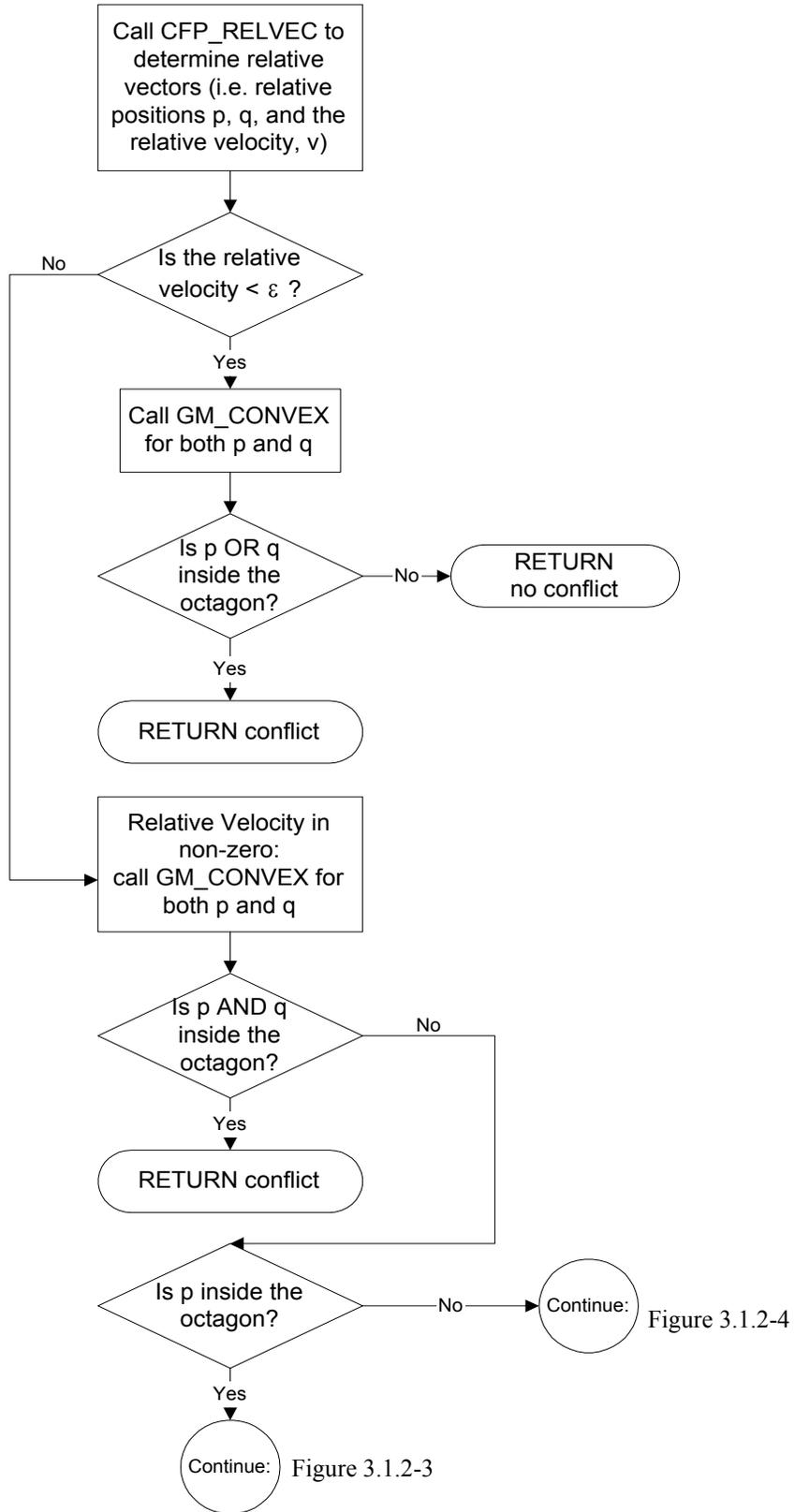


Figure 3.1.2-2: Initial Steps in CFP_FINE

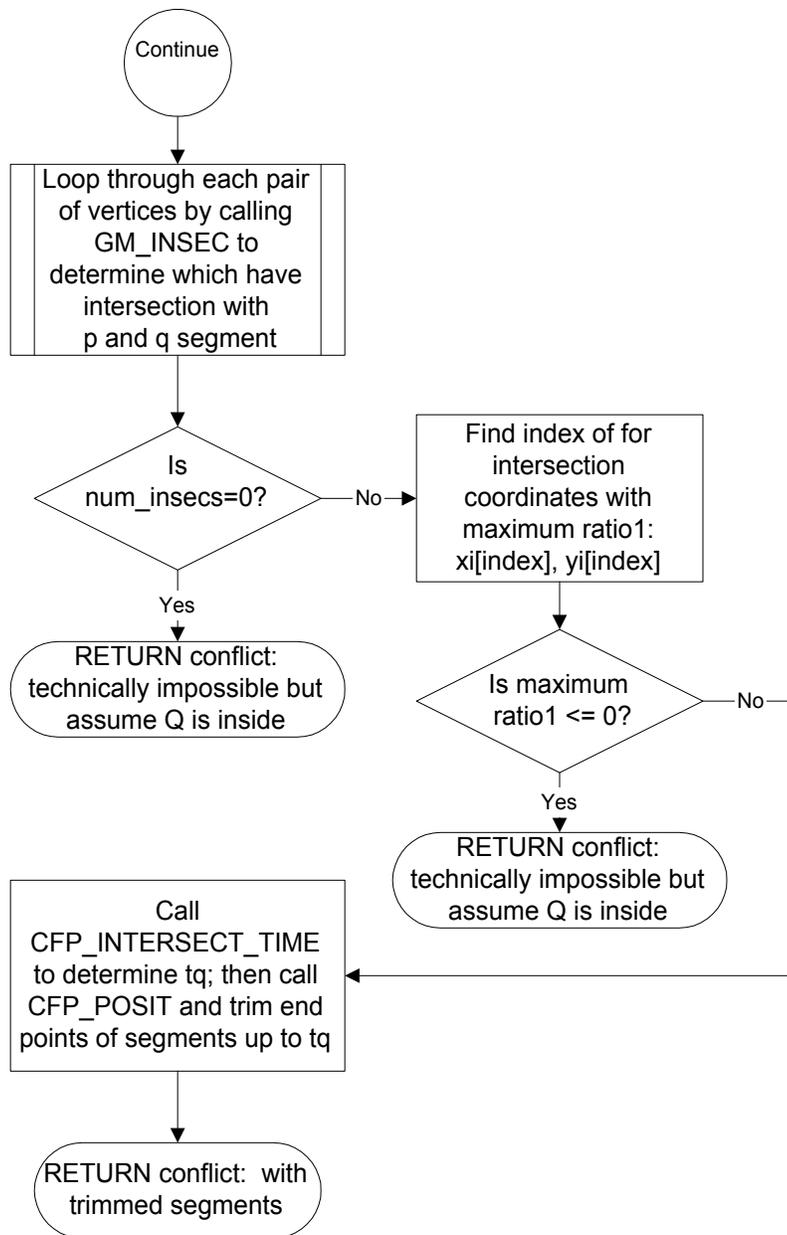


Figure 3.1.2-3: Flowchart where P is inside octagon only (continued from Figure 3.1.2-2)

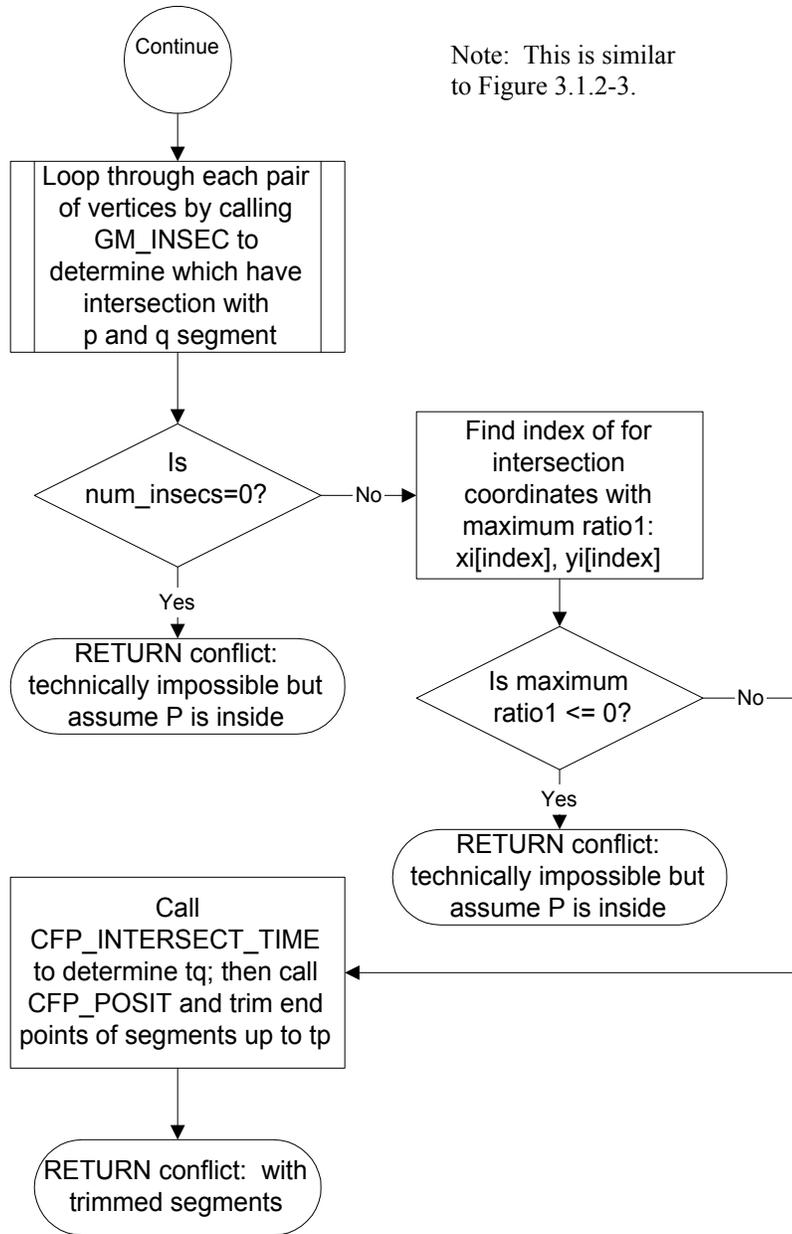


Figure 3.1.2-4: Flowchart where Q is inside octagon only (continued from Figure 3.1.2-2)

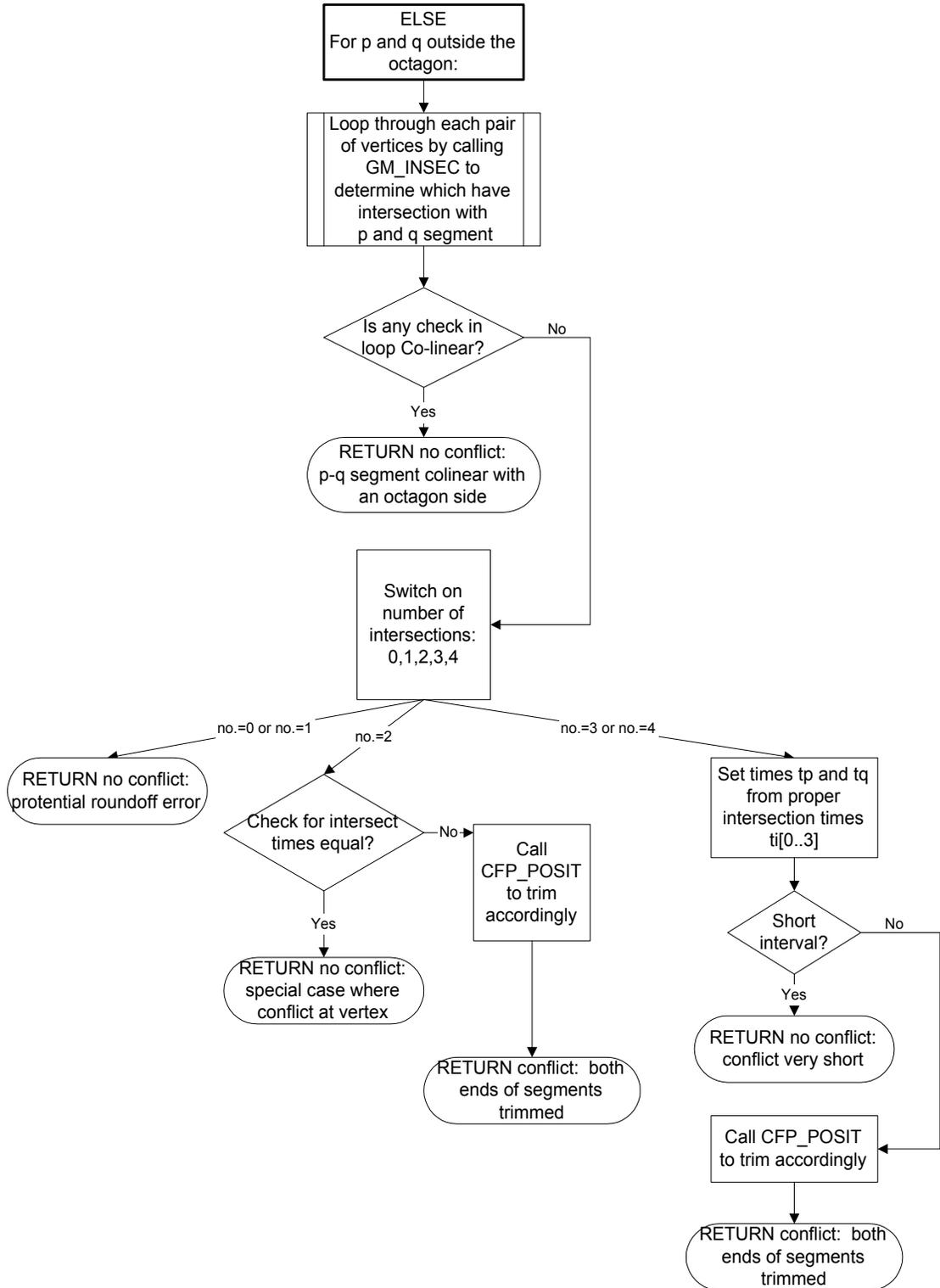


Figure 3.1.2-5: Flowchart where both P and Q are outside octagon (continued from Figure 3.1.2-2)

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.1.2-1	The relative velocity vector is determined by using the magnitude from the average ground speed over the interval and the direction from the bearing interval. This is compared to a small epsilon value to determine when to assume a check of the relative position alone is sufficient.	If the acceleration is constant over the segments and the epsilon value is small, the approximation is very reasonable. This function determines if either p or q vector are inside the octagon, which helps reduce the chance for a missed alert from round off error.	Important
R 3.1.2-2	The case where the number of intersections equals zero or the maximum ratio is ≤ 0 is checked when one of the position vectors is inside the octagon. If they do occur, a floating point error must have caused this, since they are technically impossible.	The checks are in place and the proper flags are also present. Though how often, if ever, does this occur?	Minor
R 3.1.2-3	The case where the P and Q vectors are outside the octagon and the GM_INSEC finds one intersection is an error which is protected against by an error trap. The assumption is it is caused by a round off error and no conflict result is returned.	On the surface the error seems impossible, but due to floating point round off error this may happen. The assumption is reasonable since both position vectors were already determined to not be collinear and outside the octagon. One intersection could only mean the position vector skims the surface of the octagon, but not at a vertex or it would have been two intersections. To touch the surface somewhere else, could only mean a floating point error and this is what the function checks for.	Minor

3.1.3 Function: CFP_INTERSECT_TIME (C)

This function calculates the time, given the coordinates (x and y) of a specific point, by linear interpolation between the end points of a segment.

3.1.3.1 Description:

Given the x, y, and t of the end points of a trajectory segment, the function will linear interpolate to find the time for a given x and y location along the trajectory segment. The function has simple checks to ensure the point is within the segment. If it is not within the segment, the function will use the corresponding end point time.

Table of Variable Definitions

Function Variable	Description	Math Symbol
x1, y1	x, y coordinates of first end point	x_1, y_1
t1	time of first end point	t_1
x2, y2	x, y coordinates of second end point	x_2, y_2
t2	time of second end point	t_2
xi, yi	x, y coordinates of intersection point	x_i, y_i
i	if segment is a point, which time should be used: i=1 so use first point's time; t=2 so use second end point's time	I
temp1	absolute value change in the x dimension for the segment	$temp1$
temp2	absolute value change in the y dimension for the segment	$temp2$
ratio	ratio of the difference between end point dimension and the intersection dimension by the corresponding temp1 or temp2	$ratio$

3.1.3.2 Mathematics:

The function interpolates to find a specific intersection time for a given point on a segment. The function starts by calculating the two segment difference variables.

$$temp1 = |x_1 - x_2| \quad \text{Equation 3.1.3-1}$$

$$temp2 = |y_1 - y_2| \quad \text{Equation 3.1.3-2}$$

The next statement checks for both these differences being effectively zero (as small as an epsilon value). If they are, a time is returned for one of the end points (the choice for this end point is provided as an input). However, if the both the difference equations are not effectively zero, the dimension used for the interpolation has the larger difference. Therefore, the next formula calculated is for the ratio:

$$\frac{(x_i - x_1)}{(x_2 - x_1)} \text{ or } \frac{(y_i - y_1)}{(y_2 - y_1)} \quad \text{Equation 3.1.3-3}$$

A final check is made to ensure that the ratio is within the interval (0, 1). Finally, the calculation for the time is based on the chosen ratio.

$$t = t_1 + ratio * (t_2 - t_1) \quad \text{Equation 3.1.3-4}$$

The equation above is simply the linear interpolation for the time at the given xi, yi coordinates.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.1.3-1	Equation 3.1.3-4 is a linear approximation which assumes no acceleration is present, although the real kinematics of an aircraft may have acceleration.	For the input to this function, the assumption for the relative velocity of the aircraft is that the aircraft do not have acceleration during the segment, since the relative velocity uses the average ground speed for the entire segment. A more accurate approach is to use a similar function as CFP_POSIT which uses a quadratic function to calculate the time.	Minor

3.1.4 Function: CFP_MIDDLE_HORIZ (C)

This function performs the horizontal middle filter, which uses circular conformance bounds.

3.1.4.1 Description:

The algorithm assumes circular conformance bounds around the subject and object aircraft with the radii specified from the standard horizontal conformance by taking the larger of longitudinal or lateral conformance distance. Using relative vectors, the algorithm subtracts the subject aircraft's position and velocity vectors from the object's position and velocity vectors. The circular boundary is drawn around the subject aircraft, consisting of both aircraft's conformance radii and the separation distance. The aircraft are considered not to be in a conflict if the relative position vector falls outside the subject centered circular conformance region. If it does fall inside the region, the aircraft pair flight segments are trimmed only to the inside of the circular boundary and passed to the next middle filter.

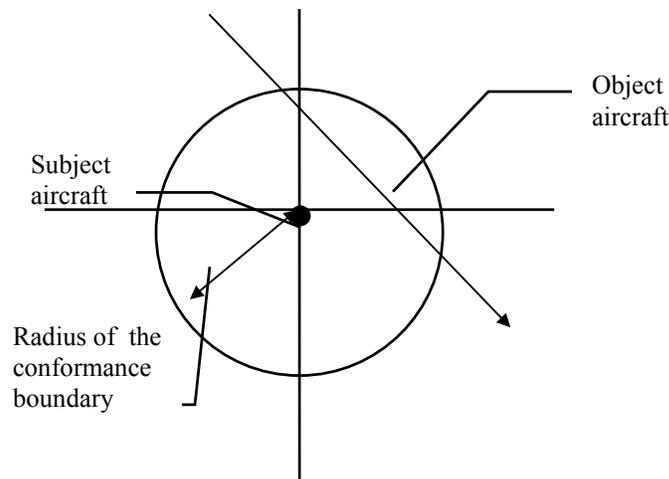


Figure 3.1.4-1: Relative Geometry with Circular Conformance Boundary

Table of Variable Definitions

Function Variable	Description	Math Symbol
xd, yd	x, y coordinates of object minus subject aircraft	x_d, y_d
tempc, temps	cosine and sine temporary variables	t_c, t_s
vdotv, vdotp, pdotp	vector dot products	$(V \bullet V), (V \bullet P), (P \bullet P)$
m	square of separation radius = (object's conformance radius + subject's conformance radius + separation distance) ²	m
qdiscr	one quarter of the discriminant	$\det^{1/4}$
beta	angle used to compute relative velocity; $\beta = \theta_o - \theta_s$ = object heading angle - subject heading angle	β
t1, t2, t3, t4	time interval endpoints	t_1, t_2, t_3, t_4
dum	dummy variable used in max3 and min3	dum
ths, tho	heading angle for subject aircraft and object aircraft, respectively	θ_s, θ_o
p[0], p[1]	relative position vector P for x-axis [0] and y-axis [1]	P_0, P_1
q[0], q[1]	relative position vector Q for x-axis [0] and y-axis [1]	Q_0, Q_1
vel[0], vel[1]	relative velocity vector for x-axis and y-axis	V_0, V_1

3.1.4.2 Mathematics:

Relative geometry calculation:

The first step is to trim the endpoints in the x and y coordinates, so the time intervals are equivalent.

Position Vectors:

The next step is to calculate the relative geometry vectors. The relative vector is calculated for the start and end of the line segments, respectively the P and Q vectors. The P and Q vectors are rotated to align the x-axis in the subject aircraft direction of travel. Therefore, the geometry vectors are calculated as follows:

$$t_c = \cos(\theta_s) \quad \text{Equation 3.1.4-1}$$

$$t_s = \sin(\theta_s) \quad \text{Equation 3.1.4-2}$$

$$x_d = (\text{x coordinate of aircraft b}) - (\text{x coordinate of aircraft a}) \quad \text{Equation 3.1.4-3}$$

$$y_d = (\text{y coordinate of aircraft b}) - (\text{y coordinate of aircraft a}) \quad \text{Equation 3.1.4-4}$$

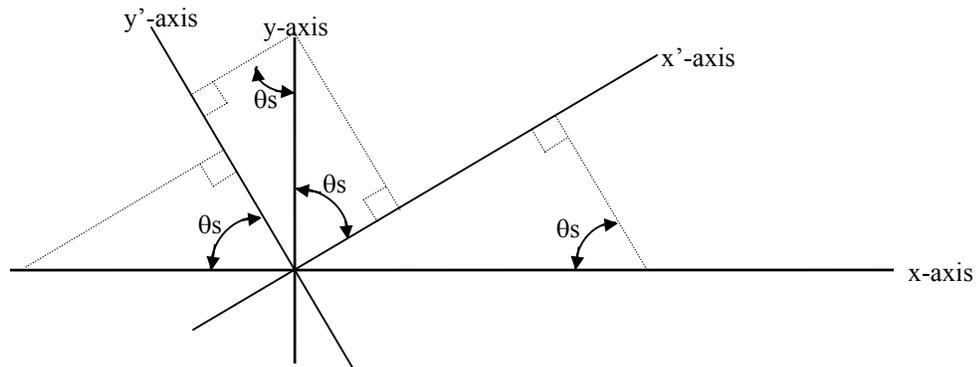


Figure 3.1.4-2: Rotating P vector to axis frame of subject aircraft

Now, rotate position vector P to define the x-axis by the subject aircraft's direction of travel (refer to Figure 3.1.4-2). Project the x and y differences on to the x and y prime axis as illustrated in the Figure 3.1.4-2 above, such that:

$$P_0 = x_d t_s + y_d t_c \quad \text{Equation 3.1.4-5}$$

$$P_1 = y_d t_s - x_d t_c \quad \text{Equation 3.1.4-6}$$

The same steps are performed for the Q vector for the point 2 coordinates, resulting with:

$$Q_0 = x_d t_s + y_d t_c \quad \text{Equation 3.1.4-7}$$

$$Q_1 = y_d t_s - x_d t_c \quad \text{Equation 3.1.4-8}$$

where x_d and y_d are again calculated using the second point on each of the position vectors

Velocity Vector:

The relative velocity vector is determined by projecting the average ground speed on to the relative position vector of the subject by the following formulas (refer to Figure 3.1.4-3):

$$V_0 = [-(\text{ground speed of aircraft a at (point 1 + point 2)}) + (\text{ground speed of aircraft b at (point 1 + point 2)}) * \cos(b)]/2$$

$$V_1 = [-(\text{ground speed of aircraft b at (point 1 + point 2)}) * \sin(b)]/2$$

where $V_a = (\text{ground speed of aircraft a at (point 1 + point 2)})/2$;
 $V_b = (\text{ground speed of aircraft b at (point 1 + point 2)})/2$;
 with a aircraft as subject aircraft and b as object aircraft

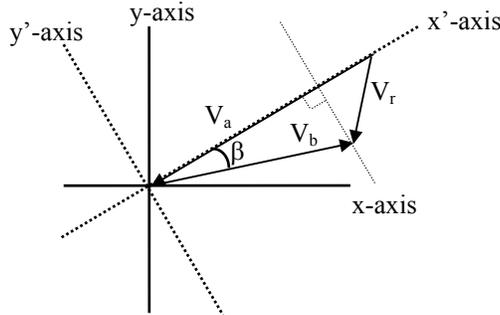


Figure 3.1.4-3: Relative velocity

Therefore, the relative velocity vector is :

$$V_0 = V_b \cos(\beta) - V_a \quad \text{Equation 3.1.4-9}$$

$$V_1 = -V_b \sin(\beta) \quad \text{Equation 3.1.4-10}$$

where the V_0 refers to x' -axis and V_1 refers to y' -axis

Dot products and separation calculated:

The dot product is calculated for the velocity and position vectors. The next check determines if the relative velocity is close to zero. The dot product of V on V is the magnitude of the relative velocity squared. This check squares the speed epsilon value (ϵ) and compares it to the dot product of V on V .

Relative Velocity close to zero:

If the dot product is smaller than the ϵ^2 , the relative velocity is essentially zero, meaning the aircraft are not moving relative to each other. Therefore, it is sufficient to test if the relative position vector is inside the circular conformance bound. The radius of the circular boundary is as follows:

$$m = (\text{radius of conformance of subject aircraft}^a + \text{radius of conformance of object aircraft}^b + \text{separation standard})^2$$

The m is compared against the dot product of the relative position vector of P on P . If the dot product is less than m , then the aircraft pair may be in conflict so are passed on to the next filter. With relative velocity at zero, the P and Q vectors should be equivalent, so only one check is

^a The radius of conformance of the subject aircraft is the hypotenuse of the right triangle formed by half the longitudinal and lateral conformance distances for the segment, $= \sqrt{(\textit{longitude})^2 + (\textit{lateral})^2}$. For example, the radius with longitudinal and lateral conformance bounds of 3 and 5 miles would be equivalent to 2.52 nautical miles from $\sqrt{(1.5nm)^2 + (2.5nm)^2}$.

^b The radius of conformance of the object aircraft is calculated in the same manner as the subject aircraft, using the longitudinal and lateral conformance distances of the object aircraft segment.

necessary if the rounding errors are minimal. *As suggested in the code's comments, the $Q \cdot Q$ should also be checked and if either are less than m , the function should result in a detected conflict. The comments suggest that only one vector check is sufficient, **however if round off problems are present both vectors should be checked against m** . The comments also state that the check is for the "norm" equal to zero, however the $v \cdot v$ is equivalent to the magnitude of the relative velocity squared not the normal vector. If this magnitude is equal to zero, the position vectors P and Q should be equivalent, since there is no relative movement for the time interval of the flight segment.*

Relative velocity greater than zero:

With the relative velocity greater than zero, the aircraft are moving relative to each other. The function needs to determine if the object aircraft is within the \sqrt{m} distance from the subject aircraft within the time interval of the segment. The vector equation for the relative distance is :

$$R(t) = P + tV \quad \text{Equation 3.1.4-11}$$

where t = time variable - time at point 1

The function must solve for the time, t , that the relative distance traveled is equal to the radius \sqrt{m} . Thus by squaring both sides, the function solves :

$$R(t)^2 = m \quad \text{Equation 3.1.4-12}$$

$$\begin{aligned} (P + tV)^2 &= m \\ (tV)^2 + 2tPV + P^2 &= m \\ (V^2)t^2 + (2PV)t + (P^2 - m) &= 0 \end{aligned}$$

$$at^2 + bt + c = 0 \quad \text{Equation 3.1.4-13}$$

where $a = V^2$, $b = 2PV$, $c = P^2 - m$

Therefore, the function must solve the quadratic Equation 3.1.4-13 for the roots (times) where the object aircraft enters the circular boundary. Using the quadratic formula, the quarter discriminant is calculated. If the quarter discriminant is negative, there are no real roots, which means no time, t , that equates the relative distance on the boundary. In other words, there would be no conflict for this case. From the quadratic equation, the expression for the quarter discriminant is derived as follows:

$$\text{Quadratic Equation: } \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

where discriminant = $b^2 - 4ac$,

$$\text{so quarter discriminant} = b^2/4 - ac \quad \text{Equation 3.1.4-14}$$

If the quarter discriminant equals zero, then there are equal roots^c to the quadratic equation and the conflict just touches the circular bound once. Since the aircraft are separated by the conformance boundary, assume no conflict. From Equation 3.1.4-13 and Equation 3.1.4-14, the quarter discriminant is equivalent to:

^c For a circular boundary with equal roots, the relative position vector touches the boundary at only one point.

$$\det_{1/4} = (PV)^2 - V^2(P^2 - m) \quad \text{Equation 3.1.4-15}$$

or by using dot products:

$$\det_{1/4} = (P \bullet V)(P \bullet V) - (V \bullet V)[(P \bullet P) - m]$$

If Equation 3.1.4-15 is greater than zero, roots do exist and are determined by the following two equations:

$$\text{Time 1} = t1 = t1_a + \left(\frac{-b - \sqrt{b^2 - 4ac}}{2a} \right) = t1_a + \left(\frac{-b/2 - \sqrt{b^2/4 - ac}}{a} \right)$$

$$\text{Time 2} = t2 = t2_a + \left(\frac{-b + \sqrt{b^2 - 4ac}}{2a} \right) = t2_a + \left(\frac{-b/2 + \sqrt{b^2/4 - ac}}{a} \right)$$

Times 1 and 2 defined in terms from the function:

$$t1 = t1_a + \left(\frac{-(P \bullet V) - \sqrt{\det_{1/4}}}{(V \bullet V)} \right) \quad \text{Equation 3.1.4-16}$$

$$t2 = t2_a + \left(\frac{-(P \bullet V) + \sqrt{\det_{1/4}}}{(V \bullet V)} \right) \quad \text{Equation 3.1.4-17}$$

If the times, t1 and t2, fall between the (t1_a, t2_a) and (t1_b, t2_b) time intervals, there exists a time interval where the object aircraft is inside the circular boundary and thus a conflict exists. The function trims the start and end points of the data set based on the intersection of the times.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.1.4-1	If the relative velocity magnitude squared (\bar{V}^2) is less than or equal to the speed epsilon squared, it sufficient to check the distance P from the subject aircraft.	If the acceleration is minor over the segment since the relative velocity is averaged over the segment, it seems reasonable. The P position vector will be sufficient to check against the circular bound radius, however a more conservative approach is to check both Q and P, returning a conflict if either is less than conformance radius.	Critical
R 3.1.4-2	Assumption is made in using the average ground speed over the length of the segment in Equation 3.1.4-9 and Equation 3.1.4-10.	Assumes linear acceleration of aircraft during the segment length. This could cause inaccuracy of the position/time estimates.	Important
R 3.1.4-3	For the trimming of the time intervals (at the end of the routine) if a conflict is found, error coded may be returned by CFP_POSIT, as stated in the comments.	Potentially rare, but rounding error due to single precision calculations may cause these error code returns.	Important
R 3.1.4-4	When the function checks for the relative velocity less than epsilon, the function calculates the dot product of V. This is referred to as the “norm” of the velocity in the comments and documentation.	The dot product of V (relative velocity) is not the normal (“norm”) of the velocity vector. It is the squared magnitude of the relative velocity vector. It is true the aircraft would be trailing or parallel with the dot product zero. However, the normal could be zero and the dot product may not be.	Minor (comment)

3.1.5 Function: CFP_MIDDLE_VERT (C)

This function performs the vertical middle filter, which trims the conflict region passed by the horizontal middle filter. The conflict region is trimmed in the vertical versus time plane. If no vertical conflict region is present, the aircraft are not in conflict and finish the detection process.

3.1.5.1 Description:

The algorithm determines the vertical region of overlap for the previously trimmed flight segments from the middle horizontal filter. The algorithm constructs an altitude (z dimension) versus time plot, determining the intersection region if present. In the z-t plane, there exist altitude conformance regions where both subject and object are present during the segment. These regions are bounded on all sides by either straight lines or continuous curves. The conflict time interval passed from the middle horizontal filter may or may not overlap in the z dimension for the entire interval. The function determines the bounds of the z dimension overlap (if at all) and trims the time interval accordingly. If there is not an overlapping region, the function returns a no conflict result. If a conflict is detected, the function trims the conflict for the z-t overlap region and passes this to the next filter.

Table of Variable Definitions

Function Variable	Description	Math Symbol
kappa	lower bound of subject aircraft interval	κ
lambda	upper bound of subject aircraft interval	λ
mu	lower bound of object aircraft interval	μ
nu	upper bound of object aircraft interval	ν
k	number of intersections found	k
roots[2]	a vector of the times of the intersections found	$roots_1$
roots[2]	a vector of the times of the intersections found	$roots_2$
cfpint->a1.z and a2.z	subject aircraft cusp 1 and 2 altitudes (z dimension, feet)	$a1z, a2z$
cfpint->b1.z and b2.z	object aircraft cusp 1 and 2 altitudes (z dimension, in feet)	$b1z, b2z$
cfpint->a1.t and a2.t	subject aircraft cusp 1 and 2 times (t dimension, in seconds)	$a1t, a2t$
cfpint->b1.t and b2.t	object aircraft cusp 1 and 2 times (t dimension, in seconds)	$b1t, b2t$
cfpint->zps	subject aircraft vertical conformance distance above trajectory altitude (in feet from trajectory altitude)	zps
cfpint->zms	subject aircraft vertical conformance distance below trajectory altitude (in feet from trajectory altitude)	zms
cfpint->zpls	subject aircraft vertical conformance bound limit above trajectory altitude (in feet from sea level)	$zpls$
cfpint->zmls	subject aircraft vertical conformance bound limit below trajectory altitude (in feet from sea level)	$zmls$
cfpint->zpo	object aircraft vertical conformance distance above trajectory altitude (in feet from trajectory altitude)	zpo
cfpint->zmo	object aircraft vertical conformance distance below trajectory altitude (in feet from trajectory altitude)	zmo
cfpint->zplo	object aircraft vertical conformance bound limit above trajectory altitude (in feet from sea level)	$zplo$
cfpint->zmlo	object aircraft vertical conformance bound limit below trajectory altitude (in feet from sea level)	$zmlo$
cfpint->zsep	aircraft vertical half separation distance (in feet)	$zsep$

3.1.5.2 Mathematics:

The code begins with error checks for synchronized segment endpoint times and positive ground speeds, which should have been completed in the Horizontal Middle Filter. Next, the code determines where in the intervals is the vertical conflict taking place, providing the following cases:

1. Intersection for the entire interval
2. Intersection takes place at the beginning of the interval
3. Intersection takes place at the end of the interval
4. No intersection takes place
5. Intersection of conformance bounds without crossing of the trajectories
6. Intersection inside the intervals with crossing of the trajectories

Table 3.1.5-1: Logical Vertical Intersection Cases

To determine each the case above, four variables are defined first for the beginning points and then for the end points.

κ	lower bound of subject aircraft interval
λ	upper bound of subject aircraft interval
μ	lower bound of object aircraft interval
ν	upper bound of object aircraft interval

Table 3.1.5-2: Beginning and End Conflict Variables

If the upper bound of the subject aircraft, λ , is above the lower bound of the object aircraft, μ , and the same for ν and κ , the location of the conflict is determined. Specifically, if the following Equation 3.1.5-1 for the given end point is true, an intersection of both aircraft's altitudes will take place at that endpoint.

$$(\lambda > \mu) \text{ and } (\nu > \kappa) \quad \text{Equation 3.1.5-1}$$

This condition statement evaluates between the following seven cases:

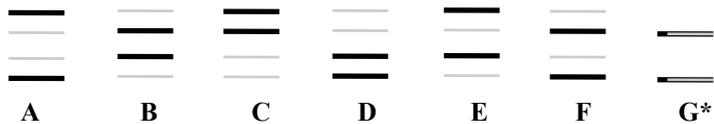


Figure 3.1.5-1: Cases of End Point Intersections

where the dark line = subject altitude
and the light line = object altitude
(note: *overlapping altitudes)

Only for cases C and D is an intersection not evaluated at the particular endpoint (i.e. cusp 1 or 2). For the remaining cases (A,B,E,F,G), the Equation 3.1.5-1 is evaluated true and does intersect in the vertical dimension.

Once the test for the beginning and ending conflicts is complete, the intersections are evaluated. As listed in Table 3.1.5-1 for *Case 1*, if the begin and end conflicts are both evaluated true, the Middle Vertical Filter ends with no trimming to the conflicts, since the trajectory segments are in vertical intersection for the entire time interval.

For *Case 2*, the beginning endpoints are in an intersection but not at the end of the intervals. There are two possible situations under this case: either the lower bound of the subject intersects the upper bound of the object, or the upper bound of the subject intersects the lower bound of the object.

The function uses the fact that the subject's lower bound intersects the object's upper bound only if the subject's cusp 2 altitude is greater than object's cusp 2. This is illustrated in Figure 3.1.5-2.

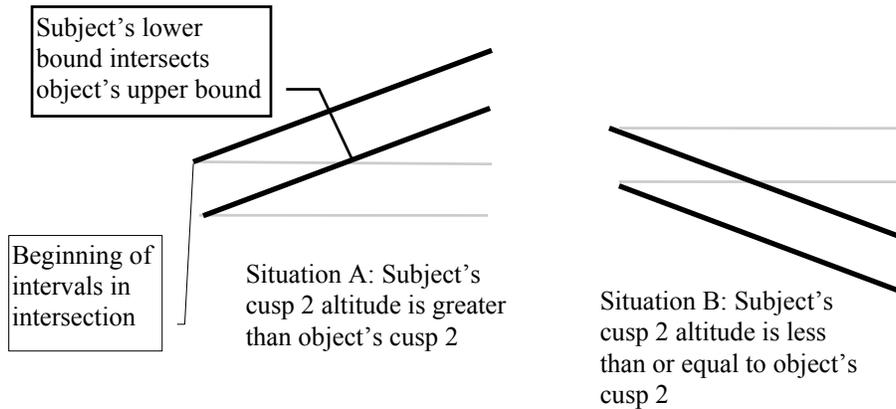


Figure 3.1.5-2: Situations for calling CFP_V_INT

Now, the actual calculation of the vertical intersection point is determined by the function CFP_V_INT. The function returns the intersection root times and the number determined between a pair of segments upper and lower bounds. There could possibly be 0, 1, or 2 intersection times for all the cases, however for *Case 2* only 1 root can be returned by CFP_V_INT. For this case, a zero root is determined only if the difference between the beginning time and the root time is less than an epsilon value. This is an approximation for a vertical conflict of a very small duration. Otherwise, the intervals are truncated from the beginning time to the new end time (i.e. root time) determined by CFP_V_INT.

For *Case 3*, the end points are where the intersection takes place and the beginning points are not. This is symmetrical to the *Case 2* comparisons. Now with the end point as the intersection side, the cusp 1 altitudes are used to determine the situation for the call to CFP_V_INT in the same manner as the cusp 2 altitudes were used for *Case 2*.

For *Cases 4* and *5*, the beginning and end altitude bounds are not intersecting, but a vertical conflict may or may not exist. These cases are evaluated by an expression that determines whether the trajectory segments cross in the vertical dimension. The expression is as follows:

$$[(a1z - b1z)(a2z - b2z)] > 0 \quad \text{Equation 3.1.5-2}$$

For the Equation 3.1.5-2 to be positive, the segments will not be crossing in the vertical dimension.



Figure 3.1.5-3: Equation 3.1.5-2 > 0 for both situations

Similar to *Cases 2* and *3*, the order of intersection (i.e. lower intersects upper bound) is determined by using the cusp 1. For *Case 4*, CFP_V_INT found zero roots, so no intersection is determined. For *Case 5*, two intersection points are found. These intersection points are within the

conformance bounds without the crossing of the trajectories, and of course without being in conflict at the segment endpoints.

For *Case 6*, the Equation 3.1.5-2 must be less than zero, which represents a pair of crossing trajectory segments. Both cases of intersecting bounds take place simultaneously. The lower bound of the subject aircraft intersects the upper bound of the object aircraft and the upper bound of the subject aircraft intersects the lower bound of the object aircraft, so two calls to the function CFP_V_INT are made. The first call is with the intersecting lower bound as the subject aircraft. The second call is with the intersecting lower bound as the object aircraft. Each call must return one root and both roots are used to trim the segments.

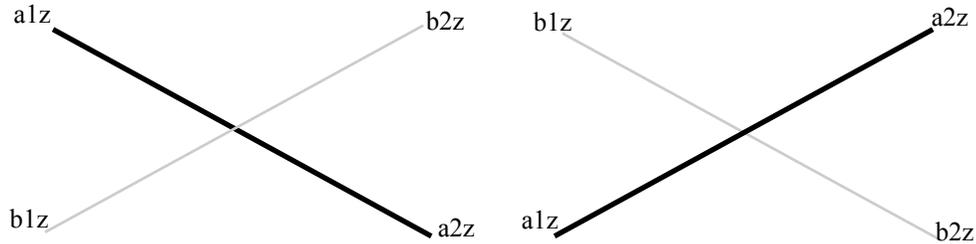


Figure 3.1.5-4: Crossing Trajectories for Case 6

A final check is made to determine if the trimmed interval is less than an epsilon value (currently 1 second). If the interval is less than or equal to the epsilon time, no conflict is returned. If the time is greater than the epsilon time, the conflict does exist and the intervals have been trimmed accordingly.

Function Logic Review:

The following flowchart summarizes the logic of the Middle Vertical Filter. The checks for the small epsilon time are not represented for each case, but assumed present for each call to CFP_V_INT.

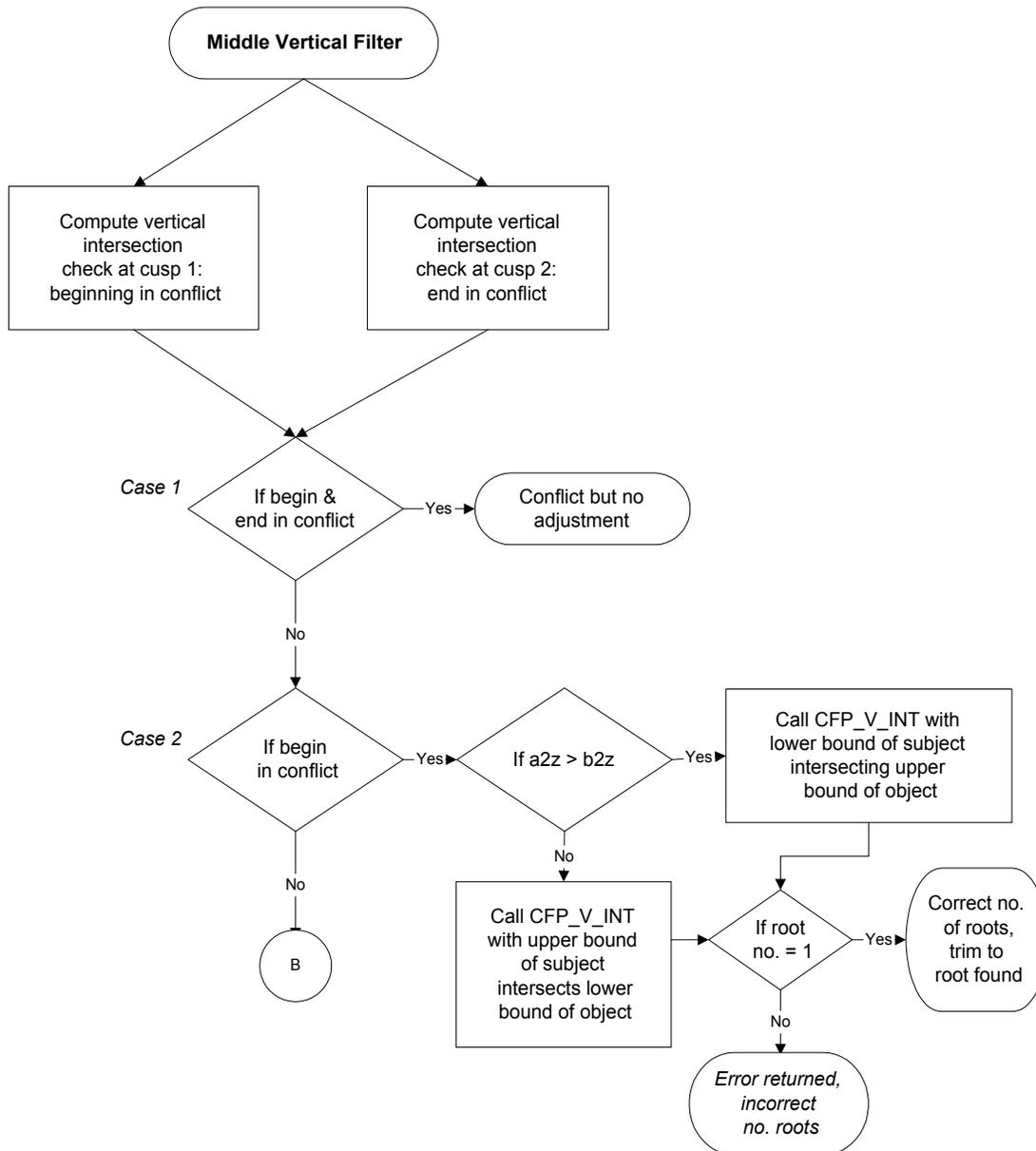
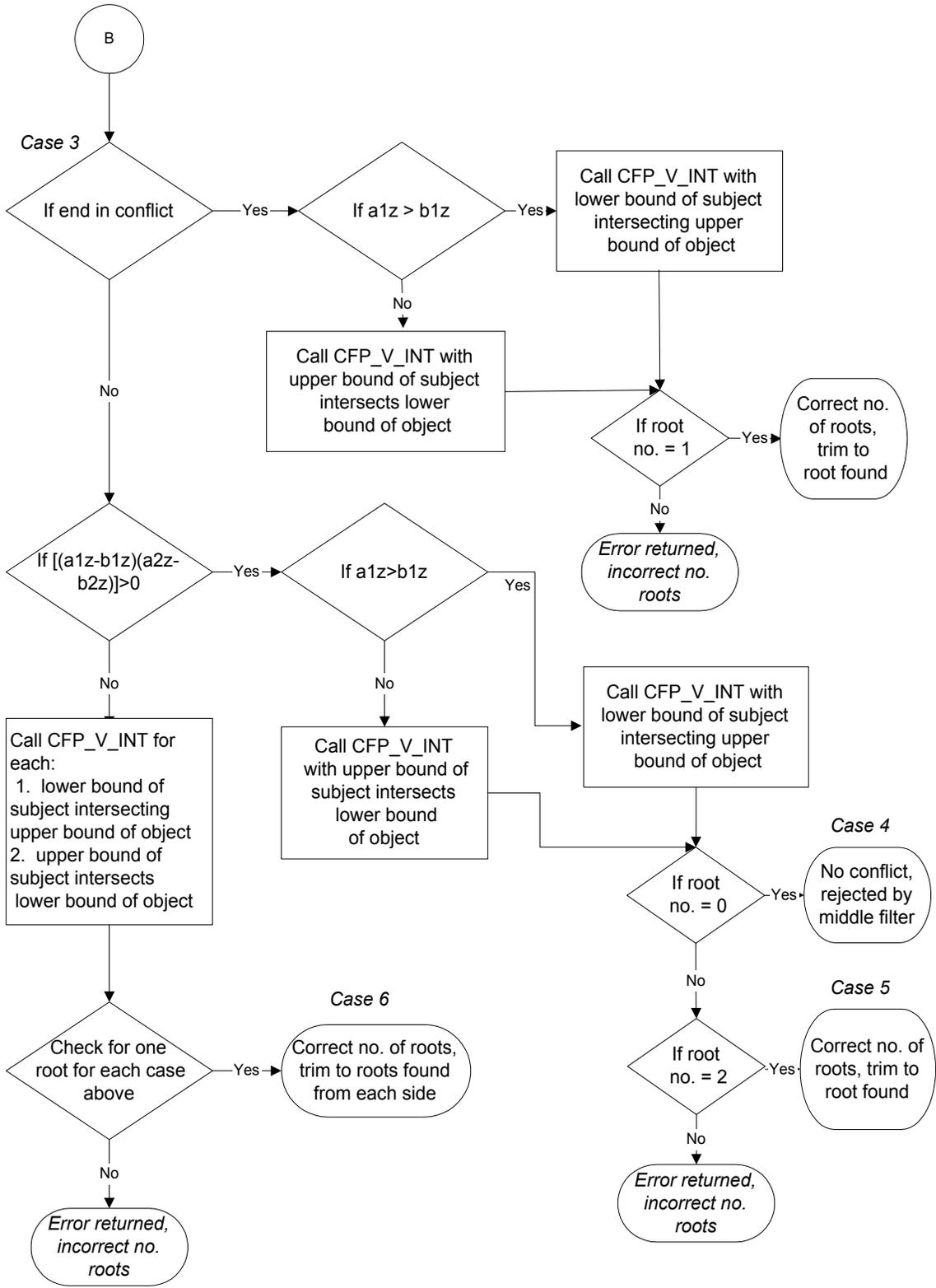


Figure 3.1.5-5: Middle Vertical Filter Logic Flowchart



Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.1.5-1	Round off problems have caused errors due to single precision accuracy as expressed in the comments. In all calls to CFP_V_INT, the middle filter checks for small interval times less than epsilon value.	The occurrence of very small conflicts cause the error checks to assume no conflict (less than acp.time_epsilon). This is a reasonable approximation if the epsilon time is relatively small (currently 1 second).	Important
R 3.1.5-2	Incorrect number of roots for particular geometric situation returned by CFP_V_INT (a sub-function call by the middle filter).	Current method is to check for the appropriate number of roots for each case. If the wrong number is returned, the middle filter will return an error code. More investigation is required to examine when and if this can occur.	Important
R 3.1.5-3	Prior filter check for equal adjusted interval cusp times.	This is checked for equal interval cusp times that must have been completed in a prior filter (middle horiz.).	Important

3.1.6 Function: CFP_RELVEC (C)

This function recalculates the relative geometry vectors, specifically the relative velocity vector and the relative position vectors.

3.1.6.1 Description:

Due to the trimming in the Middle filters the relative geometry should be recalculated before entering the Horizontal Fine Filter. Also, if acceleration is present the recalculation of the relative vectors will reduce round off in the Fine Filter. The function applies the same technique used in the Middle Horizontal filter to generate the relative vectors.

