FLIGHT CONTROL COMMAND GENERATION IN A REAL–TIME MISSION PLANNING SYSTEM

by

WILLIAM JOHN WALKER

Bachelor of Science, Electrical Engineering
Queen's University, Belfast (1980)

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

July 1990

Signature of Author_____
Department of Aeronautics and Astronautics
July 1990

Certified by_____
Prof. Wallace E. Vander Velde
Thesis Supervisor

Accepted by_____
Prof. Harold Y. Wachman
Chairman, Departmental Graduate Committee

FLIGHT CONTROL COMMAND GENERATION IN A REAL–TIME MISSION PLANNING SYSTEM

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the
degree of Master of Science
at the Massachusetts Institute of Technology

by

WILLIAM JOHN WALKER

July 1990

ABSTRACT

The task of a Mission Planning System may be defined as the specification of a desirable trajectory by means of a sufficiently detailed sequence of waypoints. However, if real–time execution of the desired trajectory is required, the Planner must communicate its intentions to the vehicle control system as a stream of essentially continuous commands. If the waypoint spacing is sufficiently small, commands can be generated using a curve–fitting process in conjunction with a tracking loop. However, construction of a suitable waypoint sequence will involve the minimization of a risk function with respect to a set of trajectory parameters. This thesis addresses the question of how control system commands can be generated directly by minimizing the risk with respect to parameters defining control command histories rather