Table of Variable Definitions

Function Variable	Description	Math Symbol
xd, yd	x, y coordinates of object minus subject aircraft	x_d, y_d
tempc, temps	cosine and sine temporary variables	t_c, t_s
beta	angle used to compute relative velocity; $b = q_0 - q_s =$ object heading angle - subject heading angle	β
ths, tho	heading angle for subject aircraft and object aircraft, respectively	q_s, q_o
p[0], p[1]	relative position vector P for x-axis [0] and y-axis [1]	P_0, P_1
q[0], q[1]	relative position vector Q for x-axis [0] and y-axis [1]	Q_0, Q_1
vel[0], vel[1]	relative velocity vector for x-axis and y-axis	V_0, V_1

3.1.6.2 Mathematics:

The function starts by calculating the relative geometry vectors. The relative vector is calculated for the start and end of the line segments, respectively the P and Q vectors. The P and Q vectors are rotated to align the x-axis in the subject aircraft direction of travel. Therefore, the geometry vectors are calculated as follows:

$$t_c = \cos(\theta_s) \quad \text{Equation 3.1.6-1}$$

$$t_s = \sin(\theta_s) \quad \text{Equation 3.1.6-2}$$

$$x_d = x_{aircraft_b} - x_{aircraft_a} \quad \text{Equation 3.1.6-3}$$

$$y_d = y_{aircraft_b} - y_{aircraft_a} \quad \text{Equation 3.1.6-4}$$

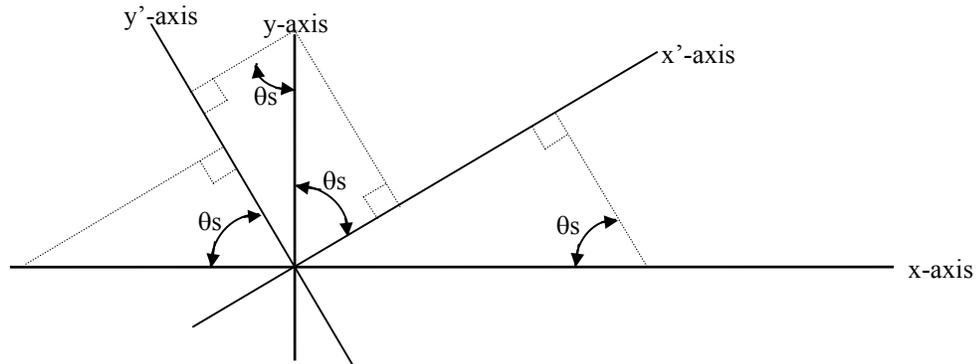


Figure 3.1.6-1: Rotating P vector to axis frame of subject aircraft

Now, rotate position vector P to define the x-axis by the subject aircraft's direction of travel (refer to Figure 3.1.6-1). Project the x and y differences on to the x and y prime axis as illustrated in the Figure 3.1.6-1 above, such that:

$$P_0 = x_d t_s + y_d t_c \quad \text{Equation 3.1.6-5}$$

$$P_1 = y_d t_s - x_d t_c \quad \text{Equation 3.1.6-6}$$

The same steps are performed for the Q vector for the point 2 coordinates, resulting with:

$$Q_0 = x_d t_s + y_d t_c \quad \text{Equation 3.1.6-7}$$

$$Q_1 = y_d t_s - x_d t_c \quad \text{Equation 3.1.6-8}$$

where x_d and y_d are again calculated using the second point on each of the position vectors

Velocity Vector:

The relative velocity vector is determined by projecting the average ground speed on to the relative position vector of the subject by the following formulas (refer to Figure 3.1.6-2):

$$V_0 = [-(\text{ground speed of aircraft a at (point 1 + point 2)}) + (\text{ground speed of aircraft b at (point 1 + point 2)}) * \cos(\beta)]/2$$

$$V_1 = [-(\text{ground speed of aircraft b at (point 1 + point 2)}) * \sin(\beta)]/2$$

where $V_a = (\text{ground speed of aircraft a at (point 1 + point 2)})/2$;
 $V_b = (\text{ground speed of aircraft b at (point 1 + point 2)})/2$;
 with a aircraft as subject aircraft and b as object aircraft

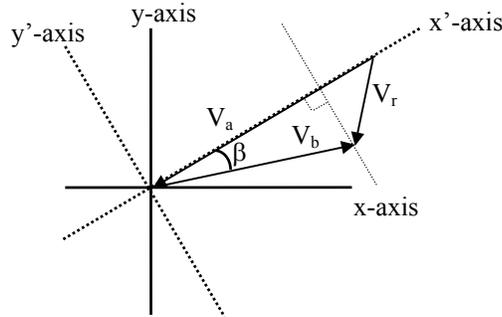


Figure 3.1.6-2: Relative velocity

Therefore, the relative velocity vector is :

$$V_0 = V_b \cos(\beta) - V_a \quad \text{Equation 3.1.6-9}$$

$$V_1 = -V_b \sin(\beta) \quad \text{Equation 3.1.6-10}$$

where the V_0 refers to x' -axis and V_1 refers to y' -axis.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.1.6-1	There is an inherent assumption of constant acceleration if acceleration is present.	The assumption is reasonable if the segments are relatively small.	Important
R 3.1.6-2	There is a check for acceleration prior to calculating the new relative velocity vector. If either aircraft has acceleration, the relative velocity is recalculated. However, if neither do, the calculation is bypassed.	The reason is sound, if no acceleration is present the calculation of relative velocity will not have changed at all.	Minor

3.1.7 Function: CFP_TRIM (C)

This routine trims two segments so they have the same start and end times, making them overlap within the same time interval.

3.1.7.1 Description:

The function is composed of two main conditional loops:

1. The first loop trims the starting points based on which segment's starting time is greater.
2. The second loop trims the ending points based on which segment's ending time is earlier.

The program returns the new set of trimmed flight segments by calling CFP_POSIT, which interpolates to find the x and y coordinates of the each trimmed endpoint.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.1.7-1	All aircraft segments that enter CFP_TRIM have overlapping time intervals.	Requirement satisfied since CFP uses coarse filter to ensure segments overlap in time.	Minor
R 3.1.7-2	All accuracy and calculation specifically carried in CFP_POSIT algorithm.	Approximation of x and y coordinates of trimmed endpoint calculated in CFP_POSIT algorithm.	Minor

3.1.8 Function: CFP_V_INT (C)

This function computes the intersections of two linear segmented curves (referred to as splines). For this application, these splines refer to the boundary lines of aircraft in the vertical dimension. Only the intersections within the interior of the interval are desired and any splines that touch but do not cross are not of interest.

3.1.8.1 Description:

The algorithm intersects the lower bound function ($z_a(t)$) with the upper bound function ($z_b(t)$) and calculates where these functions intersect. To determine these intersections, the function first computes the points of inflection, specified as the spline mesh points, in the altitude versus time dimensions. The mesh points are defined for the lower level spline and denoted $a[i]$ where $i= 1, 2, 3$. For the upper bound the spline mesh points are denoted as $b[i]$ where $i= 1, 2, 3$. The algorithm interpolates to find the same time mesh for each spline. In a sense, the spline mesh points define intervals where the algorithm checks for intersections. The times at the endpoints of the region, (where $z_a(t)$ is below $z_b(t)$ and forms the intersection), are reported back to the vertical middle filter. The algorithm may find 0, 1, or 2 intersections of the two splines.

Table of Variable Definitions

Function Variable	Description	Math Symbol
zl1	z of lower bound trajectory at t_1	z_{l1}
zl2	z of lower bound trajectory at t_2	z_{l2}
zl	lower conformance for lower bound	z_l
zll	lower conformance limit for lower bound	z_{ll}
zu1	z of upper bound trajectory at t_1	z_{u1}
zu2	z of upper bound trajectory at t_2	z_{u2}
zu	upper conformance for lower bound	z_u
zuu	upper conformance limit for lower bound	z_{uu}
zsep	vertical half separation	z_{sep}
tbegin	common start time of truncated segments	t_1
tend	common end time of truncated segments	t_2
k	number of intersections found	k
roots[2]	a vector of the times of the intersections found	$roots$
am	count of slots used in A vector	am
bm	count of slots used in B vector	bm
mesh_len	number of points in spline mesh	m_{length}
tbend	time for middle spline mesh point	$tbend$
eps	epsilon value used for machine roundoff for this routine only	ϵ

3.1.8.2 Mathematics:

This function starts by checking the times and trajectory, and performs simple error traps to ensure the segment inputs are reasonable, i.e. beginning time is before ending time. The next step is to define the ϵ value for round off checks in the calculations to follow. The ϵ value used for this function is the minimum of the following:

$$(0.000001)t_2 \text{ and } (0.1)(t_1 - t_2)$$

Since the time variables are defined by seconds (starting at zero seconds to 86400 seconds at time 2400 hours), the maximum value used for ϵ will be approximately 0.1 seconds.

Calculation of the spline mesh points for the lower bound spline starts by setting the first point to the beginning time and lower bound altitude. The function's next step is a check for level flight. For the lower bound trajectory with level flight, only two mesh points will be evaluated and am is set to 2.

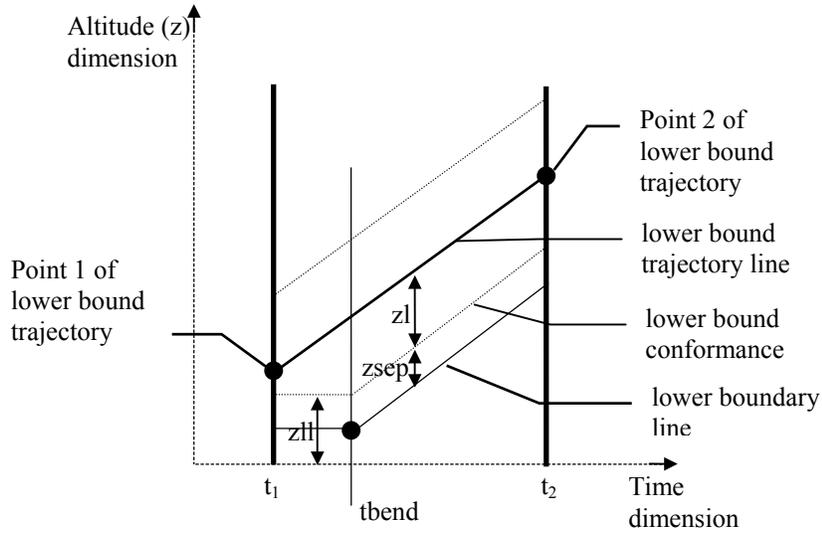


Figure 3.1.8-1: Example of lower bound trajectory and tbend

For the case where the lower bound trajectory is climbing or descending in the interval, the middle spline mesh point is calculated. The point where the trajectory line intersects the lower boundary limit is the middle spline mesh point, refer to Figure 3.1.8-1 above. There may not be a middle spline mesh point if the lower bound trajectory line intersects the lower boundary limit outside the time interval (for this case, the spline has only two mesh points). Solving for this middle spline mesh point's time, the following equation is solved:

lower boundary limit = lower boundary line

$$zll = zf(t) - zl \quad \text{Equation 3.1.8-1}$$

$$\text{where } zf(t) = [zl_1(t_2 - t) + zl_2(t - t_1)] / (t_2 - t_1) \quad \text{Equation 3.1.8-2}$$

So,

$$zll = [zl_1(t_2 - t) + zl_2(t - t_1) - zl(t_2 - t_1)] / (t_2 - t_1)$$

$$zll(t_2) - zll(t_1) = [zl_1(t_2) + t(zl_2 - zl_1) - zl_2(t_1) - zl(t_2) + zl(t_1)]$$

$$(zll + zl)(t_2 - t_1) - zl_1t_2 + zl_2t_1 = t(zl_2 - zl_1)$$

$$t = [(zll + zl)(t_2 - t_1) - zl_1t_2 + zl_2t_1] / (zl_2 - zl_1) \quad \text{Equation 3.1.8-3}$$

In the algorithm code, Equation 3.1.8-3's time t is stated as $tbend$ and examined to ensure that $tbend$ is inside the segments time interval. In other words, if $tbend$ is outside the time interval $[(t_1 + \epsilon) \text{ to } (t_2 - \epsilon)]$, then the lower boundary conformance line has only two spline mesh points at the ends of the interval. The same procedure is carried out for the upper bound trajectory and conformance line, where the $tbend$ is evaluated for this potential middle spline mesh point.

Once both sets of spline mesh points are defined, the points are combined at the same times and linearly interpolated so each uses the same mesh. Five cases of the mesh point combinations are considered by the algorithm (refer to Table 3.1.8-1). Example diagrams are illustrated in Figure 3.1.8-2. The first case is the combination of both boundary functions having 2 mesh points. For this case, there is no need for interpolation, since the points are all at the ends of the interval at equivalent times. The second and third cases have 2 mesh points for the one boundary and 3 for the other. The boundary function with the 2 mesh points is interpolated with the same middle time mesh point and stretched to have 3 mesh points. In the fourth case, both boundary functions have 3 mesh points and an equivalent middle time mesh, so no interpolation is required. In the fifth case, both boundary functions have 3 mesh points but the middle mesh point times are not equivalent. For this case, both boundary functions are interpolated and stretched to have 4 mesh points at equivalent times.

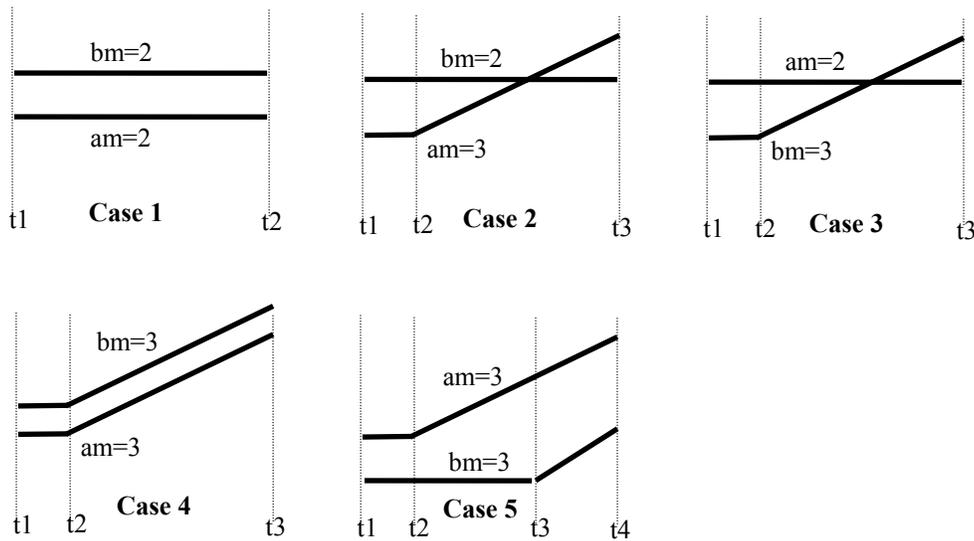


Figure 3.1.8-2: Examples of mesh point combinations for each case

No.	Variables	Mesh Length	Description
1	am=2 and bm=2	2	2 mesh points for both lines, both defined within ϵ time of the end points of the interval
2	am=2 only	3	2 mesh points for the lower bound (at the end points) while the upper bound has 3 mesh points (at ends and middle)
3	bm=2 only	3	2 mesh points for the upper bound (symmetric to above but in reverse)
4	am=3 and bm=3	3	3 mesh points for both lines and middle time mesh equivalent
5	am=3 and bm=3	4	3 mesh points for both lines and middle time mesh not equal

Table 3.1.8-1: Mesh point combinations

The mesh points are stored in two arrays of length 4 (i.e. $a(0)$, $a(1)$, $a(2)$, and $a(3)$). The a array stores the altitudes and times for the lower bound aircraft, while the b array stores the values for

the upper bound aircraft. Now, the mesh points are used to interpolate and define the intersections of the two boundary lines, $z_a(t)$ and $z_b(t)$, over the interval. Two main loops are used for either intersections at the end points of the intervals and the internal portion of the intervals.

The first loop performs two sequential checks to determine the first root times. If the mesh points meet the following conditions, the first root times are assigned:

1. The altitudes are equal at the adjacent mesh points ($a = b$), and
2. Either the next or prior mesh point altitude meets $a < b$ (excluding the beginning and end points of the interval)

These root times are assigned, when the root altitudes are equivalent and the next or prior points are not. The loop handles the cases where the splines are touching exactly (in the vertical dimension), but are not identical throughout the segment interval. If the splines are identical, they do not require this function since they are in intersection throughout the interval. If the splines are only equivalent for part of the interval, the root times are required to trim the interval and as starting and ending points for the interior mesh points.

The next loop checks for intersections in the interior portion of the segments by interpolating for the intersection point. In this loop, starting with the second mesh point (i.e. $i=1$, not 0), the b array altitudes are subtracted from the a array altitudes. For clarity in the derivation, assign the altitudes and times of the mesh points to the following variables:

$a[i][1] = a2$
$a[i-1][1] = a1$
$b[i][1] = b2$
$b[i-1][1] = b1$
$a[i][0] = t2$
$a[i-1][0] = t1$

In the code, the difference between the altitudes are calculated first and compared. Specifically, the difference between $a1-b2$ and $a2-b1$ are compared to determine if the mesh points cross. Referring to Figure 3.1.8-3, these two differences, defined in the code as $temp1 = a2-b2$ and $temp2 = a1-b1$, are multiplied and compared ($[temp1*temp2]<0$) for a negative value, which means the lines are intersecting and the interpolation computation is applicable.

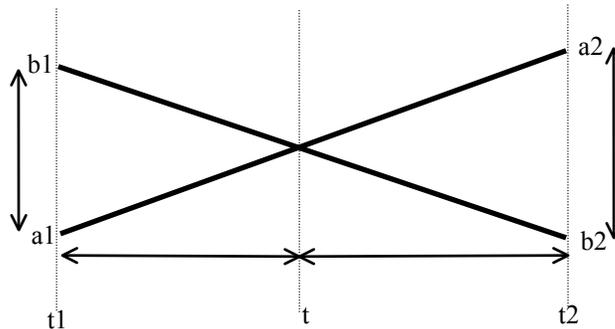


Figure 3.1.8-3: Mesh point intersection (altitude vs. time)

Again by referring to the four sides in Figure 3.1.8-3, the Equation 3.1.8-4 is defined as:

$$\left[\frac{b1 - a1}{t - t1} \right] = \left[\frac{a2 - b2}{t2 - t} \right] \quad \text{Equation 3.1.8-4}$$

Solve for t using Equation 3.1.8-4 and temp1 and temp2 definitions:

$$\begin{aligned} \left[-temp2 / (t - t1) \right] &= \left[temp1 / (t2 - t) \right] \\ -(t2)temp2 + temp2(t) &= temp1(t) - temp1(t1) \\ (temp2 - temp1)t &= (t2)temp2 - (t1)temp1 \\ t &= \left[(t2)temp2 - temp1(t1) \right] / \left[temp2 - temp1 \right] \end{aligned} \quad \text{Equation 3.1.8-5}$$

Now, using Equation 3.1.8-5 the root time t is rearranged into the same terms expressed in the code.

$$\begin{aligned} t &= \left[(t2)temp2 \right] / \left[temp2 - temp1 \right] - \left[temp1(t1) \right] / \left[temp2 - temp1 \right] \\ t &= \left[(t2)temp2 \right] / \left[temp2 - temp1 \right] - \left[temp2(t1) \right] / \left[temp2 - temp1 \right] + t1 \\ t &= t1 + \left[(t2 - t1)temp2 \right] / \left[temp2 - temp1 \right] \end{aligned} \quad \text{Equation 3.1.8-6}$$

In review, Equation 3.1.8-6 is expressed in the code and is an interpolation of the internal intersection point of the lower bound versus upper bound function as defined by the mesh points. Finally, the last loop just sorts the root times if required into the roots array. The final result is a sorted array of the intersection times in the root array.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.1.8-1	The ε value is used to define a cutoff value of a conflict in the time dimension. It is also used to provide robustness of the algorithm against roundoff error.	Reasonable approximation <i>if time values are in seconds</i> , ε for this algorithm is minimum of either: maximum of $[(0.000001)t_2] \approx 0.086400$ maximum of $[(0.1)(t_1 - t_2)] \approx -0.1$	Important
R 3.1.8-2	The climb or descent profile for the interval is assumed linear and the boundaries are assumed linear.	Reasonable approximation since the duration of the intervals are relatively small (<5000 feet in altitude change) and have approximately a constant climb rate	Important
R 3.1.8-3	For the definition of zll, zll, zul, and zu1, the variable names chosen are very difficult to distinguish between.	Relatively minor point, but for traceability and clarity changing the names or using capitol letters would be much more appropriate.	Minor

3.1.9 Function: CFP_POSIT (C)

Interpolates between two state vectors (consisting of x, y, z, t, and ground speed) to an intermediate time.

3.1.9.1 Description:

The state vector is given at both ends of a straight line segment. The function interpolates between the two state vectors to an intermediate time. The function considers both the case with no acceleration and with linear acceleration. For the case with no acceleration, the approach is a straight forward linear interpolation between the two segment end points. For the case with acceleration, the approach is based on the mapping of the along route distance formula to the x, y, and z being linear.

Table of Variable Definitions

Function Variable	Description	Math Symbol
P1	Initial node position or state vector of aircraft, contains five fields: x, y, z, t, and ground speed	x_1, y_1, z_1, t_1, g_1
P2	Final node position or state vector of aircraft, contains five fields: x, y, z, t, and ground speed	x_2, y_2, z_2, t_2, g_2
K	Index indicating which P vector is to be over written with the interpolated vector (= 1 for P1 and 2 for P2)	k
Ti	Interpolation time, which is between P1.time and P2.time	t_i
cfp_inp.apd_spd_epsilon	Ground speed difference or tolerance required to consider acceleration present	$speed_epsilon$
ALPHA	Weight for position 1 for linear interpolation; alpha + beta = 1	$\alpha.$
BETA	Weight for position 2 for linear interpolation	$\beta.$
GAMMA	Denominator for the quadratic interpolation, used in the acceleration present case	$\gamma.$
CUSP Q	Interpolated position vector	qx, qy, qz, qt
RETURN VALUE	0 = no error -1 = error in times of cusps -2 = k is 1, input time is after end cusp time -3 = k is 2, input time is prior to first cusp time	

3.1.9.2 Mathematics:

The function begins by checking for trivial or wrong input data.

The interpolation starts by defining the α and β values. For the case without acceleration, the interpolation vector is based on the following formula:

$$s(t) = [(t_2 - t)s_1 + (t - t_1)s_2] / (t_2 - t_1) \quad \text{Equation 3.1.9-1}$$

$$s(t) = (\alpha.)s_1 + (\beta.)s_2 \quad \text{Equation 3.1.9-2}$$

$$\text{where } \alpha. = (t_2 - t) / (t_2 - t_1) \text{ and } \beta. = (t - t_1) / (t_2 - t_1)$$

With:

Variable	Description
$s(t)$	Represents the x, y, or z in respect to time
t_1 or t_2	The time the aircraft crosses the segment end points 1 or 2

s_1 or s_2	The dimension parameter value (x, y, or z) at the end points 1 or 2
t	Interpolation time

The Equation 3.1.9-1 is derived from calculating the linear interpolation between the two segment endpoints, based on the formula :

$$\left[\frac{(s_2 - s_1)}{(t_2 - t_1)} \right] / \left[\frac{(s(t) - s_1)}{(t - t_1)} \right] \quad \text{Equation 3.1.9-3}$$

Solving for $s(t)$ with $t_1 < t < t_2$:

$$\left[s_2(t - t_1) \right] - \left[s_1(t - t_1) \right] + \left[s_1(t_2 - t_1) \right] = \left[s(t)(t_2 - t_1) \right]$$

$$\left[s_2(t - t_1) \right] + \left[s_1(t_2 - t) \right] = \left[s(t)(t_2 - t_1) \right]$$

$$s(t) = \left[s_1(t_2 - t) + s_2(t - t_1) \right] / (t_2 - t_1)$$

For the case where acceleration must be considered, the following equation models the position for x, y, and z:

$$s(t) = \frac{A(t) * s_1 + B(t) * s_2}{C} \quad \text{Equation 3.1.9-4}$$

where

$$A(t) = (g_1 - g_2)(t_2 - t)^2 + 2g_2(t_2 - t_1)(t_2 - t)$$

$$B(t) = (g_2 - g_1)(t - t_1)^2 + 2g_1(t_2 - t_1)(t - t_1)$$

$$C = (g_1 + g_2)(t_2 - t_1)^2$$

Therefore, Equation 3.1.9-4 is illustrated similar to Equation 3.1.9-2 with an α and β terms. The key result is the sum of both of these terms is equivalent to one.

$$\alpha + \beta = 1 \quad \text{Equation 3.1.9-5}$$

$$\text{where } \alpha = \frac{A(t)}{C} \text{ and } \beta = \frac{B(t)}{C} \text{ from Equation 3.1.9-4.}$$

From this equation and as stated in the code's comments, A and B terms sum to C ($A(t) + B(t) = C$), so the airspace segment is partitioned into two weighted parts. The denominator term, C, is a quadratic equation proportional to the average velocity times the time

interval squared. The quadratic equation (or the C term above) can be expressed as:

$$C = (g_1 + g_2)(t_2 - t_1)^2 = (g_1 + g_2)(t_2 - t)^2 + 2(g_1 + g_2)(t_2 - t)(t - t_1) + (g_1 + g_2)(t - t_1)^2$$

where $A(t) = (g_1 + g_2)(t_2 - t)^2 + 2g_2(t - t_1)(t_2 - t)$ and

$B(t) = (g_1 + g_2)(t - t_1)^2 + 2g_1(t - t_1)(t_2 - t)$. Therefore, the intermediate location along all dimensions x, y, and z are all continuous for the same relationship between time and ground speed. The result is a location along the flight segment at time t which is quadratically weighted by both sides of the interval.

For further verification, the Equation 3.1.9-4 is compared to generally derived equations of motion. The quadratic equation above interpolates between the two endpoints of the segment, giving equal weight to both sides of the time interval (or segment). If the acceleration is truly constant for the duration of the segment, the Equation 3.1.9-4 will produce the exact same result as the following equation from Calculus:

Constant acceleration is equivalent to $a = \frac{dv}{dt}$, and solving for velocity by integrating:

$$\int_{v_0}^v dv = a \int_0^t dt = at = v - v_0, \text{ the velocity becomes:}$$

$$v = v_0 + at \quad \text{Equation 3.1.9-6}$$

Now, the relation for velocity is $v = \frac{ds}{dt}$, and by integrating again, the distance, s , traveled becomes:

$$ds = v \cdot dt, \text{ so } \int_{s_0}^s ds = \int_{t_1}^{t_2} v \cdot dt \Rightarrow s - s_0 = v_0(t_2 - t_1) + \frac{a}{2}(t_2 - t_1)^2 \quad \text{Equation 3.1.9-7}$$

From numerical comparison between the distance traveled in Equation 3.1.9-4 and Equation 3.1.9-7 under various constant acceleration quantities, there was no significant difference between the distance calculated by both equations. However, if the acceleration does not exactly remain constant over the interval, the results of the two equations do diverge. In other words, Equation 3.1.9-4 is a quadratic function based on both sides of the equation, while Equation 3.1.9-7 uses the $\frac{\Delta \text{velocity}}{\Delta \text{time}}$ of the segment for the acceleration and starts from one side of the interval. Equation 3.1.9-4 uses a quadratic interpolation function that weights both sides of the interval based on the difference in the particular end point times and ground speed.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.1.9-1	For aircraft with acceleration during the interval, the function applies a quadratic interpolation. It assumes constant acceleration over the interval.	The interpolation technique does seem reasonable when acceleration is not constant over the interval, since it will still weight by ground speed and end point times. For most cases, the segment's acceleration is approximately constant anyway which is how TJM defines the state segments. For this case with constant acceleration, the function produces the same result as the general equations of motion.	Important

3.1.10 Function: ECP (PL/I)

This function checks an aircraft's trajectory for penetration of a blocked airspace. The results of the probe are returned to the external data structure ECP_OUT.

3.1.10.1 Description:

The function iterates through all the blocked airspaces inside the data base table BAS. For each BAS entry (i.e. blocked airspace), the aircraft trajectory state segments and the blocked airspace vertices are sent to ST_CHK_VP, which actually determines if the aircraft does enter the particular airspace region and what the entry and exit points are. This information is then placed into the ECP_OUT structure.

Function calls include:

- ST_CHK_VP: This is the *main function for ECP* which in turn calls GM_REGN, GM_TSTPNT and GM_INSEC which carry out the actual airspace violation calculations.
- DB_FIND_AUD_PTR: The function returns the AUD pointer and size for the specified AUD index.
- DB_LOC_FIRST: The function locates the first entry to the specified system table.
- DELTECB: The function deletes a specified ECB from the ECB list. The ECB is the environmental conflict box or data structure which contains all the aircraft-to-airspace conflict information.
- MAKE_ECB: The function loads ECB structure in the aircraft's AUD. Based on the volume (i.e. BAS) penetration data defined in the VP_OUT structure, the function creates a linked list of ECB's in the AUD. Also it builds the ECP_OUT table containing the conflicts in order of occurrence up to the table maximum.

3.2 Trajectory Modeler

The Trajectory Modeler (TJM) produces a detailed four-dimensional aircraft trajectory for use by the other URET functions. It describes the aircraft's route of flight in both horizontal and vertical dimensions. Each point (x, y, z, t) along the trajectory is referred to as a cusp and the straight line between cusps is referred to as a segment.

The trajectory modeling routine includes five basic software functions -- Horizontal Route Analysis, Horizontal Route Analysis Step B, Nominal Profile Builder, Modeler, and Hand-off. The majority of the lower level algorithmic calculations are actually performed in the library of utility functions (see Section 3.4); the higher level algorithms are omitted or only briefly described in this section.

Modeler (MDL)

The modeler produces a list of state segments (SSGs) which comprise the trajectory. It is here where aircraft positions and time are calculated in the vertical dimension.

The Modeler process will modify the start point of a gradient (or slope) to satisfy a restriction. The Modeler also determines conformance bounds for each aircraft; this includes adjusting the bounds for altitude transitions, turns, or when entering special airspaces. The Modeler also determines the required separation distances. The separation distances are based on FAA Order 7110.65.

Low level functions which actually perform the algorithmic calculations are described in the library of utility functions in Section 3.4 (DB_AIR_AT_POINT, DB_FIND_AUD_PTR, ST_MAXTAS, ST_MINTAS, GM_BRNG, CNV_GRDSPD, CNV_CNVSPD, CNV_STD_ATMOS, ST_CLIMB_GRADIENT, ST_IASALT, ST_MACHALT, ST_DESCENT_DIST, ST_DESCENT_GRADIENT, CNV_SPEED, ST_CHK_VP, ST_TIME_SSGDATA, GM_TURN, ST_ARD_SSGDATA).

Horizontal Route Analysis (HRA)

The Horizontal Route Analysis routine creates the horizontal trajectory dimension. It performs route interpretation as specified in NAS-MD-312. Fixes and route information are loaded from the Adaptation Controlled Environment System (ACES) adaptation files. It ensures that appropriate controller/facility preferred routing is assigned (i.e. Preferred Arrival Route (PAR), Preferred Departure Route (PDR), preferred Departure and Arrival Route (PDAR), etc.) as well as applicable altitude and speed restrictions. This routine breaks the flight plan route string into a descriptive list of fixes and builds a data structure (known as on-board route segments (ORSs)) which describes each segment defined between two successive fixes.

Low level functions which actually perform the algorithmic calculations are described in the library of utility functions in Section 3.4 (LO_FIND, CNV_XYLL, GM_INSEC, GM_PTLIN, CNV_RADDMS, CNV_GNOMONIC_STEREO, CNV_STEREO_GNOMONIC, GM_BRNG, DB_FIND_AUD_PTR).

Horizontal Route Analysis Step B (HRB)

Defines an ORS which joins the current position of the aircraft to the filed route. The routine is called during a route reconformance or when a trial plan is created.

Low level functions which actually perform the algorithmic calculations are described in the library of utility functions in Section 3.4 (DB_FIND_AUD_PTR, CNV_XYLL, GM_PTLIN).

Nominal Profile Builder (NPB)

The Nominal Profile Builder creates the vertical and speed dimension of the trajectory. It builds planned actions for future changes in an aircraft's speed and altitude. The process is decomposed into three subsections - altitude processing, speed processing, and delay processing. The altitude processing organizes all altitude restrictions associated with the aircraft route and builds planned actions (PAs) to transition the aircraft to the target altitudes or altitude restrictions. The speed processing does a similar process for speed restrictions.

Low level functions which actually perform the algorithmic calculations are described in the library of utility functions in Section 3.4 (DB_FIND_AUD_PTR, LO_FIND, GM_INSEC, ST_TRANSLATE_ARD, ST_DESCENT_DIST, ST_CLIMB_DIST, DB_CDMERG).

Handoff (HDO)

The Hand-Off routine computes the time and position of entry and exit from each sector, or from the AERA boundary.

Low level functions which actually perform the algorithmic calculations are described in the library of utility functions in Section 3.4 (DB_FIND_AUD_PTR, ST_CHK_VP).

3.2.1 Function: ARDXY (PL/I)

Finds x, y for a given ARD.

3.2.1.1 Description:

Given an Along Route Distance (ARD), this function will return the position data (x, y) within a given On-Board Route Segment (ORS). This function is used to support the FTME and TPRIME functions with missing x, y information. These higher level functions are used to supply the modeler with the necessary SSG end-point data.

Table of Variable Definitions

Function Variable	Description	Math Symbol
X, Y	Coordinates of the aircraft at the given ARD (ft)	x, y
ORS.XSN, ORS.YSN, ORS.XEN, ORS.YEN	Coordinates of the start and end points of the On-Board Route Segment (ft)	x_1, y_1 x_2, y_2
ORS.LNGTH	The length of the On-Board Route Segment (ft)	ℓ
ARD_IN	Along Route Distance (ft)	ard
ORD.ACCUM_DIST	ARD at the beginning of the On-Board Route Segment (ft)	ard_1

3.2.1.2 Mathematics:

The function performs the following simple calculations and logic.

First it checks if the given ARD is less than the accumulated distance up to the start of the given ORS. If it is, it simply assigns the coordinates of the starting point of the ORS to the output x, y.

$$x = x_1 \quad \text{Equation 3.2.1-1}$$

$$y = y_1$$

Otherwise the function finds the ratio of the length the aircraft traveled from the start of the ORS to the ARD over the total length of the ORS

$$r = \frac{ard - ard_1}{\ell} \quad \text{Equation 3.2.1-2}$$

The function uses this ratio to interpolate the x and y coordinates at the ARD position from the ORS endpoints.

$$x = x_1 + r(x_2 - x_1) \quad \text{Equation 3.2.1-3}$$

$$y = y_1 + r(y_2 - y_1) \quad \text{Equation 3.2.1-4}$$

The values for x, y are returned from this function.

There are no assumptions or approximations made in this module which would have significant impact on the algorithms.

3.2.2 Function: EGRAD (PL/I)

Computes the adjustment to the altitude gradient angle from the effects of wind.

3.2.2.1 Description:

A typical vertical trajectory is first associated with a predetermined nominal trajectory estimated with no wind. The nominal trajectory is stored in a table and associates altitude gradients, estimated from “normal aircraft flight characteristics”, to changes in flight levels (during ascents and descents). Since this altitude gradient is estimated for no wind, an adjustment to the gradient must be made when a known value for wind speed and direction is defined along the predicted trajectory.

Table of Variable Definitions

Function Variable	Description	Math Symbol
WIND_SPEED	Scalar value for wind speed	V_w
CROSS_WIND	Wind speed along the transverse axis of the trajectory	V_{wt}
WIND_BEARING	Direction of wind with respect to true North	θ_w
ALONG_TRACK_WIND	Wind speed along the longitudinal axis of the trajectory	V_{wl}
CURRENT.SPEED.TAS	Current true airspeed	V_t
GSPD	Ground speed	V_g
CHANGES.EGRD	The wind corrected effective gradient; the ratio of the altitude change over horizontal distance traveled (ft/ft). Note that the math symbol actually represents the angle made by the gradient	γ_i
CHANGES.GRD	Air mass gradient; the ratio of the altitude change over horizontal distance traveled (ft/ft). Note that the math symbol actually represents the angle made by the gradient	γ_a
CHANGES.BRG	Aircraft bearing with respect to true North.	ψ

3.2.2.2 Mathematics:

3.2.2.2.1 Ground Speed

The EGRAD function first defines the ground speed as:

$$V_g = \sqrt{V_t^2 - [V_w \sin(\Psi - \theta_w)]^2} + V_w \cos(\Psi - \theta_w) \quad \text{Equation 3.2.2-1}$$

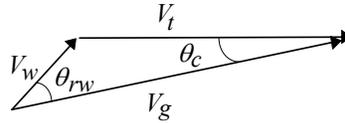
If we make the following assignment

$$\theta_{rw} = \Psi - \theta_w \quad \text{Equation 3.2.2-2}$$

where θ_{rw} is the relative angle between the course heading of the aircraft and the wind direction, Equation 3.2.2-1 can be reoriented as

$$V_g = V_t \sqrt{1 - \left(\frac{V_w}{V_t} \sin \theta_{rw}\right)^2} + V_w \cos \theta_{rw} \quad \text{Equation 3.2.2-3}$$

Equation 3.2.2-3 is based on the vector representation of ground speed



and

$$V_g \approx V_t \cos \theta_c + V_w \cos \theta_{rw} \quad \text{Equation 3.2.2-4}$$

where θ_c is the “crab” angle between the V_t -leg and V_g -leg. Equation 3.2.2-3 and Equation 3.2.2-4 are approximations because they assume a small flight path angle (i.e. negligible effect from motion in the vertical plane).

Therefore Equation 3.2.2-1 appears to be reasonable and coincides with most trajectory estimation practices that assume a small flight path angle and a horizontal wind direction.

3.2.2.2.2 Effective Gradient Angle

The effective gradient angle is then calculated as:

$$\text{EGRD} = \gamma_i = \gamma_a \frac{V_t}{V_g} \quad \text{Equation 3.2.2-5}$$

where γ_a is the gradient angle in the aerodynamic reference frame, which is equivalent to the nominal gradient angle assignment when no wind exists. The ratio of true airspeed to ground speed multiplied by the nominal gradient angle will adjust the gradient angle for the wind effect.

This relationship also appears reasonable and coincides with other trajectory assumptions. However this relationship is based on other approximations, namely:

- The flight path angle during ascent or descent is small so that

$$\begin{aligned} \sin \gamma_{a,i} &\approx \gamma_{a,i} \\ \cos \gamma_{a,i} &\approx 1 \end{aligned} \quad \text{Equation 3.2.2-6}$$

- The altitude rate is described as

$$\frac{dh}{dt} = V_t \sin \gamma_a$$

$$\approx V_t \gamma_a$$

Equation 3.2.2-7

and

$$\approx V_g \gamma_i$$

- Therefore arriving at the relationship

$$\frac{\gamma_a}{\gamma_i} \approx \frac{V_g}{V_t}$$

Equation 3.2.2-8

All of these approximations are reasonable for most commercial aircraft flight paths. For very steep ascents or descents (for example, greater than 45°) these approximations may be less accurate. Also see the analysis for the CNV_GRDSPD function, Section 3.4.2.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.2.2-1	Small inertial path angle Equation 3.2.2-6	Reasonable, for Commercial Aircraft, and a small inertial flight path angle (i.e. <= 45 degrees)	Critical
R 3.2.2-2	Derivation of ground speed, Equation 3.2.2-3	Reasonable for the given vector representation and relative wind and crab angle definitions. Assumes a small flight path angle, as in Equation 3.2.2-6, and a horizontal wind.	Important
R 3.2.2-3	Derivation of effective gradient, Equation 3.2.2-5	Reasonable, for small path angle approx.	Critical

3.2.3 Function: INTMDL (PL/I)

Function which initializes the MDL.

3.2.3.1 Description:

Function which initializes the MDL_INFO and several trajectory model (MDL) fields, deletes all state segments (SSG), sets the status of all Planned Actions (PA) to 0, checks for bad data in flight plans, and sets the modeling to begin with the Along Route Distance (ARD) set to zero. Here the CURRENT data structure is established, which describes the initial x, y, and z position, time, ground speed and air speed. The time is calculated by converting the flight plan's initial time to the time, in seconds, since the cold start (using the CNV_TIN function).

3.2.4 Function: HRB (PL/I)*

The Horizontal Route Analysis Step B (HRB) function defines an ORS which joins the current position of the aircraft to the filed route. The routine is called during a route reconformance or when a trial plan is created.

3.2.4.1 Description:

The HRB function is called when: the TKM determines that the track data is out-of-conformance with respect to the trajectory, or a trial plan requests a new route which eventually merges back with the filed route. HRB will build an ORS from the current position to the end of the next ORS which satisfies a parameter constraint. All of the remaining ORSs which were previously built for the aircraft's current trajectory will then be appended to the newly created HRB ORS.

Table of Variable Definitions

Function Variable	Description	Math Symbol
OFF_DIST	The smallest distance from the current position to the associated ORS	d_{off}
STD_LAT_CONFORMANCE	The standard lateral conformance bounds. In D1.A, this parameter is assigned a value of 2.5 nmi.	C_L
MIN_DIST	The minimum of the joining distance permitted in the code.	d_{min}
X_PROJECTION, Y_PROJECTION	The coordinates of the point which is projected on the current associated ORS line segment from the current position	x_p, y_p
MAX_JOIN_ANGLE	The maximum join angle. The D1.A database assigns this parameter equal to 15 degrees	α_{min}
MIN_JOIN_DIST	The minimum joining distance. The D1.A database assigns this parameter equal to 50 nmi.	D_{jmin}
PROJECTION_TO_FIX	Distance from the projected point on the associated ORS to the endpoint of a downroute ORS	d_{proj}
XX, YY	The coordinates of the current position of the aircraft	x, y
RTE_ORS.XEN, RTE_ORS.YEN	The coordinates of the endpoint of the ORS	$x_{\text{ors}}, y_{\text{ors}}$
ORS.LNGTH	The length of the ORS	l_{ors}

3.2.4.2 Mathematics:

HRB first determines which ORS is associated with the current position of the aircraft using the Along Route Distance (ARD) and the approximate distance (d_{off}) between the ORS and the current position (determined by using the function GM_PTLIN (Section 3.4.16)).

* Note: The following analysis was based on URET Version D1.A. Revisions which will describe this function as it appears in URET Version D1.1 are still being developed.

3.2.4.2.1 HRB for a Reconformance

If the considered route is a current plan, the HRB function assumes that the current position of the aircraft is out of lateral conformance:

$$d_{off} > C_{Lat} \quad \text{Equation 3.2.4-1}$$

Next, the minimum allowable distance⁴ from the current position to the next ORS is determined by:

$$d_{min} = \max \left[D_{jmin}, \frac{d_{off}}{\tan(\alpha_{max})} \right] \quad \text{Equation 3.2.4-2}$$

Loop through every subsequent ORS, checking to see if any satisfy the criteria

$$\text{Equation 3.2.4-3}$$

- 1) $d_{proj} > d_{min}$
- 2) The ORS does not have a delay
- 3) $d_{proj} + \varepsilon \neq 0$

Where ε is a very small parameter value.

The projection to the fix, d_{proj} , is calculated as the distance from the projected point on the current associated ORS⁵, (x_p, y_p) , to the end point of the ORS being considered.

$$d_{proj} = \sqrt{(x_{ors} - x_p)^2 + (y_{ors} - y_p)^2} \quad \text{Equation 3.2.4-4}$$

Note: It is questionable why d_{proj} is calculated using (x_p, y_p) instead of using the current position (x, y) , since the new ORS is built from the current position. See Figure 3.2.4-1.

The first ORS which satisfies the criteria in Equation 3.2.4-3 will contain the endpoint to which the new ORS will be built (see Section 3.2.4.2.2).

⁴ In D1.A, $D_{jmin} = 50$ nmi and $d_{off} > C_L (= 2.5$ nmi). Therefore $d_{off} / \tan(\alpha_{max}) \geq 9.33$ nmi. In order for $d_{off} / \tan(\alpha_{max}) \geq D_{jmin}$, d_{off} must be greater than 13.4 nmi.

⁵ (x_p, y_p) is calculated from the function GM_PTLINE

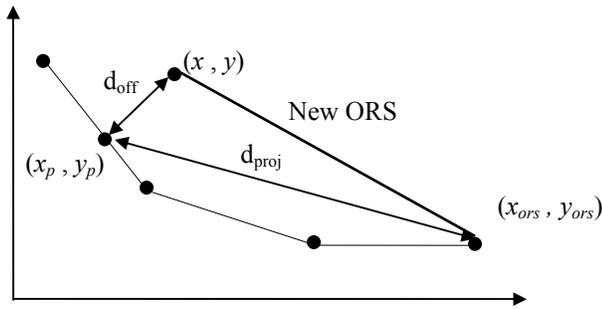


Figure 3.2.4-1: Example of an HRB Return-to-Route

3.2.4.2.2 Create New ORS

From the current position to the end point of the ORS which satisfied the criteria in Equation 3.2.4-3, make a new ORS with length

$$l_{ors} = \sqrt{(x_{ors} - x)^2 + (y_{ors} - y)^2} \quad \text{Equation 3.2.4-5}$$

Making sure that $(l_{ors} + \varepsilon) > 0$.

Append the remaining ORSs to the route string and recalculate the accumulated distance.

Note: The steps described above which involve returning the aircraft to its route are now performed in the TKM_RTR function in URET D1.1. There is additional logic in TKM_DETERMINE_CASE which considers several different flight history characteristics before deciding on a method to return the aircraft to its horizontal route. The description and analysis of this process is not included in this report.

3.3 Track Management

Track Management (TKM) monitors track reports received from the HCS against each aircraft's trajectory to determine whether an aircraft is "out-of-conformance" in a particular dimension (lateral, longitudinal, vertical), and determines the necessary maneuvers to remodel the trajectory. TKM also makes an assessment about the track categories of the flight, depending on the quality of the track data and the state of the aircraft. These categories are used as a basis for determining whether a "reconformance" should be performed and whether an aircraft is eligible for conflict probing by Automated Problem Detection.

TKM includes six major software functions: Match ID, Verify Data, Change Category, Check Airspace, Monitor Conformance, and Compute Reconformance. Because the Track Management subsystem was under continuous development throughout the course of this assessment (especially the TKM_COMPUTE_RECONFORMANCE module and its underlying functions), it was not possible to assess these functions at the same level of detail as the other algorithmic subsystems. Low level functions which actually perform the algorithmic calculations are described in the library of utility functions (see Section 3.4 - DB_AIR_AT_POINT, GM_BRNG, DB_FIND_AUD_PTR, LO_FIND, ST_TIME_SSGDATA, GM_REGN); the higher level algorithms are omitted or only briefly described in this section.

3.3.1 Function: CNV_GRD_TO_TAS (PL/I)

This function calculates True Airspeed from a given ground speed, position/altitude information and track bearing.

3.3.1.1 Description:

This function computes the aircraft's true airspeed based on the ground speed, the x, y, and z position coordinates, and the track bearing.

Table of Variable Definitions

Function Variable	Description	Math Symbol
WIND_SPEED	Scalar value for wind speed	V_w
WIND_BEARING	Direction of wind with respect to North	θ_w
TAS	True airspeed	V_t
GROUND_SPEED	Ground speed	V_g
TRACK_BEARING	The flight path bearing.	Ψ

3.3.1.2 Mathematics:

The function uses the x, y, and z position data in the DB_AIR_AT_POINT function (Section 3.4.10) to calculate the wind data (the x and y wind components, wind temperature and pressure). It then calculates the wind bearing using the GM_BRNG function (Section 3.4.13). It takes the difference between the wind bearing and the track bearing and assigns this value as the relative wind angle, θ_{rw}

$$\theta_{rw} = \Psi - \theta_w \quad \text{Equation 3.3.1-1}$$

The function uses all of these values to solve for true airspeed with the following equation

$$V_t = \sqrt{(V_g - V_w \cos \theta_{rw})^2 + (V_w \sin \theta_{rw})^2} \quad \text{Equation 3.3.1-2}$$

Equation 3.3.1-2 is simply (Equation 3.4.3-1 from Section 3.4.3) reordered to solve for true airspeed, V_t . For the derivation, refer to Section 3.4.3.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TKM
R 3.3.1-1	Small inertial path angle	During steep climbs or descents, the true airspeed will be calculated with error	Critical

3.3.2 Function: GM_PTSEG (PL/I)

Finds the relationship between a point and a line.

3.3.2.1 Description:

Refer to Section 3.4.16 which describes the GM_PTLINE function. The GM_PTSEG function is very similar to the GM_PTLINE function except that it also calculates the distance between (x_1, y_1) and (x_2, y_2) .

3.3.3 Function: LEASTSQ (C)

Finds the least square parameters for a linear first order (and second order optional) regression model.

3.3.3.1 Description:

This function performs the least squares fit of dependent data vector to independent data vector. The function can find the least square parameters for both a first order or second order regression model, however only the first order model is being used in D1.1.

This function uses the classic linear regression model solving for the parameters α_0 and α_1 .

$$\hat{Y} = \alpha_0 + \alpha_1 X$$

Where X are the independent data and \hat{Y} is the predicted value of the dependent Y data. The values for α_0 and α_1 are determined as follows:

$$\alpha_0 = \frac{(\sum X^2)(\sum Y) - (\sum XY)(\sum X)}{n\sum X^2 - (\sum X)^2}$$

$$\alpha_1 = \frac{n\sum XY - (\sum X)(\sum Y)}{n\sum X^2 - (\sum X)^2}$$

The function also checks to ensure there are no divisions by zero. This function is based on classic principles⁶ and is reasonable.

⁶ See Draper, N., and Smith, H., *Applied Regression Analysis, Second Edition*, New York, NY.: John Wiley and Sons, 1988, Chapter 1.

3.3.4 Function: TKM_CATEGORY_CHANGE (C)

Determines the category that best describes the current state of the aircraft's track and flight plan data.

3.3.4.1 Description:

This function maintains a category for each aircraft for use by TKM_Conformance_Monitor and Automated Problem Detection in determining which aircraft should be subject to those functions. The categories are defined below:

- Category A: Have flight plan data and reasonable, continuous track data
- Category B: Have flight plan data and intermittent track data or short track history
- Category C: Have flight plan data but no track data (aircraft is proposed departure)
- Category D: Have flight plan data and track data but aircraft is in a hold
- Category E: Have: 1) flight plan data and track data, but unable to model trajectory from flight plan
2) track data but no flight plan data
- Category F: Have flight plan data but no track data (aircraft is inbound from another facility or outbound from this facility)

An aircraft's category is initialized to category C, E or F outside of this function (i.e., Replan Manager (RPM)) for first flight plans. TKM_CATEGORY_CHANGE (called by TKM_CORE AND TKM_CONTROL) then determines if the category should remain the same, or be changed, via the following logic flow:

```

If there is no track history for the aircraft, then the current category will remain unchanged
    else
If there is track history, but the amount is less than or equal to a parameter representing "reasonable
history" (TKM_REASONABLE_HISTORY) or has more than a parameter number of "holes"
(HOLES_IN_TRACK_HISTORY), then the current category will be set to "B"
    else
If the input category is "D" and the aircraft is determined to still be in a hold, then the category will
remain unchanged (i.e., CURRENT_CATEGORY=IN_CATEGORY)
    else
If the input category is "F" and the aircraft is determined to be: in tactical airspace, leaving the
ARTCC airspace, or leaving the APD inhibited area (APDIA), then the category will remain
unchanged
    else
Set the current category for this aircraft to "A"

```

Table of Variable Definitions

Function Variable	Description
AC_INDEX	Index to aircraft history in TKM_HISTORY
TKM_REASONABLE_HISTORY	Minimum amount of track history that aircraft must have to be considered "reasonable" (parameter: 2)
CURRENT_TIME	Current global time
RADAR_UPDATE	Time interval between two consecutive radar updates (currently 12 seconds)
HOLES_IN_TRACK_HISTORY	Number of holes required before aircraft can be downgraded to B category (parameter: 3)
IN_CATEGORY	Input aircraft track category
CURRENT_CATEGORY	Current category indicator

3.3.5 Function: TKM_CHECK_AIRSPACE (C)

Determines whether aircraft is in special airspaces.

3.3.5.1 Description

This function checks the aircraft's current track position against several special airspaces: leaving the tactical airspace around the departure airport, entering tactical airspace around the arrival airport, and leaving the ARTCC facility boundary. If the aircraft is currently within any of these airspaces (determined by TKM_GM_REGN; returns "1" if inside tactical airspace, "0" if outside), a flag will be set to cause conformance monitoring to be discontinued for this aircraft.

Table of Variable Definitions

Function Variable	Description
AC IX	Index to aircraft's track history
TRACK_X, TRACK_Y	X, Y coordinates of TRACK_DATA
TRACK_T	Time of TRACK_DATA
TRACK_Z	Altitude of TRACK_DATA
IN_TACTICAL_AIRSPACE	Flag indicating whether aircraft is in tactical airspace or not
VERTICAL_MONITOR	Flag indicating whether vertical conformance monitoring should be performed or not.
CATEGORY	Aircraft category

3.3.6 Function: TKM_COMPUTE_RECONFORMANCE (C)

Determines if an aircraft's trajectory should be reconfirmed in accordance with its reported track position and the proper actions to bring the aircraft back into conformance.

3.3.6.1 Description:

This function, initiated when an aircraft's conformance bounds are exceeded a consecutive number of times (parameters: lateral=2, longitudinal=2, vertical=1), determines the specific actions required to remodel the trajectory to bring the aircraft's predicted position into conformance with its reported track position. If the aircraft is out of conformance in the lateral and/or longitudinal dimension(s), the aircraft will be reconfirmed in that dimension as well as in the vertical dimension. If the aircraft is out of conformance in the vertical dimension, it will be reconfirmed only in the vertical dimension.

The function first makes a general check to ensure that the aircraft is not in "vertical drift" or "far away" from its route. Vertical drift exists when the aircraft is out of vertical conformance and is beyond the minimum and maximum vertical bounds associated with the current flight characteristic. For example, an aircraft that is in the middle of an altitude transition which is either greater than the ssg_max_conform_z parameter or less than the ssg_min_conform_z parameter is considered in a vertical drift. An aircraft is considered far away from its route if its current position is greater than a parameter horizontal distance (in D1.1, the value is far_away_distance = 30 nmi) from its associated trajectory segment. If it has been determined that the aircraft is in either vertical drift or far away from its route, the function will not attempt to reconfirm the trajectory. Consequently the aircraft will no longer be probed by the conflict detection logic.

TKM reconfirms the aircraft trajectory that is out of conformance in the longitudinal dimension by calling the TKM_DETERMINE_SPEED function to compute the track speed of the aircraft. It then calculates a multiplying factor to weigh both the track speed and the trajectory speed to determine a new "reconfirmed" speed for the trajectory. Similarly, for the reconfirmation in the vertical dimension, the

TKM_DETERMINE_GRADIENT function is called to compute the actual track gradient the aircraft is flying. It then calculates a multiplying factor to weigh both the track gradient and the trajectory gradient to determine a new “reconformed” gradient for the trajectory.

Finally, if the aircraft is out of lateral conformance, the function calls the TKM_DETERMINE_CASE function. TKM_DETERMINE_CASE determines, from a history of the track report, the characteristic of the flight path with respect to the original filed horizontal route. TKM_DETERMINE_CASE uses a heuristic method based on the observed flight characteristic to determine a position along the original filed route, downstream from the current aircraft position, to reconform the aircraft’s trajectory (and in one case it determines an intermediate point before it returns the aircraft to its route).

TKM will then supply the reconformance information (i.e. new gradient, new speed, or return to route position) to the Replan Manager (RPM) and Trajectory Modeler (TJM) where a new reconformed trajectory structure can be built.

3.3.7 Function: TKM_CONFORMANCE_MONITOR (C)

Determines if an aircraft is out of conformance in longitudinal, lateral, and/or vertical dimension.

3.3.7.1 Description:

This function checks to see if the aircraft’s track position is within the defined conformance bounds (i.e., the conformance region) around its’ predicted trajectory position (for category A and B aircraft only), and maintains the out of conformance count and reason for each dimension. Following is a description of the basic logic flow of the function:

If the HORIZONTAL_MONITOR flag is TRUE, then the extrapolated position of the aircraft (difference between ARD_BY_TIME and TRAJ_ARD) is compared with the longitudinal conformance bound (SSG_LONG_CONFORM), and the appropriate direction is set in LONG_IND (blank if within conformance). Next, the DISTANCE_TO_SSG is compared with the lateral conformance bound (SSG_LAT_CONFORM), and the appropriate direction is set in LAT_IND. Likewise, if the VERTICAL_MONITOR flag is set, vertical conformance is checked by determining whether TRACK_Z is within the vertical conformance bounds, and the appropriate direction is set in VERT_IND. This is done by comparing TRACK_Z with the minimum and maximum allowable altitudes for the segment (SSG_MIN_CONFORM_Z and SSG_MAX_CONFORM_Z,) and with the trajectory altitude (TRAJ_Z) plus/minus vertical conformance bounds (SSG_BOT_CONFORM and SSG_TOP_CONFORM).

Table of Variable Definitions

Function Variable	Description
AC_INDEX	Index to TKM_HISTORY
TRACK_Z	Reported track altitude for this aircraft
TRAJ_Z	Predicted (trajectory)altitude for this aircraft
ARD_BY_TIME	Projected along route distance by track
TRAJ_ARD	Estimated along route distance from trajectory
DISTANCE_TO_SSG	Shortest distance from track to trajectory
SSG_LONG_CONFORM	Longitudinal conformance bound
SSG_LAT_CONFORM	Lateral conformance bound
SSG_TOP_CONFORM	Vertical conformance bound (above)
SSG_MAX_CONFORM_Z	Maximum allowable altitude for this segment
SSG_BOT_CONFORM	Vertical conformance bound (below)
SSG_MIN_CONFORM_Z	Minimum allowable altitude for this segment
HORIZONTAL_MONITOR	Flag indicating whether horizontal conformance should be monitored (i.e., if X, Y data “good enough”)
VERTICAL_MONITOR	Flag indicating whether vertical conformance should be monitored (i.e., if altitude data “good enough”)
LONG_IND (Output)	Indicates out of conformance direction for dimension (blank, A (ahead), or B (behind))
LAT_IND (Output)	Indicates out of conformance direction for dimension (blank, L (left), or R (right))
VERT_IND (Output)	Indicates out of conformance direction for dimension (blank, A (above), or B (below))

3.3.8 Function: TKM_CONTROL (PL/I)

This is the PL/I control function for URET. It sets up the database and TKM data structures, and then waits to receive one of the following messages from the mailbox:

1. Track Update or Progress Report message from the HCS
2. Drop Track message from the HCS
3. 12 second update message from the Clock subsystem (CLK)
4. Check category message from RPM
5. Data collection message from CLK
6. Terminate message

TKM_CONTROL also sends messages to other URET subsystems (e.g., RPM and Plan Display Manager (PDM)) as a result of processing these messages.

TKM_CONTROL invokes the following TKM modules and utility functions upon receiving the appropriate message:

Message	TKM Modules
1. Track Update/Progress Report	TKM_FOR_AERA TKM_UPDATE_CURRENT_POSITION DB_FIND_AUD_PTR
2. Drop Track	TKM_MATCH_ID TKM_FREE_HISTORY DB_FIND_AUD_PTR TKM_TRAJECTORY_EXIST TKM_GET_HOLD
3. 12 second update	TKM_FREE_HISTORY DB_FIND_AUD_PTR ST_TIME_SSGDATA TKM_UPDATE_CURRENT_POSITION TKM_TRAJECTORY_EXIST TKM_GET_HOLD TKM_CATEGORY_CHANGE
4. Check Category	TKM_MATCH_ID TKM_TRAJECTORY_EXIST TKM_GET_HOLD TKM_CATEGORY_CHANGE

3.3.9 Function: TKM_CORE (C)

This function is the top-level C control routine for all “major” TKM algorithmic functions. All data is passed to and from TKM_CORE via the TKM_RESULTS data structure. TKM_CORE calls the following TKM functions:

TKM_VERIFY_DATA
TKM_ADD_TRACK_HISTORY
TKM_FIND_ARD_BY_TIME
TKM_POINT_ON_TRAJ
TKM_CHECK_AIRSPACE
TKM_CATEGORY_CHANGE
TKM_CONFORMANCE_MONITOR
TKM_COMPUTE_RECONFORMANCE
TKM_ADD_OOC

3.3.10 Function: TKM_FOR_AERA (PL/I)

This function is called by TKM_CONTROL to process Track Update and Progress Report messages received from the HCS. This is accomplished via calls to the following TKM modules and utility functions:

```
DB_GET_CMT_INDEX
TKM_MATCH_ID
TKM_FREE_HISTORY
TKM_INIT_HISTORY
TKM_GET_TACTICAL_AS
TKM_GET_HOLD
TKM_TRAJECTORY_EXIST
TKM_CORE
CNV_GRND_TO_TAS
GM_REGN
DB_FIND_AUD_PTR
ST_TIME_SSGDATA
ST_NEAREST_FL
```

3.3.11 Function: TKM_GET_RTE_ORIS (PL/I)

This function returns the number of segments in the RTE_ORIS of an aircraft starting with the current offset.

3.3.11.1 Description:

Given the index for the central track store (CTS) data structure, this function finds the corresponding aircraft unique data table (AUD) pointer and the pointer to the RTE_ORIS (route of the aircraft based on the initial filed route string) data structure. Here the function will loop through all of the route segments, keeping a counter, while recording to a global variable array the values for the starting x, y coordinates, the segment course heading, the accumulated distance to the start of the segment, and the type of fix (i.e. 3 character, 5 character, turn fix, lat/long, or not a fix) for every segment of the aircraft's route. The call to this function will actually return the number of segments in the RTE_ORIS for the aircraft. This value can then be used later to loop through the variable array containing all of the above, detailed RTE_ORIS information.

The function never uses the ORIS_OFFSET value which is supplied as an input.

3.3.12 Function: TKM_GM_REGN (C)

In the x-y plane, this function determines if a test point (xt, yt) lies within a polygon region defined by the a set of boundary points (x[n], y[n]).

3.3.12.1 Description:

The function uses the stdlib.h random number generator to create random numbers, and by using the maximum and minimum x and y coordinates of the region, the function creates points outside the polygon region. These outside points are joined with the test point to form a line segment. These test lines are checked for intersections against the segments defining the circumference of the polygon region. The end results is a number of intersections. The function makes n number of random number calls (currently 8). The number of intersection checks is therefore n times the number of polygon vertices. If the number of intersections is even, the test point is outside the region. If the number of intersections is odd, the test point is inside the region.

Table of Variable Definitions

Function Variable	Description	Math Symbol
xmin, xmax	minimum and maximum x coordinates of the polygon region in feet	<i>xmin, xmax</i>
ymin, ymax	minimum and maximum y coordinates of the polygon region in feet	<i>ymin, ymax</i>
rv	random number value	<i>rv</i>
xr, yr	x and y coordinates of random generated point in feet	<i>xr, yr</i>
*x1	x coordinate of the start point of the line (ft)	<i>x₁</i>
y1	y coordinate of the start point of the line (ft)	<i>y₁</i>
x2	x coordinate of the end point of the line (ft)	<i>x₂</i>
y2	y coordinate of the end point of the line (ft)	<i>y₂</i>
xt	x coordinate of the test point (ft)	<i>xt</i>
yt	y coordinate of the test point (ft)	<i>yt</i>
xi	x coordinate of the intersection point (ft)	<i>xi</i>
yi	y coordinate of the intersection point (ft)	<i>yi</i>
p, t	ratio's returned by gm_insec	<i>p, t</i>
pton	point on line status; pton=1 point is on a line segment of the boundary region's polygon, pton=0 point is not on a line of this boundary	<i>pton</i>
istat	intersection status; istat=0 line segments intersect between endpoints	<i>istat</i>
i, k	loop counters	<i>i, k</i>
ptsep	constant used as delta separation allowed to consider a point on a line; currently set at 1 ft.; used for the TKM GM TSTPNT	<i>ptsep</i>
eps	epsilon value used as effective difference of zero; currently = 0.001	ϵ
nrpt	number of random point tries that function iterates	<i>nrpt</i>
n	number of polygon end points	<i>n</i>

3.3.12.2 Mathematics:

The function determines if a test point lies within a polygon region in the x-y plane defined by the array of boundary points (x[n], y[n]).

The first step in the function is to determine the minimum and maximum boundary distances. These extreme points are used to perform a gross check for the test point. If the test point lies outside the extreme points of the polygon, the function returns an outside the region result for the test point. However, if the test point is equal to or inside the extreme points, 8 random numbers are generated. These numbers are used to define random points outside the polygon region by using the extreme points defined earlier.

The function generates the random number by the "rand()" function in an overall loop structure i from 0 to 7. The value of i is MOD by 4. Therefore, the result is expressed in Table 3.3.12-1, where each MOD value is carried out twice.

Iteration	MOD value	Resulting random point is
-----------	-----------	---------------------------

number		generated
0	0	point to the left of the region
1	1	point above the region
2	2	point to the right of the region
3	3	point below the region
4	0	point to the left of the region
5	1	point above the region
6	2	point to the right of the region
7	3	point below the region

Table 3.3.12-1: Iteration key TKM_GM_REGN

For $i = 0$ or 4 , the random point is generated to the left of the polygon region. The expression to determine this point is :

$$x_r = x_{\min} - 1.005(x_{\max} - x_{\min}) \quad \text{Equation 3.3.12-1}$$

$$y_r = y_{\min} + r_v(y_{\max} - y_{\min}) \quad \text{Equation 3.3.12-2}$$

For the x dimension, the point is placed 0.5% to the left of the length of the polygon. For the y dimension, the point is placed a uniform random variable distance within the width of the polygon.

For the other directions (i.e. to the right, below, and above), the calculation is performed analogously to Equation 3.3.12-1 and Equation 3.3.12-2.

After each random point is generated, the function runs a second loop for each segment of the polygon. For each segment, the TKM_GM_TSTPNT is called to determine if the test point is on the segment. If it is, the test point is considered inside the region and the function ends with an inside result. However, if the TKM_GM_TSTPNT determines the point is not on the line the test point is combined with the current random point to form a segment.

Now, GM_INSEC is called to determine if an intersection takes place between the test point to random point segment versus the polygon segment. A counter is incremented ($icnt$) for each intersection found. If an intersection is found, but the ratio of the distance to the end point of the polygon segment is less than ϵ or greater than $1 - \epsilon$, then the loop ends without finishing the rest of the polygon segments and the next random point is generated. The i loop ends after $nrpt$ random points are generated.

The function ends by returning the MOD value of $icnt$ by 2. This value will return a 0 or 1 for the number of intersections determined to be even or odd, respectively. If an odd number of intersections are found, the test point does lie inside the region. If the number of intersections is an even number, the test point lies outside the region. Figure 3.3.12-1 illustrates the logic for this function.

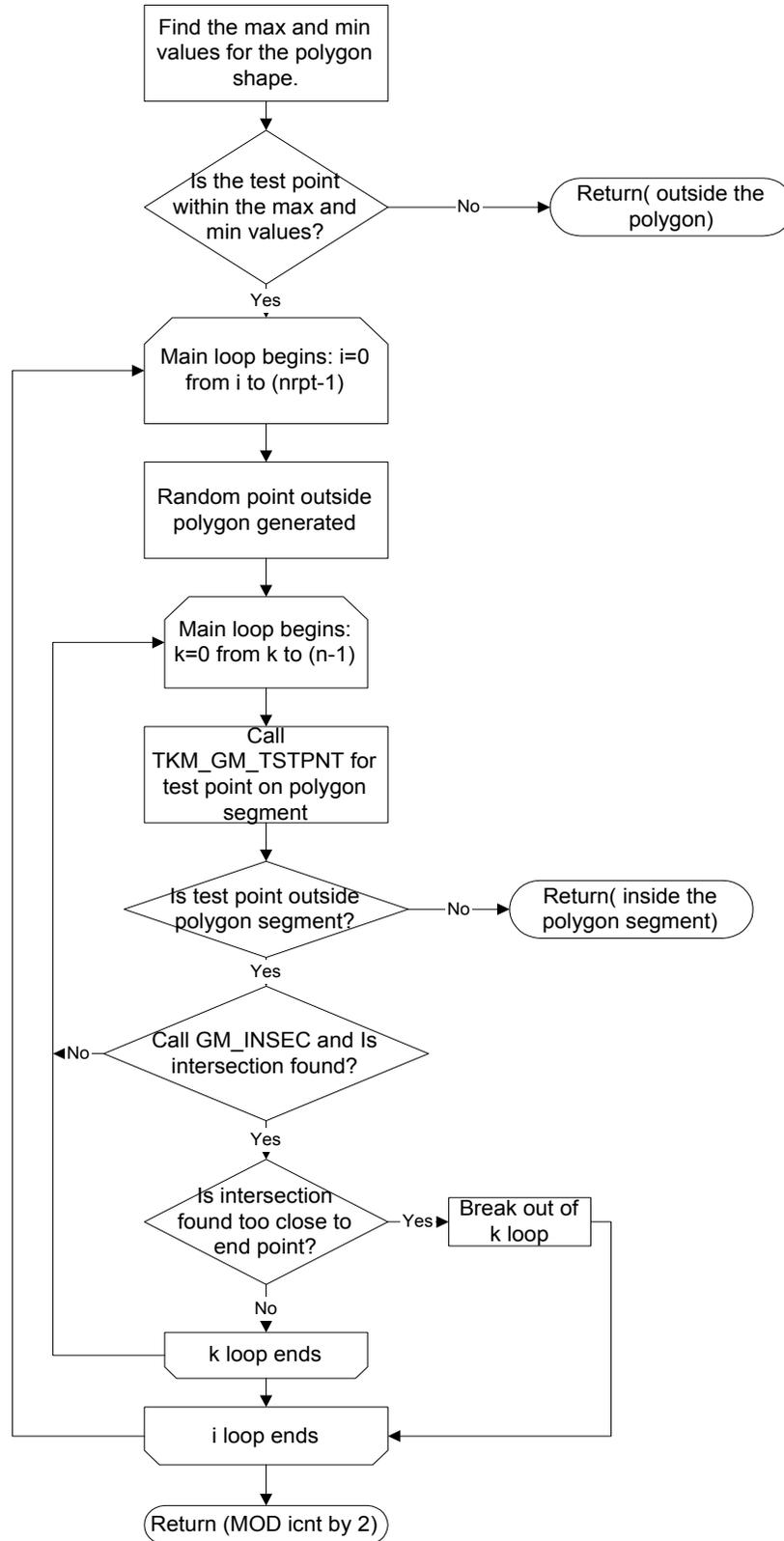


Figure 3.3.12-1: Logic Flow of TKM_GM_REGN

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TKM
R 3.3.12-1	The original PL/I version of this function was written to only generate additional random points if an intersection was found too close to the end point. This function always runs <i>nrpt</i> times <i>n</i> iterations, while the PL/I version runs a maximum of <i>nrpt</i> times <i>n</i> iterations.	This delta between the two versions will not effect the accuracy of the code, since only one random point sufficiently outside the polygon can be utilized to determine if the test point is inside the polygon. If all <i>nrpt</i> random points are generated, the result is the same, however, the code efficiency would be improved if only one were used.	Important
R 3.3.12-2	The ϵ value assumes the value for the ratio returned by GM_INSEC is approximately 1 or 0. The value currently chosen is 0.001 which is very reasonable.	A reasonable choice for the parameter has been chosen and if the intersection point is effectively on the end point of the polygon line segment, another random point is chosen (total of <i>nrpt</i> of them)	Minor
R 3.3.12-3	The choice of <i>nrpt</i> random point iterations seems reasonable, though if more are required the result will be to falsely determine the point is outside the region (return 0).	The number <i>nrpt</i> =8 chosen seems reasonable and can only be verified by unit testing.	Important
R 3.3.12-4	This function is subject to the critical and important approximations of TKM_GM_TSTPNT, since it uses this function to check if the test point lies on each polygon line segment.	Refer to the TKM_GM_TSTPNT function's Assessment Table. They will directly effect this function TKM_GM_REGN.	Critical

3.3.13 Function: TKM_GM_TSTPNT (C)

This function determines if a point lies on a line. To lie on this line, the point may be a small epsilon distance from the line and still be considered on the line.

3.3.13.1 Description:

Given the x and y coordinates (in feet) of the end points of the line and the x and y coordinates of a point (in feet), the function first determines the location of an intersection point which forms a perpendicular line from the given point to the given line (refer to Figure 3.3.13-1). Next, the function determines the distance of this perpendicular line and compares it to the minimum epsilon distance. If the distance of the normal line is less than the minimum distance, the point is considered on the line.

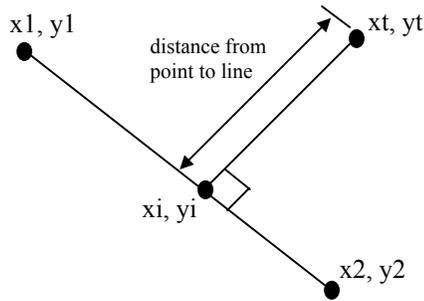


Figure 3.3.13-1: Diagram of test point to line distance

Table of Variable Definitions

Function Variable	Description	Math Symbol
x1, y1	x, y coordinates of the start point of the line (ft)	x_1, y_1
x2, y2	x, y coordinates of the end point of the line (ft)	x_2, y_2
xt, yt	x, y coordinates of the test point (ft)	xt, yt
pntsep	separation allowed between 2 points	$pntsep$
delx	delta difference of the line in x dimension (ft)	$\delta.x$
dely	delta difference of the line in y dimension (ft)	$\delta.y$
s1	slope of the line from x_1, y_1 to x_2, y_2	m
s2	slope of the normal line from xt, yt to xi, yi	$-1/m$
xi, yi	x, y coordinates (ft) of intersection point of test point to line	xi, yi
eps	epsilon value for considering the line to vertical or horizontal (currently set at 100 feet)	ϵ
d	perpendicular distance (ft) from the test point to the line	d

3.3.13.2 Mathematics:

The function starts by calculating the delta differences of each dimension of the line. These variables include the following:

$$\delta.x = x_2 - x_1 \quad \text{Equation 3.3.13-1}$$

$$\delta.y = y_2 - y_1 \quad \text{Equation 3.3.13-2}$$

These deltas are used to determine if the line is a vertical line or horizontal line. For the x dimension, if the line's $\delta.x$ is less than an ϵ value, consider the line to be a vertical line. The function checks if the test point is greater than the distance $pntsep$ in the x dimension and if so considers it not on the line. If the test point is less than the distance $pntsep$ in the x dimension and is within the y dimensions of the line, it is considered on the line. An analogous check is made for the y dimension.

Now, the line is not a vertical or horizontal line and the perpendicular distance will need to be calculated between the test point and the line. The first step is to determine the following slope equations:

$$s1 = m = \frac{\delta.y}{\delta.x} \quad \text{Equation 3.3.13-3}$$

$$s2 = -\frac{1}{m} = -\frac{\delta.x}{\delta.y} \quad \text{Equation 3.3.13-4}$$

The equation of the line is expressed for the given line and the line formed by drawing a perpendicular line from the test point to the line. By solving these two equations simultaneously for x and y, the resulting formulas give the x and y coordinates of the intersection point used in the function to solve for the distance d .

The given line:

$$y - y_1 = m(x - x_1) \quad \text{Equation 3.3.13-5}$$

The normal line from the test point to the line:

$$y - y_t = -\frac{1}{m}(x - x_t) \quad \text{Equation 3.3.13-6}$$

Solving them simultaneously for x (note the x below is equivalent to xi in the code):

$$\begin{aligned} m(x - x_1) + y_1 - y_t &= \left(-\frac{1}{m}\right)(x - x_t) \\ mx - mx_1 + \frac{x}{m} - \frac{x_t}{m} &= y_t - y_1 \\ x\left(m + \frac{1}{m}\right) &= y_t - y_1 + \frac{x_t}{m} + x_1m \\ x &= \frac{y_t - y_1 + \frac{x_t}{m} + x_1m}{\left(m + \frac{1}{m}\right)} \end{aligned} \quad \text{Equation 3.3.13-7}$$

Now, solve for y (or yi in the code) for the intersection point using the x value in Equation 3.3.13-7 and use Equation 3.3.13-5 to solve for y.

The last check determines the distance of the intersection point using the general distance formula:

$$d^2 = \sqrt{(xt-x)^2 + (yt-y)^2} \quad \text{Equation 3.3.13-8}$$

The function proceeds by checking this distance d against the $ptsep$ distance, and if this distance is greater than the $ptsep$ distance the test point is evaluated as not on the given line. However, if the distance is less than $ptsep$, the test point is checked to determine if it is between the end points of the line. For example, this checks for cases when the distance in Equation 3.3.13-8 is zero because the test point is collinear with the given line, but not within the line segment. It could actually be a large distance from the endpoints of the line. (NOTE: To determine if the test point is within the line segment, the function extends the line by the ϵ value, currently 100 feet.)

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TKM
R 3.3.13-1	If the test point is less than a distance $ptsep$ from the given line, the point is evaluated to be between the end points of the given line segment. However, the line is extended by ϵ for the TKM version of GM_TSTPNT in C, but for the PL/I version the line segment is extended only by $ptsep$. The $ptsep$ value is 1 foot and the ϵ value is 100 feet.	The transfer from C to PL/I will provide different results not because of coding in a different language, but because different comparison values are used. An investigation in the potential reasons for the change are necessary.	Important
R 3.3.13-2	For each check (includes three in this function), the “else return(false);” should be added to protect against an undetermined return from the function. For example, the last case where the distance equation returned a value of zero because the test point is collinear with the line, but not within the endpoints of the segment. The result will end the function without specifically assigning the value FALSE.	The specific compiler by default may or may not assign a zero value (which will return the correct value) or the return value may be reinitialized before the call to this function, but this is not sufficient for portable ANSI C code. (NOTE: The original PL/I version was written differently to protect under this case.)	Critical
R 3.3.13-3	As a result of the unprotected return in the function for the horizontal line case, a horizontal line checked against a point outside the endpoints of the line segment but on the line will return a division by zero (s1 = 0.0 while s2 will be in error...). The corresponding problem is present for the vertical case as well.	The original PL/I version had goto statements to protect under this case. This is not necessary, but a simple “else statement” with a return of false would protect against the problem.	Critical
R 3.3.13-4	The check carried out to determine if a point is between the end points of the line segment when the line segment is either vertical or horizontal uses the $ptsep$ value to extend the lines under the PL/I version and not for the C version here.	It is actually more accurate not to use the $ptsep$ value, but this may cause errors due to round off during floating point arithmetic. Therefore, an investigation is required to determine why this was not used in this function.	Important

3.3.14 Function: TKM_MATCH_ID (PL/I)

Attempts to associate track data with flight plan data in the URET database.

3.3.14.1 Description:

Upon receipt of a track update message or progress report message from the HCS, this function attempts to associate the track data with flight plan data in the URET database. This is accomplished by calls to utility functions LO_FINDAC and DB_FIND_AUD_PTR. If a match is found, the output variables identified will be returned and further TKM processing can continue.

Table of Variable Definitions

Function Variable	Description
ACID (Input)	Aircraft identification
CID (Input)	Computer identification
ORIGIN (Input)	Origin airport of this flight
CTS_IX (Output)	CTS index
AUD_IX (Output)	AUD index
AMC_PTR (Output)	Pointer to AMC structure
FLP_PTR (Output)	Pointer to flight plan structure
THE_FIRST_FLP_PTR (Output)	Pointer to the first flight plan

3.3.15 Function: TKM_TK_HDG (C)

This function determines the course heading of a set of track position reports.

3.3.15.1 Description:

Given the number of needed track position reports (or all of the points from the newest given index to the oldest given index, whichever is less), this function will determine the heading of this series of points. If there is only one point, the function determines the heading by taking the \tan^{-1} of the ratio of the given track velocities at this newest point. If there is more than one track report available, the function determines the least square parameters for the linear first order regression model. The line defined by these parameters is the linear relationship of the dependent variable (the x or y coordinate) to the independent variable (time). The slope (the second parameter) is actually the velocity component of the x and y, and is later used to determine the heading using the \tan^{-1} function as above. Since the track reports are extracted in reverse order, π is added to the final heading result (only if multiple track reports are available) to reverse the direction of the heading vector to correspond to the aircraft movement.

Table of Variable Definitions

Function Variable	Description	Math Symbol
tkm_track[i].x, tkm_track[i].y	Coordinates of the aircraft at the given track index i (ft)	x_i, y_i
number_reports_to_use	The parameter number of TK reports to use to determine the track heading	m
index_newest_tk, index_oldest_tk	Indices of the newest (most current) and oldest (first) report of the aircraft's track data	$0, k$
temp	The course heading of the aircraft with respect to true North (radians)	Ψ

3.3.15.2 Mathematics:

The function performs the following simple calculations and logic:

First the function extracts the track data (x , y , and t) in reverse order starting with the newest, most recent, report and ending with oldest report or until the given parameter number of reports (m) has been found. These values are stored in an array for each variable (x_i , y_i , and t_i); however t_i is converted from a clock value to the time increment starting from newest point. The increment is calculated by $(t_0 - t_i)$ to reflect the reverse order sequence.

Next the function supplies the LEASTSQ function with the number of track reports to use (m) the independent variable array (t) and either dependent variable array (x or y). The function returns the least square parameters for a linear first order regression model, in terms of arrays, α_0 and α_1 or β_0 and β_1 , where α_1 and β_1 actually represent the slopes of the regression lines.

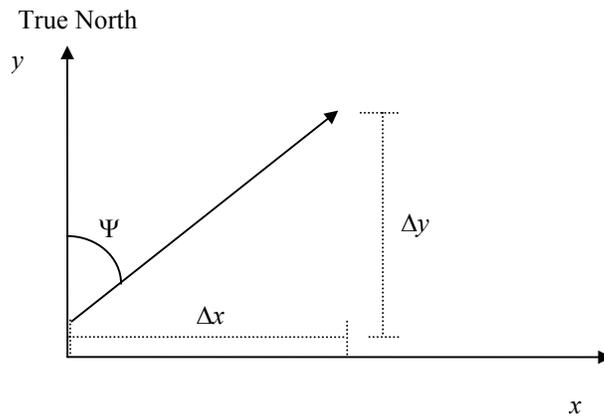
$$\alpha_1 = \frac{\Delta x}{\Delta t} \quad \text{Equation 3.3.15-1}$$

$$\beta_1 = \frac{\Delta y}{\Delta t} \quad \text{Equation 3.3.15-2}$$

The function uses these slopes, which are the x , y component velocities, to calculate the heading angle using the following equation.

$$\Psi = \pi + \tan^{-1}\left(\frac{\alpha_1}{\beta_1}\right) \quad \text{Equation 3.3.15-3}$$

Where Ψ is the course heading of the aircraft with respect to true north. π is added to the resultant angle to correct for the reverse order of the time increment (i.e. to reverse the vector heading).



The function also checks:

If $k=0$ (newest index = oldest index) then there is only one available track point. The function will then use the corresponding x and y velocity components (as long as they are not zero) given with the track report to calculate the heading.

$$\begin{aligned} \text{tkm_track}[0].x_velocity &= V_x \\ \text{tkm_track}[0].y_velocity &= V_y \end{aligned}$$

$$\Psi = \tan^{-1}\left(\frac{V_y}{V_x}\right) \quad \text{Equation 3.3.15-4}$$

The above method (i.e. Equation 3.3.15-4) is also used if $\alpha_1 = \beta_1 = 0$ and there is more than one track report and the least square function has been called.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TKM
R 3.3.15-1	The function does not protect against a β_1 value equal to zero in the denominator of the inverse tangent function, Equation 3.3.15-3.	This coding error could cause a floating point error while processing	Important
R 3.3.15-2	All headings are with respect to true North.	Any advisories or future resolutions to the aircraft would need to be converted back to magnetic North	Minor

3.3.16 Function: TKM_VERIFY_DATA (C)

This function verifies the quality and reasonableness of track data received from the HCS, based on track history. The validity of the data determines the degree it can be used for further processing by TKM. The horizontal distance, and the vertical gradient, between the current and previous track positions are computed. The horizontal distance is calculated using the standard distance formula, and is compared with a parameter representing the distance that this aircraft could reasonably travel in one radar scan (currently set to 100 nm). If the distance is within this parameter, a flag is set to indicate that horizontal conformance should be monitored by TKM_CONFORMANCE_MONITOR. The vertical gradient is calculated in TKM_CHECK_POSITIVE_Z, and is compared with a parameter representing the change in altitude this aircraft could achieve in one radar scan (currently set to 1 ft/ft). If the gradient is within this parameter, another flag is set to indicate that vertical conformance should be monitored by TKM_CONFORMANCE_MONITOR.

3.3.17 Function: UTL_XY_ARD_BY_RTE (PL/I)

This function determines the closest distance from a given point to the original route in the horizontal plane.

3.3.17.1 Description:

This function determines the closest original route segment (RTE_ORIS) to the given track (x, y) point. The closest distance calculation is determined by calling the GM_PTSEG for every segment in the RTE_ORIS data structure. The minimum distance is returned from the function in the variable MIN_D.

NOTE: The comments and name of this function are misleading. The function does not calculate the ARD at the x, y position. It only computes the minimum distance from the given point to any point along the original route (RTE_ORIS).

3.4 General Purpose Utilities

The developer's library of functions are contained in two general purpose utility directories. The assessment tables at the end of the following sections address the impact of the identified

assumptions/approximations on the indicated algorithm set (i.e., TJM, TKM or APD) although the utility function itself may be called by more than the specified function.

3.4.1 Function: CNV_CNVSPD (PL/I)

Converts airspeed from one form to another.

3.4.1.1 Description:

This function converts a given airspeed to either true airspeed, indicated airspeed, or Mach. The necessary inputs supplied to this function are:

Input Speed	ISPD
Conversion Code	CODE
Coordinates at Conversion	XX, YY, ZZ

If there is atmospheric data available, the function retrieves the temperature and pressure from the DB_AIR_AT_POINT function (See Section 3.4.10).

If there is no atmospheric data available, the function uses the CNV_STD_ATMOS function (See Section 3.4.7) to calculate the standard temperature and pressure at that altitude.

The function then converts the CODE number to correspond to an input airspeed type (i.e. TAS, IAS or Mach) and supplies this to the CNV_SPEED function (See Section 3.4.6) along with the input speed, altitude, temperature and pressure. CNV_SPEED then returns the desired output speed (TAS, IAS or Mach).

3.4.2 Function: CNV_GNOMONIC_STEREO (PL/I)

Converts gnomonic X,Y coordinates to stereographic X,Y coordinates. X and Y are in nautical miles.

3.4.2.1 Description:

The descriptive information contained here has been obtained from the computer code for this PL/I procedure, from the *URET Trajectory Modeling Algorithmic Definition* document (MTR 96W0000072) and from the reference stated in the text.

3.4.2.1.1 Stereographic Coordinates

The HCS uses the stereographic coordinate system to locate the aircraft. It is a Cartesian, planar coordinate system. The points on the surface of the earth are projected onto a plane which is tangent to the earth at a point within the ARTCC airspace. This representation of points on a plane when they are really on an ellipsoid introduces distortion or errors in position. However, for points close to the point of tangency, the distortion is small and can be ignored.

The latitude and longitude of a point on the earth's surface are based on a model of the earth as an ellipsoid of revolution. Points on the surface of the ellipsoid are projected onto the surface of a sphere (the conformal sphere) having the same center as the ellipsoid. Then the points on the sphere are projected onto the tangent plane using as a focal point the point on the sphere which is directly opposite the point of tangency (the antipode). This projection is illustrated in Figure 3.4.2-1. The projection is also described and illustrated in the description of the function CNV_LLXY. The location of a point thus placed on the tangent plane is specified by its stereographic coordinates.

3.4.2.1.2 Gnomonic Coordinates

The gnomonic projection is similar to the stereographic projection. Points on the sphere are projected onto the same tangent plane. For a gnomonic projection, the focal point is the center of the sphere. The

location of a point projected onto the sphere in this way is specified by its gnomonic coordinates. The gnomonic projection is also illustrated in Figure 3.4.2-1.

3.4.2.1.3 Coordinate Conversion

Conceptually, this function, referring to Figure 3.4.2-1, takes a gnomonic point C on the tangent plane, reverse projects it to the point B on the conformal sphere, and then projects it (the point B) back onto the tangent plane as a stereographic point A. Given the gnomonic coordinates of the point C, the function calculates the stereographic coordinates of the point A.

3.4.2.1.4 Use of the Gnomonic Projection

A gnomonic projection of the surface of a sphere onto a plane introduces more distortion than a stereographic projection. However, great circle routes on the sphere are projected as straight lines on the tangent plane. This characteristic is used in the following way.

Let V_S and Z_S be the starting and ending points of a route. When an aircraft flies from V_S to Z_S , it flies the shortest distance from V_S to Z_S which is the great circle route. V_S and Z_S are defined by their stereographic coordinates. Great circle routes are approximated on the stereographic plane by a series of straight line segments. The stereographic coordinates of the ends of these line segments are found by first converting the stereographic coordinates of V_S and Z_S to gnomonic coordinates V_G and Z_G . The great circle route from V_S to Z_S in gnomonic coordinates is the straight line $V_G Z_G$. This straight line from V_G to Z_G is easily divided into a series of shorter line segments - $V_G W_G$, $W_G X_G$, $X_G Y_G$, and $Y_G Z_G$. The gnomonic coordinates for W_G , X_G , Y_G , and Z_G are then converted to their corresponding stereographic coordinates. In this way the great circle route from V_S to Z_S is approximated in the stereographic plane by the straight line segments $V_S W_S$, $W_S X_S$, $X_S Y_S$, and $Y_S Z_S$.

3.4.2.1.5 Note

The function being described in this section, `CNV_GNOMONIC_STEREO`, converts gnomonic coordinates to stereographic coordinates. A companion function, `CNV_STEREO_GNOMONIC`, converts stereographic coordinates to gnomonic coordinates.

3.4.2.1.6 Conversion Equations

The equations, listed later on in the section, are given in NAS-MD-312, Appendix C, page C-2.

The conversion functions in Equation 3.4.2-1 and Equation 3.4.2-2 convert the gnomonic X and Y coordinates to stereographic X and Y coordinates relative to the point of tangency. It is necessary to add the values of the stereographic coordinates of the point of tangency to these values to get the values of X and Y relative to the origin of the stereographic coordinate system.

3.4.2.1.7 Constants

One constant is necessary for the computation. It is the conformal radius of the earth.

3.4.2.1.8 Units

The gnomonic coordinates, the stereographic coordinates, and the radius of the earth are all in nautical miles.

Table of Variable Definitions

Function Variable	Description	Math Symbol
GX, GY	Input variables - Gnomonic X, Y coordinates of the point being converted	X_G, Y_G
ACP.XTANG, ACP.YTANG	Input parameter - Stereographic X, Y coordinates of the point of tangency	X_t, Y_t
ACP.RAD_EARTH	Input parameter - Conformal radius of the earth	R
SX, SY	Output variable - Stereographic X, Y coordinates of the point being converted	X_S, Y_S

3.4.2.2 Mathematics:

The following equations are used in the function CNV_GNOMONIC_STEREO to calculate the stereographic coordinates of a point, given its gnomonic X and Y coordinates.

$$X_S = \frac{2X_G}{1 + \sqrt{1 + \frac{X_G^2 + Y_G^2}{R^2}}} + X_t \quad \text{Equation 3.4.2-1}$$

$$Y_S = \frac{2Y_G}{1 + \sqrt{1 + \frac{X_G^2 + Y_G^2}{R^2}}} + Y_t \quad \text{Equation 3.4.2-2}$$

The function correctly calculates the stereographic coordinate values given the gnomonic coordinate values.

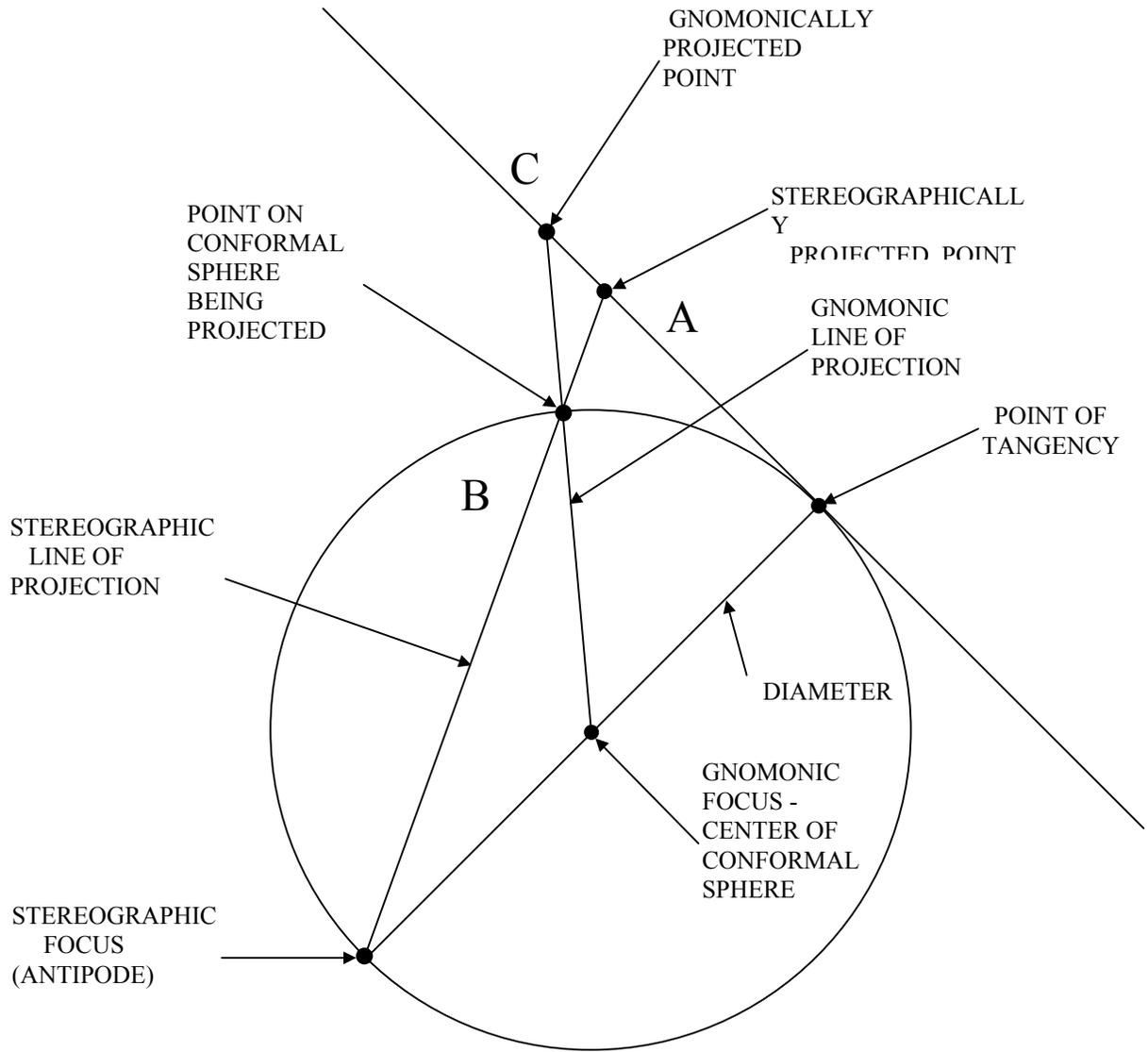


Figure 3.4.2-1: Gnomonic to Stereographic Projection

3.4.3 Function: CNV_GRDSPD (PL/I)

Computes the aircraft's ground speed based on the true airspeed and the wind speed.

3.4.3.1 Description:

This function computes the aircraft's ground speed based on the true airspeed and the wind speed using the vector representation. The two vectors, true airspeed and wind speed, determine the resultant ground speed vector. The effects of wind in both the cross-track and along-track dimensions (cross winds and head/tail winds) are considered and a special allowance is made when the cross-track component of wind is greater than true airspeed.

Table of Variable Definitions

Function Variable	Description	Math Symbol
WIND_SPEED	Scalar value for wind speed	V_w
WIND_BEARING	Direction of wind with respect to North	θ_w
TAS	True airspeed	V_t
GROUND_SPEED	Ground speed	V_g
TRACK_BEARING	The flight path bearing.	ψ

3.4.3.2 Mathematics:

This function defines the ground speed as:

$$V_g = \sqrt{V_t^2 - [V_w \sin(\Psi - \theta_w)]^2} + V_w \cos(\Psi - \theta_w) \quad \text{Equation 3.4.3-1}$$

with the exception when

$$V_t^2 - [V_w \sin(\Psi - \theta_w)]^2 < 0$$

which indicates that the cross wind speed component is greater than the true airspeed. In this case, ground speed is approximated with the contribution of the along track wind component only

$$V_g = V_w \cos(\Psi - \theta_w) \quad \text{Equation 3.4.3-2}$$

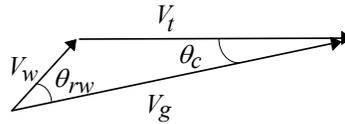
If we make the following assignment

$$\theta_{rw} = \psi - \theta_w \quad \text{Equation 3.4.3-3}$$

where θ_{rw} is the relative angle between the course heading of the aircraft and the wind direction, Equation 3.4.3-1 can be reoriented as

$$V_g = V_t \sqrt{1 - \left(\frac{V_w}{V_t} \sin \theta_{rw}\right)^2} + V_w \cos \theta_{rw} \quad \text{Equation 3.4.3-4}$$

Equation 3.4.3-4 is based on the vector representation of ground speed



and

$$V_g \approx V_t \cos \theta_c + V_w \cos \theta_{rw} \quad \text{Equation 3.4.3-5}$$

where θ_c is the “crab” angle between the V_t -leg and V_g -leg.

The relationship

$$\cos \theta_c = \sqrt{1 - \sin^2 \theta_c} = \sqrt{1 - \left(\frac{V_w}{V_t} \sin \theta_{rw} \right)^2}$$

holds true because of the law of sines $\left(\frac{\sin \theta_{rw}}{V_t} = \frac{\sin \theta_c}{V_w} \right)$ and the basic trigonometric identity.

Equation 3.4.3-4 and Equation 3.4.3-5 are approximations because they assume a small flight path angle (i.e. negligible effect from motion in the vertical plane).

Therefore Equation 3.4.3-1 and Equation 3.4.3-2 appear to be reasonable and coincides with most trajectory estimation practices that assume a small flight path angle (rate of descent) and a horizontal wind direction.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.3-1	Small path angle	Reasonable, for Commercial Aircraft, and a small flight path angle (i.e. small rate of descent)	Critical
R 3.4.3-2	Large cross wind approximation Equation 3.4.3-2	Wind speeds that large are unlikely and make standard calculations very difficult. Some tests on this effect could be made.	Important
R 3.4.3-3	Derived ground speed	Reasonable for the given vector representation and relative wind and crab angle definitions. Assumes a small flight path angle and a horizontal wind.	Minor

3.4.4 Function: CNV_LLXY (PL/I)

Converts horizontal position coordinates given as a geodetic latitude and a geodetic longitude to stereographic X and Y coordinates. Latitude and longitude are radians. X and Y are in nautical miles.

3.4.4.1 Description:

The descriptive information contained here has been obtained from the computer code for this PL/I procedure, from the *URET Trajectory Modeling Algorithmic Definition* document (MTR 96W0000072), and from the references stated in the text.

3.4.4.1.1 Geodetic Coordinates

Aircraft are located on the surface of the earth by a latitude and a longitude. The earth is almost a sphere, but not quite. It is modeled by an ellipsoid of revolution. This ellipsoid is created by rotating an ellipse about its minor axis. The axis becomes the earth's polar axis - the line connecting the north pole to the south pole. The International Ellipsoid of Reference is used. The distance in this model from the center of the earth to the north (or to the south) pole is 3432.4579 nautical miles. The distance from the center to the equator is 3444.0540 nautical miles. Latitudes and longitudes referred to this ellipsoid model are called the geodetic latitudes geodetic longitudes.

3.4.4.1.2 Stereographic Coordinates

The HCS uses a different coordinate system to locate the aircraft. It is called the stereographic coordinate system. It is a Cartesian, planar coordinate system. The points on the surface of the earth are projected onto a plane which is tangent to the earth at a point within the ARTCC airspace. This representation of points on a plane when they are really on an ellipsoid introduces distortion or errors in position. However, for points close to the point of tangency, the distortion is small and can be ignored.

The stereographic coordinate system is an XY plane with the XY grid lined up approximately with the lines of latitude and longitude for points near the point of tangency. The parallel of constant latitude passing through the point of tangency is projected onto the stereographic plane as a line of constant Y value. The meridian of constant longitude passing through the point of tangency is projected onto the stereographic plane as a line of constant X value. At other points on the stereographic plane, the projections of lines of constant latitude or of constant longitude are curved and do not exactly line up with the lines of constant Y or X.

In the northern hemisphere, the line of constant X value on the stereographic plane passing through the point of tangency points to true north. As the Y coordinate of a point increases in value, the point gets closer to the north pole. The projection of the north pole is on this line through the point of tangency. Similarly the line of constant Y value passing through the point of tangency points due east.

The origin of the stereographic coordinate system is usually not the point of tangency but, in the continental US, is a point in the southwest corner of the ARTCC airspace.

The point of tangency of the stereographic plane is defined by a geodetic latitude, a geodetic longitude, and a distance from the center of the ellipsoid. The location of the point of tangency is determined by the locations of the ARTCC's surveillance radars and is chosen to minimize position errors caused by the projection of an ellipsoid onto a plane.

3.4.4.1.3 Conformal Coordinates

It is convenient to do the coordinate conversion in two steps. A point is first converted to an intermediate coordinate system, and then converted from this intermediate system to the stereographic system. The intermediate coordinate system is a sphere whose center is the same as the center of the ellipsoid and whose radius is based on the locations of the ARTCC's surveillance radars. The angles between lines on the ellipsoid remain unchanged when they are projected onto the sphere. Therefore the coordinate transformation is a conformal transformation and the values of the latitudes and longitudes in this

intermediate coordinate system are referred to as conformal latitudes and longitudes. The radius of the sphere is called the conformal radius of the earth. The geodetic latitude and longitude are first converted to a conformal latitude and longitude, and then the conformal latitude and longitude are converted to the stereographic X and Y coordinates. The conformal longitude is the same as the geodetic longitude; only the latitude is changed upon converting from geodetic to conformal coordinates.

3.4.4.1.4 Geodetic to Conformal Projection

The ellipsoid and the conformal sphere are concentric - that is, they have a common center. The radius of the sphere determines the scale of the projection in the stereographic plane and is chosen to minimize the errors introduced by the projection. The geodetic latitude of a point on the ellipsoid is the elevation above the equatorial plane of a line perpendicular to a plane tangent to the ellipsoid at that point.

The geometry of the conformal spherical projection is shown in Figure 3.4.4-1. The eccentricity of the earth is greatly exaggerated in the figure to show the geometry more clearly. A point P_g on the surface of the ellipsoid has a geodetic latitude of ϕ_g . This is the angle that a normal to the tangent at the point P_g makes with the equator. The conformal projection of the point P_g onto the sphere is the point P . The conformal latitude of the point P is ϕ .

A conformal projection is one in which the change in scale at a given point is the same in all directions. The equation for the projection is derived by making the change in scale in mapping from the ellipsoid to the sphere along a meridian of longitude on the sphere equal to the change in scale along a parallel of latitude on the sphere. Since the lines of longitude are orthogonal to the lines of latitude, making the scale equal in the directions of constant latitude and constant longitude makes the scale equal in all directions.

The conformal value of a latitude when converted depends only on its original geodetic value and on the eccentricity of the ellipsoid.

3.4.4.1.5 Conformal to Stereographic Projection

A point on the conformal sphere is projected onto the stereographic plane in the following way. See Figure 3.4.4-2. The focal point for the projection is the point on the sphere which is directly opposite the point of tangency (the antipode). A line is drawn from the focal point to the point on the sphere being projected. This line is then extended to intersect the stereographic plane which is tangent to the sphere at the point of tangency. The point of intersection of the line with the tangent plane is the projection of the conformal point onto the stereographic plane.

3.4.4.1.6 Conversion Equations

The equations described here are listed later in Section 3.4.4.2. The conversion of the geodetic latitude to a conformal latitude is done using a two term power series approximation (Equation 3.4.4-2) to the conversion equation. This equation is given in NAS-MD-312, Appendix D, Section 2. The original equation being approximated is given in NAS-MD-320, Appendix A, Section 3.2. Both the sine of the conformal latitude of the point being converted and the sine of the conformal latitude of the point of tangency are calculated using the power series equation. The conversion of the conformal latitude and longitude of a point to its XY stereographic coordinates is done using the two equations found also in NAS-MD-312, Appendix D, Section 2 (Equation 3.4.4-6 and Equation 3.4.4-7 in this document) and in NAS-MD-320, Appendix A, Section 3.1. Conversion functions give the X and Y coordinates relative to the point of tangency. It is necessary to add to these values the coordinates of the point of tangency in the stereographic plane to get the values of X and Y relative to the origin of the stereographic coordinate system.

3.4.4.1.7 Constants

Three constants are necessary for the computation. They are the conformal radius of the earth, and the coefficients of the two terms of the power series equation.

3.4.4.1.8 Units

The latitudes and longitudes are angles and are measured in radians. The conformal radius of the earth is measured in nautical miles and therefore the X and Y coordinate values are in nautical miles also.

3.4.4.1.9 Internal Error Checking

The absolute values for the sines of the latitudes are checked to make sure that they are less than or equal to 1. Imprecision in the computation may cause a value to exceed 1. When this occurs the value is reset to 1.

3.4.4.1.10 Unit Testing

A limited amount of unit testing was performed on this function. It performed correctly for all of the cases tested.

Table of Variable Definitions

Function Variable	Description	Math Symbol
LATR, LONGR	Input data - Geodetic latitude and longitude of the point being converted (radians)	ϕ_g, λ
ACP.COORDS.LATIT	Input parameter - Geodetic latitude of the point of tangency	ϕ_{0g}
LONGITT	Input parameter - Geodetic (& conformal) longitude of the point of tangency	λ_0
XT, YT	Input parameter - Stereographic X, Y coordinates of the point of tangency	X_t, Y_t
RE	Input parameter - Conformal radius of the earth	R
0.9932773	Constant - the first order coefficient in the power series expression for the conformal latitude in terms of the geodetic latitude	A
0.0066625	Constant - the third order coefficient in the power series expression for the conformal latitude in terms of the geodetic latitude	B
N/A	Conformal latitude of the point being converted	ϕ
SIN_PHI	Sine of the conformal latitude of the point being converted	$\sin \phi$
N/A	Conformal latitude of the point of tangency	ϕ_0
SIN_PHI0	Sine of the conformal latitude of the point of tangency	$\sin \phi_0$
COS_PHI	Cosine of the conformal latitude of the point being converted	$\cos \phi$
COS_PHI0	Cosine of the conformal latitude of the point of tangency	$\cos \phi_0$
DLONG	The difference in longitude (both geodetic and conformal) between the point being converted and the point of tangency	$\Delta\lambda$
X, Y	Output data - Stereographic X, Y coordinates of the point being converted	X, Y

3.4.4.2 Mathematics:

The following equations are used in this function to calculate the stereographic X and Y coordinates of a point, given its geodetic latitude and longitude. First calculate the difference in the longitudes.

$$\Delta\lambda = \lambda_0 - \lambda \quad \text{Equation 3.4.4-1}$$

Next, convert the geodetic latitudes of the point being converted and the point of tangency to their corresponding conformal latitudes. Actually the conformal latitudes are not explicitly calculated. Their sines and cosines are calculated.

$$\sin \phi = A \sin \phi_g + B \sin^3 \phi_g \quad \text{Equation 3.4.4-2}$$

$$\sin \phi_0 = A \sin \phi_{0g} + B \sin^3 \phi_{0g} \quad \text{Equation 3.4.4-3}$$

$$\cos \phi = \sqrt{1 - \sin^2 \phi} \quad \text{Equation 3.4.4-4}$$

$$\cos \phi_0 = \sqrt{1 - \sin^2 \phi_0} \quad \text{Equation 3.4.4-5}$$

The next two equations convert the conformal latitude ϕ and conformal longitude λ to the stereographic coordinates X and Y . The first term in each equation is the value of the coordinate relative to the point of tangency. Since the origin of the stereographic coordinate system is not at the point of tangency, it is necessary to add to this term the coordinate of the point of tangency in the stereographic coordinate system.

$$X = 2R \left(\frac{\sin \Delta\lambda \cos \phi}{1 + \sin \phi \sin \phi_0 + \cos \phi \cos \phi_0 \cos \Delta\lambda} \right) + X_t \quad \text{Equation 3.4.4-6}$$

$$Y = 2R \left(\frac{\sin \phi \cos \phi_0 - \cos \phi \sin \phi_0 \cos \Delta\lambda}{1 + \sin \phi \sin \phi_0 + \cos \phi \cos \phi_0 \cos \Delta\lambda} \right) + Y_t \quad \text{Equation 3.4.4-7}$$

The power series used in Equation 3.4.4-2 and in Equation 3.4.4-3 is a satisfactory approximation. The function correctly calculates the values of x and y for a given geodetic latitude/longitude coordinate pair.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.4-1	The point being converted is sufficiently near the point of tangency for the algorithm to work.	The points that can be stereographically projected from a sphere are limited to the hemisphere centered on the point of tangency. The point being converted must be within 90 degrees of the point of tangency. For robustness, the function should do this bounds check before proceeding with the calculation of X and Y.	Minor
R 3.4.4-2	$\cos \phi_{0g}$ and $\cos \phi_g$ are calculated	The function unnecessarily calculates the cosine of the geodetic latitude of the point of tangency and the cosine of the geodetic latitude of the point being converted. This code should be deleted.	Minor
R 3.4.4-3	$\cos \phi$ and $\cos \phi_0$ are compared to 0	The bounds check on the cosine function is unnecessary because the bounds check has already been run on the sine calculation.	Minor
R 3.4.4-4	The conformal latitude of the point of tangency is calculated every time the function is called.	This calculation should be done once (for a given ARTCC) and the result saved for future use.	Minor

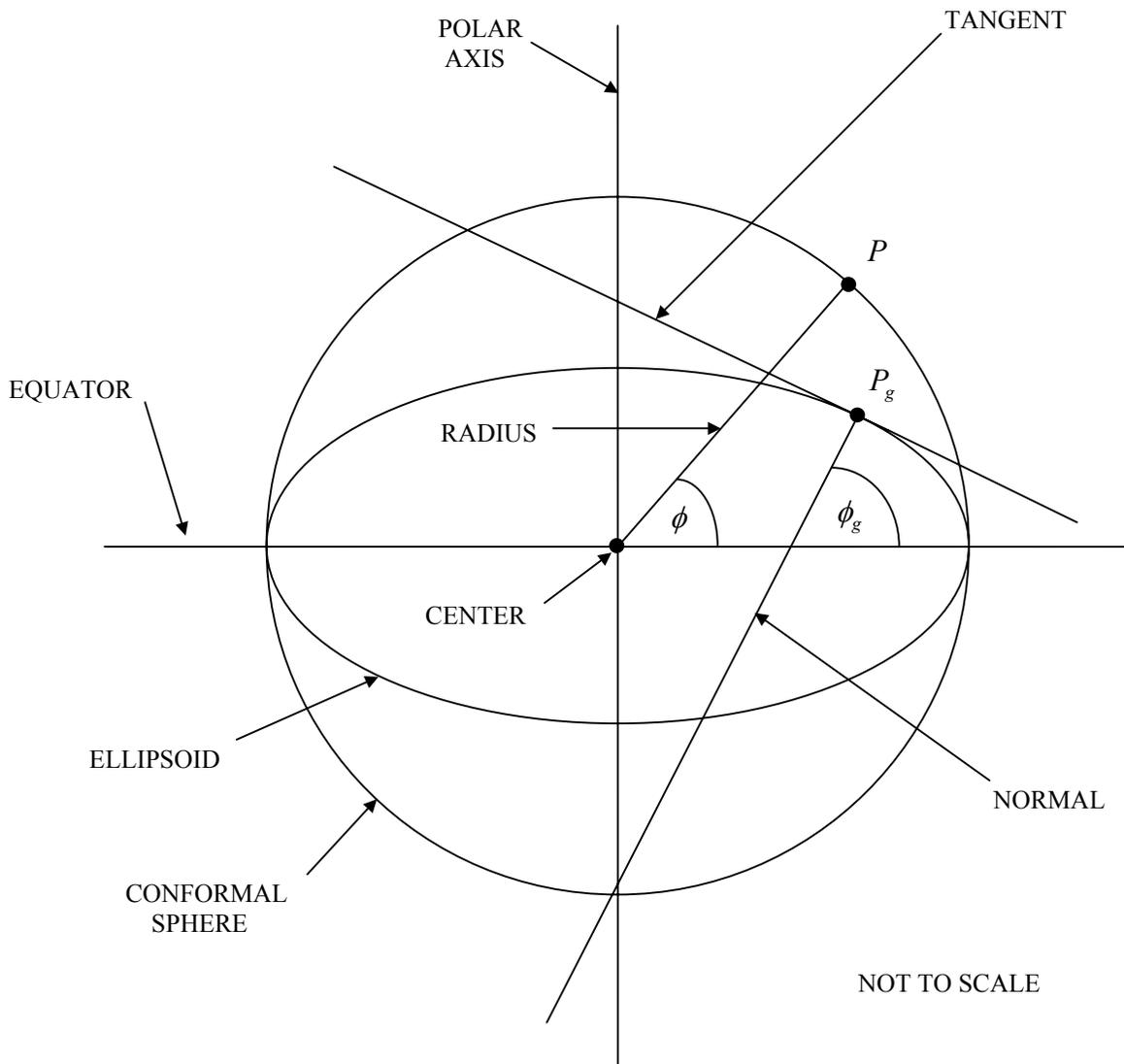


Figure 3.4.4-1: Mapping Geometry - Geodetic to Conformal

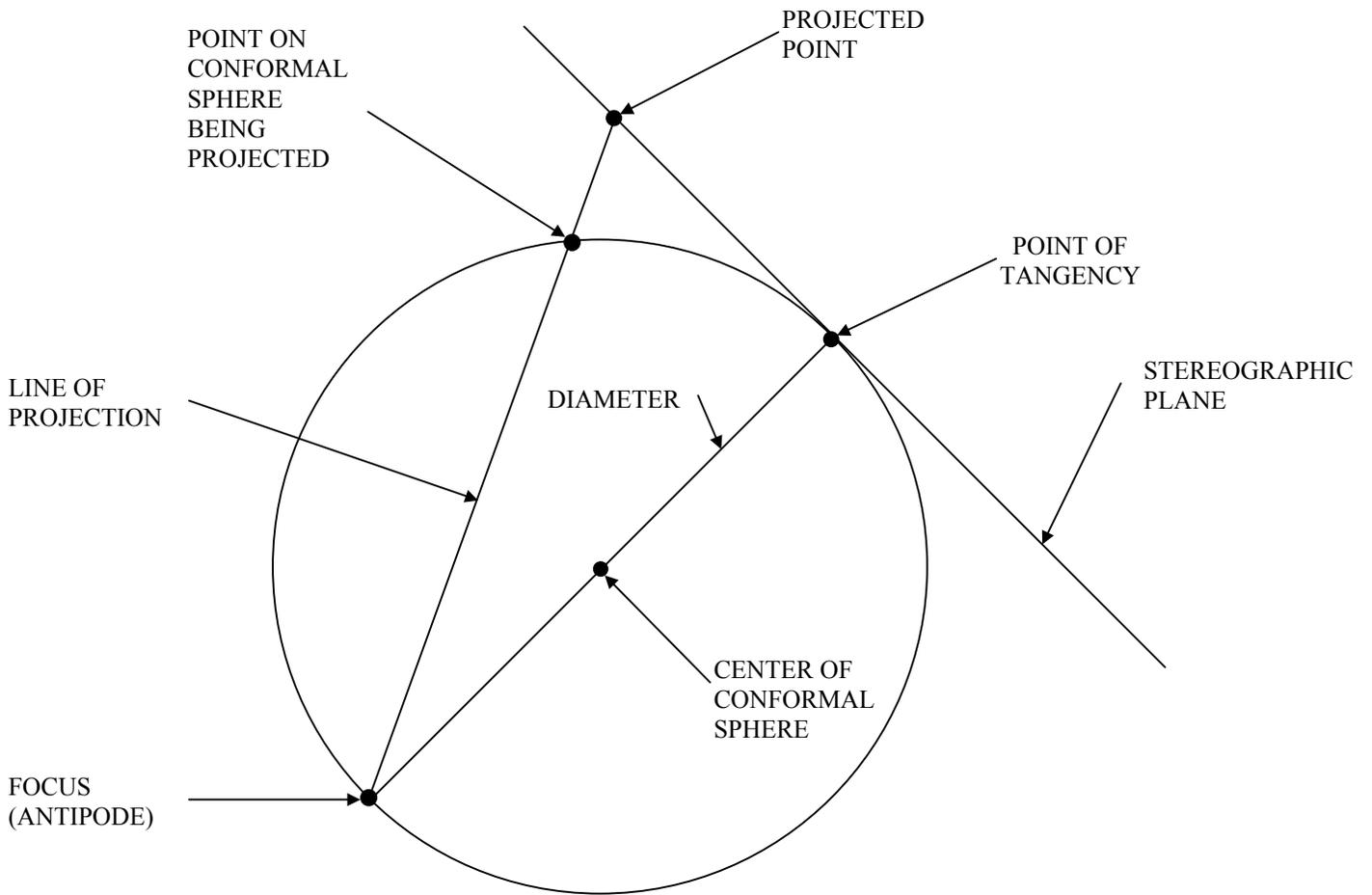


Figure 3.4.4-2: Stereographic Projection

3.4.5 Function: CNV_RADDMS (PL/I)

Converts a latitude or a longitude in radians to degrees, minutes, and seconds.

3.4.5.1 Description:

The descriptive information contained here has been obtained from the computer code for this PL/I procedure. The variable names in capital letters in parentheses are from the code.

Internally in URET an angle is represented both in radians and in degrees, minutes, and seconds, and it is necessary to convert back and forth between the two formats. This function converts an angle in radian measure to an angle in degrees, minutes, and seconds. The radian representation is in floating point format. The degrees, minutes, and seconds format is a character string (7 characters). The first 3 characters are the degree measure, the next 2 characters are the minutes measure, and the last 2 characters are the seconds measure. The procedure converts the radians into a floating point representation of the degrees, minutes, and seconds and then converts the floating point number into a character string.

First the radian value is converted to degrees by dividing by the number of degrees in a radian (DG_RAD). This resulting number is rounded up to the nearest second by adding a half second and then converting the number of seconds calculated later in the procedure from floating point format to fixed point format. (Conversion from floating point format to fixed point format truncates the fractional part of the number.) Negative angles are similarly rounded downward.

The integral number of degrees is found by converting the floating point representation of the number of degrees (DEGS) to fixed point format (D). Subtracting the integral number of degrees D from the total number of degrees and multiplying by 60 (the number of minutes in a second) leaves the minutes and seconds as a remainder (X) expressed as minutes. Again, converting to fixed point, the integral number of minutes are found (M). Subtracting both the integral number of degrees and the integral number of minutes (converted to degrees) from the value for the total degrees and multiplying by 3600, the number of seconds in a degree, leaves the number of seconds (X again). Converting this value into fixed point format gives the integral number of seconds (S).

Positive angles have been rounded up to the nearest second; negative angles have been rounded down to the nearest second.

The three values, the number of degrees (D), the number of minutes (M), and the number of seconds (S), are combined into a single floating point number (FLTDMS) in which the decimal digits for the 1s and 10s give the number of seconds, the decimal digits for the 100s and 1000s give the number of minutes, and the decimal digits for the 10,000s, 100,000s, and 1,000,000s give the number of degrees.

The floating point representation of the degrees, minutes, and seconds are converted into a character string (DMS).

3.4.5.1.1 Constants

The procedure uses two constants. The number of radians in a degree (DG_RAD) is an included external constant ; the value of $\frac{1}{2}$ a second expressed in degrees (RND) is an internal constant.

3.4.5.1.2 Units

The input angle is in radians in floating point format. The output angle is in degrees, minutes, and seconds - represented as a character string.

Table of Variable Definitions

Function Variable	Description	Math Symbol
RAD	Input variable - Angle in radians	ϕ_r
DG_RAD	Input constant - Number of radians in a degree	K_1
RND	Internal constant - 1/2 of angular second expressed in degrees	K_2
DEGS	Number of degrees in the input angle	ϕ_d
D	Integral number of degrees in the input angle	D
X	Remainder of minutes and seconds after the integral number of degrees have been subtracted from the input angle, units are minutes	R_1
M	Integral number of minutes in the input angle	M
X	Remainder of seconds after the integral number of degrees and the integral number of minutes have been subtracted from the input angle, units are seconds	R_2
S	Integral number of seconds in the input angle (rounded up or rounded down)	S
FLTDMS	Degrees, minutes, and seconds in the input angle expressed in floating point format	DMS_{fl}
DMS	Degrees, minutes, and seconds in the input angle expressed as a character string	DMS_{st}

3.4.5.2 Mathematics:

The angular measure in radians is converted to angular measure in degrees by dividing by K_1 , the number of radians in a degree, and then is prepared to be rounded up by adding K_2 , the value of 1/2 a second expressed in degrees. If ϕ_r is negative, the constant K_2 is subtracted to round down.

$$\phi_d = \frac{\phi_r}{K_1} + K_2 \quad \text{Equation 3.4.5-1}$$

Then the integral number of degrees D is extracted from ϕ_d by using the PL/I function FIXED. This function truncates the fractional part of the number.

$$D = \text{FIXED}(\phi_d) \quad \text{Equation 3.4.5-2}$$

The remainder of minutes and seconds is obtained by subtraction and is converted from degrees to minutes by multiplying by 60.

$$R_1 = (\phi_d - D) * 60 \quad \text{Equation 3.4.5-3}$$

The integral number of minutes M is extracted from R_1 by using the PL/I function FIXED.

$$M = \text{FIXED}(R_1) \quad \text{Equation 3.4.5-4}$$

The remainder of seconds is obtained by subtraction and is converted from degrees to seconds by multiplying by 3600.

$$R_2 = (\phi_d - D - \frac{M}{60}) * 3600 \quad \text{Equation 3.4.5-5}$$

The integral number of seconds S is extracted from R_2 by using the PL/I function FIXED.

$$S = \text{FIXED}(R_2) \quad \text{Equation 3.4.5-6}$$

The values obtained for the integral number of degrees, minutes, and seconds are combined into a single floating point number as follows.

$$DMS_{fl} = D * 10000 + M * 100 + S \quad \text{Equation 3.4.5-7}$$

This floating point number is converted into a character string by a string operation.

$$DMS_{st} = \text{STRING}(DMS_{fl}) \quad \text{Equation 3.4.5-8}$$

The function correctly converts a radian angular value to degrees, minutes, and seconds representation.

3.4.6 Function: CNV_SPEED (C)

Converts the given speed to its equivalent True Airspeed, Indicated Airspeed, and Mach at the given altitude, temperature and pressure.

3.4.6.1 Description:

This function converts the given airspeed, at a given altitude, temperature, and pressure, returning Mach, True Airspeed and Indicated Airspeed. Calculation of Indicated Airspeed assumes the speed of sound, temperature and pressure at sea level are all standard.

Table of Variable Definitions

Function Variable	Description	Math Symbol
*temp	Temperature at the current altitude (Supplied as an input, °K)	T_1
pr	The pressure ratio of current altitude pressure over sea level standard pressure.	$\frac{p_1}{p_s}$
_P0	Standard sea level pressure (= 29.92126 inches of mercury)	p_s
*pres	Freestream pressure at the current altitude (Supplied as an input, inches of mercury)	p_1
R	Universal Gas Constant of Air (= 287 m ² /s ² °K)	R
CONSTANT_1	Constant (=38.967876 kts/ $\sqrt{^\circ K}$)	$\sqrt{\gamma R}$
_Cso	Standard speed of sound at sea level (=661.47862 kts)	a_s
cs	Speed of sound at the current altitude	a_1
*tas	True airspeed (If supplied as an input, ft/s. It is returned as ft/s)	$V_1 = V_t$
*ias	Indicated airspeed (kts)	V_{ias}
*mach	Mach number (between 0 and 1)	M

3.4.6.2 Mathematics:

This function uses three equations which define Mach, True Airspeed, and Indicated Airspeed (Equation 3.4.6-11, Equation 3.4.6-12, and Equation 3.4.6-13) as a basis. All three of these equations are expressed with common terms, namely: the speed of sound, a , and the ratio of the total and static pressure difference over a static pressure, $(p_0 - p)/p$. However, the values of a and p may be different depending on the airspeed definition of assumed altitude (i.e. Indicated Airspeed assumes sea level altitude while Mach and True Airspeed assume the current altitude of the aircraft).

When this function is given a value for an airspeed, it will solve the corresponding airspeed equation for $(p_0 - p)/p$. Using this term, the function then solves the remaining two airspeed equations by making the necessary corrections for the altitude assumptions. (This is shown in Sections 3.4.6.2.1, 3.4.6.2.2, and 3.4.6.2.3).

The derivation of the three airspeed equations from classic thermodynamic and atmospheric relationships is provided below. Sections 3.4.6.2.1, 3.4.6.2.2, and 3.4.6.2.3 then show how the process of airspeed conversions is performed in the CNV_SPEED function.

NOTE: This function begins by determining the geopotential altitude h , as in Section 3.4.7, but then never uses this result anywhere else in the code.

Derivation:

Start with the energy equation for a subsonic, isentropic, compressible flow⁷

$$c_p T_1 + \frac{1}{2} V_1^2 = c_p T_0 \quad \text{Equation 3.4.6-1}$$

which can also be expressed as

$$\frac{T_0}{T_1} = 1 + \frac{V_1^2}{2c_p T_1} \quad \text{Equation 3.4.6-2}$$

where

- c_p = the specific heat for a perfect gas at constant pressure,
- T_1 = the temperature at a point in the freestream flow
- V_1 = the velocity at a point in the freestream flow. This is True Velocity.
- T_0 = the temperature at the stagnation point⁸.

By definition, c_p is expressed as

$$c_p = \frac{\gamma R}{\gamma - 1} \quad \text{Equation 3.4.6-3}$$

where

- γ = the constant which represents the ratio of specific heats at constant pressure to constant volume c_p/c_v
- R = the universal gas constant.

Substitute the expression for c_p from Equation 3.4.6-3 into Equation 3.4.6-2, which gives the result

$$\frac{T_0}{T_1} = 1 + \frac{(\gamma - 1)}{2} \frac{V_1^2}{\gamma R T_1} \quad \text{Equation 3.4.6-4}$$

The speed of sound at a point in the freestream flow is defined as

$$a_1^2 = \gamma R T_1 \quad \text{Equation 3.4.6-5}$$

⁷ An isentropic process is one in which there is neither heat exchange nor effect due to friction. A compressible flow is one in which the density of the fluid (i.e. air) can change from point to point

⁸ Velocity at the stagnation point is zero.

Since the right hand side of Equation 3.4.6-5 is found in Equation 3.4.6-4, the term a_1^2 can be substituted, giving the result

$$\frac{T_0}{T_1} = 1 + \frac{(\gamma - 1)}{2} \frac{V_1^2}{a_1^2} \quad \text{Equation 3.4.6-6}$$

Mach is defined as

$$M = \frac{V_1}{a_1} \quad \text{Equation 3.4.6-7}$$

Therefore, since the expression for Mach is found in Equation 3.4.6-6, M_1^2 can be substituted, giving the result

$$\frac{T_0}{T_1} = 1 + \frac{(\gamma - 1)}{2} M_1^2 \quad \text{Equation 3.4.6-8}$$

An isentropically compressed gas has the following relationship

$$\frac{p_0}{p_1} = \left(\frac{T_0}{T_1} \right)^{\frac{\gamma}{\gamma - 1}} \quad \text{Equation 3.4.6-9}$$

Where p_0 and p_1 represents the pressure at the stagnation and freestream point, respectively.

Therefore, combining Equation 3.4.6-8 and Equation 3.4.6-9 and solving for Mach gives the result

$$M_1^2 = \frac{2}{\gamma - 1} \left[\left(\frac{p_0}{p_1} \right)^{\frac{\gamma - 1}{\gamma}} - 1 \right] \quad \text{Equation 3.4.6-10}$$

which is equivalent to

$$M_1^2 = \frac{2}{\gamma - 1} \left[\left(\frac{p_0 - p_1}{p_1} + 1 \right)^{\frac{\gamma - 1}{\gamma}} - 1 \right] \quad \text{Equation 3.4.6-11}$$

Next, substitute Equation 3.4.6-7 into Equation 3.4.6-11 and solve for V_1

$$V_1^2 = \frac{2a_1^2}{\gamma - 1} \left[\left(\frac{(p_0 - p_1)}{p_1} + 1 \right)^{\frac{\gamma-1}{\gamma}} - 1 \right] \quad \text{Equation 3.4.6-12}$$

Calibrated airspeed is the speed the aircraft would be flying if it were at sea level. By assigning

$a_1 = a_s =$ the standard sea level value for the speed of sound

$p_s =$ the standard sea level value for pressure

calibrated airspeed V_{cas} can be defined using Equation 3.4.6-12

$$V_{cas}^2 = \frac{2a_s^2}{\gamma - 1} \left[\left(\frac{(p_0 - p_1)}{p_s} + 1 \right)^{\frac{\gamma-1}{\gamma}} - 1 \right] \quad \text{Equation 3.4.6-13}$$

URET assumes no instrument error and makes no distinction between indicated airspeed and calibrated airspeed.

$$V_{cas} = V_{ias}$$

For all of the following conversions, pressure and temperature are calculated using the CNV_STD_ATMOS function (see Section 3.4.7) or is measured using the DB_AIR_AT_POINT function. The above derivation was based on Anderson's *Introduction to Flight*, 1989.

3.4.6.2.1 Given Indicated Airspeed, Determine Mach and True Airspeed

Solve Equation 3.4.6-13 for $(p_0 - p_1)/p_s$

$$A_1 = \frac{(p_0 - p_1)}{p_s} = \left(\frac{(\gamma - 1) V_{ias}^2}{2 a_s^2} + 1 \right)^{\frac{\gamma}{\gamma-1}} - 1 \quad \text{Equation 3.4.6-14}$$

and substitute into Equation 3.4.6-11. Adjust this result by dividing by the pressure ratio

$$M^2 = \frac{2}{\gamma - 1} \left[\left(\frac{A_1 p_s}{p_1} + 1 \right)^{\frac{\gamma-1}{\gamma}} - 1 \right] \quad \text{Equation 3.4.6-15}$$

True airspeed is then calculated using Equation 3.4.6-7

$$V_t = a_1 M$$

3.4.6.2.2 Given True Airspeed, Determine Mach and Indicated Airspeed

Mach is determined by Equation 3.4.6-7. Solving Equation 3.4.6-11 for $(p_0 - p_1)/p_1$

$$A_2 = \frac{(p_0 - p_1)}{p_1} = \left(\frac{(\gamma - 1)}{2} M^2 + 1 \right)^{\frac{\gamma}{\gamma - 1}} - 1 \quad \text{Equation 3.4.6-16}$$

and adjusting A_2 by the pressure ratio to get

$$A_2 PR = \frac{(p_0 - p_1)}{p_1} \frac{p_1}{p_s} = \frac{(p_0 - p_1)}{p_s} \quad \text{Equation 3.4.6-17}$$

Substitute this result into Equation 3.4.6-13

$$V_{cas}^2 = \frac{2a_s^2}{\gamma - 1} \left[\left(A_2 \frac{p_1}{p_s} + 1 \right)^{\frac{\gamma - 1}{\gamma}} - 1 \right] \quad \text{Equation 3.4.6-18}$$

3.4.6.2.3 Given Mach, Determine True Airspeed and Indicated Airspeed

Solve Equation 3.4.6-7 for V_1 , then use the method in Equation 3.4.6-16 through Equation 3.4.6-18 to determine V_{ias}

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.6-1	Indicated airspeed is equivalent to calibrated airspeed. $V_{cas} = V_{ias}$. Assumes no instrument error	Reasonable. Instrument error is usually negligible in current aircraft.	Minor
R 3.4.6-2	Subsonic airspeeds	Reasonable for aircraft flying within controlled airspace	Important
R 3.4.6-3	Air flow is isentropic and compressible.	Reasonable for subsonic aircraft. These assumptions are needed to simplify mathematical modeling	Minor

3.4.7 Function: CNV_STD_ATMOS (PL/I)

Determines the standard atmospheric temperature and pressure at a specified altitude.

3.4.7.1 Description:

Temperature and pressure are functions of altitude for the standard atmosphere. The variation of temperature with respect to altitude is based on experimental evidence (see Figure 3.4.7-1). Pressure is derived from temperature and altitude using the equation of state for a gas and the hydrostatic equation (shown in Equation 3.4.7-4 and Equation 3.4.7-3, respectively). The measured altitude is the geometric altitude above sea level and is supplied as an input. To simplify the calculations, the geometric altitude is converted to a geopotential altitude, which assumes that the effect of gravity remains constant over all altitudes.

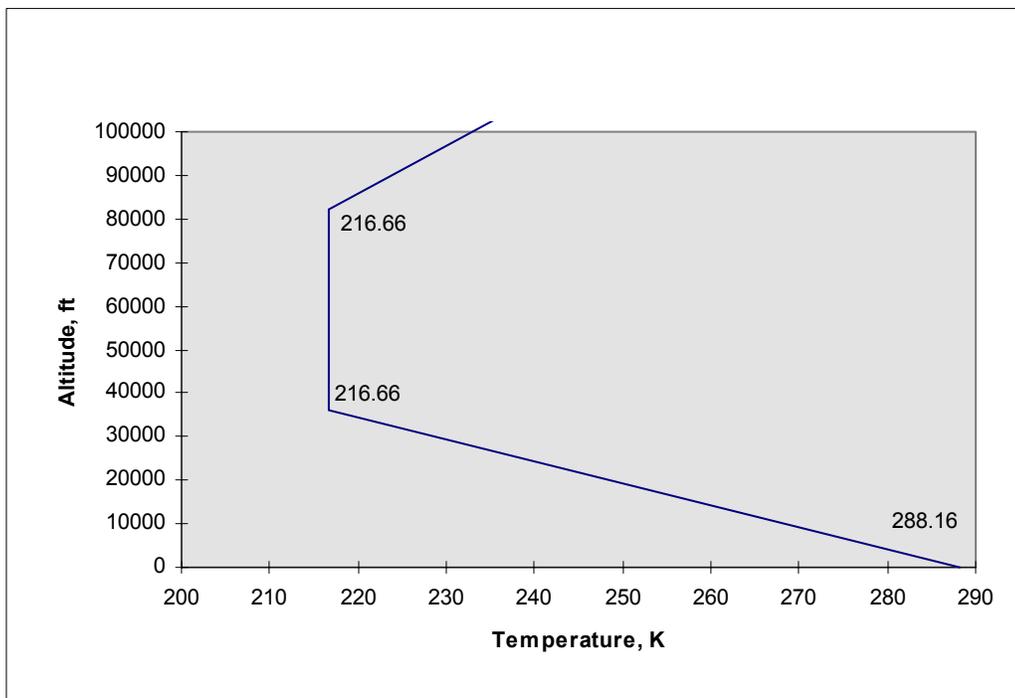


Figure 3.4.7-1: Temperature Distribution in the Standard Atmosphere

Table of Variable Definitions

Function Variable	Description	Math Symbol
ALT	The geometric altitude (Supplied as an input, ft)	h_G
HG	The geopotential altitude (nmi)	h
HB	Base of second altitude layer (= 5.9395248 nmi)	h_B
ERAD	The radius of the earth at latitude 45 (nmi). From the North American Datum, 1983, $r_{45} = 3438.1693$ nmi	r
TM0	Sea level standard temperature (= 288.15 K)	T_0
TB	Standard Temperature at the base of the second altitude layer (= 216.66 K)	T_B
TEMP	Standard temperature at the current altitude (Returned as an output, K)	T
LB	Temperature gradient of the first altitude layer (= -12.038K/nmi)	a
PR	The pressure ratio of current altitude pressure over sea level standard pressure.	$\frac{p}{p_0}$
P0	Standard sea level pressure (= 29.92126 inches of mercury)	p_0
PRES	Standard pressure at the current altitude (Returned as an output, inches of mercury)	p
R	Universal Gas Constant of Air (= 287 m ² /s ² K)	R
G	Sea level gravitational acceleration	g_0
C2	Constant (= -5.2558761)	$\frac{g_0}{aR}$
C3	Constant (= 0.223361). The pressure ratio of the standard pressure at the base of the second altitude layer over the standard sea level pressure	$\frac{p_B}{p_0}$
C4	Constant (= 0.29203894)	$\frac{g_0}{RT_B}$

3.4.7.2 Mathematics:

The CNV_STD_ATMOS function first converts the input geometric altitude from feet to nautical miles (h_G), then defines geopotential altitude as:

$$h = \left(\frac{r}{r + h_G} \right) h_G \quad \text{Equation 3.4.7-1}$$

This equation makes the assumption that gravitational acceleration is independent of altitude.

If the altitude is above sea level but below the first isothermal layer of the standard atmosphere ($0 \leq h \leq h_B$), there exists a temperature gradient with respect to altitude and temperature which is calculated as follows:

$$T = 288.16 - \left(\frac{h}{h_B} \right) (T_B - T_0) \quad \text{Equation 3.4.7-2}$$

However, if the altitude is greater than the first isothermal layer of the standard atmosphere ($h \geq h_B$), temperature remains constant⁹ ($T = 216.66$ K)

Pressure is derived by dividing the hydrostatic equation

$$dp = -\rho g_0 dh \quad \text{Equation 3.4.7-3}$$

by the equation of state for a perfect gas

$$p = \rho RT \quad \text{Equation 3.4.7-4}$$

which gives the result

$$\frac{dp}{p} = -\frac{g_0}{RT} dh \quad \text{Equation 3.4.7-5}$$

where $\rho =$ air density.

When altitude is above sea level but below the first isothermal layer of the standard atmosphere ($0 \leq h \leq h_B$), temperature variation follows a constant rate:

$$a \equiv \frac{dT}{dh} \quad \text{Equation 3.4.7-6}$$

or

$$dh = \frac{1}{a} dT \quad \text{Equation 3.4.7-7}$$

Substitute Equation 3.4.7-7 into Equation 3.4.7-5 and integrate to determine the pressure ratio

$$\int_{p_0}^p \frac{dp}{p} = -\frac{g_0}{aR} \int_{T_0}^T \frac{dT}{T} \quad \text{Equation 3.4.7-8}$$

$$\frac{p}{p_0} = \left(\frac{T}{T_0} \right)^{-\left(\frac{g_0}{aR} \right)} \quad \text{Equation 3.4.7-9}$$

⁹ The temperature remains constant until a geopotential altitude exceeds 25 km (82000 ft). It is assumed that controlled aircraft will not exceed this altitude.

These terms can be re-arranged to reflect the equation within the function

$$\frac{p}{p_0} = \left(\frac{T_0}{T_0 + ah} \right)^{\left(\frac{g_0}{aR} \right)} \quad \text{Equation 3.4.7-10}$$

When the altitude is greater than the first isothermal layer of the standard atmosphere ($h \geq h_B$), the pressure ratio is determined by integrating Equation 3.4.7-5.

$$\int_{p_B}^p \frac{dp}{p} = -\frac{g_0}{RT_B} \int_{h_B}^h dh \quad \text{Equation 3.4.7-11}$$

$$\frac{p}{p_B} = e^{-\left(\frac{g_0}{RT_B} \right) (h-h_B)} \quad \text{Equation 3.4.7-12}$$

These terms can be re-arranged to reflect the equation within the function

$$\frac{p}{p_0} = \frac{p_B}{p_0} e^{\left(\frac{g_0}{RT_B} \right) (h_B-h)} \quad \text{Equation 3.4.7-13}$$

Finally, Equation 3.4.7-10 or Equation 3.4.7-13 are solved for p , which is the standard atmospheric pressure at that altitude. Temperature (T) and pressure (p) are returned as outputs of the function.

This function appears to be reasonable and follows the classic derivations for the standard atmosphere¹⁰.

¹⁰ Refer to Anderson's *Introduction to Flight*, 1989 Chapter 3.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.7-1	Equation 3.4.7-1, geopotential altitude. Approximates that the gravitational acceleration is a constant, independent of altitude.	Reasonable. This is a classic assumption to simplify calculations and should have very little impact.	Minor
R 3.4.7-2	Equation 3.4.7-2, Equation 3.4.7-3, and Equation 3.4.7-4 all assume that the geopotential altitude will not exceed 82021 ft	Reasonable for aircraft in controlled airspace.	Minor

3.4.8 Function: CNV_STEREO_GNOMONIC (PL/I)

Converts stereographic X,Y coordinates to gnomonic X,Y coordinates. X and Y are in nautical miles.

3.4.8.1 Description:

The descriptive information contained here has been obtained from the computer code for this PL/I procedure, from the *URET Trajectory Modeling Algorithmic Definition* document (MTR 96W0000072) and from the reference stated in the text.

3.4.8.1.1 Stereographic Coordinates

The HCS uses the stereographic coordinate system to locate the aircraft. It is a Cartesian, planar coordinate system. The points on the surface of the earth are projected onto a plane which is tangent to the earth at a point within the ARTCC airspace. This representation of points on a plane when they are really on an ellipsoid introduces distortion or errors in position. However, for points close to the point of tangency, the distortion is small and can be ignored.

The latitude and longitude of a point on the earth's surface are based on a model of the earth as an ellipsoid of revolution. Points on the surface of the ellipsoid are projected onto the surface of a sphere (the conformal sphere) having the same center as the ellipsoid. Then the points on the sphere are projected onto the tangent plane using as a focal point the point on the sphere which is directly opposite the point of tangency (the antipode). This projection is illustrated in Figure 3.4.8-1. The projection is also described and illustrated in the description of the function CNV_LLXY. The location of a point thus placed on the tangent plane is specified by its stereographic coordinates.

3.4.8.1.2 Gnomonic Coordinates

The gnomonic projection is similar to the stereographic projection. Points on the sphere are projected onto the same tangent plane. For a gnomonic projection, the focal point is the center of the sphere. The location of a point projected onto the sphere in this way is specified by its gnomonic coordinates. The gnomonic projection is also illustrated in Figure 3.4.8-1.

3.4.8.1.3 Coordinate Conversion

Conceptually, this function, referring to Figure 3.4.8-1, takes a stereographic point A on the tangent plane, reverse projects it to the point B on the conformal sphere, and then projects it (the point B) back onto the tangent plane as a gnomonic point C. Given the stereographic coordinates of the point A, the function calculates the gnomonic coordinates of the point C.

3.4.8.1.4 Use of the Gnomonic Projection

A gnomonic projection of the surface of a sphere onto a plane introduces more distortion than a stereographic projection. However, great circle routes on the sphere are projected as straight lines on the tangent plane. This characteristic is used in the following way.

Let V_S and Z_S be the starting and ending points of a route. When an aircraft flies from V_S to Z_S , it flies the shortest distance from V_S to Z_S which is the great circle route. V_S and Z_S are defined by their stereographic coordinates. Great circle routes are approximated on the stereographic plane by a series of straight line segments. The stereographic coordinates of the ends of these line segments are found by first converting the stereographic coordinates of V_S and Z_S to gnomonic coordinates V_G and Z_G . The great circle route from V_S to Z_S in gnomonic coordinates is the straight line V_GZ_G . This straight line from V_G to Z_G is easily divided into a series of shorter line segments - V_GW_G , W_GX_G , X_GY_G , and Y_GZ_G . The gnomonic coordinates for W_G , X_G , Y_G , and Z_G are then converted to their corresponding stereographic coordinates. In this way the great circle route from V_S to Z_S is approximated in the stereographic plane by the straight line segments V_SW_S , W_SX_S , X_SY_S , and Y_SZ_S .

3.4.8.1.5 Note

The function being described in this section, `CNV_STEREO_GNOMONIC`, converts the stereographic coordinates to gnomonic coordinates. A companion function, `CNV_GNOMONIC_STEREO`, converts gnomonic coordinates to stereographic coordinates.

3.4.8.1.6 Conversion Equations

The equations, listed later on in the section, are given in NAS-MD-312, Appendix C, page C-1. (The equations are misprinted in the 10 May 1991 edition.)

The conversion functions (Equation 3.4.8-3 and Equation 3.4.8-4) convert X and Y coordinates relative to the point of tangency. It is necessary to subtract the values of the stereographic coordinates of the point of tangency from the values of the stereographic coordinates X and Y to get the values of X and Y relative to the origin of the stereographic coordinate system (Equation 3.4.8-1 and Equation 3.4.8-2).

3.4.8.1.7 Constants

One constant is necessary for the computation. It is the conformal radius of the earth

3.4.8.1.8 Units

The stereographic coordinates, the gnomonic coordinates, and the radius of the earth are all in nautical miles.

Table of Variable Definitions

Function Variable	Description	Math Symbol
SX, SY	Input variable - Stereographic X, Y coordinates of the point being converted	X_S, Y_S
DX, DY	Stereographic X, Y coordinates of the point being converted relative to the point of tangency	X_r, Y_r
ACP.XTANG, ACP.YTANG	Input parameters - Stereographic X, Y coordinates of the point of tangency	X_t, Y_t
ACP.RAD_EARTH	Input parameter - Conformal radius of the earth	R
GX, GY	Output variables - Gnomonic X, Y coordinates of the point being converted	X_G, Y_G

3.4.8.2 Mathematics:

The following equations are used in the function CNV_STEREO_GNOMONIC to calculate the gnomonic coordinates of a point, given its stereographic X and Y coordinates. First the coordinate values relative to the point of tangency are obtained.

$$X_r = X_S - X_t \quad \text{Equation 3.4.8-1}$$

$$Y_r = Y_S - Y_t \quad \text{Equation 3.4.8-2}$$

Then the conversion functions are applied.

$$X_G = \frac{X_r}{1 - \frac{X_r^2 + Y_r^2}{4R^2}} \quad \text{Equation 3.4.8-3}$$

$$Y_G = \frac{Y_r}{1 - \frac{X_r^2 + Y_r^2}{4R^2}} \quad \text{Equation 3.4.8-4}$$

The function correctly calculates the gnomonic coordinate values given the stereographic coordinate values.

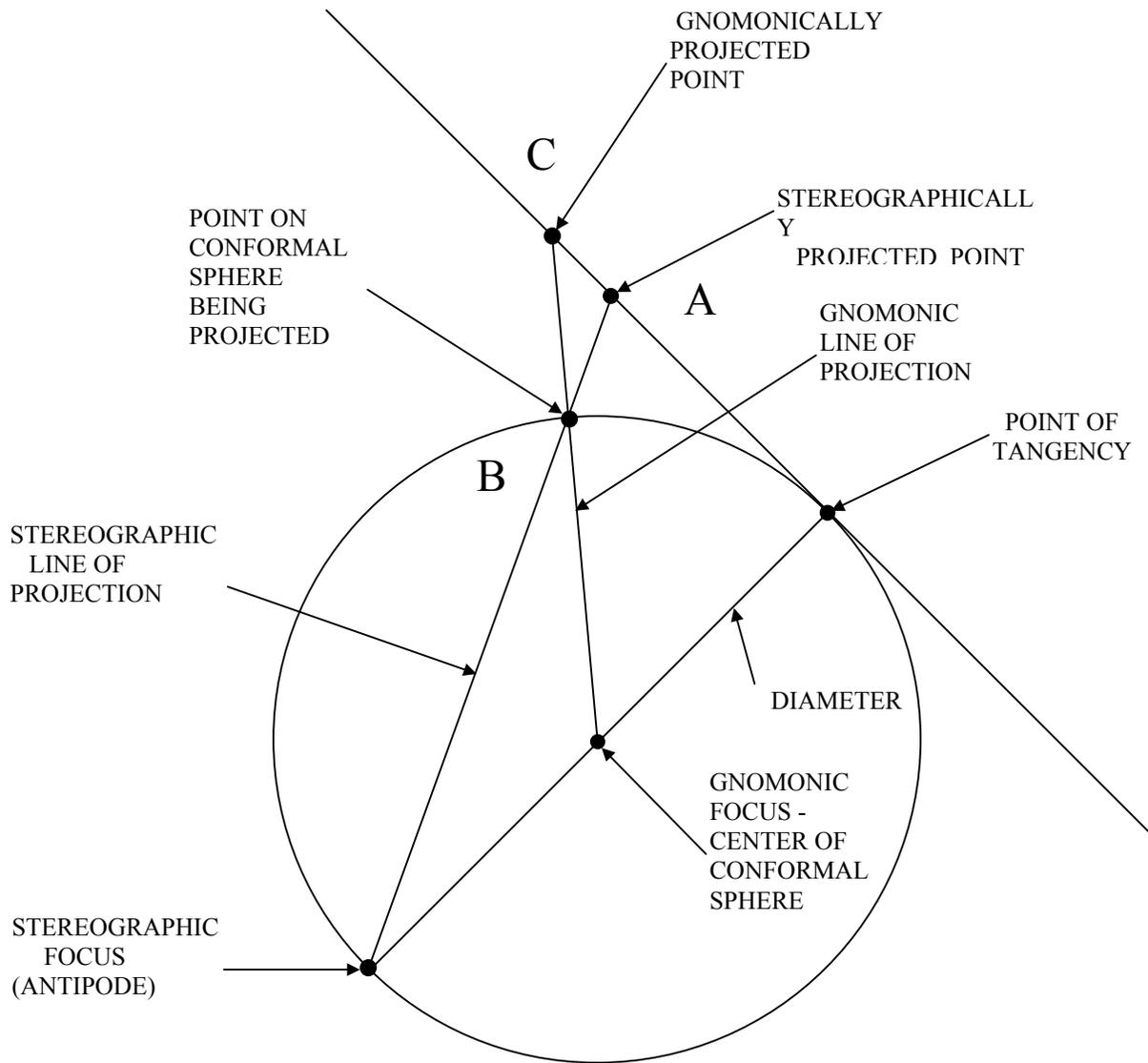


Figure 3.4.8-1: Stereographic to Gnomonic Projection

3.4.9 Function: CNV_XYLL (PL/I)

Converts the stereographic coordinates X, Y of a point to a geodetic latitude and a geodetic longitude. The X and Y coordinates are in nautical miles from the origin of the ARTCC coordinate system. Latitude and longitude are in radians in the International Ellipsoid of Reference.

3.4.9.1 Description:

The descriptive information contained here has been obtained from the computer code for this PL/I procedure, from the *URET Trajectory Modeling Algorithmic Definition* document (MTR 96W0000072), and from the references stated in the text.

3.4.9.1.1 Stereographic Coordinates

The HCS uses a stereographic coordinate system to locate the aircraft. The stereographic coordinate system is a Cartesian, planar coordinate system. The points on the surface of the earth are projected onto a plane which is tangent to the earth at a point within the ARTCC airspace. This representation of points on a plane when they are really on an ellipsoid introduces distortion or errors in position. However, for points close to the point of tangency, the distortion is small and can be ignored.

The stereographic coordinate system is an XY plane with the XY grid lined up approximately with the lines of latitude and longitude for points near the point of tangency. The parallel of constant latitude passing through the point of tangency is projected onto the stereographic plane as a line of constant Y value. The meridian of constant longitude passing through the point of tangency is projected onto the stereographic plane as a line of constant X value. At other points on the stereographic plane, the projections of lines of constant latitude or of constant longitude are curved and do not exactly line up with the lines of constant Y or X.

In the northern hemisphere, the line of constant X value on the stereographic plane passing through the point of tangency points to true north. As the Y coordinate of a point increases in value, the point gets closer to the north pole. The projection of the north pole is on this line through the point of tangency. Similarly the line of constant Y value passing through the point of tangency points due east.

The origin of the stereographic coordinate system is usually not the point of tangency but, in continental US, is a point in the southwest corner of the ARTCC airspace.

The point of tangency of the stereographic plane is defined by a latitude, a longitude, and a distance from the center of the ellipsoid. The location of the point of tangency is determined by the locations of the ARTCC's surveillance radars.

3.4.9.1.2 Geodetic Coordinates

Aircraft are located on the surface of the earth by a latitude and a longitude. The earth is almost a sphere, but not quite. It is modeled by an ellipsoid of revolution. This ellipsoid is created by rotating an ellipse about its minor axis. The axis becomes the earth's polar axis - the line connecting the north pole to the south pole. The International Ellipsoid of Reference is used. The distance in this model from the center of the earth to the north (or to the south) pole is 3432.4579 nautical miles. The distance from the center to the equator is 3444.0540 nautical miles. Latitudes and longitudes referred to this ellipsoid model are called the geodetic latitudes and geodetic longitudes.

3.4.9.1.3 Conformal Coordinates

It is convenient to do the conversion in two steps. A point's stereographic coordinates are first converted to a pair of intermediate coordinates, and then the intermediate coordinates are converted to the geodetic coordinates. The intermediate coordinate system is a sphere whose center is the same as the center of the ellipsoid and whose radius is based on the locations of the ARTCC's surveillance radars. The angles between lines on the sphere remain unchanged when they are projected onto the ellipsoid. Therefore the coordinate transformation is a conformal transformation and the values of the latitudes and longitudes in

this intermediate coordinate system are referred to as conformal latitudes and longitudes. The radius of the sphere is called the conformal radius of the earth. The geodetic longitude is the same as the conformal longitude; only the latitude is changed upon converting from conformal to geodetic coordinates. The stereographic X and Y coordinates are first converted to a conformal latitude and longitude, and then the conformal latitude and longitude are converted to the geodetic latitude and longitude.

3.4.9.1.4 Stereographic to Conformal Projection

The projection of a point on the stereographic plane onto a point on the conformal sphere is illustrated in Figure 3.4.9-1. The focal point (labeled “FOCUS” in the figure) for the projection is the point on the sphere which is directly opposite the point of tangency. A line of projection is drawn from the focal point to the point on the stereographic plane being projected (the “STEREOGRAPHIC POINT”). The point of intersection of this line with the conformal sphere (the “CONFORMAL POINT”) is the projected point.

3.4.9.1.5 Conformal to Geodetic Projection

The ellipsoid and the conformal sphere are concentric - that is, they have a common center. The radius of the sphere determines the scale of the projection in the stereographic plane and is chosen to minimize the errors introduced by the projection. The geodetic latitude of a point on the ellipsoid is the elevation above the equatorial plane of a line perpendicular to a plane tangent to the ellipsoid at that point.

The geometry of the conformal spherical projection is shown in Figure 3.4.9-2. The eccentricity of the earth is greatly exaggerated in the figure to show the geometry more clearly. A point P on the surface of the sphere has a conformal latitude of ϕ . This is the angle a radius vector makes with the equatorial plane of the sphere. A point P_g on the surface of the ellipsoid has a geodetic latitude of ϕ_g . In the two dimensional drawing of the figure this is the angle that the normal to the tangent at the point P_g makes with the equator. The conformal projection of the point P on the sphere onto the ellipsoid is the point P_g .

A conformal projection is one in which the change in scale at a given point is the same in all directions. The equation for the projection is derived by making the change in scale in mapping from the ellipsoid to the sphere along a meridian of longitude on the sphere equal to the change in scale along a parallel of latitude on the sphere. Since the lines of longitude are orthogonal to the lines of latitude, making the scale equal in the directions of constant latitude and constant longitude makes the scale equal in all directions.

The geodetic value of a latitude when converted depends only on its original conformal value and on the eccentricity of the ellipsoid.

3.4.9.1.6 Conversion Equations

The equations described here are listed later on in the section. Unless otherwise noted the equations in this function are found in *URET Trajectory Modeling Algorithmic Definition* document (MTR 96W0000072, Appendix A.4).

The coordinate conversion is done in two steps. The stereographic coordinates are converted into conformal (spherical) coordinates, and then the conformal coordinates are converted into geodetic coordinates. The second conversion is done by two equations (Equation 3.4.9-15 and Equation 3.4.9-20). The remaining equations are required to convert the stereographic coordinates to spherical coordinates.

A spherical triangle is formed on the surface of the conformal sphere by the three principal points. The geometry of the triangle is shown in Figure 3.4.9-3. The three points are the point being converted, the point of tangency, and the north pole. Spherical trigonometry is used to calculate the projection of the point on the stereographic plane onto the sphere. The triangle will be degenerate if the three points are not distinct. If the point of tangency is coincident with the north pole the triangle is degenerate and cannot be solved. Similarly if the point being converted is coincident with the north pole the triangle is degenerate and cannot be solved. In both cases the Law of Cosines has a divide by zero. If the point being converted

is coincident with the point of tangency, the triangle is again degenerate. However, the equations in this case are still valid and will yield a solution. The value of the angle β is defined in the code to be zero when both X_r and Y_r are zero in Equation 3.4.9-7.

In the spherical triangle in Figure 3.4.9-3, the lengths of the sides are defined by angular measure. The length of a side is the angle it subtends on the sphere. The diagram shows that the meridian of longitude from the point of tangency to the north pole subtends an angle of γ radians. This angle is the complement of the latitude of the point of tangency (See Equation 3.4.9-3).

3.4.9.1.7 Constants

Three constants are necessary for the computation. They are the conformal radius of the earth, and the coefficients of the two terms of the power series equation for the sine of the conformal latitude in terms of the sine of the geodetic latitude.

3.4.9.1.8 Units

The X and Y coordinate values and the conformal radius of the earth are measured in nautical miles.

3.4.9.1.9 Internal Error Checking

The absolute values for the sines and cosines are checked to make sure that they are less than or equal to 1. This is done after the calculations performed by the two power series approximations, the Law of Cosines, and by the Law of Sines. Imprecision and/or approximation in the computation may cause a value to exceed 1. When this occurs the value is reset to 1.

3.4.9.1.10 Unit Testing

A limited amount of unit testing was performed on this function. It performed correctly for all of the cases tested.

Note in the following table that ALPHA is used to represent both α and 2α , and that DLATC is used to represent ϕ , $\sin\phi_g$ and ϕ_g .

Table of Variable Definitions

Function Variable	Description	Math Symbol
XPOS, YPOS	Input data - Stereographic X, Y coordinates of the point being converted (nautical miles)	X, Y
ACP.COORDS.LATIT, ACP.COORDS.LONGIT	Input parameter - Geodetic latitude and longitude of the point of tangency	ϕ_{0g}, λ_0
ACP.COORDS.XTANG, ACP.COORDS.YTANG	Input parameter - Stereographic X, Y coordinates of the point of tangency	X_t, Y_t
RAD_EARTH	Input parameter - Conformal radius of the earth	R
PI	Input parameter - number of radians in one half of a revolution	π
CON_A	Constant - the first order coefficient in the power series expression for the conformal latitude in terms of the geodetic latitude	A
CON_B	Constant - the third order coefficient in the power series expression for the conformal latitude in terms of the geodetic latitude	B
CON_C	Constant - the first order coefficient in the power series expression for the geodetic latitude in terms of the conformal latitude	C
CON_D	Constant - the third order coefficient in the power series expression for the geodetic latitude in terms of the conformal latitude	D
CON_E	Constant - the fifth order coefficient in the power series expression for the geodetic latitude in terms of the conformal latitude	E
CON_F	Constant - the seventh order coefficient in the power series expression for the geodetic latitude in terms of the conformal latitude	F
SIN_PHI _g	Sine of the geodetic latitude of the point of tangency	$\sin \phi_{0g}$
SIN_PHI ₀	Sine of the conformal latitude of the point of tangency	$\sin \phi_0$
PHI ₀	Conformal latitude of the point of tangency	ϕ_0
GAMMA	Angular distance from the north pole to the point of tangency	γ
COS_GAMMA	Cosine of the angular distance from the north pole to the point of tangency	$\cos \gamma$
SIN_GAMMA	Sine of the angular distance from the north pole to the point of tangency	$\sin \gamma$
X, Y	Stereographic X, Y coordinates of the point being converted relative to the point of tangency	X_r, Y_r
ALPHA	One half of the angular distance from the point of tangency to the point being converted	α
COS_ALPHA	Cosine of the angular distance from the point of tangency to the point being converted	$\cos 2\alpha$
SIN_ALPHA	Sine of the angular distance from the point of tangency to the point being converted	$\sin 2\alpha$
BETA	Angle between the meridian of longitude passing through the point of tangency and the line joining the point of tangency to the point being converted	β

COS_DELTA	Cosine of the angular distance from the north pole to the point being converted	$\cos \delta$
DELTA	Angular distance from the north pole to the point being converted	δ
SIN_DELTA	Sine of the angular distance from the north pole to the point being converted	$\sin \delta$
DLATC	Conformal latitude of the point being converted	ϕ
SIN_EPS	Sine of the angle at the north pole between the meridian of longitude going through the point of tangency and the meridian of longitude going through the point being converted	$\sin \varepsilon$
COS_EPS	Cosine of the angle at the north pole between the meridian of longitude going through the point of tangency and the meridian of longitude going through the point being converted	$\cos \varepsilon$
EPSILON	Angle at the north pole between the meridian of longitude going through the point of tangency and the meridian of longitude going through the point being converted	ε
LONGC	Output data - conformal (and geodetic) longitude of the point being converted	λ
SIN_PHI	Sine of the conformal latitude of the point being converted	$\sin \phi$
DLATC	Sine of the geodetic latitude of the point being converted	$\sin \phi_g$
DLATC	Geodetic latitude of the point being converted	ϕ_g
LATC	Output data - geodetic latitude of the point being converted	ϕ_g

3.4.9.2 Mathematics:

The following equations are used in the function CNV_XYLL to calculate the geodetic latitude and longitude of a point, given its stereographic X and Y coordinates.

The angle γ is needed to solve the spherical triangle connecting the point being converted, the point of tangency, and the north pole. γ is found from the conformal latitude of the point of tangency ϕ_0 whose sine is given by the following equation from NAS-MD-312, Appendix D, Section 2.

$$\sin \phi_0 = A \sin \phi_{0g} + B \sin^3 \phi_{0g} \quad \text{Equation 3.4.9-1}$$

The angle γ is given then by these two equations

$$\phi_0 = \sin^{-1} \phi_0 \quad \text{Equation 3.4.9-2}$$

$$\gamma = \frac{\pi}{2} - \phi_0 \quad \text{Equation 3.4.9-3}$$

The stereographic coordinates X and Y are translated to an X and Y measured relative to the point of tangency by subtracting the stereographic coordinates of the point of tangency.

$$X_r = X - X_t \quad \text{Equation 3.4.9-4}$$

$$Y_r = Y - Y_t \quad \text{Equation 3.4.9-5}$$

The geometry for the angle α can be seen in Figure 3.4.9-1. On the stereographic plane the point being converted, called the stereographic point in the figure, is offset from the point of tangency by its relative coordinates X_r and Y_r . The equation for calculating α is

$$\alpha = \sin^{-1} \sqrt{\frac{X_r^2 + Y_r^2}{X_r^2 + Y_r^2 + 4R^2}} \quad \text{Equation 3.4.9-6}$$

The geometry for calculating β can be seen in Figure 3.4.9-3. β is the angle between the meridian of longitude passing through the point of tangency and the line on the sphere joining the point of tangency with the point being converted. This angle is unchanged when these two lines are projected onto the stereographic plane. On the stereographic plane, note that β is measured from the positive Y axis. The equation for calculating β is then

$$\beta = \tan^{-1} \left(\frac{X_r}{Y_r} \right) \quad \text{Equation 3.4.9-7}$$

If X_r and Y_r are both zero, β is defined to be zero.

The angle δ is the angular distance of the point being converted from the north pole. The Law of Cosines for spherical triangles is used to find its value using the previously calculated values for α and β . See Figure 3.4.9-3.

$$\cos \delta = \cos 2\alpha \cos \gamma + \sin 2\alpha \sin \gamma \cos \beta \quad \text{Equation 3.4.9-8}$$

The conformal latitude of the point being converted is found from the angle δ .

$$\delta = \cos^{-1}(\cos \delta) \quad \text{Equation 3.4.9-9}$$

$$\phi = \frac{\pi}{2} - \delta \quad \text{Equation 3.4.9-10}$$

The difference in longitudes of the point of tangency and the point being converted is the interior angle of the spherical triangle at the north pole - ε . See Figure 3.4.9-3. Its value is found by applying both the Law of Sines and the Law of Cosines for spherical triangles in the following way.

$$\sin \varepsilon = \frac{\sin 2\alpha \sin \beta}{\sin \delta} \quad \text{Equation 3.4.9-11}$$

$$\cos \varepsilon = \frac{\cos 2\alpha - \cos \gamma \cos \delta}{\sin \gamma \sin \delta} \quad \text{Equation 3.4.9-12}$$

$$\varepsilon = \tan^{-1} \left(\frac{\sin \varepsilon}{\cos \varepsilon} \right) \quad \text{Equation 3.4.9-13}$$

The longitude, conformal and geodetic, is obtained from the angle ε and the longitude of the point of tangency.

$$\lambda = \lambda_0 - \varepsilon \quad \text{Equation 3.4.9-14}$$

The conformal latitude has been found above in Equation 3.4.9-10. It is converted to the geodetic latitude by the following Equation 3.4.9-15. This equation is the reversion of Equation 3.4.9-1 above. That is the power series in Equation 3.4.9-1 is inverted to obtain Equation 3.4.9-15.

$$\sin \phi_g = C \sin \phi - D \sin^3 \phi + E \sin^5 \phi - F \sin^7 \phi \quad \text{Equation 3.4.9-15}$$

The coefficients C, D, E, and F are functions of the coefficients A and B and are calculated from the following equations.

$$C = \frac{1}{A} \quad \text{Equation 3.4.9-16}$$

$$D = \frac{B}{A^4} \quad \text{Equation 3.4.9-17}$$

$$E = \frac{3B^2}{A^7} \quad \text{Equation 3.4.9-18}$$

$$F = \frac{12B^3}{A^{10}} \quad \text{Equation 3.4.9-19}$$

The geodetic latitude ϕ_g is obtained by the inverse sine function.

$$\phi_g = \sin^{-1}(\sin \phi_g) \quad \text{Equation 3.4.9-20}$$

The function correctly calculates the geodetic latitude and longitude of a given stereographic X and Y coordinate pair.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.9-1	The conformal latitude ϕ_0 and colatitude γ of the point of tangency are calculated by Equation 3.4.9-1, Equation 3.4.9-2, and Equation 3.4.9-3 every time the function is called.	This calculation should be done once (for a given ARTCC) and the results saved for future use.	Minor
R 3.4.9-2	The variable names ALPHA and DLATC are used to represent more than one variable.	Distinct variables should have distinct variable names.	Minor
R 3.4.9-3	Truncated power series are used to calculate angles.	The power series approximations, Equation 3.4.9-1 and Equation 3.4.9-15 are adequate.	Critical. Coordinate conversion is basic to the conflict probe calculations.
R 3.4.9-4	It is assumed that neither the point being converted nor the point of tangency are at the north pole.	Neither the point of tangency nor the point being converted may be at the north pole. The function should check the input data for these two cases.	Minor This case does not occur within the U.S.

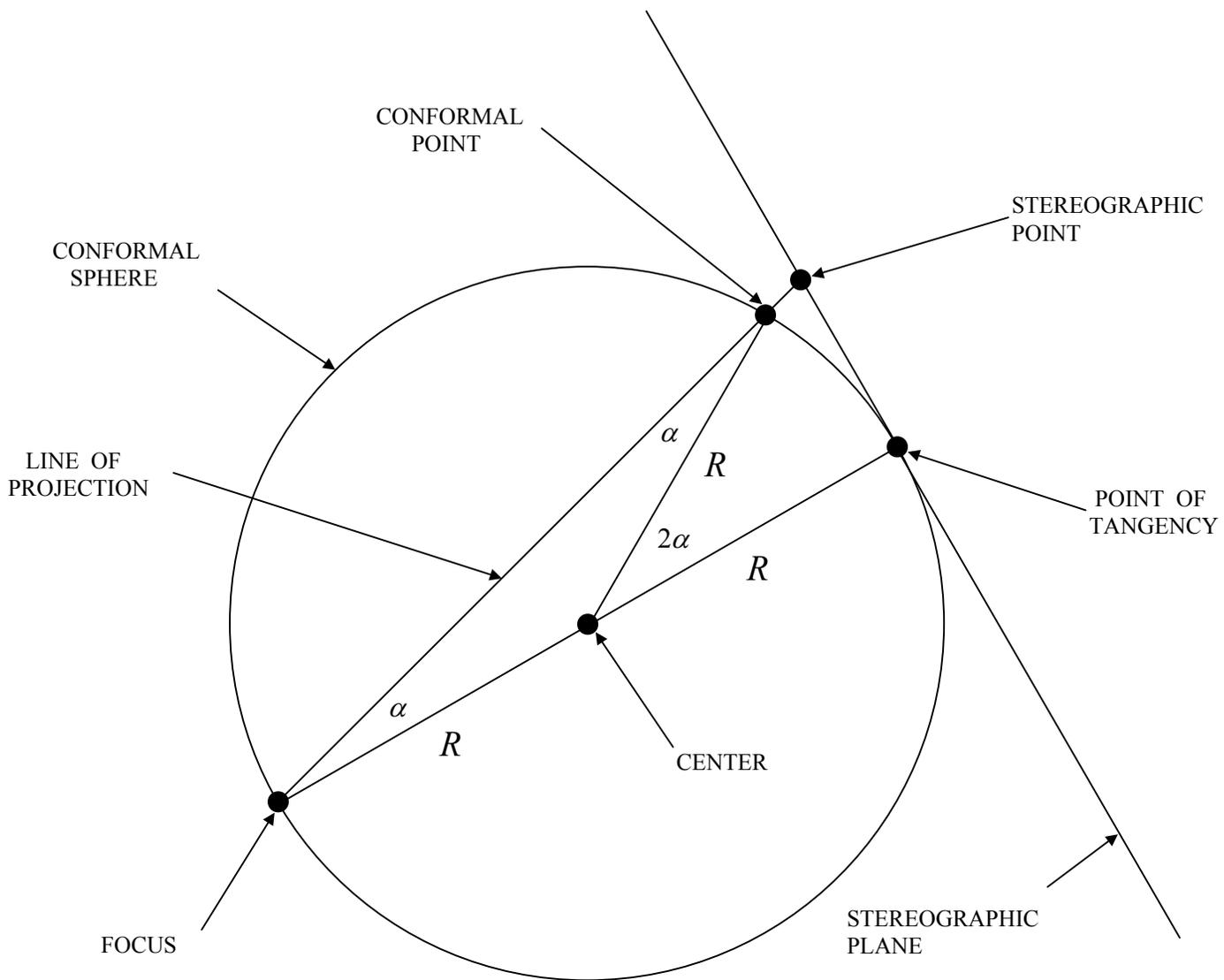


Figure 3.4.9-1: Stereographic Projection Details

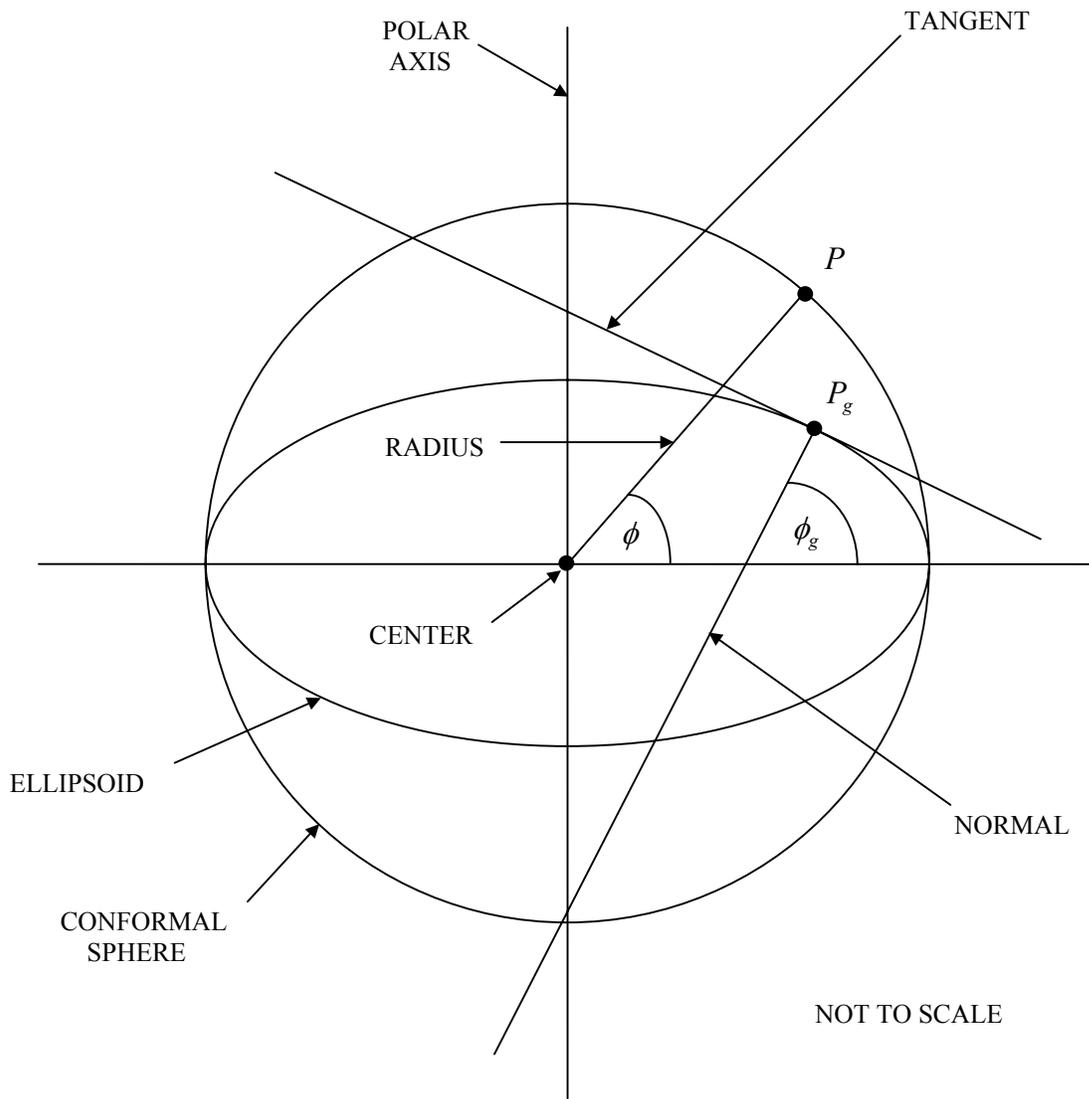


Figure 3.4.9-2: Mapping Geometry - Conformal to Geodetic

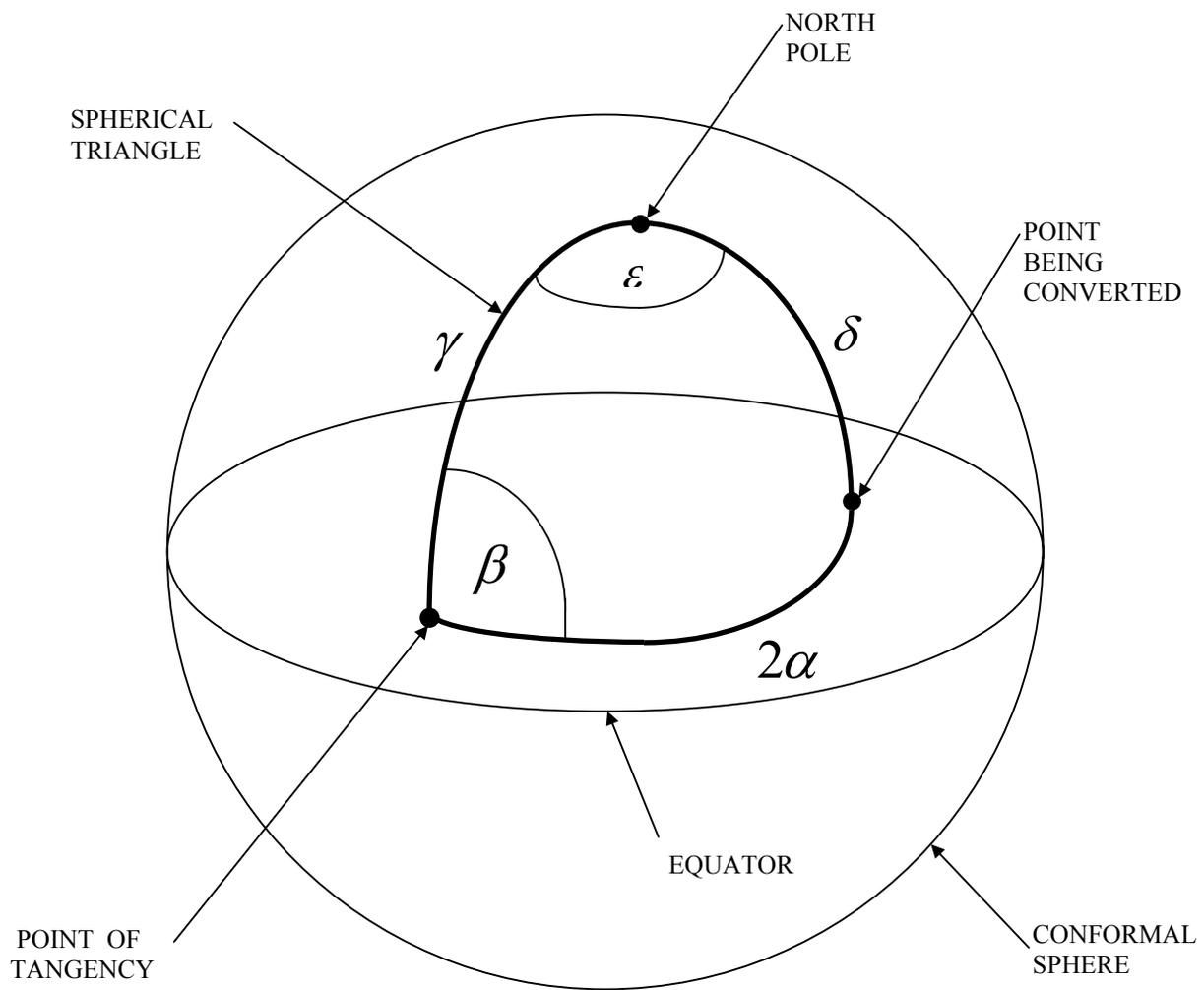


Figure 3.4.9-3: Spherical Triangle on the Conformal Sphere

3.4.10 Function: DB_AIR_AT_POINT (PL/I)

Determines the wind, temperature, and pressure at the specified point.

3.4.10.1 Description:

Determines the wind, temperature, and pressure at the specified point. It uses the values at the closest grid point.

3.4.10.2 Mathematics:

-Interpolation is turned off in D1.1

-Indices are calculated based on max, min and increments in the x, y, and z coordinates

-All indices truncate the decimals during calculation, therefore a 1 is added to the expression to ensure there is no zero index calculated

-If $(x - x_{\min}) \geq 1/2 x_{\text{inc}}$, use the next increment index

-If $(x - x_{\min}) < 1/2 x_{\text{inc}}$, use the current increment index

Use the index values for

WIND_X_AT_PT
WIND_Y_AT_PT
TEMPERATURE_AT_PT
PRESSURE_AT_PT

Indices are calculated as:

$$I = \frac{\left(x - x_{\min} + \frac{xy_{\text{inc}}}{2} \right)}{xy_{\text{inc}}} + 1$$

$$J = \frac{\left(y - y_{\min} + \frac{xy_{\text{inc}}}{2} \right)}{xy_{\text{inc}}} + 1$$

$$K = \frac{\left(z - z_{\min} + \frac{z_{\text{inc}}}{2} \right)}{z_{\text{inc}}} + 1$$

These indices correspond to the values in the AIR database

Assumes that weather is a constant throughout a weather grid, 50nmi x 50nmi x 1000ft in (x, y, z).

3.4.11 Function: DB_CDMERG (PL/I)

This function inserts a Clearance Directive (CD, otherwise known as a Planned Action, PA) into a linked list of CDs.

3.4.12 Function: DB_FIND_AUD_PTR (PL/I)

This function returns a pointer to a specified aircraft unique data (AUD) data structure as well as the structure's size. The AUD data structure is specified by an index number.

3.4.13 Function: GM_BRNG (PL/I)

Computes the bearing from the origin to a specified point.

3.4.13.1 Description:

Computes the bearing from the origin to a specified point.

Table of Variable Definitions

Function Variable	Description	Math Symbol
X, Y	Components of the specified point	x, y
MAGNITUDE	The scalar magnitude of the heading vector	$\sqrt{x^2 + y^2}$
BEARING	Bearing with respect to North	ψ

3.4.13.2 Mathematics:

The function performs the following simple calculations and logic.

If $x = 0$

If $y \geq 0$ then $\psi = 0$

If $y < 0$ then $\psi = \pi$

Else

If $\sqrt{x^2 + y^2} > 0$ then $\psi = \cos^{-1}\left(\left(\frac{x}{|x|}\right)\frac{y}{\sqrt{x^2 + y^2}}\right)$

Else $\psi = 0$

If $x < 0$ then add π to ψ

These calculations and logic appear to be reasonable and are based on sound trigonometric identities.

3.4.14 Function: GM_CONVEX (C)

This function determines if a test point (xt, yt) is inside a polygon region defined by the boundary points (vertices: ver[n][2]).

3.4.14.1 Description:

The algorithm to detect if a point lies within an octagon is based on the following theorem:

A point Pt(xt, yt) is outside an octagon if and only if:

$$Q(V_{i-1}, V_i, Pt) \leq 0;$$

$$i = 1, 2, \dots, n$$

$$V_0 = V_n$$

where the octagon vertices are listed in a counterclockwise manner

The function checks the inequality for every point in the octagon, ensuring that the point is not outside the polygon region.

Table of Variable Definitions

Function Variable	Description	Math Symbol
xt, yt	x, y coordinates of the test point	x_t, y_t
ver	2 dimensional array containing the coordinates of the matrices (i.e. ver[0][0] = X ₀ , ver[0][1]=Y ₀ , ...)	X_i, Y_i
n	number of boundary points (or number of vertices)	n
OUT	status return variable of function, OUT=0 means test point is outside octagon	
IN	status return variable of function, IN=1 means test point is inside octagon	

3.4.14.2 Mathematics:

As stated previously in the description, the function utilizes the following theorem to evaluate if a test point lies outside an octagon.

$$Q(V_{i-1}, V_i, Pt) \leq 0;$$

$$i = 1, 2, \dots, n$$

$$V_0 = V_n$$

Equation 3.4.14-1

(where the octagon vertices are listed in a counterclockwise manner)

For the octagon polygon, the matrix determinant Q is evaluated for each i=1, 2, .. 8 and if any of the boundary points has a determinant less than or equal to zero, then the test point is outside the octagon. The function uses a fairly straightforward loop illustrated in Figure 3.4.14-2.

Derivation:

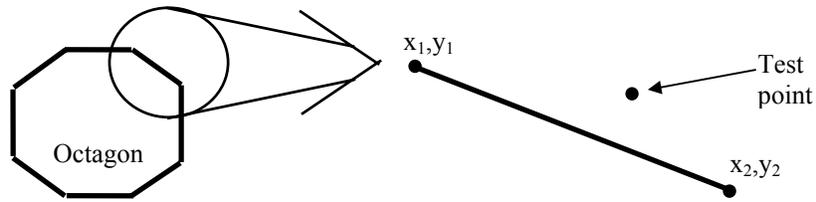


Figure 3.4.14-1: Side of octagon and test point

The function loops around each linear segment of the octagon in a counterclockwise direction. The loop starts with the first linear segment, illustrated in Figure 3.4.14-1 above. The vertices are the end points of the linear segment. These end points are used to set up a linear equation:

$$\left(\frac{y_1 - y_2}{x_1 - x_2}\right) = \left(\frac{y - y_2}{x - x_2}\right) \quad \text{Equation 3.4.14-2}$$

where x, y are the coordinates of the test point and the vertices are the coordinates x_1, y_1 to x_2, y_2

The above equation can be expanded to a similar form as the determinant, as shown below.

$$\begin{aligned} \left(\frac{y_1 - y_2}{x_1 - x_2}\right)x - \left(\frac{y_1 - y_2}{x_1 - x_2}\right)x_2 + y_2 &= y \\ \left(\frac{y_1 - y_2}{x_1 - x_2}\right)x - \left(\left(\frac{y_1 - y_2}{x_1 - x_2}\right)x_2 - y_2\right) &= y \\ \left(\frac{y_1 - y_2}{x_1 - x_2}\right)x - \left(\frac{(y_1 - y_2)x_2 - y_2(x_1 - x_2)}{x_1 - x_2}\right) &= y \\ \left(\frac{y_1 - y_2}{x_1 - x_2}\right)x + \left(\frac{x_1y_2 - y_1x_2}{x_1 - x_2}\right) &= y \end{aligned}$$

$$(y_1 - y_2)x + (x_1y_2 - y_1x_2) = y(x_1 - x_2) \quad \text{Equation 3.4.14-3}$$

The equation above expressed in determinant becomes:

$$\begin{vmatrix} x & y & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} = 0, \text{ if } x, y \text{ on the line segment}$$

$$x \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} - y \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} + \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} = 0$$

$$x(y_1 - y_2) - y(x_1 - x_2) + (x_1y_2 - y_1x_2) = 0$$

Referring back to Equation 3.4.14-3, if the equation is rearranged and the difference in the y axis between the line and the test point is calculated, the equation becomes:

$$\Delta.y = \left(\frac{y_1 - y_2}{x_1 - x_2} \right) x + \left(\frac{x_1y_2 - y_1x_2}{x_1 - x_2} \right) - y \quad \text{Equation 3.4.14-4}$$

Referring back to Figure 3.4.14-1, there are three cases that the above Equation 3.4.14-4 will evaluate:

1. If $\Delta.y < 0$, then the test point y is above the line and outside the octagon.
2. If $\Delta.y > 0$, then the test point is below the line and inside the octagon.
3. If $\Delta.y = 0$, then the test point is on the line and on the octagon exactly.

The function iterates for each side of the octagon and the orientation remains the same, since the direction of the iteration is constant. For the case 3, the function considers the test point outside the octagon (a conflict on the octagon exactly meets separation standards and is not a violation).

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.4.14-1	The method description listed in the comment section of the function states that the test point is inside the polygon if the Q determinant is less than or equal to zero. This is exactly opposite the code what the code does.	Needs correction.	Minor

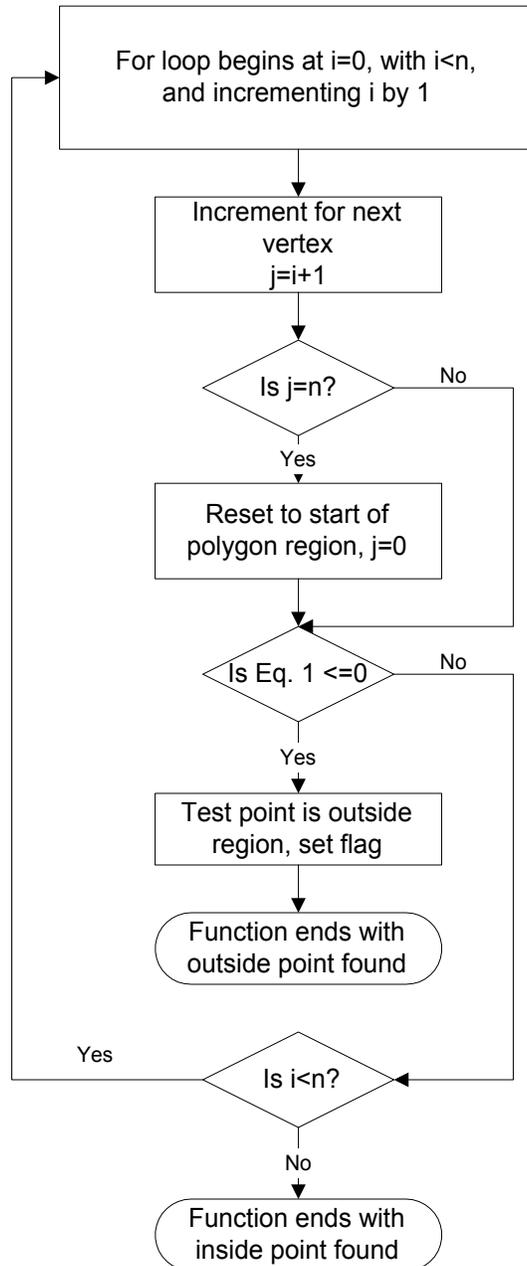


Figure 3.4.14-2: GM_CONVEX Main Function Loop

3.4.15 Function: GM_INSEC (C)

Computes the intersection point of two line segments.

3.4.15.1 Description:

The function receives the end point x and y coordinates for two line segments or points. It first determines if the two line segments or points intersect and then categorizes for the following cases:

0. Line 1 is a point and line 2 is also a point.
1. Line 1 is a point and line 2 is a line.
2. Line 1 is a line and line 2 is a point.
3. Line 1 and line 2 are parallel or collinear.
4. Line 1 and line 2 intersect.

The function calculates the intersection point based on the case type defined above. There is a distinction made between the given line segments and the calculated infinite lines containing those segments. For example, an intersection may take place on the line, without taking place on the line segment, and if that case was determined, the status variable would return a value 1 or 2.

Table of Variable Definitions

Function Variable	Description	Math Symbol
x1, y1	Initial x, y coordinates of segment 1	x_1, y_1
x2, y2	Second x, y coordinates of segment 1	x_2, y_2
x3, y3	Initial x, y coordinates of segment 2	x_3, y_3
x4, y4	Second x, y coordinates of segment 2	x_4, y_4
ratio1	ratio of intersection to line 1; = distance between one end point and the intersection point / length of the line segment; ratio = 9999, indicates ratio has no meaning	ratio ₁
ratio2	ratio of intersection to line 2; (same as above)	ratio ₂
delta1, delta2, delta3, delta4	difference values for each set of line end points, = $(y_1 - y_2)$, = $(x_1 - x_2)$, = $(y_3 - y_4)$, = $(x_3 - x_4)$, respectively	$\delta_1, \delta_2, \delta_3, \delta_4$
delta	= $(\delta_1) * (\delta_4) - (\delta_2) * (\delta_3)$	δ

3.4.15.2 Mathematics

Initial Function Check

The function begins by checking if the two given lines do not intersect. Using the following expression, the two lines will not intersect if δ equals zero.

$$\delta = (\delta_1)(\delta_4) - (\delta_2)(\delta_3)$$

Equation 3.4.15-1

Case 0

For Case 0 (TWO_POINTS) both lines are points, therefore if both points are equal an intersection is evaluated at that point. If both points do not equal, then no intersection status is evaluated.

Case 1-2

For Case 1 and 2 (ONE_POINT_TWO_LINE or ONE_LINE_TWO_POINT), one of the lines is a point and the other is a line segment. For Case 1, the point at x_1 and y_1 is determined to be on the other line, on the line segment, or not on the line at all. If the point 1 at (x_1, y_1) lies on the line 2 from (x_3, y_3) to (x_4, y_4) , the slope of a line from point 1 to (x_3, y_3) must be equal to the slope of line 2. Therefore, the function assumes an intersection does occur if the following equation is evaluated true.

$$(y_3 - y_1)(x_4 - x_3) = (y_4 - y_3)(x_3 - x_1) \quad \text{Equation 3.4.15-3}$$

This is true only if the slopes are equivalent, specifically:

$$(y_4 - y_3)/(x_4 - x_3) = (y_3 - y_1)/(x_3 - x_1) \quad \text{Equation 3.4.15-4}$$

If the intersection point lies on the line segment, the point 1 must be between the points 3 and 4 or (x_3, y_3) and (x_4, y_4) , which returns the SEG_INSEC value. If the point 1 lies outside the line segment's end points, the intersection point is the same, but the status is returned with the intersection on the extension (SEG1_EXT2_INSEC).

For the Case 2, the line is defined for line segment 1 (points x_1, y_1 to x_2, y_2) and the point 2 (x_3, y_3) . The function loop is identical to the description for Case 1.

Case 3

For Case 3, the two lines are parallel and the function determines if they are collinear. The sum for the x coordinates is calculated and checked for the zero condition. If the sum adds to zero, the lines are evaluated as collinear, since the slopes are equivalent. If the sum is not equal to zero, the lines may still be collinear only if the following equation for δ is equal to zero.

$$\delta = (\delta_4) (y_3 - y_1) - (\delta_3)(x_3 - x_1) \quad \text{Equation 3.4.15-5}$$

Derivation:

Show that the two point slope line equations with equal slopes and solved simultaneously for the same x and y (representing the collinear case) reduce to Equation 3.4.15-5.

$$\text{Equation for line 1: } (y - y_1) / (x - x_1) = (y_2 - y_1) / (x_2 - x_1)$$

$$\text{Equation for line 2: } (y - y_3) / (x - x_3) = (y_4 - y_3) / (x_4 - x_3)$$

Rearrange the terms for line 1 equation and substitute δ values:

$$y = [((y_2 - y_1) / (x_2 - x_1))x] - [((y_2 - y_1) / (x_2 - x_1))x_1 - y_1]$$

$$y = [(\delta_1 / \delta_2)x] - [(\delta_1 / \delta_2)x_1 - y_1] \quad \text{Equation 3.4.15-6}$$

It can also be shown for line 2 equation that:

$$y = [(\delta_3 / \delta_4) x] - [(\delta_3 / \delta_4)x_3 - y_3] \quad \text{Equation 3.4.15-7}$$

By equating Equation 3.4.15-6 and Equation 3.4.15-7, the two line equations are being set collinear, if the slopes are equal. With the slopes equal, let $m = (\delta_1 / \delta_2) = (\delta_3 / \delta_4)$. Therefore, solve this equation to determine the δ from Equation 3.4.15-5.

$$\begin{aligned} [mx] - [(mx_1) - y_1] &= [mx] - [(mx_3) - y_3] \\ 0 &= mx_1 - mx_3 + y_3 - y_1 \\ m(x_3 - x_1) &= (y_3 - y_1) \end{aligned} \quad \text{Equation 3.4.15-8}$$

By using $m = (\delta_3 / \delta_4)$ from above, Equation 3.4.15-8 reduces to Equation 3.4.15-5:

$$(\delta_3 / \delta_4) (x_3 - x_1) = (y_3 - y_1)$$

$$0 = [\delta_4(y_3 - y_1)] - [\delta_3(x_3 - x_1)] \quad \text{Equation 3.4.15-9}$$

To show that the x coordinate sum can be used to check for collinear lines, determine if the line equations for both lines are equivalent and thus collinear when the sum of the x coordinates is equal to zero.

Let, $xs = x_1 + x_2 + x_3 + x_4 = 0$, so :

For the x values to sum to zero, they either all must be zero or the variables must have both positive and negative values. Unless there are other assumptions relating to the source of the x coordinates, the sum and the equivalent slopes do not ensure that the lines are collinear. This check may only be an error trap for all zero values for the x coordinates and used for single precision arithmetic, but this assumes all the x coordinates are positive (in the first quadrant). Therefore, there is no reason for keeping this portion of the source code at this time.

Case 4

For Case 4, the lines do intersect, so the function will return both the intersection coordinates and the status category. The status is based on whether the two lines intersect either within the two segments (SEG_INSEC), on one segment and the other line extension (i.e. SEG1_EXT2_INSEC), or both line extensions (EXT_NO_INSEC). The function calculates the ratios between the distance of the intersection point to the beginning segment point and the total segment length. This ratio is calculated for both lines and used to determine the intersection point and what status is returned. The function expresses the following formula for the ratio of the intersection distance to the segment length of line 1.

$$\text{ratio}_1 = [(x_3 - x_1)(\delta_3) - (y_3 - y_1)(\delta_4)] / (\delta) \quad \text{Equation 3.4.15-10}$$

Derivation:

To solve for Equation 3.4.15-10, it is first necessary to derive general intersection equations from the two line equations, again:

$$\text{Equation for line 1: } (y - y_1) / (x - x_1) = (y_2 - y_1) / (x_2 - x_1)$$

$$\text{Equation for line 2: } (y - y_3) / (x - x_3) = (y_4 - y_3) / (x_4 - x_3)$$

Solve for y for each line:

$$y = m_1x - m_1x_1 + y_1 \quad \text{Equation 3.4.15-11}$$

$$y = m_2x - m_2x_3 + y_3 \quad \text{Equation 3.4.15-12}$$

As in Equation 3.4.15-2, the slopes are:

$$m_1 = (y_2 - y_1) / (x_2 - x_1); \quad m_2 = (y_4 - y_3) / (x_4 - x_3)$$

Equate Equation 3.4.15-11 and Equation 3.4.15-12 and then solve for x:

$$x = (m_1x_1 - m_2x_3 + y_3 - y_1) / (m_1 - m_2) \quad \text{Equation 3.4.15-13}$$

Now, it is necessary to represent the ratio₁ in terms of the x coordinates. This can be accomplished by using similar triangles as illustrated in the following diagram.

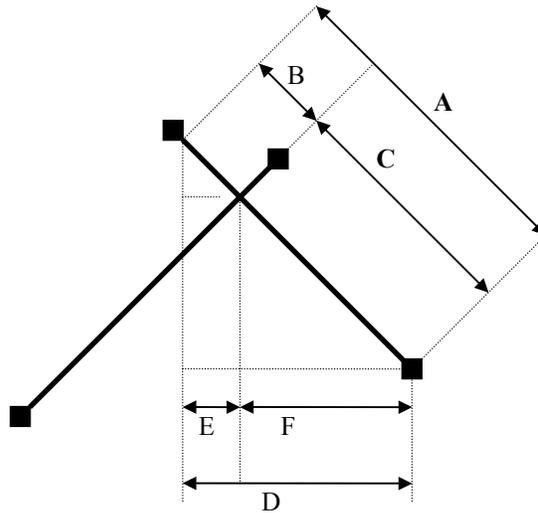


Figure 3.4.15-2: Ratio of Distance B to A

As illustrated in Figure 3.4.15-2 above, the ratio of Equation 3.4.15-10 can be expressed as the ratio of distance B to A. From similar triangles, the ratio B/A is equivalent to E/D. This second ratio can be expressed as:

$$\text{ratio}_1 = E / D = (x - x_1) / (x_2 - x_1) \quad \text{Equation 3.4.15-14}$$

Using Equation 3.4.15-13, the ratio can be shown to be equivalent to Equation 3.4.15-10. First, substitute the variable x from Equation 3.4.15-13 into the following equation for the ratio E/D:

$$\begin{aligned} \text{ratio}_1 &= E / D = (x - x_1) / (x_2 - x_1) = \\ &= [(y_3 - y_1 + m_1x_1 - m_2x_3) - x_1(m_1 - m_2)] / [(m_1 - m_2)(x_2 - x_1)] \end{aligned} \quad \text{Equation 3.4.15-15}$$

Multiply and cancel the terms in the numerator to get the following,

$$\text{ratio}_1 = [m_2(x_1 - x_3) + y_3 - y_1] / [(m_1 - m_2)(x_2 - x_1)]$$

Now, multiplying the slopes: $m_1 = (y_2 - y_1) / (x_2 - x_1)$ and $m_2 = (y_4 - y_3) / (x_4 - x_3)$ in the numerator and denominator, provides the following expression for the ratio₁:

$$\begin{aligned} &= [((y_4 - y_3)/(x_4 - x_3))(x_1 - x_3) + y_3 - y_1] / [((y_2 - y_1)(x_4 - x_3) + (y_4 - y_3)(x_1 - x_2))/(x_4 - x_3)] \\ &= [(y_4 - y_3)(x_1 - x_3) + (y_3 - y_1)(x_4 - x_3)] / [(y_2 - y_1)(x_4 - x_3) + (y_4 - y_3)(x_1 - x_2)] \\ &= [(y_3 - y_4)(x_3 - x_1) - (y_3 - y_1)(x_3 - x_4)] / [(y_1 - y_2)(x_3 - x_4) - (y_3 - y_4)(x_1 - x_2)] \end{aligned}$$

Thus, by substituting the δ terms the Equation 3.4.15-10 is returned.

$$\text{ratio}_1 = [(\delta_3)(x_3 - x_1) - (y_3 - y_1)(\delta_4)] / [\delta]$$

An analogous argument can be shown for the ratio₂ for line 2's intersection. The ratio is the intersection to endpoint distance to segment distance, so if the ratio is greater than one, the intersection is certainly outside the segment. The function evaluates the ratios to determine if the intersection is outside the segment. The function uses the following expression, returning a true value when the intersection is outside the given segment:

$$[\text{Absolute value}(\text{ratio}_2 - 0.5)] > [0.5 + \epsilon] \quad \text{Equation 3.4.15-16}$$

The epsilon (ϵ) value is a small value (i.e. 0.0001) for approximation in the comparison. If the intersection is outside one of the line segments and located on the line extension, the ratio may not be greater than one (although a ratio greater than one always proves that the intersection is outside the line segment). When the intersection distance is within the segment length but outside the line segment, depending on the orientation of the points, the ratio will either be greater than 1 or a negative number. This negative value will ensure that Equation 3.4.15-16 returns true, the correct result.

Note: In GM_PTLINE, another approach was used to calculate the intersection point to a line. In summary, GM_PTLINE used Equation 3.4.15-13 and the point slope equation of the line to find the intersection point. A ratio was not used in this algorithm to determine if the intersection took place inside the line segment, but a simple check in the x coordinates was utilized.

Assessment Table

REF#	Approximation / Assumption	Assessment	Impact on APD
R 3.4.15-1	Two flight segments have the same slope, but may not have the same sum for collinear lines. It assumes the coordinates are only positive, which stems from an early version of URET.	Incorrect assumption.	Minor*
R 3.4.15-2	The check for the intersection of the lines, the check for parallel/collinear line pairs, and the final determination of the intersection point all incorporate adjustments to minimize the effect of floating point arithmetic error in single precision. The problem is that these adjustments are undocumented in the code.	Need more documentation or comments explaining these adjustments.	Minor

*Minor impact in APD since the consequence may produce either a parallel or collinear line which results in the same outcome in only one APD function call, the CFP_FINE function.

However, the impact of GM_INSEC's assumptions on other module's functions is yet to be determined.

3.4.16 Function: GM_PTLINE (PL/I)

Finds the relationship between a point and a line.

3.4.16.1 Description:

This function will calculate the minimum distance between a point and a line and will indicate if the point is actually on the line. The GM_PTLINE function takes the coordinates of two end-points of a line segment and the coordinates of the point as inputs. The function returns the shortest distance from the point to the line, the coordinates of the point where the normal line, from the point to the line, intersects the line segment, and a status indicator which signals if the point lies on the line segment.

Table of Variable Definitions

Function Variable	Description	Math Symbol
X1, Y1	The two dimensional Cartesian coordinates of the first point which defines the line segment	x_1, y_1
X2, Y2	The two dimensional Cartesian coordinates of the last point which defines the line segment	x_2, y_2
X3, Y3	The two dimensional Cartesian coordinates of the point	x_3, y_3
XI, YI	The two dimensional Cartesian coordinates of the point projected onto the line segment	x_{int}, y_{int}
D	The minimum distance from the point to the line segment	d
S1	Slope of the line segment	m_1
S2	Slope of the normal line to the line segment	m_2

3.4.16.2 Mathematics:

Find the slope an equation for the line (L) which includes the line segment.

$$\text{Slope} = m_1 = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{Equation 3.4.16-1}$$

$$L: y = y_1 + m_1(x - x_1) \quad \text{Equation 3.4.16-2}$$

Find the equation for the line (L') through the point (x_3, y_3) which is perpendicular to L .

$$\text{The slope of } L' = -\frac{1}{m_1} = m_2 \quad \text{Equation 3.4.16-3}$$

$$L': y = y_3 + m_2(x - x_3) \quad \text{Equation 3.4.16-4}$$

Find the point where L' crosses L by solving their equations simultaneously.

$$x_{\text{int}} = \frac{y_3 - y_1 + m_1 x_1 - m_2 x_3}{m_1 - m_2} \quad \text{Equation 3.4.16-5}$$

$$y_{\text{int}} = m_1(x_{\text{int}} - x_1) + y_1 \quad \text{Equation 3.4.16-6}$$

Determine if $(x_{\text{int}}, y_{\text{int}})$ fall on the line segment.

In this function, if the line segment is not determined to be vertical or horizontal, there seems to be an inaccurate assumption that $(x_{\text{int}}, y_{\text{int}})$ will be on the line segment, between (x_1, y_1) and (x_2, y_2) , if one of the following equations are satisfied:

$$\begin{aligned} (x_1 - 1) &\leq x_{\text{int}} \leq (x_2 + 1) \\ \text{or} & \\ (x_2 - 1) &\leq x_{\text{int}} \leq (x_1 + 1) \end{aligned} \quad \text{Equation 3.4.16-7}$$

This assumption is incorrect anytime x_{int} is greater than the largest x value along the segment (either x_1 or x_2) or if it is less than the smallest x value along the segment.

However, if the function determines that $(x_{\text{int}}, y_{\text{int}})$ is indeed on the line segment then calculate the distance

$$d = \sqrt{(x_3 - x_{\text{int}})^2 + (y_3 - y_{\text{int}})^2} \quad \text{Equation 3.4.16-8}$$

If the function determines that $(x_{\text{int}}, y_{\text{int}})$ is not on the line segment, it will return the smaller distance between the point and either end-point of the line segment.

$$d = \min \left[\sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2} \text{ or } \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2} \right] \quad \text{Equation 3.4.16-9}$$

The function then returns d , the coordinates of the point $(x_{\text{int}}, y_{\text{int}})$, and a status indicator whether $(x_{\text{int}}, y_{\text{int}})$ is on the line segment.

This approach is closed form and correct except for the assumption made in Equation 3.4.16-7, which will need further explanation.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.16-1	Equation 3.4.16-7	Assumes a larger line segment in the x dimension. Appears to be incorrect.	Minor

3.4.17 Function: GM_REGN (PL/I)

In the x - y plane, this function determines if a test point (x_t, y_t) lies within a polygon region defined by the a set of boundary points $(x[n], y[n])$.

3.4.17.1 Description:

The function uses the PL/I random number generator to create a random number. The function uses this random number and the maximum and minimum x and y coordinates of the region to create a point outside the polygon region. The outside point is joined with the test point to form a line segment. The test line is checked for intersections against the segments defining the circumference of the polygon region. The function results in a number of intersections. The function makes n number of random number calls (currently 8 maximum) if the intersection point is too close to the end of a polygon segment. The maximum number of intersection checks is therefore n times the number of polygon vertices. If the number of intersections is even, the test point is outside the region. If the number of intersections is odd, the test point is inside the region.

Table of Variable Definitions

Function Variable	Description	Math Symbol
xmin, xmax	minimum and maximum x coordinates of the polygon region in feet	<i>xmin, xmax</i>
ymin, ymax	minimum and maximum y coordinates of the polygon region in feet	<i>ymin, ymax</i>
rv	random number value	<i>rv</i>
xr, yr	x, y coordinates of random generated point in feet	<i>xr, yr</i>
*x1, y1	x, y coordinates of the start point of the line (ft)	<i>x1, y1</i>
x2, y2	x, y coordinates of the end point of the line (ft)	<i>x2, y2</i>
xt, yt	x, y coordinates of the test point (ft)	<i>xt, yt</i>
xi, yi	x, y coordinates of the intersection point (ft)	<i>xi, yi</i>
p, t	ratio's returned by gm_insec	<i>p, t</i>
pton	point on line status; pton=1 point is on a line segment of the boundary region's polygon, pton=0 point is not on a line of this boundary	<i>pton</i>
istat	intersection status; istat=0 line segments intersect between endpoints	<i>istat</i>
i, k	loop counters	<i>i, k</i>
ptsep	constant used as delta separation allowed to consider a point on a line; currently set at 1 ft.; used for the TKM_GM_TSTPNT	<i>ptsep</i>
eps	epsilon value used as effective difference of zero; currently = 0.001	ϵ
nrpt	number of random point tries that function iterates	<i>nrpt</i>
n	number of polygon end points	<i>n</i>

3.4.17.2 Mathematics:

The function determines if a test point lies within a polygon region in the x-y plane defined by the arrays of boundary points (x[n], y[n]).

The first step in the function is to determine the minimum and maximum boundary distances. These extreme points are used to perform a gross check for the test point. If the test point lies outside the extreme points of the polygon, the function returns an outside the region result for the test point. However, if the test point is equal to or inside the extreme points, a random number is generated. The number is used to define a random point outside the polygon region by using the extreme points already defined.

The function generates the random number by the RAND function in an overall loop structure i from 1 to 8. The value of i is MOD by 4. Therefore, the result is expressed in Table 3.4.17-1 where each MOD value is carried out a maximum of twice.

Iteration number	MOD value	Resulting random point is generated
1	1	point above the region
2	2	point to the right of the region
3	3	point below the region
4	0	point to the left of the region
5	1	point above the region
6	2	point to the right of the region
7	3	point below the region
8	0	point to the left of the region

Table 3.4.17-1: Iteration key GM_REGN

For $i =$ equals 4 or 8, the random point is generated to the left of the polygon region. The expression to determine this point is :

$$xr = x \text{ min} - 1.005(x \text{ max} - x \text{ min}) \quad \text{Equation 3.4.17-1}$$

$$yr = y \text{ min} + rv(y \text{ max} - y \text{ min}) \quad \text{Equation 3.4.17-2}$$

For the x dimension, the point is placed 0.5% to the left of the length of the polygon. For the y dimension, the point is placed a uniform random variable distance within the width of the polygon.

For the other directions (i.e. to the right, below, and above), the calculation is performed analogously to Equation 3.4.17-1 and Equation 3.4.17-2.

After each random point is generated, the function runs a second loop for each segment of the polygon. For each segment, the GM_TSTPNT is called to determine if the test point is on the segment. If it is, the test point is considered inside the region and the function ends with an inside result. However, if the GM_TSTPNT determines the point is not on the line the test point is combined with the current random point to form a segment.

Now, GM_INSEC is called to determine if an intersection takes place between the test point to random point segment versus the polygon segment. A counter is incremented (*icnt*) for each intersection found. If an intersection is found, but the ratio of the distance to the end point of the polygon segment is less than ε or greater than $1 - \varepsilon$, then the loop ends without finishing the rest of the polygon segments and the next random point is generated. The *i* loop ends with either *nrpt* random points generated or a successful iteration through the *k* loop, where each polygon segment is checked.

The function ends by returning the MOD value of *icnt* by 2. This value will return a 0 or 1 for the number of intersections determined to be even or odd, respectively. If an odd number of intersections are found, the test point does lie inside the region. If the number of intersections is an even number, the test point lies outside the region. The flow chart in Figure 3.4.17-1 illustrates the logic for this function.

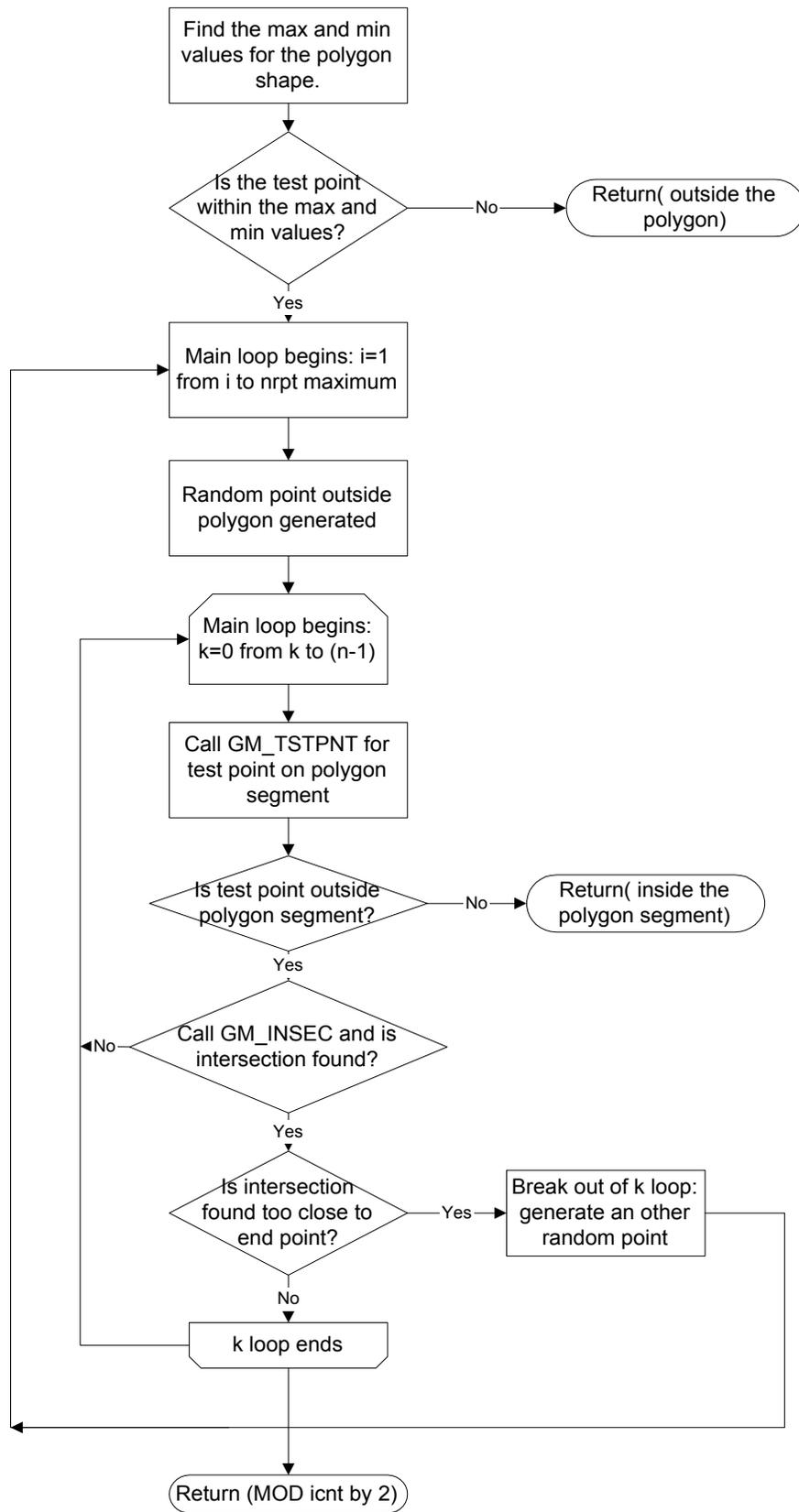


Figure 3.4.17-1: Logic of GM_REGN

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.4.17-1	The ϵ value assumes the value for the ratio returned by GM_INSEC is approximately 1 or 0. The value currently chosen is 0.001 which is very reasonable.	A reasonable choice for the parameter has been chosen and if the intersection point is effectively on the end point of the polygon line segment, an other random point is chosen (total of <i>nrpt</i> of them)	Minor
R 3.4.17-2	The choice of <i>nrpt</i> random point iterations seems reasonable, though if more are required the result will be to falsely determine the point is outside the region (return 0).	The number <i>nrpt</i> =8 chosen seems reasonable and can only be verified by unit testing.	Important
R 3.4.17-3	This function is subject to the critical and important approximations of TKM_GM_TSTPNT, since it uses this function to check if the test point lies on each polygon line segment.	Refer to the GM_TSTPNT function's Assessment Table. They will directly effect this function GM_REGN.	Critical

3.4.18 Function: GM_TSTPNT (PL/I)

This function determines if a point lies on a specified line. To lie on this line, the point may be a small epsilon distance from the line and still be considered on the line.

3.4.18.1 Description:

Given the x and y coordinates (in feet) of the end points of the line and the x and y coordinates of a point (in feet), the function first determines the location of an intersection point which forms a perpendicular line from the given point to the given line (refer to Figure 3.4.18-1). Next, the function determines the distance of this perpendicular line and compares it to the minimum epsilon distance. If the distance of the normal line is less than the minimum distance, the point is considered on the line.

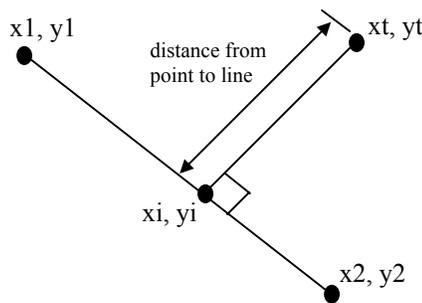


Figure 3.4.18-1: Diagram of test point to line distance

Table of Variable Definitions

Function Variable	Description	Math Symbol
x1, y1	x, y coordinates of the start point of the line (ft)	x_1, y_1
x2, y2	x, y coordinates of the end point of the line (ft)	x_2, y_2
xt, yt	x, y coordinates of the test point (ft)	x_t, y_t
pntsep	separation allowed between 2 points	$pntsep$
deltx	delta difference of the line in x dimension (ft)	$\delta.x$
dely	delta difference of the line in y dimension (ft)	$\delta.y$
s1	slope of the line from x_1, y_1 to x_2, y_2	m
s2	slope of the normal line from x_t, y_t to x_i, y_i	$-1/m$
xi, yi	x, y coordinates (ft) of intersection point of test point to line	x_i, y_i
eps	epsilon value for considering the line to vertical or horizontal (currently set at 100 feet)	ϵ
d	perpendicular distance (ft) from the test point to the line	d

3.4.18.2 Mathematics:

The function starts by calculating the delta differences of each dimension of the line. These variables include the following:

$$\delta.x = x_2 - x_1 \quad \text{Equation 3.4.18-1}$$

$$\delta.y = y_2 - y_1 \quad \text{Equation 3.4.18-2}$$

These deltas are used to determine if the line is a vertical line or horizontal line. For the x dimension, if the line's $\delta.x$ is less an ϵ value, consider the line to be a vertical line. The function checks if the test point is greater than the distance $pntsep$ in the x dimension and if so considers it not on the line. If the test point is less than the distance $pntsep$ in the x dimension and is within the y dimensions of the line, it is considered on the line. An analogous check is made for the y dimension.

Now, the line is not a vertical or horizontal line and the perpendicular distance will need to be calculated between the test point and the line. The first step is to determine the following slope equations:

$$s1 = m = \frac{\delta.y}{\delta.x} \quad \text{Equation 3.4.18-3}$$

$$s2 = -\frac{1}{m} = -\frac{\delta.x}{\delta.y} \quad \text{Equation 3.4.18-4}$$

The equation of the line is expressed for the given line and the line formed by drawing a perpendicular line from the test point to the line. By solving these two equations simultaneously for x and y, the resulting formulas give the x and y coordinates of the intersection point used in the function to solve for the distance d .

The given line:

$$y - y_1 = m(x - x_1) \quad \text{Equation 3.4.18-5}$$

The normal line from the test point to the line:

$$y - y_t = -\frac{1}{m}(x - x_t) \quad \text{Equation 3.4.18-6}$$

Solving them simultaneously for x (note the x below is equivalent to xi in the code):

$$\begin{aligned} m(x - x_1) + y_1 - y_t &= \left(-\frac{1}{m}\right)(x - x_t) \\ mx - mx_1 + \frac{x}{m} - \frac{x_t}{m} &= y_t - y_1 \\ x\left(m + \frac{1}{m}\right) &= y_t - y_1 + \frac{x_t}{m} + x_1m \\ x &= \frac{y_t - y_1 + \frac{x_t}{m} + x_1m}{\left(m + \frac{1}{m}\right)} \end{aligned} \quad \text{Equation 3.4.18-7}$$

Now, solve for y (or yi in the code) for the intersection point using the x value in Equation 3.4.18-7 and use Equation 3.4.18-5 to solve for y.

The last check determines the distance of the intersection point using the general distance formula:

$$d^2 = \sqrt{(x_t - x)^2 + (y_t - y)^2} \quad \text{Equation 3.4.18-8}$$

The function proceeds by checking this distance d against the *pntsep* distance, and if this distance is greater than the *pntsep* distance the test point is evaluated as not on the given line. However, if the distance is less than *pntsep*, the test point is checked to determine if it is between the end points of the line. For example, this checks for cases when the distance in Equation 3.4.18-8 is zero because the test point is collinear with the given line, but not within the line segment and could actually be a large distance from the endpoints of the line. (NOTE: To determine if the test point is within the line segment, the function extends the line by the *pntsep* value, currently 1 foot.)

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.18-1	If the test point is less than a distance <i>pntsep</i> from the given line, the point is evaluated to be between the end points of the given line segment. However, the line is extended by <i>pntsep</i> for the PL/I version and by ϵ for the TKM version of GM_TSTPNT in C. The <i>pntsep</i> value is 1 foot and the ϵ value is 100 feet.	The transfer from C to PL/I will provide different results not because of coding in a different language, but because different comparison values are used. An investigation into the potential reasons for the change are necessary and should be documented.	Important
R 3.4.18-2	The check carried out to determine if a point is between the end points of the line segment when the line segment is either vertical or horizontal uses the <i>pntsep</i> value to extend the lines under the PL/I version but not for the C version here.	It is actually more accurate not to use the <i>pntsep</i> value, but this may cause errors due to round off during floating point arithmetic. Therefore, an investigation is required to determine why this was not used in the C version and used in this PL/I version only.	Important

3.4.19 Function: GM_TURN (PL/I)

Determines the turn angle to go from one bearing to another.

3.4.19.1 Description:

This function will take the difference between an initial bearing and a final bearing, supplied as inputs, and return the turn angle, where right turns are positive and left turns are negative.

Table of Variable Definitions

Function Variable	Description	Math Symbol
INITIAL_BRNG	The initial bearing, in radians, from north	Ψ_i
FINAL_BRNG	The final bearing, in radians, from north	Ψ_f
TURN_ANGLE	The turn angle, in radians. Right turns are positive, left turns are negative	ϕ

3.4.19.2 Mathematics:

The function performs the following simple calculations and logic.

First it determines the difference between the two bearings.

$$\phi = \Psi_f - \Psi_i \qquad \text{Equation 3.4.19-1}$$

The function then limits the range to be between -2π and 2π by using the MOD function

$$\phi = \text{sign}(\phi) \text{MOD}(|\phi|, 2\pi) \quad \text{Equation 3.4.19-2}$$

It then guarantees that any angle in the first or second quadrant is positive. Conversely, it ensures any angle in the third or fourth quadrant is negative. For instance, if the angle was greater than π , the function subtracts 2π , resulting in a negative (left turn). If the angle was less than $-\pi$, the function adds 2π , resulting in a positive (right turn).

The value for turn angle, in radians, is returned from this function. There are no assumptions or approximations of significance to be noted.

3.4.20 Function: LO_FIND (PL/I)

Finds the index of a specified name in a specified table.

3.4.20.1 Description:

This function finds the index of a specified name in a specified database table.

The following is a list of the database tables which can currently be searched using this function. This list includes the database name, description, and the name of the subfunction, if any, which is invoked to search the particular database table.

Database	Description	Calling Sub-Function
ACD	Aircraft Class Data	LO_BSACD
ALINE	A-Line preferred arrival routes	LO_BSALINE
APT	Airports	LO_APT
BAS	Blocked Airspace	DB_LOC_FIRST
DLINE	D-Line preferred departure routes	LO_BSDLINE
NODE	Fixes	LO_BSFIX
RTE	Routes	LO_BSRTE
SATAPT	Satellite Airports	LO_BSSATAPT
SEC	Sectorization Data	

All of the “LO_” subdirectories use a binary search technique to speed the search.

3.4.21 Function: ST_ARD_SSGDATA (PL/I)

Finds various SSG values (time, x, y, altitude, ground speed, true airspeed, pointer to the SSG) for a given ARD.

3.4.21.1 Description:

Given an ARD, this function will return the time, position data (x, y), altitude, ground speed, and true airspeed from the SSG which encompasses the given ARD. There is a check to ensure the ARD does not go beyond the total length of the route. Otherwise, the above values are interpolated (non-linearly in the case of accelerations) from the start and end point data of the SSG. Since aircraft which are in a hold are assigned a value of zero for their segment length, hold segments will not be considered.

Table of Variable Definitions

Function Variable	Description	Math Symbol
X, Y	Coordinates of the aircraft at the given ARD (ft)	x, y
SSG.X(1), SSG.Y(1), SSG.X(2), SSG.Y(2)	Coordinates of the start and end points of the state segment (ft)	x_1, y_1 x_2, y_2
ALT	Altitude of the aircraft at the given ARD (ft)	z
SSG.Z(1), SSG.Z(2)	Altitude of the aircraft at the start and end points of the state segment (ft)	z_1, z_2
GSPD_ACC	Ground Speed acceleration (ft/s/s)	a_g
TAS_ACC	Aircraft true airspeed acceleration (ft/s/s)	a_t
GSPD	Ground speed of the aircraft at the given ARD (ft/s)	V_g
SSG.GSPD(1), SSG.GSPD(2)	Ground speed of the aircraft at the start and end points of the state segment (ft/s)	V_{g1}, V_{g2}
TSPD	True airspeed of the aircraft at the given ARD (ft/s)	V_t
SSG.TSPD(1), SSG.TSPD(2)	True airspeed of the aircraft at the start and end points of the state segment (ft/s)	V_{t1}, V_{t2}
XTIME	The time the aircraft will be at the given ARD (seconds)	t
SSG.TIME(1), SSG.TIME(2)	Time associated with the start and end points of the state segment (seconds)	t_1, t_2
SSG.SEG_LNG	The length of the state segment (ft)	ℓ
SSG.ARD	Along Route Distance (ft)	ard
SSG.TOTDIST	ARD at the beginning of the SSG (ft)	ard_1
SSG.GRAIENT	The gradient value of the SSG (ft/ft)	g

3.4.21.2 Mathematics:

The function performs the following simple calculations and logic.

First it determines which SSG intervals includes the given ARD.

$$ard < ard_1 + \ell \quad \text{Equation 3.4.21-1}$$

The first SSG that satisfies the above condition will then be used in the following processing.

Initially the function finds the ratio of the length the aircraft traveled from the start of the SSG to the ARD over the total length of the SSG

$$r = \frac{ard - ard_1}{\ell} \quad \text{Equation 3.4.21-2}$$

The function uses this ratio to interpolate the x and y coordinates at the ARD position from the SSG endpoints.

$$x = x_1 + r(x_2 - x_1) \quad \text{Equation 3.4.21-3}$$

$$y = y_1 + r(y_2 - y_1) \quad \text{Equation 3.4.21-4}$$

Next, the function supplies the ST_XYTOTIME function (See 3.4.34) with the x, y coordinates and SSG pointer to determine the time, t , at the given ARD position.

Altitude is simply determined by multiplying the gradient by the increment of horizontal distance traveled from the start of the SSG to the ARD and then added to the altitude at the start of the SSG.

$$z = z_1 + g(ard - ard_1) \quad \text{Equation 3.4.21-5}$$

Finally true airspeed and ground speed are calculated. First the function determines the accelerations in true airspeed or ground speed by finding the difference of these values at the start and end points of the SSG and dividing by the SSG time interval.

$$a_g = \frac{V_{g2} - V_{g1}}{t_2 - t_1} \quad \text{Equation 3.4.21-6}$$

$$a_t = \frac{V_{t2} - V_{t1}}{t_2 - t_1} \quad \text{Equation 3.4.21-7}$$

The function uses these values in a simple kinematic equation of motion to determine the current velocities at the ARD position.

$$V_g = V_{g1} + a_g(t - t_1) \quad \text{Equation 3.4.21-8}$$

$$V_t = V_{t1} + a_t(t - t_1) \quad \text{Equation 3.4.21-9}$$

The values for x, y, z, V_g, V_t, t and a pointer to the associated SSG are returned from this function.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.21-1	Equation 3.4.21-6 and Equation 3.4.21-7 assume a constant acceleration over the entire length of the segment.	This assumption could cause accuracy problems when complex accelerations and variable winds are present	Important

3.4.22 Function: ST_CHK_VP (PL/I)

This function determines any entry and exit points of an aircraft into the airspace defined in the VP_IN structure given the aircraft's trajectory represented by SSG data structure.

3.4.22.1 Description:

The function first determines if the first trajectory segment is inside the airspace volume, and if it is indeed inside, marks it as the entry point into the airspace. The next step is to enter a loop on the trajectory state segments. Next, the function proceeds into the two filters. First, it enters the gross filter to determine if each segment potentially is inside the airspace volume. If the state segment passes the gross filter, it enters the fine filter to determine the specific intersection coordinates.

- *GROSS FILTER TEST*: The gross filter part of this function first assigns the minimum and maximum boundary points of the airspace from the VP_IN structure. It also assigns the minimum and maximum end point dimensions of the particular iteration's state segment. The next step is to compare each dimension's (i.e. x, y, z) minimum and maximum boundaries for overlap. If the state segment has overlap with the airspace, it proceeds in the function to the next filter. If the state segment does not have any overlap, the loop proceeds to the next state segment.
- *FINE FILTER TEST*: The fine filter part of this function loops through the airspace segments and calls GM_INSEC to determine if the flight state segment intersects with one or more of the airspace segments. The fine filter determines if there are any intersections with the horizontal dimensions in the x-y plane, then determines if there are intersections with the top of the volume (maximum altitude), and finally determines if there are intersections with the bottom of the volume

Following the fine filter, the last segment is checked and flagged if inside the airspace volume. The records in the VP_OUT structure which contain the entry and exit points of the aircraft into the airspace are sorted by time. Any duplicate intersection points are eliminated as well.

Table of Variable Definitions

Function Variable	Description	Math Symbol
VP_IN	Structure data variable which contains the pre-calculated information on the airspace volume's boundary points	VP_IN
VP_OUT	Structure data variable which is filled with entry and exit information into the given airspace	VP_OUT
VP_MAX_PTS	Maximum and minimum coordinates of the airspace volume	$maxz, minz, maxx, minx, maxy, miny$
SSG.X(1), SSG.Y(1), SSG.X(2), SSG.Y(2)	Coordinates of the start and end points of the state segment (ft)	x_1, y_1 x_2, y_2
SSG.Z(1), SSG.Z(2)	Altitude of the aircraft at the start and end points of the state segment (ft)	z_1, z_2
XI, YI, ZI	Temporary variables for the coordinates of the intersection point	xi, yi, zi
MIN_SSG_X	Minimum x distance (ft) within the given state segment	min_ssgx
MAX_SSG_X	Maximum x distance (ft) within the given state segment	max_ssgx
SSG_RATIO	Ratio along the state segment that marks the horizontal intersection point	ssg_ratio
RATIO	Ratio for the distance in the vertical dimension of the determined altitude intersection used in Fine Filter to interpolate for the corresponding x and y coordinates	$ratio$
REGN_IND	Flag variable used as output of the GM_REGN function where 0 = outside, 1 = inside the airspace	$regn_ind$
SSG.TIME(1)	Time associated with the start point of the state segment (seconds)	ti
SSG.SEG_LNG	The length of the state segment (ft)	ℓ
SSG.ARD	Along Route Distance (ft)	ard
SSG.TOTDIST	ARD at the beginning of the SSG (ft)	ard_1

3.4.22.2 Mathematics:

Pre-Filter Processing

The function starts by initializing a few variables and then checking to see if the first state segment is inside the airspace volume. This step calls GM_REGN to determine if the point is inside the airspace volume. If this point is within the volume, the VP_OUT structure is updated with this point and defined as an END_IN_AIRSPACE condition. The next step is the main loop which starts the iteration through the state segments of the aircraft trajectory (*The current function only examines linear segments not segments associated with a hold called box state segments.*).

Gross Filter

Now, the function starts the check for an intersection with the volume. Any intersections found in the gross filter or fine filter algorithms are defined by the CROSSING condition. First, the flag variable FILTER is initialized to zero. There are three checks in the Gross Filter algorithm, one for each dimension x, y, and z. For example, the check for the x dimension first defines the minimum and maximum x value for the state segment, min_ssgx and max_ssgx . These values are compared to the airspace volumes minimum and maximum x values and if the check is true the

FILTER variable is incremented by 1. The check is as follows:

$$NOT[(\max_ssgx < \min x) \text{ or } (\max x < \min_ssgx)] \quad \text{Equation 3.4.22-1}$$

The same comparison is made in the y and z dimensions and the FILTER variable is incremented accordingly.

Fine Filter

1. Check for horizontal intersections with the sides of the airspace volume.

If the state segment passed the Gross Filter with the FILTER variable resulting in 3, the Fine Filter begins with a loop to check for horizontal intersections with the state segment and each segment of the airspace volume. It calls the GM_INSEC function to determine if and where a horizontal (x, y) intersection takes place between the trajectory state segment and the airspace segment. If an intersection does take place, the GM_INSEC function provides the ratio along the trajectory state segment. This ratio is used to interpolate in the altitude dimension (z), as follows:

$$z_i = z_1 + ssg_ratio(z_2 - z_1) \quad \text{Equation 3.4.22-2}$$

Therefore, z_i is the altitude that the state segment intersects a segment of the airspace volume. The z_i altitude is checked to be within the vertical limits of the airspace volume and if so, the intersection is stored in the VP_OUT structure. The function ST_FINDARD is called to determine the along route distance of the intersection point and ST_ARD_SSGDATA is called to find the time of the intersection based on the along route distance.

2. Check for an intersection with the top of the airspace volume.

If the end point altitudes of the trajectory state segment intersect the top of the airspace volume's altitude ($maxz$), the altitude z_1 must be greater than $maxz$ while the z_2 altitude must be less than $maxz$ or altitude z_2 must be greater than $maxz$ and the z_1 altitude must be less than $maxz$. The expression for the top intersection check is as follows:

$$[(z_1 > \max z) \text{ and } (z_2 \leq \max z)] \text{ or } [(z_2 > \max z) \text{ and } (z_1 \leq \max z)] \quad \text{Equation 3.4.22-3}$$

The ratio of the distance between z_1 and the $maxz$ to the altitude range of the state segment is calculated next. The equation for this ratio is presented in Equation 3.4.22-4.

$$ratio = (\max z - z_1) / (z_2 - z_1) \quad \text{Equation 3.4.22-4}$$

The ratio is used to interpolate for the x and y coordinates of the altitude intersection. The two linear equations are solved for the x_i and y_i coordinates of the altitude intersection and z_i is set to $maxz$.

$$x_i = x_1 + ratio(x_2 - x_1) \quad \text{Equation 3.4.22-5}$$

$$y_i = y_1 + ratio(y_2 - y_1) \quad \text{Equation 3.4.22-6}$$

The function GM_REGN is called to determine if the altitude intersection is within the airspace boundary in the x-y plane. If it is, the along route distance (ARD) and time are calculated by ST_FINDARD and ST_ARD_SSGDATA. Finally, the ARD, time, and the intersection coordinates are stored in the VP_OUT structure.

3. Check for an intersection with the bottom of the airspace volume.

For the bottom intersection, the function is calculated in the same manner as the top intersection was calculated. The check and interpolation equations are analogous to the top intersection and listed here only for reference.

$$\left[(z_1 \geq \min z) \text{ and } (z_2 < \min z) \right] \text{ or } \left[(z_2 \geq \min z) \text{ and } (z_1 < \min z) \right] \quad \text{Equation 3.4.22-7}$$

$$\text{ratio} = (\min z - z_1) / (z_2 - z_1) \quad \text{Equation 3.4.22-8}$$

Post Filter Processing

1. Following the Fine Filter test, the last end point of the trajectory is checked, using GM_REGN, to determine whether it is inside or outside the airspace volume. If it is inside, the state segment end point information is stored in the VP_OUT structure.
2. The next step is the sort loop (Bubble Sort technique) that sorts the intersection points in ascending order of time.
3. Next, the intersection points are examined for duplicates and if found they are eliminated from the list. Both points must be category Crossing, the X and Y dimensions must both be within 1000 feet of each other, and the intersection must occur within 10 seconds of each other.
4. If the trajectory begins inside the airspace and has a crossing point near this entry point (X and Y within 1000 feet and within 10 seconds time of the intersections) and there are more than two intersection points, eliminate these two points. In other words, eliminate the beginning entry point and crossing point if there are more than two points in the VP_OUT structure and are close to the boundary.
5. If the trajectory ends inside the airspace and there exists a crossing point that is close to the end point (within 1000 feet horizontally and 10 seconds in time) and there is greater than or equal to 2 points in VP_OUT, these two points are eliminated. In other words, eliminate a pair of near intersections if they are at the end of the trajectory and are very close to the boundary.
6. The next loop defines the type field of the VP_OUT structure for each intersection point. The type field defines whether each intersection is associated with an entry or an exit (the VOL_ENTRY and VOL_EXIT fields).
7. If there is only one intersection point (a crossing point), it must be a point of tangency so the point is eliminated.
8. A final check is made for other points of tangency (crossing points only). If the function determines an odd number of crossing intersections, it determines if the trajectory is within 10 seconds before the crossing point and outside the boundary 10 seconds after the crossing point occurs. The ST_TIME_SSGDATA function is called to determine the coordinates along the trajectory for the specific time. The GM_REGN function determines if these points (trajectory location 10 seconds before and after the intersection) are within the airspace volume. If the point does not meet the conditions of being inside and outside the airspace within the 10 second window, the crossing intersection point is eliminated from the VP_OUT structure, since it must be a point of tangency.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD
R 3.4.22-1	Linear interpolation is used to estimate the specific coordinates of each intersection point.	The solution of a given set of coordinates based on the end point coordinates and the known distance of one dimension (X, Y, or Z) is a reasonable approximation assuming the segment is small and the aircraft is only accelerating moderately.	Important
R 3.4.22-2	A small parameter distance (1000 feet horizontally and 10 seconds in time) is used to assume a pair of intersection points are alike.	The small parameter distance allows an aircraft to be considered outside the airspace if it is inside by a distance smaller than the parameter distance. The 1000 feet / 10 seconds seem like reasonable choices for an epsilon value.	Important
R 3.4.22-3	A global variable should be used as the small parameter distance.	The small parameter value 1000 feet / 10 seconds should be defined by a global variable to be consistent with other algorithms and improve the readability of the code.	Minor

3.4.23 Function: ST_CLIMB_DIST (PL/I)

Computes the distance required for a climb

3.4.23.1 Description:

This function calculates the horizontal distance required for a climb without the effect of wind.

Table of Variable Definitions

Function Variable	Description	Math Symbol
NEXT_ALT	The lower level of the altitude layer (ft)	h_1
ALT	The current altitude of the aircraft (ft)	h_0
TARGET_ALT	The target altitude for the descent (ft)	h_t
GRADIENT	The ratio of the change in altitude over the distance traveled in the horizontal plane. (ft/ft)	g_r
CLIMB_GRAD_FACTOR	A multiplier which effects the climb gradient. This value is found from the AMC table for a particular aircraft.	C_{GF}
DIST	The total horizontal distance the aircraft would fly in still air (ft)	D_{tot}

3.4.23.2 Mathematics:

This function uses Equation 3.4.23-2 to compute the horizontal distance an aircraft would be required to travel to complete a given climb. In the Aircraft Control Characteristics (ACC) table each aircraft has various altitude layers and corresponding climb gradients assigned to each layer. Therefore, this function must calculate the horizontal distance required for each altitude layer the aircraft climbs through, and aggregate these distances into a total.

The method used is described below. Since the method is simple in form, no derivations were needed for this description.

The process begins at the current altitude and loops through every altitude layer until the target altitude is reached. At each altitude layer, the function calls ST_CLIMB_GRADIENT (see

Section 3.4.24) to get the assigned altitude gradient, g_r . If the higher level of the altitude layer, h_1 , is higher than the target altitude, h_t , then assign

$$h_1 = h_t \quad \text{Equation 3.4.23-1}$$

Add the distance required for the aircraft to climb through the current climb layer

$$D_{tot} = D_{prev} + \frac{(h_1 - h_0)}{C_{GF}} \frac{1}{g_r} \quad \text{Equation 3.4.23-2}$$

where C_{GF} is the climb gradient factor found in the Aircraft Modeling Characteristics (AMC) table for the particular aircraft.

This process continues to “loop” through every altitude layer between the starting altitude and the target altitude, keeping track of the total horizontal distance traveled by the aircraft. At the end of each iteration, the current altitude, h_0 , is assigned the value of the higher level of the current altitude layer, h_1 (i.e. let $h_0 = h_1$, then the next loop starts with the next altitude layer).

This function’s accuracy completely depends on the accuracy of the climb gradients and the climb gradient factor. The accuracy of these values and their impact on TJM will need to be studied, tested and proven.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.23-1	The accuracy of the altitude layer climb gradients supplied by ST_DESCENT_GRADIENT.	This issue must be verified to be as close to “typical” as possible to minimize reconformances. Justification of these values may fall under an analysis of the Aircraft Characteristics and Adaptation subsystems	Critical
R 3.4.23-2	The accuracy of the aircraft’s climb gradient factor supplied by the AMC table	This issue must be verified to be as close to “typical” as possible to minimize reconformances. Justification of these values may fall under an analysis of the Aircraft Characteristics and Adaptation subsystems	Critical

3.4.24 Function: ST_CLIMB_GRADIENT (PL/I)

Computes the climb gradient and IAS (or Mach) for a given aircraft type, altitude, engine, temperature and aircraft weight, assuming zero wind.

3.4.24.1 Description:

This function simply searches the ACC table for the climb gradient and IAS (or Mach) which corresponds to the given altitude and temperature and the aircraft type, engine, and weight. The processing begins by finding the tables which are associated with the given aircraft engine type. Then it searches for the table with the temperatures and weights which represent the given input temperature and weight. If the input values fall between two table values, then the function uses linear interpolation to calculate the climb gradient and airspeed from the two closest table values.

In the current URET version (D1.1), each aircraft type is given a default value for weight found in the AMC table. This default weight (or effective weight) is the simple average of the Nominal Takeoff Weight and the Nominal Landing Weight found in the ACD data table.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.24-1	The accuracy of the altitude layer climb gradients and the IAS (or Mach) in the ACC table.	This issue must be verified to be as close to “typical” as possible to minimize reconfirmances. Justification of these values may fall under an analysis of the Aircraft Characteristics and Adaptation subsystems	Critical

3.4.25 Function: ST_DESCENT_DIST (PL/I)

Computes the distance required for a descent

3.4.25.1 Description:

This function calculates the distance required for a descent. If the aircraft descends through a parameter altitude which has an associated speed limit restriction (for example, the 250 knot indicated airspeed (IAS) speed limit for aircraft flying below 10000ft, *CFR 91.117*) the distance traveled during the deceleration is also calculated.

Table of Variable Definitions

Function Variable	Description	Math Symbol
NEXT_ALT	The lower level of the altitude layer (ft)	h_1
ALT	The current altitude of the aircraft (ft)	h_0
TARGET_ALT	The target altitude for the descent (ft)	h_t
DESCENT_IASMACH	The current aircraft indicated airspeed (kts)	V_{ias}
DESCENT_TAS	The current aircraft true airspeed (ft/s)	V_t
SPEED_LIMIT	The altitude speed limit (in indicated airspeed, kts)	V_{sl}
SPEED_LIMIT_TAS	The altitude speed limit (in true airspeed, ft/s)	$V_{t\ sl}$
DECEL_TIME	The time needed for deceleration (secs)	T
IDLE_DECEL	The idle deceleration rate for the aircraft	$a_{i\ dec}$
DECEL_DIST	The horizontal distance the aircraft traveled during the deceleration (ft)	D_{decel}
GRADIENT	The ratio of the change in altitude over the distance traveled in the horizontal plane. (ft/ft)	g_r
DESCENT_GRAD_FACTOR	A multiplier which affects the descent gradient. This value is found from the AMC table for a particular aircraft.	C_{gf}
DIST	The total horizontal distance the aircraft would fly in still air (ft)	D_{tot}

3.4.25.2 Mathematics:

This function uses Equation 3.4.25-6 to compute the horizontal distance an aircraft would be required to travel to complete a given descent. In the ACC table each aircraft has various altitude layers and corresponding descent gradients assigned to each layer. Therefore, this function must calculate the horizontal distance required for each altitude layer the aircraft descends through, and aggregate these distances into a total.

This function also takes into account the horizontal distance an aircraft travels while it decelerates to a speed limit at or below a parameter speed limit altitude (Equation 3.4.25-2 to Equation 3.4.25-5).

The method used is described below. Since the method is simple in form, no derivations were needed for this description.

The process begins at the current altitude and loops through every altitude layer until the target altitude is reached. At each altitude layer, the function calls ST_DESCENT_GRADIENT (see Section 3.4.26) to get the assigned altitude gradient, g_r . If the lower level of the altitude layer, h_1 , is lower than the target altitude, h_t , then assign

$$h_1 = h_t \qquad \text{Equation 3.4.25-1}$$

3.4.25.2.1 Calculate the Distance Traveled During Deceleration

The function determines if the aircraft crosses the speed limit altitude (10000 ft) and its current speed exceeds the speed limit (250 kts)¹¹

$$\begin{aligned} &\text{If} && h_0 \geq 10000 \\ &\text{and} && h_1 < 10000 \\ &&& \text{and } V_{ias} > 250 \end{aligned} \quad \text{Equation 3.4.25-2}$$

then, convert the IAS and speed limit to True Airspeed (V_t and V_{t_sl}) at that altitude, using (CNV_CNVSPD, see Section 3.4.1) and calculate the time needed for the deceleration.

$$T = \frac{(V_t - V_{t_sl})}{|a_{i_dec}|} \quad \text{Equation 3.4.25-3}$$

where $|a_{i_dec}|$ is the absolute value of the idle deceleration rate found in the ACC table for each particular aircraft type.

Next, calculate the horizontal distance, D_{decel} , the aircraft traveled during the deceleration by finding the average deceleration speed over the deceleration time

$$D_{decel} = \left(\frac{V_t + V_{t_sl}}{2} \right) \times T \quad \text{Equation 3.4.25-4}$$

and add this result to the total horizontal distance traveled

$$D_{tot} = D_{prev} + D_{decel} \quad \text{Equation 3.4.25-5}$$

where D_{tot} is the total horizontal distance traveled up to and including this point and D_{prev} is the total horizontal distance traveled previous to this point.

3.4.25.2.2 Calculate the Total Distance Traveled

Add the distance required for the aircraft to descend through the current descent layer

$$D_{tot} = D_{prev} + \frac{(h_0 - h_1)}{C_{GF}} \frac{1}{|g_r|} \quad \text{Equation 3.4.25-6}$$

where C_{GF} is the descent gradient factor found in the AMC table for the particular aircraft.

This process continues to “loop” though every altitude layer between the starting altitude and the target altitude, keeping track of the total horizontal distance traveled by the aircraft. At the end of

¹¹ This restriction is based on the Code of Federal Regulations, Title 14, part 91.117, May 1996.

each iteration, the current altitude, h_0 , is assigned the value of the lower level of the current altitude layer, h_1 (i.e. let $h_0 = h_1$, then the next loop starts with the next altitude layer).

This function's accuracy completely depends on the accuracy of the descent gradients, the descent gradient factor, and the estimation of the aircraft's idle deceleration rate. The accuracy of these values and their impact on TJM will need to be studied, tested and proven.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.25-1	The accuracy of the altitude layer descent gradients and the aircraft's idle deceleration rate supplied by ST_DESCENT_GRADIENT and the ACC table.	This issue must be verified to be as close to "typical" as possible to minimize reconfirmances. Justification of these values may fall under an analysis of the Aircraft Characteristics and Adaptation subsystems	Critical
R 3.4.25-2	The accuracy of the aircraft's descent gradient factor supplied by the AMC table	This issue must be verified to be as close to "typical" as possible to minimize reconfirmances. Justification of these values may fall under an analysis of the Aircraft Characteristics and Adaptation subsystems	Critical

3.4.26 Function: ST_DESCENT_GRADIENT (PL/I)

Computes the descent gradient and IAS (or Mach) for a given aircraft type, altitude and aircraft weight assuming zero wind.

3.4.26.1 Description:

This function simply searches the ACC table for the descent gradient and IAS (or Mach) which corresponds to the given altitude and the aircraft type and weight. If the weight of the aircraft is a value which falls between two given weight values in the table, this function will linearly interpolate values for the descent gradient and IAS (or Mach) using these two weights.

In the current URET version (D1.1), each aircraft type is given a default value for weight found in the AMC table. This default weight (or effective weight) is the simple average of the Nominal Takeoff Weight and the Nominal Landing Weight found in the ACD data table.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.26-1	The accuracy of the altitude layer descent gradients and the IAS (or Mach) in the ACC table.	This issue must be verified to be as close to “typical” as possible to minimize reconfirmances. Justification of these values may fall under an analysis of the Aircraft Characteristics and Adaptation subsystems	Critical

3.4.27 Function: ST_FINDARD (PL/I)

This function computes the ARD of a given point in the x and y plane for a specific aircraft trajectory. The function uses as input the minimum and maximum ARD that the given point should be along the trajectory.

3.4.27.1 Description:

Given the x and y coordinates of a given point and the minimum and maximum ARD for the specified aircraft trajectory, the function will compute the ARD of the aircraft to reach this point. The function uses a minimum and maximum parameter set to start and end the search. This prevents errors due to a circular route in which the route may be at a particular point several times during the route and thus have multiple locations at the given point.

Table of Variable Definitions

Function Variable	Description	Math Symbol
XX, YY	Coordinates of the given point (ft)	<i>xx, yy</i>
SSG.X(1), SSG.Y(1), SSG.X(2), SSG.Y(2)	Coordinates of the start and end points of the state segment (ft)	<i>x₁, y₁</i> <i>x₂, y₂</i>
MIN_ARD	Minimum ARD the given point can result in	<i>min_ard</i>
MAX_ARD	Maximum ARD the given point can result in	<i>max_ard</i>
AUD_PTR	Pointer to aircraft’s unique data structure	<i>aud_ptr</i>
ARD	Along route distance of given point	<i>ard</i>
DARD	Distance from point to closest state segment (or ORS) in the x and y dimensions	<i>dard</i>
D	Distance from point to current ORS	<i>d</i>
XI, YI	Coordinates in x and y dimensions of point of closest approach (returned from GM_PTLINE function)	<i>xi, yi</i>
PTLINE_STAT	Status of point returned by GM_PTLINE function which returns 1 if the point is on the segment and 0 otherwise	<i>pt_status</i>

3.4.27.2 Mathematics:

The function searches the ORS structure of the specific aircraft for the ARD for a given point. The function starts by initializing a few flag variables and starting the main function loop which increments through the segment list. The next step checks the current ORS ARD’s against the minimum and maximum ARD’s (*min_ard* and *max_ard*). If the current ORS is not within the limits, the function iterates to the next ORS.

If current ORS ARD's are within the limits, the function first assigns the end point coordinate variables (i.e. x_1, y_1, x_2, y_2). The next step is a call to the GM_PTLINE function which returns the closest distance of the point to the line and the status (i.e. pt_status) of whether the closest approach point is on the segment or not. It also returns the coordinates of the closest approach point. If this closest approach point is on the segment, the function increments the ard distance as follows:

$$ard = ors.accum_dist + \sqrt{(xi - x_1)^2 + (yi - y_1)^2} \quad \text{Equation 3.4.27-1}$$

The Equation 3.4.27-1 adds the ORS current ARD to the distance from the segments beginning point of closest approach point. Under this case, the closest approach point is the point of intersection of a line from the given point and perpendicular to the flight segment line. For a situation where the given point is on the line segment, the closest approach point is equivalent to the given point's coordinates (i.e. $xx = xi$ and $yy = yi$), and the distance d is 0. This is illustrated in the Figure 3.4.27-1.

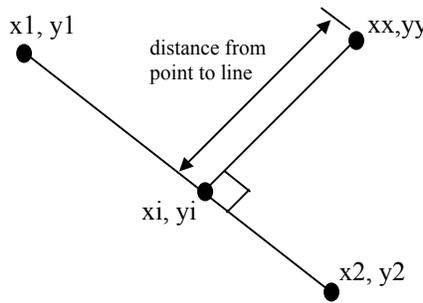


Figure 3.4.27-1: Diagram of closest approach point (xi, yi) which is on the flight segment

If the closest approach point is not on the line segment but somewhere on the line, the distance from each end point (i.e. x_1, y_1 and x_2, y_2) is calculated to determine the minimum distance between the given point and line. The following equations are calculated and the minimum $dard$ chosen:

Minimum of:

$$dard = \sqrt{((xx - x_1)^2 + (yy - y_1)^2)} \quad \text{Equation 3.4.27-2}$$

$$dard = \sqrt{((xx - x_2)^2 + (yy - y_2)^2)} \quad \text{Equation 3.4.27-3}$$

For the equation of $dard$ for the first end point, the function will calculate the ard as the $ors.accum_dist$ for that flight segment. For the $dard$ for the second end point, the function simply adds the length of the flight segment to the $ors.accum_dist$.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on APD and TJM
R 3.4.27-1	For the second loop where the GM_PTLINE found the closest approach point not on the flight segment, the distance calculation for the first end point has an error. The second term for the y dimension of the first end point in the code is listed as y_2 where it should be y_1 .	The result of this error could return an incorrect ARD by as much as one segment length, since the minimum and maximum ARD values restricts the search.	Critical

3.4.28 Function: ST_IASALT (PL/I)

Iteratively searches for the altitude at which an aircraft, which is accelerating and either climbing or descending, attains a desired IAS.

3.4.28.1 Description:

This function performs an iterative search to determine the altitude an aircraft will be when it attains a desired IAS. The acceleration supplied to the function describes the TAS acceleration. This acceleration is calculated from both the change in altitude and the aircraft's speed change.

Since the function is attempting to capture a desired IAS, there is no closed form solution to solving the motion equations as long as the acceleration is with respect to TAS. Instead, this function logically searches for the desired IAS and altitude by trying to surround the solution with a narrowing time interval.

Table of Variable Definitions

Function Variable	Description	Math Symbol
ACCEL	The aircraft's acceleration due to both a level cruise acceleration and a TAS acceleration due to the change in altitude. This is given in units of (ft/s ²)	a
EXTREME_TAS	The maximum TAS possible with the given target IAS. In this case, the corresponding TAS associated with a target IAS at the maximum altitude possible (MAX_ALTITUDE = 60000ft)	V_{t_extr}
TMAX, TMIN	Iterative time variables used to locate the time the aircraft captures the target IAS	t_{max}, t_{min}
TT	The test time which the function tries in determining when the aircraft captures the target IAS. This is calculated as the average of t_{max} and t_{min}	t
TEST_IAS	The resultant IAS that the function calculates from the given iteration	V_{i_test}
TARGET_IAS	The desired IAS	V_i
TEST_HT	The test altitude that the function tries in determining where the aircraft captures the target IAS	h_t
TEST_TAS	The test TAS that the function tries in determining where the aircraft captures the target IAS	V_{t_test}
IAS_EPSILON	A small parameter value used to determine if two IAS values are close (currently set to 1)	ϵ
CURRENT_Z	The current altitude of the aircraft (ft)	h
CURRENT_TAS	The current TAS at the current altitude	V_t
GRADIENT	The altitude gradient. This value is the ratio of the change in altitude over the change in horizontal distance traveled. (ft/ft)	g

3.4.28.2 Mathematics:

The function begins by determining if the aircraft is accelerating, decelerating, or has a constant true airspeed (TAS). If the aircraft is accelerating, it is known that the final TAS should be larger than the current TAS. The function must ensure that the estimated time interval includes the solution. Therefore, the function calculates the amount of time it would take for the aircraft to accelerate from its current TAS to the largest TAS that could be obtained using the desired IAS at the maximum allowable altitude (60000 ft). Conversely, if the aircraft were decelerating the function would calculate the amount of time it would take for the aircraft to decelerate from its current TAS to a minimal TAS value of 0. The above time is then assigned to the variable t_{max} and the corresponding airspeed is referred to as the "extreme TAS". The variables t_{min} and the initial test IAS, V_{i_test} are both initialized to zero.

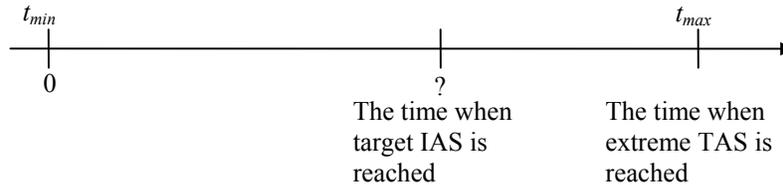
$$t_{max} = \frac{V_{t_extr} - V_t}{a}$$

Equation 3.4.28-1

$$t_{min} = 0$$

$$V_{i_test} = 0$$

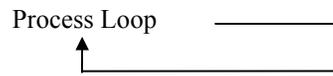
As a result, the following initial time interval is created. Within this time interval exists the time which it would take the aircraft to accelerate or decelerate to the target IAS.



As long as the difference between the target IAS and the test IAS is greater than a small epsilon value (in this case 0.5) and the interval between t_{max} and t_{min} is greater than 1, the function will continue to iteratively search for a solution within a logical loop.

While

$$(V_i - V_{i_test}) > \frac{\epsilon}{2} \text{ and } (t_{max} - t_{min}) > 1$$



BEGIN LOOP

The first step in this loop takes the average (midpoint) of the current time interval [t_{min}, t_{max}].

$$t = \frac{t_{max} + t_{min}}{2} \tag{Equation 3.4.28-2}$$

Next, the function extrapolates the altitude at which the aircraft would be after this amount of time, t , has elapsed. The extrapolation uses the following equation in the function's source code:

$$h_t = h + \left[V_i t + \frac{1}{2} a t^2 \right] g \tag{Equation 3.4.28-3}$$

Where the gradient, g , is the change in altitude over the change in the horizontal distance

$$g = \frac{\Delta h}{\Delta x} \tag{Equation 3.4.28-4}$$

and the change in horizontal distance is based on the basic Newtonian motion equation

$$\Delta x = V_i t + \frac{1}{2} a t^2 \tag{Equation 3.4.28-5}$$

Therefore, Equation 3.4.28-3 can be expressed as

$$h_t = h + \Delta x \frac{\Delta h}{\Delta x} \quad \text{Equation 3.4.28-6}$$

or simply,

$$h_t = h + \Delta h \quad \text{Equation 3.4.28-7}$$

It is important to note here that Equation 3.4.28-5 assumes that true airspeed and acceleration exist completely in the horizontal plane and have no component in the vertical dimension. This assumption is probably made because the descent/ascent angles are usually small resulting in very small vertical velocity and acceleration components. However, there are cases when the angle of climb or descent is significant (i.e. as much as 25°). By ignoring the vertical component of true airspeed and acceleration, Equation 3.4.28-3 could be in error by as much as 6%.

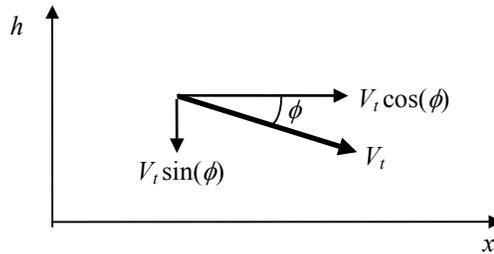
The solution to this anomaly is to address the vertical and horizontal components of true airspeed separately. The angle of descent or ascent is simply a trigonometric function of the gradient.

$$\phi = \arcsin(g)$$

The horizontal and vertical components of true airspeed are expressed as

$$\text{vertical: } V_t \sin(\phi)$$

$$\text{horizontal: } V_t \cos(\phi)$$



This approach should also be used when the acceleration, a , is calculated. Acceleration could then be expressed in both horizontal and vertical components, a_h and a_v , respectively. Subsequently, Equation 3.4.28-3 would be written as

$$h_t = h + \left[V_t \cos(\phi)t + \frac{1}{2} a_h t^2 \right] g \quad (\text{suggested Equation 3.4.28-3})$$

For the special case when the test altitude is above the maximum allowable altitude (parameter ACP.MAX_ALTITUDE) or below an altitude of 0, the function solves Equation 3.4.28-3 for the time when either $h_t = [\text{MAX_ALTITUDE or } 0]$. Since Equation 3.4.28-3 is quadratic, the function uses the quadratic formula to solve for the time roots.

Equation 3.4.28-3 is solved for 0

$$\frac{1}{2}at^2 + V_t t + \frac{(h - h_t)}{g} = 0 \quad \text{Equation 3.4.28-8}$$

Where

$$AA = \frac{1}{2}a$$

$$BB = V_t$$

$$CC = \frac{h - h_t}{g}$$

AA, *BB*, and *CC* are then used in the GM_QUADRATIC function as the quadratic coefficients. GM_QUADRATIC will return the number of real roots (either 1, 2 or 0). ST_IASALT will then use the smallest, non-negative root as the test time, *t*. *Note that there is no error condition handler here if GM_QUADRATIC returns 0 real roots. This error could cause an infinite loop since the variable TT would never change its value.*

Next, the true airspeed at the altitude, *h_t*, is calculated using the given acceleration, *a*, time, *t*, and the current true airspeed.

$$V_{t_test} = V_t + at \quad \text{Equation 3.4.28-9}$$

Using *V_{t_test}* and *h_t*, the associated indicated airspeed, *V_{i_test}*, is calculated using the CNV_CNVSPD function.

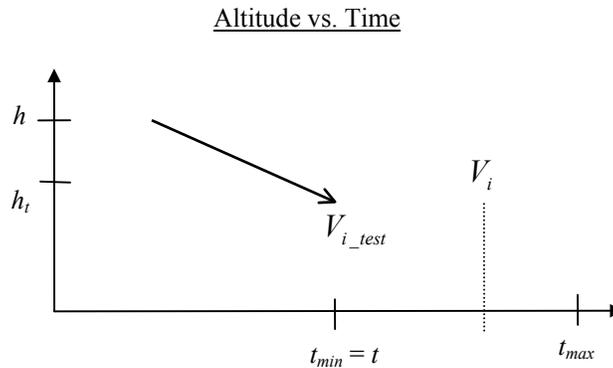
The function then makes the following logical assignments:

- **Condition 1:** If the aircraft has a positive acceleration, and the test IAS is less than the target IAS, then the aircraft was not given enough time to accelerate to the target IAS; therefore initialize the start of the new time interval with this time.

$$\text{Condition 1: } a > 0 \text{ and } V_{i_test} < V_i$$

$$\text{Result: } t_{min} = t$$

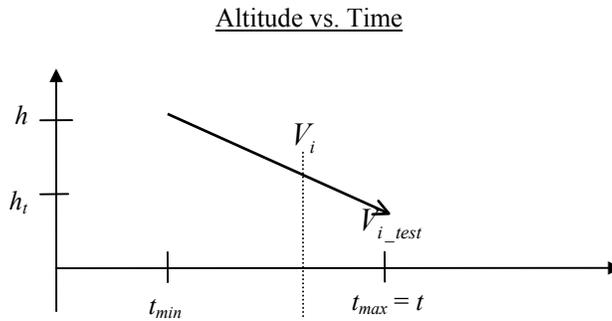
Condition 1 example: Aircraft descending



- **Condition 2:** If the aircraft has a positive acceleration, and the test IAS is greater than (or equal to) the target IAS, then the aircraft may have been given too much time, and accelerated to a speed faster than the target IAS; therefore initialize the end of the new time interval with this time.

Condition 2: $a > 0$ and $V_{i_test} \geq V_i$
 Result: $t_{max} = t$

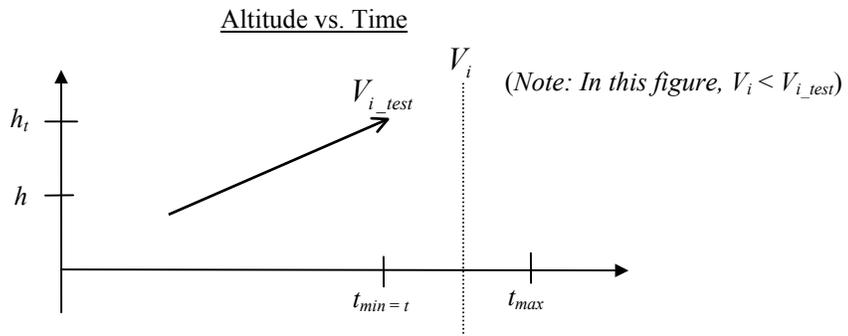
Condition 2 example: Aircraft descending



- **Condition 3:** If the aircraft is decelerating (or has no acceleration), and the test IAS is greater than the target IAS, then the aircraft was not given enough time to decelerate to the target IAS; therefore initialize the start of the new time interval with this time.

Condition 3: $a < 0$ and $V_{i_test} > V_i$
 Result: $t_{min} = t$

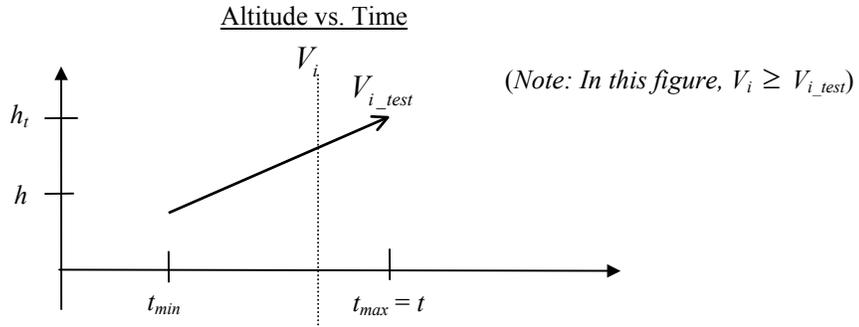
Condition 3 example: Aircraft ascending



- Condition 4: If the aircraft is decelerating (or has no acceleration), and the test IAS is less than (or equal to) the target IAS, then the aircraft may have been given too much time, and decelerated to a speed slower than the target IAS; therefore initialize the end of the new time interval with this time.

Condition 4: $a \leq 0$ and $V_{i_test} \leq V_i$
 Result: $t_{max} = t$

Condition 4 example: Aircraft ascending



Conditions 1 and 3 include an error trapping routine which checks for impossible situations, such as a climbing aircraft exceeding its maximum allowable altitude or a descending aircraft going below an altitude of 0. If either of these conditions are true, it makes the determination that the target IAS is unattainable with the given constraints and terminates the function. The last h_t and t that were calculated are then returned from this function.

It should be noted that this “unattainable condition” does not record an error code. Therefore it is unknown how many times this situation takes place.

END LOOP

The following example shows the iteration steps ST_IASALT uses to find a solution

Numerical Example:

Aircraft: B737
 Current Altitude = 31000ft Current TAS = 643 ft/s
 Target IAS = 280 nm/h Gradient = -0.07 ft/ft
 Acceleration = 1.227 ft/s/s

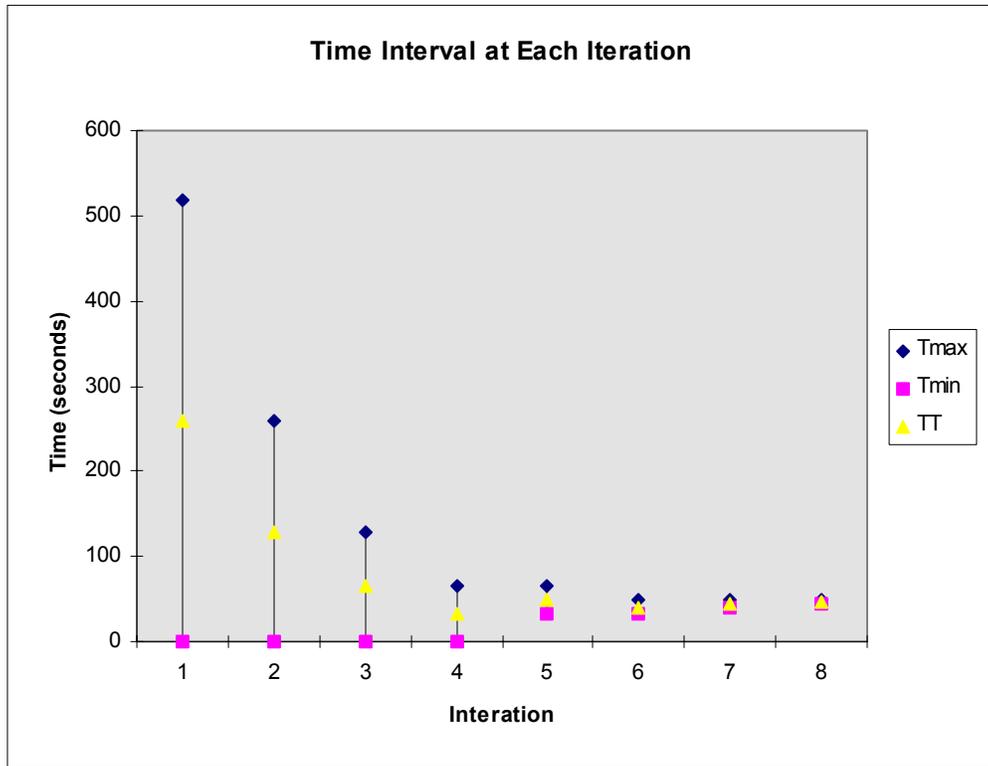
Initial steps:

Extreme TAS = Target IAS at 60000 ft = 1279 ft/s
 $T_{max} = (1279-643)/1.227 = 517.9$ s

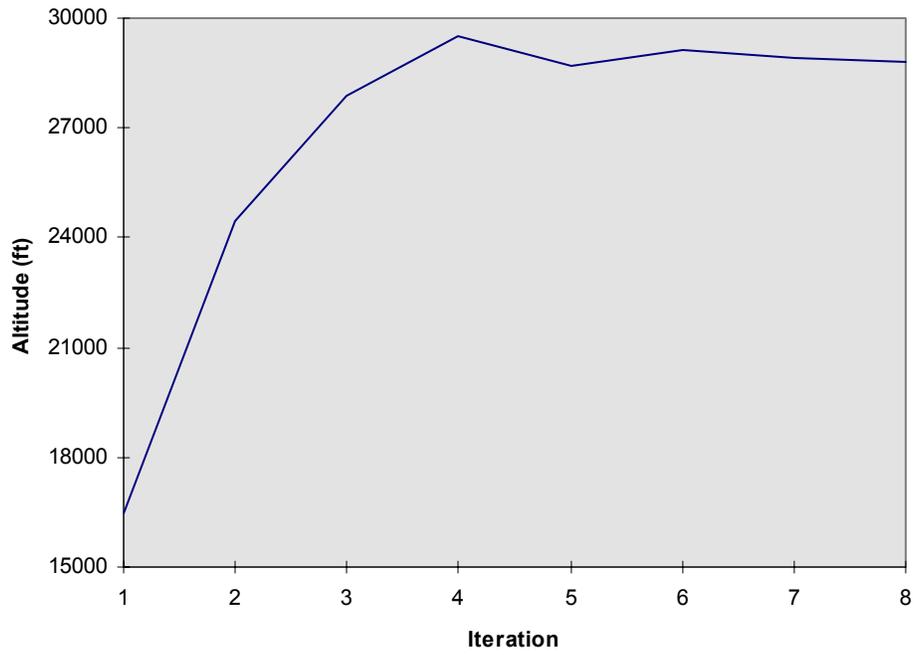
Values at Each Iteration

Iteration #	Tmax	Tmin	TT	Test HT	Test TAS (ft/s)	Test IAS (nm/h)
1	517.9	0	258.9	16454.9	961.3	460.4
2	258.9	0	129.5	24445.3	802.5	336.1
3	129.5	0	64.8	27902.7	723.1	283.7
4	64.8	0	32.4	29496.4	683.4	259.8
5	64.8	32.4	48.6	28710.9	703.2	271.6
6	48.6	32.4	40.5	29106.3	693.3	265.7
7	48.6	40.5	44.5	28909.1	698.3	268.66
8	48.6	44.52	46.5	28810.2	700.7	270.1

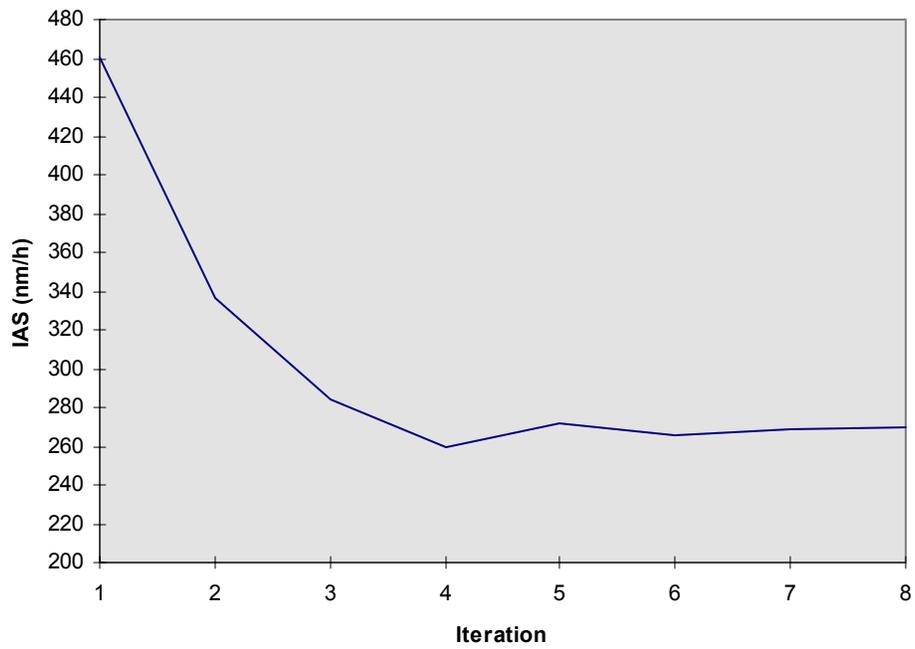
Time Interval at Each Iteration



Test Altitude at Each Iteration



Test IAS at Each Iteration



Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.28-1	Equation 3.4.28-5 assumes true airspeed and acceleration exist only in the horizontal plane	During steep ascent or descents this assumption could cause estimation error. A simple solution is provided in the analysis	Critical
R 3.4.28-2	For the special case when GM_QUADRATIC is called to solve for the time roots (Equation 3.4.28-8), there is no condition handler for the case when there are zero real roots returned.	This could cause an infinite loop error. A test could be made where $V_t^2 < 2a \frac{(h - h_t)}{g}$ <i>(A slow, decelerating aircraft climbing several thousand feet)</i>	Critical
R 3.4.28-3	The function will not model aircraft trajectories above 60,000 ft or below 0 ft.	It is possible for aircraft to exceed these constraints, however unlikely. Aircraft exceeding these altitude constraints would not be probed anyway.	Minor
R 3.4.28-4	There are no monitors to determine the number of “unattainable conditions” that occur when altitude constraints are exceeded	Extreme altitude conditions may be poorly modeled. URET has no way to record this problem.	Important

3.4.29 Function: ST_MACHALT (PL/I)

Iteratively searches for the altitude at which an aircraft, which is accelerating and either climbing or descending, attains a desired mach airspeed.

3.4.29.1 Description:

This function performs an iterative search to determine the altitude an aircraft will be when it attains a desired mach airspeed . The acceleration supplied to the function describes the true airspeed (TAS) acceleration. This acceleration is calculated from both the change in altitude and the aircraft’s speed change.

Since the function is attempting to capture a desired mach, there is no closed form solution to solving the motion equations as long as the acceleration is with respect to TAS. Instead, this function logically searches for the desired mach and altitude by trying to surround the solution with a narrowing time interval.

Table of Variable Definitions

Function Variable	Description	Math Symbol
ACCEL	The aircraft's acceleration due to both a level cruise acceleration and a TAS acceleration due to the change in altitude. This is given in units of (ft/s ²)	a
EXTREME_TAS	The maximum TAS possible with the given target mach. In this case, the corresponding TAS associated with a target mach at the maximum altitude possible (MAX_ALTITUDE = 60000ft)	V_{t_extr}
TMAX, TMIN	Iterative time variables used to locate the time the aircraft captures the target mach	t_{max}, t_{min}
TT	The test time which the function tries in determining when the aircraft captures the target mach. This is calculated as the average of t_{max} and t_{min}	t
TEST_MACH	The resultant mach that the function calculates from the given iteration	V_{m_test}
TARGET_MACH	The desired mach	V_m
TEST_HT	The test altitude that the function tries in determining where the aircraft captures the target mach	h_t
TEST_TAS	The test TAS that the function tries in determining where the aircraft captures the target mach	V_{t_test}
MACH_EPSILON	A small parameter value used to determine if two mach values are close (currently set to 0.0001)	ϵ
CURRENT_Z	The current altitude of the aircraft (ft)	h
CURRENT_TAS	The current TAS at the current altitude	V_t
GRADIENT	The altitude gradient. This value is the ratio of the change in altitude over the change in horizontal distance traveled. (ft/ft)	g

3.4.29.2 Mathematics:

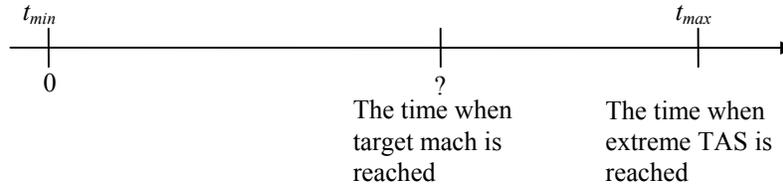
The function begins by determining if the aircraft is accelerating, decelerating, or has a constant TAS. If the aircraft is accelerating, it is known that the final TAS should be larger than the current TAS. The function must ensure that the estimated time interval includes the solution. Therefore, the function calculates the amount of time it would take for the aircraft to accelerate from its current TAS to the largest TAS that could be obtained using the desired mach at sea level (0 ft). Conversely, if the aircraft were decelerating the function would calculate the amount of time it would take for the aircraft to decelerate from its current TAS to a minimal TAS value of 0. The above time is then assigned to the variable t_{max} and the corresponding airspeed is referred to as the "extreme TAS". The variables t_{min} and the initial test mach, V_{m_test} are both initialized to zero.

$$t_{max} = \frac{V_{t_extr} - V_t}{a} \quad \text{Equation 3.4.29-1}$$

$$t_{min} = 0$$

$$V_{m_test} = 0$$

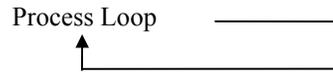
As a result, the following initial time interval is created. Within this time interval exists the time which it would take the aircraft to accelerate or decelerate to the target mach.



As long as the difference between the target mach and the test mach is greater than a small epsilon value (in this case 0.0001) and the interval between t_{max} and t_{min} is greater than 1, the function will continue to iteratively search for a solution within a logical loop.

While

$$(V_m - V_{m_test}) > \varepsilon \text{ and } (t_{max} - t_{min}) > 1$$



BEGIN LOOP

The first step in this loop takes the average (midpoint) of the current time interval $[t_{min}, t_{max}]$.

$$t = \frac{t_{max} + t_{min}}{2} \quad \text{Equation 3.4.29-2}$$

Next, the function extrapolates the altitude at which the aircraft would be after this amount of time, t , has elapsed. The extrapolation uses the following equation in the function's source code:

$$h_t = h + \left[V_t t + \frac{1}{2} a t^2 \right] g \quad \text{Equation 3.4.29-3}$$

Where the gradient, g , is the change in altitude over the change in the horizontal distance

$$g = \frac{\Delta h}{\Delta x} \quad \text{Equation 3.4.29-4}$$

and the change in horizontal distance is based on the basic Newtonian motion equation

$$\Delta x = V_t t + \frac{1}{2} a t^2 \quad \text{Equation 3.4.29-5}$$

Therefore, Equation 3.4.29-3 can be expressed as

$$h_t = h + \Delta x \frac{\Delta h}{\Delta x} \quad \text{Equation 3.4.29-6}$$

or simply,

$$h_t = h + \Delta h \quad \text{Equation 3.4.29-7}$$

It is important to note here that Equation 3.4.29-5 assumes that true airspeed and acceleration exist completely in the horizontal plane and have no component in the vertical dimension. This assumption is probably made because the descent/ascent angles are usually small resulting in very small vertical velocity and acceleration components. However, there are cases when the angle of climb or descent is significant (i.e. as much as 25°). By ignoring the vertical component of true airspeed and acceleration, Equation 3.4.29-3 could be in error by as much as 6%.

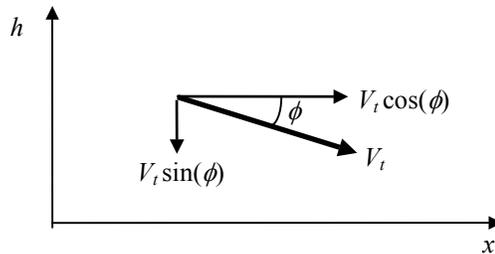
The solution to this anomaly is to address the vertical and horizontal components of true airspeed separately. The angle of descent or ascent is simply a trigonometric function of the gradient.

$$\phi = \arcsin(g)$$

The horizontal and vertical components of true airspeed are expressed as

$$\text{vertical: } V_t \sin(\phi)$$

$$\text{horizontal: } V_t \cos(\phi)$$



This approach should also be used when the acceleration, a , is calculated. Acceleration could then be expressed in both horizontal and vertical components, a_h and a_v , respectively.

Subsequently, Equation 3.4.29-3 would be written as

$$h_t = h + \left[V_t \cos(\phi)t + \frac{1}{2} a_h t^2 \right] g \quad (\text{suggested Equation 3.4.29-3})$$

For the special case when the test altitude is above the maximum allowable altitude (parameter ACP.MAX_ALTITUDE) or below an altitude of 0, the function solves Equation 3.4.29-3 for the time when either $h_t = [\text{MAX_ALTITUDE or } 0]$. Since Equation 3.4.29-3 is quadratic, the function uses the quadratic formula to solve for the time roots.

Equation 3.4.29-3 is solved for 0

$$\frac{1}{2}at^2 + V_t t + \frac{(h-h_t)}{g} = 0 \quad \text{Equation 3.4.29-8}$$

Where

$$AA = \frac{1}{2}a$$

$$BB = V_t$$

$$CC = \frac{h-h_t}{g}$$

AA, *BB*, and *CC* are then used in the GM_QUADRATIC function as the quadratic coefficients. GM_QUADRATIC will return the number of real roots (either 1, 2 or 0). ST_MACHALT will then use the smallest, non-negative root as the test time, *t*. *Note that there is no error condition handler here if GM_QUADRATIC returns 0 real roots. This error could cause an infinite loop since the variable TT would never change its value.*

Next, the true airspeed at the altitude, *h_t*, is calculated using the given acceleration, *a*, time, *t*, and the current true airspeed.

$$V_{t_test} = V_t + at \quad \text{Equation 3.4.29-9}$$

Using *V_{t_test}* and *h_t*, the associated mach airspeed, *V_{m_test}*, is calculated using the CNV_CNVSPD function.

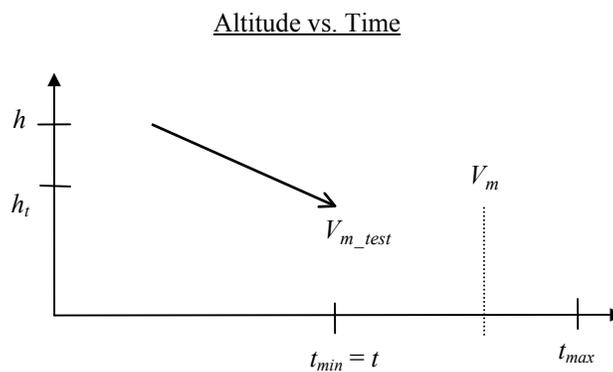
The function then makes the following logical assignments:

- **Condition 1:** If the aircraft has a positive acceleration, and the test mach is less than the target mach, then the aircraft was not given enough time to accelerate to the target mach; therefore initialize the start of the new time interval with this time.

$$\text{Condition 1: } a > 0 \text{ and } V_{m_test} < V_m$$

$$\text{Result: } t_{min} = t$$

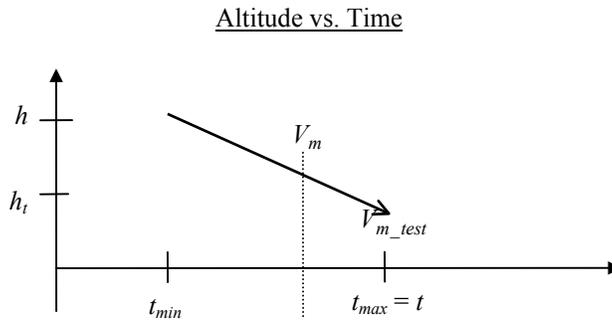
Condition 1 example: Aircraft descending



- **Condition 2:** If the aircraft has a positive acceleration, and the test mach is greater than (or equal to) the target mach, then the aircraft may have been given too much time, and accelerated to a speed faster than the target mach; therefore initialize the end of the new time interval with this time.

Condition 2: $a > 0$ and $V_{m_test} \geq V_m$
 Result: $t_{max} = t$

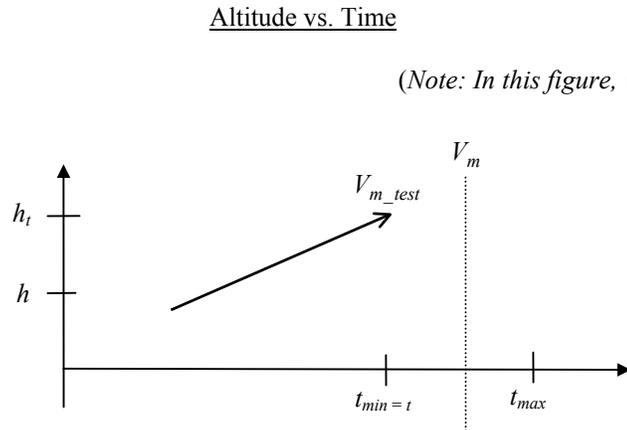
Condition 2 example: Aircraft descending



- **Condition 3:** If the aircraft is decelerating (or has no acceleration), and the test mach is greater than the target mach, then the aircraft was not given enough time to decelerate to the target mach; therefore initialize the start of the new time interval with this time.

Condition 3: $a < 0$ and $V_{m_test} > V_m$
 Result: $t_{min} = t$

Condition 3 example: Aircraft ascending

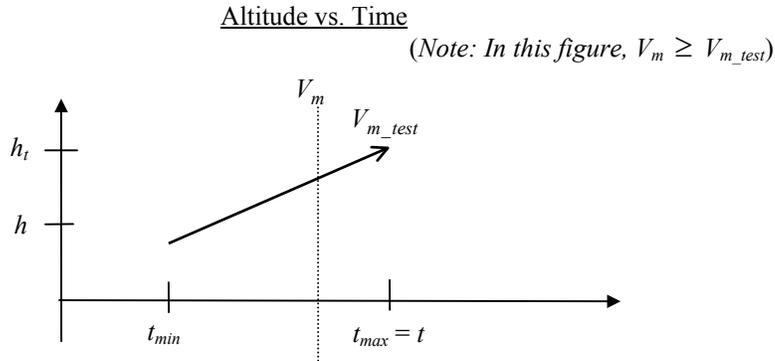


- Condition 4: If the aircraft is decelerating (or has no acceleration), and the test mach is less than (or equal to) the target mach, then the aircraft may have been given too much time, and decelerated to a speed slower than the target mach; therefore initialize the end of the new time interval with this time.

$$\text{Condition 4: } a \leq 0 \text{ and } V_{m_test} \leq V_m$$

$$\text{Result: } t_{max} = t$$

Condition 4 example: Aircraft ascending



Conditions 1 and 3 include an error trapping routine which checks for impossible situations, such as a climbing aircraft exceeding its maximum allowable altitude or a descending aircraft going below an altitude of 0. If either of these conditions are true, it makes the determination that the target mach is unattainable with the given constraints and terminates the function. The last h_i and t that were calculated are then returned from this function.

It should be noted that this “unattainable condition” does not record an error code. Therefore it is unknown how many times this situation takes place.

END LOOP

The following example shows the iteration steps ST_MACHALT uses to find a solution

For a numerical example, see Section 3.4.28, ST_IASALT.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.29-1	Equation 3.4.29-5 assumes true airspeed and acceleration exist only in the horizontal plane	During steep ascent or descents this assumption could cause estimation error. A simple solution is provided in the analysis	Critical
R 3.4.29-2	For the special case when GM_QUADRATIC is called to solve for the time roots (Equation 3.4.29-8), there is no condition handler for the case when there are zero real roots returned.	This could cause an infinite loop error. A test could be made where $V_t^2 < 2a \frac{(h - h_t)}{g}$ <i>(A slow, decelerating aircraft climbing several thousand feet)</i>	Critical
R 3.4.29-3	The function will not model aircraft trajectories above 60,000 ft or below 0 ft.	It is possible for aircraft to exceed these constraints, however unlikely. Aircraft exceeding these altitude constraints would not be probed anyway.	Minor
R 3.4.29-4	There are no monitors to determine the number of “unattainable conditions” that occur when altitude constraints are exceeded	Extreme altitude conditions may be poorly modeled. URET has no way to record this problem.	Important

3.4.30 Function: ST_MAXTAS (PL/I)

Finds the maximum TAS for an aircraft at a given altitude.

3.4.30.1 Description:

This function simply searches the ACC table for the maximum True Airspeed which corresponds to the given altitude and the aircraft type. The processing begins by finding the tables which are associated with the given aircraft engine type. Then it searches the Max_TAS_in_Alt_Layer table for the range of altitude layers which include the given altitude. If the input values fall between two table values, then the function uses linear interpolation to calculate maximum TAS value from the two closest table values.

The maximum TAS values are based on the aircraft manufacturers’ tables.

3.4.31 Function: ST_MINTAS (PL/I)

Finds the minimum TAS for an aircraft at a given altitude.

3.4.31.1 Description:

This function simply searches the ACC table for the minimum True Airspeed which corresponds to the given altitude and the aircraft type. The processing begins by finding the tables which are associated with the given aircraft engine type. Then it searches the Min_TAS_in_Alt_Layer table for the range of altitude layers which include the given altitude. If the input values fall between two table values, then the function uses linear interpolation to calculate minimum TAS value from the two closest table values.

The minimum TAS values are based on the aircraft manufacturers’ tables.

3.4.32 Function: ST_TIME_SSGDATA (PL/I)

Finds various SSG values (ARD, x, y, altitude, ground speed, true airspeed, pointer to the SSG) for a given time.

3.4.32.1 Description:

Given a time, this function will return the ARD, position data (x, y), altitude, ground speed, and true airspeed from the appropriate SSG which encompasses the given input time. If the time is between the start and end time of the SSG, but not actually coincident with the start or end time, this function will interpolate the output values over the segment interval. If the aircraft is determined to be in a hold at this time, the modeled hold values are assigned and ground and true airspeeds are taken from the state segment either before or after the hold state segment.

Table of Variable Definitions

Function Variable	Description	Math Symbol
X, Y	Coordinates of the aircraft at the given time (ft)	x, y
SSG.X(1), SSG.Y(1), SSG.X(2), SSG.Y(2)	Coordinates of the start and end points of the state segment (ft)	x_1, y_1 x_2, y_2
SSG_BOX.XPOS, SSG_BOX.YPOS	Coordinates of the assumed aircraft position during a hold (ft)	x_h, y_h
ALT	Altitude of the aircraft at the given time (ft)	z
SSG.Z(1), SSG.Z(2)	Altitude of the aircraft at the start and end points of the state segment (ft)	z_1, z_2
SSG_BOX.ZPOS(1)	The assumed aircraft altitude at the beginning of a hold (ft). Since altitude is not modeled during holds, the altitude at the start and end of the hold are assumed to be equal	z_h
A	Ground Speed acceleration (ft/s/s)	a_g
SSG.ACC	Aircraft true airspeed acceleration parameter (ft/s/s)	a_t
GSPD	Ground speed of the aircraft at the given time (ft/s/s)	V_g
SSG.GSPD(1), SSG.GSPD(2)	Ground speed of the aircraft at the start and end points of the state segment (ft/s/s)	V_{g1}, V_{g2}
TSPD	True airspeed of the aircraft at the given time (ft/s/s)	V_t
SSG.TSPD(1), SSG.TSPD(2)	True airspeed of the aircraft at the start and end points of the state segment (ft/s/s)	V_{t1}, V_{t2}
XTIME	Given time at which to find the resultant output values (seconds)	t
SSG.TIME(1), SSG.TIME(2)	Time associated with the start and end points of the state segment (seconds)	t_1, t_2
SSG.SEG_LNG	The length of the state segment (ft)	ℓ
SSG.ARD	Along Route Distance (ft)	ard
SSG.TOTDIST	ARD at the beginning of the SSG (ft)	ard_1
SSG_BOX.TOTDIST	ARD at the beginning of the hold SSG (ft)	ard_h

3.4.32.2 Mathematics:

The function performs the following simple calculations and logic.

First it determines which SSGs start and end time interval includes the given input time.

The function also checks if the SSG is in a hold or not. If the SSG is not in a hold, the following process takes place.

Not in a Hold

The ground speed acceleration over that SSG is calculated by the following equation:

$$a_g = \frac{V_{g2} - V_{g1}}{t_2 - t_1} \quad \text{Equation 3.4.32-1}$$

Then the time interval, from the start of the SSG to given time, is calculated

$$\Delta t = t - t_1 \quad \text{Equation 3.4.32-2}$$

The above two values are then used in the simple kinematic equation for constant acceleration to get the ground distance traveled over the above time interval

$$d = V_{g1} \Delta t + \frac{1}{2} a_g (\Delta t)^2 \quad \text{Equation 3.4.32-3}$$

and added to the total distance traveled by the aircraft at the beginning of the SSG to get the ARD

$$ard = ard_1 + d \quad \text{Equation 3.4.32-4}$$

This distance is then used in calculating a ratio of the distance traveled over the total length of the state segment.

$$r = \frac{d}{\ell} \quad \text{Equation 3.4.32-5}$$

Note: There is no check made here to ensure that the total length of the state segment is not zero (as in the ST_ARD_SSGDATA module). This is a software issue that could cause errors under the right conditions.

This ratio is then used as a multiplier in calculating the new coordinates of the aircraft position, along the state segment, at the given time.

$$\begin{aligned} x &= x_1 + r(x_2 - x_1) \\ y &= y_1 + r(y_2 - y_1) \end{aligned} \quad \text{Equation 3.4.32-6}$$

The new altitude can also be calculated with this multiplier, since the trajectory modeler models the change in altitude based on a gradient. This gradient is defined as the change in vertical distance over the change in horizontal distance traveled. This gradient is constant over the entire segment, therefore the change in altitude up to the given time can be calculated with the same ratio multiplier (which is based on horizontal movement, in units of feet). Therefore, the following equation can be used to calculate the new altitude at the given time.

$$z = z_1 + r(z_2 - z_1) \quad \text{Equation 3.4.32-7}$$

Next the function performs the following logic:

If the True Airspeed acceleration is equal to zero

If

$$a_t = 0, \quad \text{Equation 3.4.32-8}$$

Then assign the ground speed and true airspeed at the first cusp of the state segment equal to the ground speed and true airspeed at the given time.

Then

$$V_g = V_{g1} \quad \text{Equation 3.4.32-9}$$

$$V_t = V_{t1} \quad \text{Equation 3.4.32-10}$$

Note: This appears to be an incorrect assumption. If the true airspeed acceleration is zero, the ground speed could still change over the segment because of changes in wind. The ground speed should be calculated with the ground speed acceleration calculated earlier in the code (i.e. Equation 3.4.32-1). A more accurate, however more processing intensive, method would be to use the same true airspeed along with the DB_AIR_AT_POINT, GM_BRNG, and CNV_GRDSPD functions to calculate the exact ground speed at the given time with the known wind conditions.

Otherwise, the function will use the calculated accelerations for ground speed and true airspeed to determine the new ground speed and true airspeed at the given time

Else

$$V_g = V_{g1} + a_g \Delta t \quad \text{Equation 3.4.32-11}$$

$$V_t = V_{t1} + \left(\frac{V_{t2} - V_{t1}}{t_2 - t_1} \right) \Delta t \quad \text{Equation 3.4.32-12}$$

If the SSG is in a hold, the following process takes place.

In a Hold

The position, altitude and ARD are all assigned the assumed aircraft positions during the hold.

$$\begin{aligned}
 ard &= ard_h \\
 x &= x_h \\
 y &= y_h \\
 z &= z_h
 \end{aligned}
 \tag{Equation 3.4.32-13}$$

The ground speed and the true airspeed are simply assigned either the values of the true airspeed and ground speed of the previous SSG (previous to the hold SSG) if it exists, or the next SSG (after the hold) if it exists.

Finally, the function returns the values for position (x, y) , altitude (z) , along route distance (ard) , ground speed (V_g) , true airspeed (V_t) , and a pointer the state segment which contained the given input time.

Note: In the beginning comments of this module, under outputs, the description for FOUND states “Indicates whether the ARD exists on Route” This should say “Indicates whether the XTIME exists during the route”.

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.32-1	Equation 3.4.32-9 assumes ground speed remains constant if true airspeed remains constant	This assumption appears to be incorrect if wind values were to change over the length of the state segment.	Important
R 3.4.32-2	Equation 3.4.32-5 does not protect against division by a zero segment length	This is a software (robustness) issue which could cause processing errors under the proper conditions. There are other areas in this module which also do not protect against dividing by zero (i.e. Equation 3.4.32-1 and Equation 3.4.32-12)	Important

3.4.33 Function: ST_TRANSLATE_ARD (PL/I)

This function translates a full route ARD to an ORS ARD and vice versa.

3.4.33.1 Description:

This function will either translate a full route ARD (defined by the RTE_OR data structure) to the current ORS ARD (defined by the ORS data structure) or vice versa, depending on the value of an input variable (TRAN_TYPE). The supplied ARD need not be at the end point of either the RTE_OR or ORS segment. The RTE_OR data structure is defined by the initial trajectory which was built based on the flight plan (or amendment) route string. The ORS data structure could be different from the RTE_OR because of a later reconformance. This function will take the difference between the full route ARD (defined by the RTE_OR data structure) and the ORS ARD (defined by the ORS data structure) and either add or subtract this difference to the given ARD (supplied as an input), depending on the TRAN_TYPE value.

Table of Variable Definitions

Function Variable	Description	Math Symbol
IN_ARD	The ARD given as the input	ard_{in}
OUT_ARD	The ARD supplied as the output	ard_{out}
ORS_END_ARD	ARD at the end of the given ORS segment	ard_{oe}
RTE_ORIS_END_ARD	ARD at the end of the full route, RTE_ORIS, segment	ard_{re}
ORS.ACCUM_DIST	ARD at the beginning of the given ORS segment	ard_{os}
RTE_ORIS.ACCUM_DIST	ARD at the beginning of the full route, RTE_ORIS, segment	ard_{rs}
ORS.LNGTH	The length of the ORS segment	l_o
RTE_ORIS.LNGTH	The length of the RTE_ORIS segment	l_r

3.4.33.2 Mathematics:

For instance, if TRAN_TYPE = 1 the function will translate a full route ARD to a ORS ARD, if TRAN_TYPE = 2 the function will translate a ORS ARD to a full route ARD.

For TRAN_TYPE = 1, the function first finds the ARD at the end point of both the ORS and RTE_ORIS segments by adding the length of the segment to the ARD at the beginning of the segment.

$$ard_{oe} = ard_{os} + l_o \quad \text{Equation 3.4.33-1}$$

$$ard_{re} = ard_{rs} + l_r \quad \text{Equation 3.4.33-2}$$

The function then takes the difference of these two values

$$d = ard_{re} - ard_{oe} \quad \text{Equation 3.4.33-3}$$

and subtracts this difference from the given ARD

$$ard_{out} = ard_{in} - d \quad \text{Equation 3.4.33-4}$$

The function will then check to ensure ard_{out} is not less than zero or greater than ard_{oe} . If it is the value of zero or the ard_{oe} would be assigned, respectively, in its place.

In the case when TRAN_TYPE = 2, all the above steps are about the same except that the function adds the difference to the given ARD

$$ard_{out} = ard_{in} + d \quad \text{Equation 3.4.33-5}$$

and checks are made with respect to the RTE_ORIS endpoints.

There are no assumptions or approximations made in this module which would have significant impact on the algorithms.

3.4.34 Function: ST_XYTOTIME (PL/I)

Finds the time of a given (x, y) from an SSG.

3.4.34.1 Description:

This function will take a pointer to a particular state segment and a set of x, y coordinates and returns the time it would take for the aircraft to travel along its route to the given coordinates. This function calculates this time by representing the motion of the aircraft with a simple kinematic equation. If ground speed acceleration is close to zero, the function simply calculates an increment of time to be added to the start of the state segment by using a distance ratio. If there is an acceleration, the kinematic equation becomes a quadratic, and the roots of the quadratic are then solved (using the quadratic formula). Any meaningful roots are used as a solution. If there are no meaningful roots (i.e. both roots fall outside the state segment time interval), then the time associated with state segment end-point that is closest to one of the roots is used.

This function is based on classic physics and mathematics. It assumes a constant ground speed acceleration over the entire length of the segment.

Table of Variable Definitions

Function Variable	Description	Math Symbol
XX, YY	Coordinates of the aircraft position supplied as inputs to the function (nm)	x, y
SSG.X(1), SSG.Y(1), SSG.X(2), SSG.Y(2)	Coordinates of the start and end points of the given SSG, respectively (nm)	x_1, y_1 x_2, y_2
SSG.SEG LNG	The horizontal length of the SSG	l
SSG.GSPD(1), SSG.GSPD(2)	The ground speed of the aircraft at the start and end points of the given SSG, respectively (ft/s/s)	V_{g1}, V_{g2}

3.4.34.2 Mathematics:

This function solves a basic kinematic equation of motion in the form of a quadratic to determine the time roots. The motion equation used is

$$d = V_{g1}t + \frac{1}{2}at^2 \quad \text{Equation 3.4.34-1}$$

or equivalently,

$$\frac{1}{2}a_g t^2 + V_g t - d = 0 \quad \text{Equation 3.4.34-2}$$

where a_g is the calculated ground speed acceleration over the course of the segment, and is

calculated as (*assumes constant acceleration*)

$$a_g = \frac{\Delta V}{\Delta t} = \frac{V_{g2} - V_{g1}}{t_2 - t_1} \quad \text{Equation 3.4.34-3}$$

and d is the horizontal distance traveled by the aircraft from the start of the SSG to the given x, y position.

$$d = \sqrt{(x - x_1)^2 + (y - y_1)^2} \quad \text{Equation 3.4.34-4}$$

The function attempts to solve the quadratic in Equation 3.4.34-2 by breaking it up into the appropriate terms of the quadratic formula.

$$A = \frac{1}{2} a$$

$$B = V_{g1}$$

$$C = -d$$

solve for t in

$$At^2 + Bt + C = 0 \quad \text{Equation 3.4.34-5}$$

If $A \approx 0$ the function assumes a constant ground speed and uses a simple ratio of distance over segment length to interpolate a time increment.

$$t = (t_2 - t_1) \frac{d}{l} \quad \text{Equation 3.4.34-6}$$

Otherwise, the function solves the two roots of the classic quadratic formula

$$t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \quad \text{Equation 3.4.34-7}$$

Upon solving this formula, there are several conditions that are handled that should be noted

- If the term inside the radical, $B^2 - 4AC$, is less than 0, the function assigns a zero value to this term and the quadratic simplifies to

$$t = \frac{-B}{2A}$$

- If the first root is between the SSG time interval, the function uses this real root.

$$\text{If } 0 \leq t_{root1} \leq (t_2 - t_1)$$

- Else if the second root is between the SSG time interval, the function uses this real root.
- If neither root is within the SSG time interval, the function uses the time associated with the SSG endpoint which is closest to one of the root solutions.

Finally, the calculated time increment, t , is added to the time at the start of the SSG, to arrive at the total time for the aircraft to reach the given coordinates along its route.

$$t_{total} = t_1 + t$$

Equation 3.4.34-8

Assessment Table

REF#	Approximation/Assumption	Assessment	Impact on TJM
R 3.4.34-1	Equation 3.4.34-3 assumes a constant ground speed acceleration	This assumption may be incorrect when various winds could have inconsistent effects on the ground speed over the length of a segment. The proper modeling of this could result in a considerable increase in processing (i.e. a new segment at every change in wind direction/speed)	Important

4. Assessment Findings and Observations

This section provides a summary of the assessment findings and observations. The URET D1.1 limitations and assumptions determined through the independent assessment effort are provided, as well as a summary of the algorithmic assessment tables contained in Section 3. ACT-250's suggested improvements to the source code are also provided, as well as recommendations for future analysis.

4.1 URET D1.1 Limitations

The following limitations should be considered restricted or non-existing capabilities of the URET D1.1 conflict probe. This section lists these limitations to give the reader a clearer understanding of what URET can and cannot do. It is not intended to imply that these limitations are right or wrong, but rather leaves these issues open for future discussion. (Note: if it is known that the limitation is being addressed in URET Delivery 2 (D2), this is so noted).

- URET D1.1 does not probe all aircraft in the AERA boundary (only categories A, B and inbound F)
- URET D1.1 does not monitor the conformance of, or reconfirm, all aircraft trajectories (only categories A and sometimes B).
- URET D1.1 does not probe or reconfirm aircraft determined to be in vertical drift
- URET D1.1 does not probe the holding trajectories
- URET D1.1 performs an event driven conflict probe for a fixed distance into the future. It does not continuously reprobe all aircraft pairs (URET D2 performs a periodic reprobe). The events that cause a conflict probe of an aircraft are: entry into the ARTCC airspace where the trajectory is first created, or a trajectory re-modeling. The probe does not examine the entire trajectory for aircraft-to-aircraft conflicts. It examines only a window of time into the future, called the look ahead time. Therefore, if an event does not take place over the length of the look ahead time window, any future conflicts will not be detected. The consequence is a missed or delayed alert.
- In the lateral reconfirmation logic, there is no unique solution to reconfirm unreported holds or a track which is offset parallel to the trajectory.
- The coordinate system used in D1.1 allows for the system to only function in one ARTCC airspace. Significant changes must be made to enhance URET D1.1 to work in a multi-ARTCC environment
- URET D1.1 assumes that all SUAs are active. The system does not allow the user to disable conflict detection for inactive sectors.
- URET D1.1 notifies the user about any detected conflicts, regardless of the likelihood of the conflict (URET D2 incorporates new notification logic which considers conflict likelihood).
- The horizontal route conversions cannot account for Type 2 or 4 coded routes.
- The trajectory modeler can only model one interim altitude at a time per flight plan.
- URET D1.1 has problems with modeling climb interim altitudes since it assumes the aircraft will remain at that altitude for the remainder of the flight.
- TKM documentation was minimal or outdated. TKM was continuously evolving through multiple URET deliveries during the course of the assessment.
- The time and position of the minimum separation is not an option to be presented by the display.

4.2 URET D1.1 Assumptions

The following are some noteworthy assumptions made in URET D1.1.

Trajectory modeling assumptions:

- Aircraft weight is assumed to be constant throughout the entire flight.
- The value of 3.00 is used for the maximum descent factor for all aircraft types.
- The value of 1.67 is used for cruise acceleration and idle deceleration for all aircraft types.

- The temperature determined at the beginning of the State Segment is assumed to be constant over the length of the segment.
- Acceleration and altitude gradient are constant over the entire trajectory segment.
- Trajectories are built with a geometric altitude, while aircraft actually fly pressure altitudes above 18000 feet.
- The use of stereographic projection introduces small errors in aircraft position estimates because it is approximating part of the surface of a sphere with a plane. The errors increase toward the edges of the ARTCC airspace.
- Aircraft descent and climb characteristics (gradient, speeds) are dependent on temperature deltas from standard atmosphere. If the temperature falls between two delta values, linear interpolation is done to approximate these values.
- Turns are modeled as an instantaneous change in heading instead of modeling a turn radius. Aircraft trajectories are approximated with a series of linear segments.
- The climb, descent, and speed profiles are based on the aircraft manufacturer's recommended profiles for that aircraft type.
- External data sources, such as wind and track positions, are assumed to be accurate. However, inaccuracies in the external data will have significant impact on the accuracy of the trajectories modeled.

Conflict Prediction Assumptions:

- The conflict probe assumes accurate trajectory and track data within conformance bounds.
- APD always assumes aircraft are within the rectangular uncertainty region (i.e. conformance bounds) which is centered on the monitored trajectory of the aircraft.
- At any specific unit of time, URET assumes an aircraft is within a region of uncertainty. These regions are represented by a series of conformance boxes along the predicted trajectory of the aircraft. The conflicts predicted are based on whether the pair of aircraft conformance boxes are less than separation distance, not whether the pair of trajectory centerlines are less than separation.
- URET assumes larger conformance regions around the trajectory centerline during a turn, climb, or descent. These larger distances account for the increased uncertainty associated with the maneuver of the aircraft.

4.3 Summary of Algorithmic Assessment Tables

As described in Section 3, the actual source code was examined for assumptions and approximations. These assumptions and approximations are presented in tables following each specific algorithm analyzed. The following table consolidates and summarizes all these assessment tables. It is sorted first by impact category, then module name, and finally function name. The reference number for each assessment item refers to the section number where the item first appears in Section 3.

REF#	Function Name	Impact Category	Module	Brief Description
R 3.1.4-1	CFP_MIDDLE_HORIZ (C)	Critical	APD	Acceleration is assumed minor over the segment and only one of the end point position vectors is used to check the separation.
R 3.4.17-3	GM_REGN (PL/I)	Critical	APD	Accuracy of algorithm directly related to the GM_TSTPNT function.
R 3.4.27-1	ST_FINDARD (PL/I)	Critical	APD, TJM	Closest approach point not on the flight segment has an error in the distance calculation.
R 3.3.1-1	CNV_GRD_TO_TAS (PL/I)	Critical	TJM	Assumes small inertial path angle.
R 3.4.3-1	CNV_GRDSPD (PL/I)	Critical	TJM	Assumes small path angle.
R 3.4.9-3	CNV_XYLL (PL/I)	Critical	TJM	Approximates angles using a power series.
R 3.2.2-1	EGRAD (PL/I)	Critical	TJM	A small flight path angle is assumed.
R 3.2.2-3	EGRAD (PL/I)	Critical	TJM	Approximation in gradient calculation.
R 3.4.23-1	ST_CLIMB_DIST (PL/I)	Critical	TJM	Assumes altitude layer climb gradients.
R 3.4.23-2	ST_CLIMB_DIST (PL/I)	Critical	TJM	Assumes the aircraft's climb gradient factor supplied by the AMC table.
R 3.4.24-1	ST_CLIMB_GRADIENT (PL/I)	Critical	TJM	Assumes altitude layer climb gradients and the IAS (or Mach) in the ACC table.
R 3.4.25-1	ST_DESCENT_DIST (PL/I)	Critical	TJM	Assumes altitude layer descent gradients and the idle deceleration rate.
R 3.4.25-2	ST_DESCENT_DIST (PL/I)	Critical	TJM	Assumes descent gradient factor supplied by the AMC table.
R 3.4.26-1	ST_DESCENT_GRADIENT (PL/I)	Critical	TJM	Assumes altitude layer descent gradients and the IAS (or Mach) in the ACC table.
R 3.4.28-1	ST_IASALT (PL/I)	Critical	TJM	Assumes true airspeed and acceleration exist only in the horizontal plane.
R 3.4.28-2	ST_IASALT (PL/I)	Critical	TJM	There is no condition handler for the case when there are zero real roots returned for the quadratic equation calculation.

REF#	Function Name	Impact Category	Module	Brief Description
R 3.4.29-1	ST_MACHALT (PL/I)	Critical	TJM	Assumes true airspeed and acceleration exist only in the horizontal plane.
R 3.4.29-2	ST_MACHALT (PL/I)	Critical	TJM	There is no condition handler for the case when there are zero real roots returned for the quadratic equation calculation.
R 3.3.12-4	TKM_GM_REGN(C)	Critical	TKM	Accuracy of algorithm directly related to the TKM_GM_TSTPNT function.
R 3.3.13-2	TKM_GM_TSTPNT (C)	Critical	TKM	Unprotected return case for an if statement.
R 3.3.13-3	TKM_GM_TSTPNT (C)	Critical	TKM	Potential division by zero problem in code.
R 3.1.1-1	CFP_COARSE_HORIZ (C)	Important	APD	Assumes aircraft in a hold are stationary not flying a holding pattern.
R 3.1.2-1	CFP_FINE (C)	Important	APD	Approximates the velocity in the relative velocity vector calculation with average ground velocity.
R 3.1.2-2	CFP_FINE (C)	Important	APD	Checks for the round off case where the number of intersections equals zero or the maximum ratio is ≤ 0 .
R 3.1.2-3	CFP_FINE (C)	Important	APD	The case where the P and Q vectors are outside the octagon and the GM_INSECS finds one intersection is assumed round off error.
R 3.1.4-2	CFP_MIDDLE_HORIZ (C)	Important	APD	Assumes constant acceleration to approximate the velocity in the relative velocity vector calculation with average ground velocity.
R 3.1.4-3	CFP_MIDDLE_HORIZ (C)	Important	APD	Checks in place for floating point rounding errors in CFP_POSIT, which may cause failure of algorithm.
R 3.1.5-1	CFP_MIDDLE_VERT (C)	Important	APD	Round off problems have caused errors due to single precision accuracy as expressed in the comments.
R 3.1.5-2	CFP_MIDDLE_VERT (C)	Important	APD	Potential for incorrect number of roots for particular geometric situation returned by CFP_V_INT (a sub-function call by the middle filter).
R 3.1.5-3	CFP_MIDDLE_VERT (C)	Important	APD	Assumes prior filter check for equal adjusted interval cusp times.

REF#	Function Name	Impact Category	Module	Brief Description
------	---------------	-----------------	--------	-------------------

R 3.1.9-1	CFP_POSIT (C)	Important	APD	Assumes constant acceleration over the interval.
R 3.1.6-1	CFP_RELVEC (C)	Important	APD	Assumes constant acceleration for the segment.
R 3.1.8-1	CFP_V_INT (C)	Important	APD	Algorithm uses an ϵ value to define a time cutoff value of a conflict.
R 3.1.8-2	CFP_V_INT (C)	Important	APD	Both the aircraft altitude changes and boundaries are assumed linear.
R 3.4.17-2	GM_REGN (PL/I)	Important	APD	The choice of the cut off value for the number of algorithm iterations (<i>nrpt</i> =8).
R 3.4.18-1	GM_TSTPNT (PL/I)	Important	APD	Difference in ϵ and <i>ptsep</i> variable distances between C and PL/I version.
R 3.4.18-2	GM_TSTPNT (PL/I)	Important	APD	Difference in the TKM C version of the algorithm and the PL/I version, <i>ptsep</i> value was not used in C version.
R 3.4.22-1	ST_CHK_VP (PL/I)	Important	APD	Linear interpolation is used to estimate the specific coordinates of each intersection point.
R 3.4.22-2	ST_CHK_VP (PL/I)	Important	APD	Small parameter distances are used to assume intersection points are alike.
R 3.4.3-2	CNV_GRDSPD (PL/I)	Important	TJM	Uses large cross wind approximation.
R 3.4.6-2	CNV_SPEED (C)	Important	TJM	Models subsonic airspeeds only.
R 3.2.2-2	EGRAD (PL/I)	Important	TJM	Approximation in ground speed calculation.
R 3.4.21-1	ST_ARD_SSGDATA (PL/I)	Important	TJM	Assumes a constant acceleration over the entire length of the segment.
R 3.4.28-4	ST_IASALT (PL/I)	Important	TJM	There are no monitors to determine the number of “unattainable conditions” that occur when altitude constraints are exceeded.
R 3.4.29-4	ST_MACHALT (PL/I)	Important	TJM	There are no monitors to determine the number of “unattainable conditions” that occur when altitude constraints are exceeded.
R 3.4.32-1	ST_TIME_SSGDATA (PL/I)	Important	TJM	Assumes ground speed remains constant if true airspeed remains constant.
R 3.4.32-2	ST_TIME_SSGDATA (PL/I)	Important	TJM	Does not protect against division by a zero segment length.
R 3.4.34-1	ST_XYTOTIME (PL/I)	Important	TJM	Assumes a constant ground speed acceleration.

REF#	Function Name	Impact Category	Module	Brief Description
------	---------------	-----------------	--------	-------------------

R 3.3.12-1	TKM_GM_REGN(C)	Important	TKM	Algorithm efficiency difference between TKM version and general utility version
R 3.3.12-3	TKM_GM_REGN(C)	Important	TKM	The choice of the cut off value for the number of algorithm iterations (<i>nrpt=8</i>).
R 3.3.13-1	TKM_GM_TSTPNT (C)	Important	TKM	Difference in ϵ variable distances between C and PL/I version.
R 3.3.13-4	TKM_GM_TSTPNT (C)	Important	TKM	Difference in the TKM C version of the algorithm and the PL/I version, <i>ptsep</i> value was not used in C version.
R 3.3.15-1	TKM_TK_HDG (C)	Important	TKM	The function does not protect against a β_1 value equal to zero in a denominator.
R 3.3.15-2	TKM_TK_HDG (C)	Important	TKM	Assumes all headings are with respect to true North.
R 3.1.1-2	CFP_COARSE_HORIZ (C)	Minor	APD	Since aircraft is assumed at a point in a holding pattern, the strip distance is calculated using a slightly different perpendicular line than the other cases.
R 3.1.1-3	CFP_COARSE_HORIZ (C)	Minor	APD	Approximation of minimum segment length to determine if aircraft is in a hold.
R 3.1.1-4	CFP_COARSE_HORIZ (C)	Minor	APD	Misleading comments for the description of perpendicular distances.
R 3.1.3-1	CFP_INTERSECT_TIME (C)	Minor	APD	Assumes no acceleration is present since only called for relative velocity calculation.
R 3.1.4-4	CFP_MIDDLE_HORIZ (C)	Minor	APD	The dot product of V is not the normal of the velocity vector. It is the squared magnitude of the relative velocity vector. Comment needs adjustment.
R 3.1.6-2	CFP_RELVEC (C)	Minor	APD	Assumes current relative velocity vector if no acceleration is present.
R 3.1.7-1	CFP_TRIM (C)	Minor	APD	Assumes all aircraft segments that enter algorithm have overlapping time intervals.
R 3.1.7-2	CFP_TRIM (C)	Minor	APD	All accuracy and calculation specifically carried in CFP_POSIT algorithm.
R 3.1.8-3	CFP_V_INT (C)	Minor	APD	The variable names, <i>zll</i> , <i>zll</i> , <i>zul</i> , and <i>zul</i> , are very difficult to distinguish between when reviewing the code.

REF#	Function Name	Impact Category	Module	Brief Description
R 3.4.14-1	GM_CONVEX (C)	Minor	APD	Comment is incorrect and needs

				correction.
R 3.4.15-1	GM_INSEC (C)	Minor	APD	Assumes the coordinates are only positive.
R 3.4.15-2	GM_INSEC (C)	Minor	APD	Floating point adjustments need documentation for the function.
R 3.4.17-1	GM_REGN (PL/I)	Minor	APD	The ε value provides the ratio of the segment distance which considers a random point is on the polygon.
R 3.4.22-3	ST_CHK_VP (PL/I)	Minor	APD	A consistent global variable should be used as the small parameter distance
R 3.4.3-3	CNV_GRDSPD (PL/I)	Minor	TJM	Approximation in derivation of ground speed.
R 3.4.4-1	CNV_LLXY (PL/I)	Minor	TJM	The point being converted is sufficiently near the point of tangency.
R 3.4.4-2	CNV_LLXY (PL/I)	Minor	TJM	The function unnecessarily calculates the $\cos \phi_{0g}$ and $\cos \phi_g$.
R 3.4.4-3	CNV_LLXY (PL/I)	Minor	TJM	The check for bounds on the cosine function is unnecessary.
R 3.4.4-4	CNV_LLXY (PL/I)	Minor	TJM	The conformal latitude of the point of tangency can be a stored value.
R 3.4.6-1	CNV_SPEED (C)	Minor	TJM	Assumes no instrument error.
R 3.4.6-3	CNV_SPEED (C)	Minor	TJM	Air flow is isentropic and compressible.
R 3.4.7-1	CNV_STD_ATMOS (PL/I)	Minor	TJM	Approximates the gravitational acceleration as a constant, independent of altitude.
R 3.4.7-2	CNV_STD_ATMOS (PL/I)	Minor	TJM	Assumes that the geopotential altitude will not exceed 82021 ft.
R 3.4.9-1	CNV_XYLL (PL/I)	Minor	TJM	The conformal latitude and colatitude of can be calculated only once and stored.
R 3.4.9-2	CNV_XYLL (PL/I)	Minor	TJM	The variable names ALPHA and DLATC are used for multiple variables.
R 3.4.9-4	CNV_XYLL (PL/I)	Minor	TJM	It is assumed that neither the point being converted nor the point of tangency are at the north pole.
R 3.4.16-1	GM_PTLNE (PL/I)	Minor	TJM	Assumes larger line segment for x.

REF#	Function Name	Impact Category	Module	Brief Description
R 3.4.28-3	ST_IASALT (PL/I)	Minor	TJM	The function will not model aircraft trajectories above 60,000 ft or below 0 ft.
R 3.4.29-3	ST_MACHALT (PL/I)	Minor	TJM	The function will not model aircraft trajectories above 60,000 ft or below 0 ft.
R 3.3.12-2	TKM_GM_REGN(C)	Minor	TKM	The ϵ value provides the ratio of the segment distance which considers a random point is on the polygon.

4.4 Suggested Improvements

It is expected that with prototype code there will be areas of dead code, lack of proper error trapping, inconsistent variable definitions, inaccurate source code comments and even some low impact logic errors. While reviewing the source code of the URET D1.1 algorithms, several anomalies were observed and recorded. This section lists the name of the source code modules where anomalies were observed, a brief description of the anomaly, and recommended solutions (if any) to the problem. If the problem was also described in the assessment table of the module, the reference number used in the assessment table is included.

This section provides the developer an easy way to identify and trace any coding issues raised in this report. It assumes that the reader has some knowledge of the concepts in these functions and its code structure. For a more detailed description of each of the referenced source code modules, refer to Section 3.

4.4.1 CFP_COARSE_HORIZ

There are misleading comments and documentation description of Case 2 and 3 perpendicular distance. The numerators: Z3, Z4, H3, and H4 are not equivalent to Case 1 perpendicular, but defined as the adjacent side of the right triangle (i.e. A1 to P to B1). The code needs more descriptive comments and documentation for the use of the adjacent side distance. This comment addresses the code's clarity and readability not it's performance. (See R 3.1.1-4)

4.4.2 CFP_MIDDLE_HORIZ

As suggested in the code's comments, the Q dot Q should be checked and if either are less than m, the function should result in a detected conflict. The comments suggest that only one vector check is sufficient, however if round off problems are present both vectors should be checked against m. The comments also state that the check is for the "norm" equal to zero, however the V dot V (vdotv variable) is equivalent to the magnitude of the relative velocity squared not the normal vector. If this magnitude is equal to zero, the position vectors P and Q should be equivalent, since there is no relative movement for the time interval of the flight segment (the aircraft would be trailing or parallel). (See R 3.1.4-4).

4.4.3 CFP_V_INT

The variables should be renamed for better readability. For the definition of zll, z1l, zul, and zu1, the variable names chosen are very difficult to distinguish between. For traceability and clarity changing the names or using capitol letters would be much more appropriate. (See R 3.1.8-3)

4.4.4 CNV_LLXY

- (1) The points that can be stereographically projected from a sphere are limited to the hemisphere centered on the point of tangency. The coordinate point being converted by the function must be within 90 degrees of the point of tangency. For robustness, the function should do this bounds check before proceeding with the calculation of X and Y. (See R 3.4.4-2)
- (2) The function unnecessarily calculates the cosine of the geodetic latitude of the point of tangency and the cosine of the geodetic latitude of the point being converted. This code should be deleted. (See R 3.4.4-3)
- (3) The bounds check on the cosine function is unnecessary because the bounds check has already been run on the sine calculation. (See R 3.4.4-4)

4.4.5 CNV_SPEED

The code which determines the geopotential altitude in the beginning of this module is never used and should be eliminated.

4.4.6 CNV_XYLL

- (1) The calculation of the conformal latitude of the point of tangency of the stereographic plane by CNV_LLXY and CNV_XYLL should be done only once for a given ARTCC and the results saved for future use. (See R 3.4.9-2)
- (2) Distinct variables should have distinct variable names. (See R 3.4.9-3)
- (3) Neither the point of tangency of the stereographic plane nor a coordinate point being converted may be at the north pole. The function should check its input data for these two cases. (See R 3.4.9-5)

4.4.7 GM_CONVEX

The method description listed in the comment section of the function states that the test point is inside the polygon if the Q determinant is less than or equal to zero; the code does exactly the opposite. (See R 3.4.14-1)

4.4.8 GM_INSEC

- (1) The function determines if the line equations for two lines are equivalent and thus collinear when the sum of the x coordinates is equal to zero.

$$xs = x_1 + x_2 + x_3 + x_4 = 0$$

For the x values to sum to zero, they either all must be zero or the variables must have both positive and negative values. Unless there are other assumptions relating to the source of the x coordinates, the sum and the equivalent slopes do not ensure that the lines are collinear. This check may only be an error trap for all zero values for the x coordinates and used for single precision arithmetic, but this assumes all the x coordinates are positive (in the first quadrant). Therefore, there is no reason for keeping this portion of the source code at this time.

This is probably a minor impact in APD since the consequence may produce either a parallel or collinear line which results in the same outcome in only one APD function call, the CFP_FINE function. However, the impact of GM_INSEC's assumptions on other module's functions is yet to be determined. (See R 3.4.15-1)

- (2) In GM_PTLNE, another approach was used to calculate the intersection point to a line. In summary, GM_PTLNE uses the point slope equation of the line to find the intersection point. A ratio was not used in this algorithm to determine if the intersection took place inside the line segment, but a simple check in the x coordinates was utilized. Since both approaches accomplish the same results, the simpler, more efficient approach should be the only method used (probably the GM_PTLNE approach).
- (3) The check for the intersection of the lines, the check for parallel/collinear line pairs, and the final determination of the intersection point all incorporate adjustments to minimize the effect of floating point arithmetic error in single precision. The problem is that these adjustments are undocumented in the code. This function needs more documentation or comments explaining these adjustments. (See R 3.4.15-2).

4.4.9 GM_PTLNE

The source code which accounts for the case when the line segment is determined to be neither vertical or horizontal, checks if the intersecting x value falls within

$$(x_1 - 1) \leq x_{\text{int}} \leq (x_2 + 1)$$

or

$$(x_2 - 1) \leq x_{\text{int}} \leq (x_1 + 1)$$

This assumes a larger line segment in the x dimension than what actually exists. This was done to correct for the inaccuracy of the single precision assignments. Future revisions of this module should use only double precision and eliminate the extensions the x dimensions. (See R 3.4.16-1).

4.4.10 ST_CHK_VP

The small parameter value 1000 feet / 10 seconds should be defined by a global (or shared) variable to be consistent with other algorithms and improve the readability of the code. (See R 3.4.22-3).

4.4.11 ST_FIND_ARD

An incorrect statement was found in the code. For the second loop where the GM_PTLNE found the closest approach point not on the flight segment, the distance calculation for the first end point has an error. The second term for the y dimension of the first end point in the code is listed as y_2 where it should be y_1 . The result of this error could return an incorrect ARD by as much as one segment length, since the minimum and maximum ARD values restricts the search. This may not cause a current problem in APD and TJM since all calls are from ST_CHK_VP which never result in a calculation in the second loop. (See R 3.4.27-1).

4.4.12 ST_IASALT

- (1) This function assumes that true airspeed and acceleration exist completely in the horizontal plane and have no component in the vertical dimension. This assumption is probably made because the descent/ascent angles are usually small resulting in very small vertical velocity and acceleration components. However, there are cases when the angle of climb or descent is significant (i.e. as much as 25°). By ignoring the vertical component of true airspeed and acceleration, these values could be in error by as much as 6%.

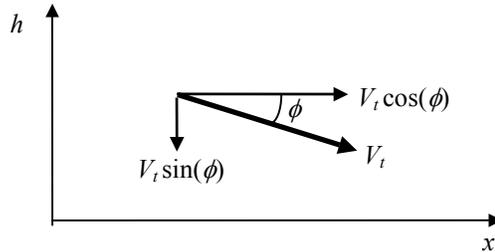
The solution to this anomaly is to address the vertical and horizontal components of true airspeed separately. The angle of descent or ascent is simply a trigonometric function of the gradient.

$$\phi = \arcsin(g)$$

The horizontal and vertical components of true airspeed are then expressed as

$$\text{vertical: } V_t \sin(\phi)$$

$$\text{horizontal: } V_t \cos(\phi)$$



This approach should also be used when the acceleration, a , is calculated. Acceleration could then be expressed in both horizontal and vertical components, a_h and a_v , respectively. Subsequently, the equation which determines the test altitude would be written as

$$h_t = h + \left[V_t \cos(\phi)t + \frac{1}{2} a_h t^2 \right] g$$

(See R 3.4.28-1).

- (2) This function calls the GM_QUADRATIC function to determine quadratic coefficients. GM_QUADRATIC returns the number of real roots (either 1, 2 or 0) and the corresponding values. ST_IASALT will then use the smallest, non-negative root as the test time, t , however there is no error condition handler here if GM_QUADRATIC returns 0 real roots. This error could cause an infinite loop since the variable TT would never change its value. It may be feasible for this condition to exist for a slow, decelerating aircraft climbing several thousand feet (i.e. practicing stalls), where

$$V_t^2 < 2a \frac{(h - h_t)}{g}$$

(See R 3.4.28-2)

- (3) The function contains error trapping routines that check for impossible situations, such as a climbing aircraft exceeding its maximum allowable altitude or a descending aircraft going below an altitude of 0. If either of these conditions are true, it makes the determination that the target IAS is unattainable with the given constraints and terminates the function. The last h_t and t that were calculated are then returned from this function. It should be noted that this “unattainable condition” does not record an error code. Therefore it is unknown how many times this situation takes place. (See R 3.4.28-4)

4.4.13 ST_TIME_SSGDATA

- (1) In the beginning comments of this module, under outputs, the description for FOUND states “Indicates whether the ARD exists on Route”. This should say “Indicates whether the XTIME exists during the route”.
- (2) There are areas in this module which do not protect against dividing by zero. This is a software (robustness) issue which could cause processing errors under the proper conditions. (See R 3.4.32-2)
- (3) If the True Airspeed acceleration is equal to zero

If $a_t = 0$,

Then assign the ground speed and true airspeed at the first cusp of the state segment equal to the ground speed and true airspeed at the given time.

Then

$$V_g = V_{g1}$$

$$V_t = V_{t1}$$

This appears to be an incorrect assumption. If the true airspeed acceleration is zero, the ground speed could still change over the segment because of changes in wind. The ground speed should be calculated with the ground speed acceleration calculated earlier in the code (i.e. Equation 3.4.32-1). A more accurate, however more processing intensive, method would be to use the same true airspeed along with the DB_AIR_AT_POINT, GM_BRNG, and CNV_GRDSPD functions to calculate the exact ground speed at the given time with the known wind conditions.

4.4.14 TKM_GM_REGN

The function converted from the earlier PL/I version is less efficient than the earlier code and may need further investigation on how it was reprogrammed in C. The original PL/I version of this function was written to only generate additional random points if an intersection was found too close to the end point. This function always runs *nrpt* times *n* iterations, while the PL/I version runs a maximum of *nrpt* times *n* iterations. This delta between the two versions will not effect the accuracy of the code, since only one random point sufficiently outside the polygon can be utilized to determine if the test point is inside the polygon. If all *nrpt* random points are generated, the result is the same, however, the code efficiency would be improved if only one were used. (See R 3.3.12-1)

4.4.15 TKM_GM_TSTPNT

The following items highlight some of the deficiencies when the code was converted from PL/I to C.

- (1) If the test point is less than a distance *pntsep* from the given line, the point is evaluated to be between the end points of the given line segment. However, the line is extended by \mathcal{E} for the TKM version of GM_TSTPNT in C, but for the PL/I version the line segment is extended only by *pntsep*. The *pntsep* value is 1 foot and the \mathcal{E} value is 100 feet. The transfer from C to PL/I will provide different results not because of coding in a different language, but because different comparison values are used. An investigation into the potential reasons for the change are necessary. (See R 3.3.13-1).
- (2) For each check (includes three in this function), the “else return(false);” should be added to protect against an undetermined return from the function. For example, the last case where the distance equation returned a value of zero because the test point is collinear with the line, the result will end the function without specifically assigning the value FALSE. The specific compiler by default may or may not assign a zero value (which will return the correct value) or the return value may be reinitialized before the call to this function, but this is not sufficient for portable ANSI C code. (NOTE: The original PL/I version was written differently to protect under this case.) (See R 3.3.13-2)
- (3) As a result of the unprotected return in the function for the horizontal line case, a horizontal line checked against a point outside the endpoints of the line segment but on the line will return a division by zero ($s1 = 0.0$ while $s2$ will be in error...). The corresponding problem is present for the vertical case as well. The original PL/I version had goto statements to protect under this case. This is not necessary, but a simple “else statement” with a return of false would protect against the problem. (See R 3.3.13-3).

- (4) The check carried out to determine if a point is between the end points of the line segment when the line segment is either vertical or horizontal uses the *pntsep* value to extend the lines under the PL/I version but not for the C version here. It is actually more accurate not to use the *pntsep* value, but this may cause errors due to round off during floating point arithmetic. Therefore, an investigation is required to determine why this was not used in this function. (See R 3.3.13-4).

4.4.16 TKM_TK_HDG

The function does not protect against a β_1 value equal to zero in the denominator of the inverse tangent function. This code could cause a floating point error while processing. (See R 3.3.15-1).

4.4.17 TKM_GET_RTE_ORIS

The function never uses the ORS_OFFSET value which is supplied as an input.

4.4.18 UTL_XY_ARD_BY_RTE

The source code comments and name of this function are misleading. The function does not calculate the ARD at the x, y position. It only computes the minimum distance from the given point to any point along the original route (RTE_ORIS).

4.5 Conclusions

This document reports the results of an assessment of the core algorithms found within the URET D1.1 source code. The source code of the algorithms that were assessed was found to be based on sound engineering principles. The assumptions and approximations made by MITRE/CAASD are reasonable for the current prototype software requirements.

Since the scope of this assessment was scaled back to an analytic assessment of the algorithmic source code modules, there is no empirical data derived from simulations or live data to validate the algorithms¹². Therefore, the assumptions and approximations should still be independently validated with a stringent set of simulations and live data tests to ensure the robustness and accuracy of the algorithms (e.g., Many assumptions have been made in the design of the URET trajectory prediction algorithms. These assumptions need to be validated by comparing the URET-predicted aircraft trajectories with the actual aircraft tracks reported by the HCS under a variety of scenarios).

The information provided by this report is valuable information to both the developer of the URET prototype and a production contractor. Section 3 bridges the documentation gap between the source code and existing software design and algorithmic definition documents. While the high level algorithmic functions were adequately documented by MITRE/CAASD, there are many algorithmic details not covered in the prototype documentation that cannot be easily derived by reading the source code. ACT-250's assessment approach of reviewing, analyzing, and often re-deriving the algorithm's mathematics revealed many of these undocumented assumptions and approximations; these details are now documented in this report. Sections 4.1 and 4.2 outline URET D1.1 limitations and assumptions, and Section 4.4 identifies suggested improvements to specific source code modules.

This report should be used as a source for any future independent assessments of the URET algorithms; particularly for sensitivity or algorithmic accuracy assessments. The assessment of the design and

¹² However in some cases during the assessment, several algorithms were rewritten in the C language. Limited unit testing was performed to validate the following functions: CNV_LLXY, CNV_XYLL, ST_MACHALT, ST_IASALT, CFP_POSIT, CFP_V_INT, CNV_SPEED, GM_REGN, GM_TSTPNT, ST_FINDARD, GM_CONVEX, GM_INSEC.

implementation of the algorithms should be completed for the current version of the URET prototype and this version of the software should be rigorously tested.

Appendix A: Simulation Experimentation

A.1 Simulation Environment

The URET system installed in the TATCA/AERA laboratory is depicted in Figure A.1-1. This system is functionally equivalent to the URET system installed at ZID, except that the TATCA/AERA laboratory URET system is a “scaled down version” of the fielded URET system. Specifically, the printer and interface to the National Weather Service depicted in Figure A.1-1 are not included, and only a subset of the full complement of workstations installed at ZID are contained in the laboratory (a system control position, a system supervisor position, and two sector controller positions (there are up to eight of these positions at ZID)). The URET system is interfaced to the Host Computer System (HCS) at the Technical Center via the same interface being used at ZID: the General Purpose Output (GPO) Interface Unit (GPOIU) with the associated HCS software patch.

A.2 Simulation Approach

A rigorous and efficient method of examining the effect of certain independent variables on one or more dependent variables can be accomplished by a well designed experiment. The designed experiment is achieved by manipulating the independent variables and studying the effect on the dependent variables. In ACT-250’s planned simulation experiment, the independent variables were the major factors associated with the assumptions and approximations which were identified as **Critical** to the algorithms during the algorithmic analysis (summarized in Section 4.3). The dependent variables are expressed by the performance measure under consideration. A test matrix (see Table A.2-1) was developed based upon these independent variables and would be used to design simulation scenarios to either verify or contradict these assumptions and approximations (see Section A.2.1.2). For example, to study the performance of the trajectory model algorithm (TJM) the difference between the trajectory predicted position of the aircraft and the actual track reporting point (defined as the track-to-trajectory deviation) is a measure of the performance. The smaller the value of the track-to-trajectory deviation, the better the performance of URET TJM.

Special types of designed experiments, called factorial experiments, are very useful in the analysis of a system’s performance. One of these is the factorial design with two response levels for each factor, usually at the extreme high and low levels of the factor. The factorial design would be used to determine the statistically significant factors on response variables and to estimate the average quantitative effects of these factors in terms of the performance variable. To minimize the amount of simulation runs and maximize the information gained by these runs, various factorial designed experiments would be applied to examine identified factors relating to several performance variables (refer to Section A.2.1.2).

Based on the test matrix (Table A.2-1), ACT-250 planned to design a set of test flight tracks to demonstrate the functions of the URET algorithms and to determine the robustness of the algorithms if the approximations and assumptions were tested using the extremes of the expected tolerances. A simulation capability would create the simulated aircraft tracks which would then be provided from the Technical Center HCS to the URET system in the TATCA/AERA laboratory via the GPOIU. The HCS would be run using the ZID system build in use with URET D1.1. The simulation-generated tracks would be the baseline “truth” during the simulations. All URET generated trajectories, reconformances, problem detections, etc. would be recorded and compared to the baseline simulation-generated tracks.

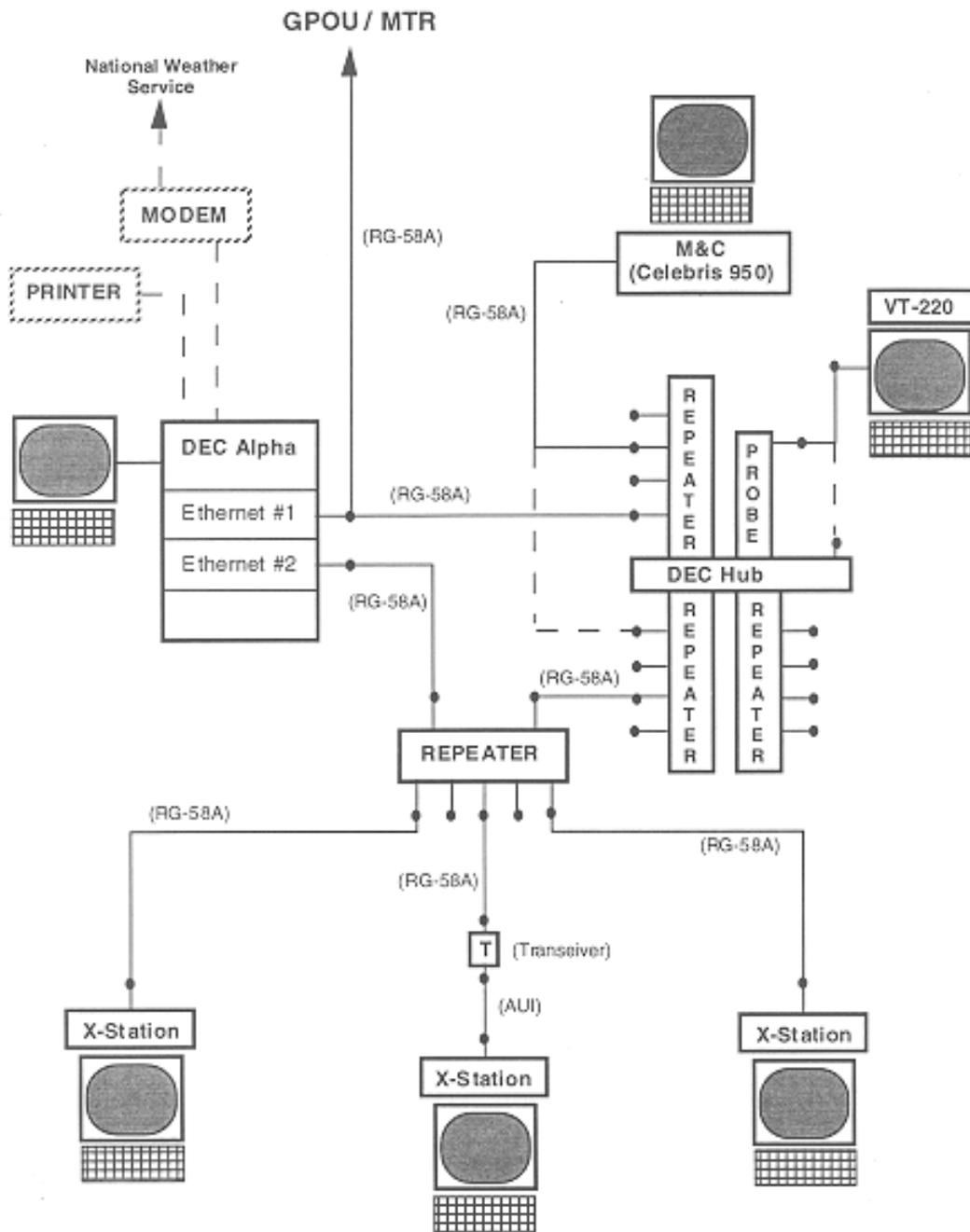


Figure A.1-1: TATCA/AERA Laboratory URET Configuration

REF#	Impact Category	Affected Algorithm	Relevant Factor:							Description of Misc.
			Gradient	Inertial Path Angle	Accel	Speed	Altitude Range	Flight State	Misc	
R 3.1.4-1	Critical	APD			X					
R 3.3.1-1	Critical	TJM	X	X						
R 3.4.3-1	Critical	TJM	X	X						
R 3.4.9-3	Critical	TJM							X	approximation of angles
R 3.2.2-1	Critical	TJM	X	X						
R 3.2.2-3	Critical	TJM	X	X						
R 3.4.17-3	Critical	APD							X	approximations associated to lower level algorithms (i.e. whether point is on a line)
R 3.4.23-1	Critical	TJM	X							
R 3.4.23-2	Critical	TJM	X							
R 3.4.24-1	Critical	TJM	X							
R 3.4.25-1	Critical	TJM	X							
R 3.4.25-2	Critical	TJM	X							
R 3.4.26-1	Critical	TJM	X							
R 3.4.27-1	Critical	APD, TJM							X	along route distance calculation
R 3.4.28-1	Critical	TJM			X	X				
R 3.4.28-2	Critical	TJM							X	zero routes in quadratic equation
R 3.4.29-1	Critical	TJM			X	X				
R 3.4.29-2	Critical	TJM							X	zero routes in quadratic equation
R 3.3.12-4	Critical	TKM							X	approximations associated to lower level algorithms (i.e. whether point is on a line)
R 3.3.13-2	Critical	TKM							X	unprotected return case
R 3.3.13-3	Critical	TKM							X	potential division by zero

NOTE: This matrix contains only the assessment items considered as critical under their impact category. This is presented as an example only. If both the important and minor items were presented, they would have provided additional relevant factors (e.g. flight state, altitude range, etc.).

Table A.2-1 Assessment Matrix

A.2.1 Simulation Design and Scenarios Development Example

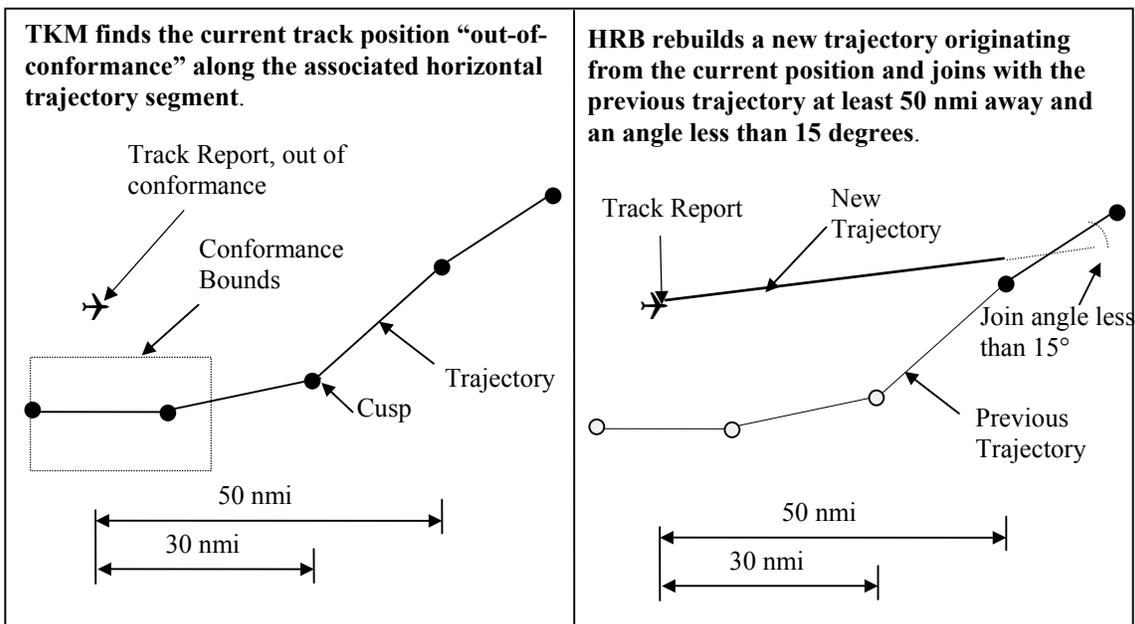
The following provides an example of the structured simulation scenarios which could be designed to exercise various aspects of the URET algorithms. There were two approaches planned for this assessment: Section A.2.1.1 presents an example of specific scenarios planned to validate and/or contradict referenced assessments considered to be **Critical** and the method by which the algorithms' performance was to be measured. Section A.2.1.2 describes a classical 2 level factorial experiment that could be conducted to measure the system responses to variations in global independent variables.

The basis for modeling these simulations is depicted in Table A.2-1, which is a collection of all of the critical items from the assessment tables defined in Section 3 and lists the important factors associated with the system.

A.2.1.1 Test Design Example for the HRB Function*

A.2.1.1.1 Definition

The Trajectory Modeler (TJM) routine Horizontal Route Analysis Step B (HRB) builds the remaining horizontal route segments from the current position of the aircraft. This routine is called when an aircraft is laterally reformed and a new trajectory must be calculated or a request for direct routing is submitted in the form of a trial plan.



In the case of a lateral reconformance, HRB will create a horizontal trajectory from the current position of the aircraft to the next horizontal route segment which is at least a minimum "join" distance (50 nmi) from the current position and a "rejoin" angle less than a maximum "join" angle (15 degrees). However, HRB will not bypass a fix with a delay.

* Note: The following test design was based on URET Version D1.A of the HRB function. There are significant changes to this function in D1.1, and the associated revisions are still being developed.

A.2.1.1.2 Example Development of Flight Track Scenarios

A.2.1.1.2.1 Validate the algorithm

A test flight track would be designed to validate that the algorithm functions as specified. A test would be designed to demonstrate the lateral reconformance utility of the HRB function. A simulated flight route would purposely deviate from its filed route and would head directly to the next down-route fix. **(1)**

A.2.1.1.2.2 Assumptions

- 1) The out-of-conformance aircraft would head to the next down-route fix that is greater than 50 nmi from the current position and a join angle less than 15 degrees.

Implications:

- a) The aircraft would not head directly to a fix less than 50 nmi from the current position 50 nmi from the current position. **(1a)**
- b) The aircraft would not head directly to a down-route fix which is farther than the next down-route fix that is more than 50 nmi from the current position and a join angle less than 15 degrees. **(1b)**
- c) The aircraft would not join a route at a down-route fix if the join angle is greater than or equal to 15 degrees. **(1c)**
- d) There is a down-route fix greater than 50 nmi from the current position **(1d)**
- e) There is a down-route fix with a join angle less than 15 degrees. **(1e)**
- f) There is a down-route fix greater than 50 nmi from the current position and a join angle less than 15 degrees. **(1f)**

A.2.1.1.2.2.1 Simulate Contradictions to the Assumptions

A separate simulated flight path would be created to contradict each of the implications derived from the assumptions. These tests would determine if the algorithms were robust enough to recover if an assumption is untrue.

A.2.1.1.3 Simulated Flight Tracks

Sim #	Description	Involves Alg. Set	Tests Assumption	Verifies or Contradicts Assumptions
S1	Sim. To verify the algorithm, assumption 1	TJM, TKM	1	Verify
S1a	Sim. To contradict assumption 1a	TJM [HRB]	1a	Contradict
S1b	Sim. To contradict assumption 1b	TJM [HRB]	1b	Contradict
S1c	Sim. To contradict assumption 1c	TJM [HRB]	1c	Contradict
S1d	Sim. To contradict assumption 1d	TJM [HRB]	1d	Contradict
S1e	Sim. To contradict assumption 1e	TJM [HRB]	1e	Contradict
S1f	Sim. To contradict assumption 1f	TJM [HRB]	1f	Contradict

Table A.2-2: Simulation Tests for the Trajectory Modeler (TJM) Horizontal Route Analysis Step B (HRB) function when it is invoked because of a Reconformance

S1. Simulation to validate the algorithm.

Input to the HOST a flight plan with a filed route which traverses a known set of fixes (i.e. ..Fix1.Fix2.Fix3.Fix4..). Supply HCS simulation capability with the same flight plan but with an altered route string (i.e. ..Fix1.Fix3.Fix4..). The filed flight plan in the HCS would be delivered to URET, and TJM would create a trajectory using the flight plan information. However, the simulation capability would supply the HCS with aircraft positions which correspond to the altered route string. Record when and where URET determines the aircraft is out-of-conformance. Determine if URET creates a new trajectory to the proper down-route fix . Figures AA.1-4 demonstrate this simulation.

S1 Route String

..Fix1.Fix3.Fix4..

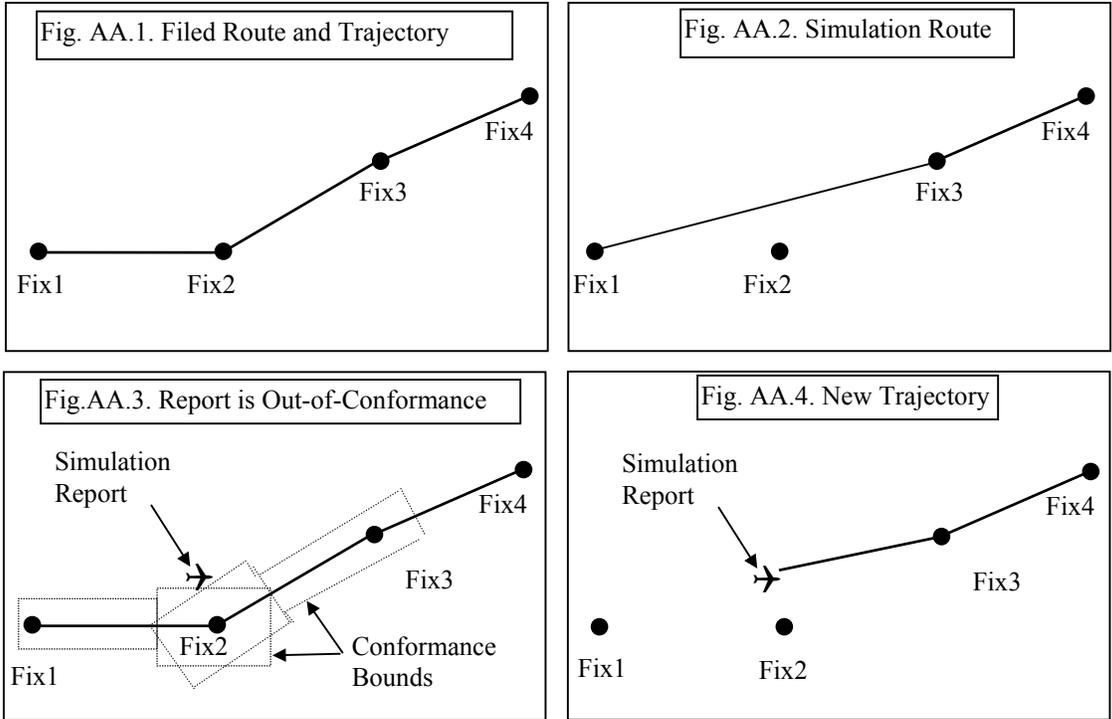


Figure AA.1 represents the filed route of the aircraft and the horizontal trajectory generated by URET (the Baseline Flight Plan Route String). Figure AA.2 represents the simulated route which the simulation capability will supply for the aircraft. Figure AA.3 shows an example of when the simulation capability report will fall out-of-conformance. It is here that TKM should recognize the deviation from the trajectory and ask HRB to regenerate a new trajectory. Figure AA.4 represents the new trajectory HRB should generate for the aircraft.

S1a. Simulation to contradict assumption 1a. This would simulate an aircraft heading directly to a fix (Fixa) which is less than 50 nmi from the current position. Fixa would be a fix only known to the simulation capability.

S1a Route String

..Fix1.Fixa.Fix3.Fix4..

A.2.1.1.4 Measurements

Trajectory-to-track deviations would be calculated during every simulation as a performance measurement of the Trajectory Modeler. Reconformances would also be recorded as a performance measurement of the TKM.

A.2.1.2 The Designed Experiment Approach

From the algorithm analysis described in Section 3, several factors were determined that may effect the performance of the trajectory. For the initial simulation experiment, these factors would include the acceleration of the aircraft, the gradient of the aircraft, the state of flight of the aircraft (i.e., turn in horizontal, climb in vertical), and the altitude range of the trajectory. The factorial experimental design described in Section A.2 should be applied (see Table A.2-3).

Factor Description	Assessment References (examples)	High Level	Low Level
aircraft acceleration	R 3.1.4-1, R 3.4.28-1, R 3.4.29-1	maximum acceleration for a/c	nominal acceleration for a/c
aircraft gradient	R 3.3.1-1, R 3.4.3-1, R 3.2.2-1, R 3.2.2-3	maximum gradient	nominal gradient
altitude range	R 3.4.7-2, R 3.4.28-3, R 3.4.29-3, R 3.4.29-4	> 50 FL	< 50 FL
aircraft flight state	general	maneuver (i.e. turn, climb, etc.)	level cruise

Table A.2-3: Definition of Factors and Levels

The experimental unit defines the measurement of the performance variable. The smallest experimental unit is the measure of track-to-trajectory deviation during the transition state of the aircraft (i.e., turn, climb, etc.). Since there may be multiple changes from one transition state to another during the flight, several measures could be made on one aircraft trajectory. For the initial TJM experiment, replications of each treatment combination could be achieved by utilizing information from several specific aircraft trajectories.

The factorial design can be used to determine the statistically significant factors on the response variable and to estimate the average quantitative effects of these factors in terms of the performance variable. This experiment intended to use track-to-trajectory deviation for the primary response variable, but other response variables were to be collected simultaneously, including:

- the number of false alerts and missed alerts for the APD algorithm
- the difference in absolute track data separation distances to the minimum trajectory-based aircraft separation distance reported by APD
- the number of reconformances per dimension

The APD statistics could provide additional information without running more treatments (or flights in URET). The actual experimental unit could be changed to include the entire trajectory, which would change the number of degrees of freedom of the experiment, but this would not require additional simulation runs.

The factors evaluated to be statistically significant could also be examined by estimating the average effect on the response variable. The average effect by the acceleration factor could be estimated by using a maximum acceleration rate compared to the nominal acceleration rate in all the treatment combinations. The cause for these effects are related to the assumption and approximation associated to the particular factor as determined in the algorithm analysis listed in Table A.2-1.

The specific mathematics involved with this factorial designed experiment are based on the model of the randomized block design at two levels. Table A.2-4 defines the specific variables. The formula that

follows is the mathematical model for a factorial experiment. This formula expresses the effects of the four factors and interactions as 16 treatment combinations (2^4).

No.	Factor Description	Factor Variable	Run Variable (at high level)
1	aircraft acceleration	A	a
2	aircraft gradient	B	b
3	altitude range	C	c
4	state of flight	D	d

Table A.2-4: Factor/Level Key

$$Y_{ijkl} = \mu + A_i + B_j + C_k + D_l + AB_{ij} + AC_{ik} + AD_{il} + BC_{jk} + BD_{jl} + CD_{kl} + ABC_{ijk} + ABD_{ijl} + ACD_{ikl} + BCD_{jkl} + ABCD_{ijkl} + \mathcal{E}_{ijkl}$$

The 16 treatment combinations represent the specific combination of factors and levels that should be performed in running the complete designed experiment. The treatment combinations, as well as the coefficients for the effects, are listed in Table A.2-5. The coefficients are used to calculate the sum of the squares for each contrast of the experiment¹³ (the contrast is a measure of the difference of a factor or combination of factors from the high and low levels). The effect of a factor or interaction of several factors is directly proportional to the contrast statistic. To calculate the contrast for the factor A, the results of the 16 treatment combinations were summed using the coefficients in the A column from Table A.2-5. The sum of the squares are calculated by squaring the contrast and dividing it by the product of the number of replications of each treatment by the number of treatments (2^f , where f is the number of factors). As illustrated in the ANOVA table (refer to Table A.2-6), the F statistic is calculated and compared to the Cumulative F Distribution.

¹³ Using Yates Method, the computation for the contrast and effects is described in detail in Hicks, Fundamental Concepts in the Design of Experiments, 1993.

Treatment Combination	Effect														
	A	B	AB	C	AC	BC	ABC	D	AD	BD	ABD	CD	ACD	BCD	ABCD
(1)	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1
a	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1
b	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1
ab	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
c	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1
ac	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1
bc	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1
abc	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
d	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1
ad	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1
bd	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1
abd	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1
cd	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1
acd	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1
bcd	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1
abcd	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A.2-5: Treatment Combinations Versus Effects

Source	Degrees of Freedom (dof)	Sum of Squares	MS	F
A	1	$(\text{contrast})^2/2n^4$	SS/(dof)	MS/error
B	1			
AB	1			
C	1			
AC	1			
BC	1			
ABC	1			
D	1			
AD	1			
BD	1			
ABD	1			
CD	1			
ACD	1			
BCD	1			
ABCD	1			
Error	$(n-1)(2^4)$			

Table A.2-6: ANOVA Table for a 4 Factor Factorial Designed Experiment

The simulation would provide the data for each treatment combination (Table A.2-6). The designed experiment would provide the analysis tool to determine which factors and interactions significantly effect the response variable. The example presented illustrates the application of a factorial designed experiment to the assessment of the URET algorithms.

Appendix B: LIST OF ACRONYMS

ACC	Aircraft Control Characteristics
ACD	Automated Conflict Detection/Aircraft Characteristics Directory
ACES	Adaptation Controlled Environment System
ACT	Active List
AERA	Automated En Route Air Traffic Control
AMC	Aircraft Modeling Characteristics
APD	Automated Problem Detection
APDIA	APD Inhibited Area
ARD	Along Route Distance
ARTCC	Air Route Traffic Control Center
ATC	Air Traffic Control
ATM	Air Traffic Management
AUD	Aircraft Unique Data
BAS	Blocked Airspace
CAASD	Center for Advanced Aviation System Development
CD	Clearance Directive
CFP	Conflict Probe
CTS	Central Track Store
ECB	Environmental Conflict Box
ECP	Environmental Conflict Probe
GPO	General Purpose Output
GPOIU	General Purpose Output Interface Unit
HCS	Host Computer System
HDO	Handoff
HRA	Horizontal Route Analysis
HRB	Horizontal Route Analysis Step B
IAS	Indicated Airspeed
IFR	Instrument Flight Rules
IPT	Integrated Product Team
JRC	Joint Resources Council
kts	Nautical miles/hour
MDL	Modeler
nm	Nautical Miles
NPB	Nominal Profile Builder
ORS	Onboard Route Segment
PA	Planned Action
PAR	Preferred Arrival Route
PDAR	Preferred Departure Arrival Route
PDR	Preferred Departure Route
RPM	Replan Manager
SSG	State Segment
SUA	Special Use Airspace
TAS	True Airspeed
TATCA	Terminal Air Traffic Control Automation
TJM	Trajectory Modeler
TK	Track
TKM	Track Management
UPR	User Preferred Routing
URET	User Request Evaluation Tool
ZID	Indianapolis ARTCC

REFERENCES

MITRE/CAASD Documentation

- AERA Algorithmic Specifications: Flight Plan Conflict Probe, Volume 3, Rationale and Mathematical Derivations of Algorithmic Enhancements* (MTR83W152-03), Frolow & Hannet, 1/85
- AERA Algorithmic Specifications: Trajectory Estimation* (MTR83W152), J. A. Kingsbury, 9/83
- AERA Automated Problem Detection (APD) Algorithmic Definition* (draft), M. Ricker, 6/95
- AERA Smoothing and Prediction* (F042-M-003), D. Brudnicki, 1/95
- AERA Track Management Algorithmic Definition* (draft), D. Brudnicki, 5/95
- AERA Trajectory Modeling Algorithmic Definition* (draft), P. S. Johnson, 5/95
- AERA Trajectory Modeling Algorithmic Guidance Document* (draft - F042-M-186), P. S. Johnson, 11/21/94
- Algorithm Evaluation Capability (AEC) Evaluation Plan* (F022-L-010), D. Brudnicki, 3/9/95
- AEC Set 1 Report* (F022-L-042), D. Brudnicki, 8/7/95
- AEC Set 2 Report* (F022-L-504), D. Brudnicki, 10/9/95
- AEC Set 3 Report* (F022-L-521), D. Brudnicki, 3/25/96
- Description of AERA Trajectory Modeling* (F042-M-124), P. S. Johnson, 8/15/94
- Performance Analysis Results for the User Request Evaluation Tool* (MTR96W0000066), W. C. Arthur et. al., 8/96
- Track Manager (TKM) Detailed Design* (F042-M-017), M. Tucker, 2/3/95
- URET Adaptation Data Definition* (MTR96W0000078), J. T. Cochrane, Jr. et. al., 9/96
- URET Automated Problem Detection Algorithmic Definition DRAFT* (MTR96W0000038), Dr. M. Ricker, 9/96
- URET Delivery Version 1A System Data Structures Overview* (F022-M-551), D. E. Bellamy & W. Poteat, 12/22/95
- URET Delivery 1 System Segment Specification* (WN95W0000078), J. Celio & S. Schultheis, April 1995 (and Change Packages - 7/10/95 & 9/18/95)
- URET Delivery 1A Track Manager (TKM) Subsystem SW Design* (F022-M-575), C. V. Fong, 1/30/96
- URET Delivery 1A Automated Problem Detection (APD) Subsystem SW Design* (F022-M-576), C. V. Fong, 1/30/96

URET Delivery 1A Trajectory Modeler (TJM) Subsystem SW Design (F022-M-570), J. Summers, 1/23/96

URET Delivery 1A High Level Subsystem Description (F022-M-554), V. Fong, 12/27/95

URET Delivery 1A System Event Thread Charts (F022-M-556), E. L. Williams, 12/20/95

Environment Builder User's Guide for URET Delivery 1 (F022-M-523), J. A. Summers, 11/13/95

URET Delivery 1A Replan Manager (RPM) Subsystem SW Design (F022-M-562), B. C. Giller, 12/20/95

URET Delivery 1A System Segment Specification (WN95W0000078-R), J.C. Celio et. al, 1/96

URET Delivery 1.1 System Testing: Test Report (WN96W0000066), R. Katkin, 7/96

URET Trajectory Modeling Algorithmic Definition Draft (MTR 96W-DRAFT), D. L. Bashioum & J. J. Mayo III, 6/96

URET Trajectory Modeling Algorithmic Definition (MTR 96W0000072), D. L. Bashioum & J. J. Mayo III, 9/96

URET with DSR System Level Requirements Draft (MTR-96W-0000049), J. C. Celio et. al., 7/96

Other Documentation

NAS-MD-312, *National Airspace System Configuration Management Document, Route Conversion and Posting*, U. S. Department of Transportation, Federal Aviation Administration.

NAS-MD-312, *National Airspace System Configuration Management Document, Multiple Radar Data Processing*, U. S. Department of Transportation, Federal Aviation Administration.

Anderson, John D. Jr., *Introduction to Flight, Third Edition*, McGraw-Hill, 1989.

Draper, N. and Smith, H., *Applied Regression Analysis, Second Edition*, New York, NY.: John Wiley and Sons, 1988, Chapter.

Hicks, Charles, R., *Fundamental Concepts in the Design of Experiments, Fourth Edition*, Saunders College Publishing, 1993.

Montgomery, Douglas C., *Introduction to Statistical Quality Control, Second Edition*, John Wiley and Sons, Inc., 1991.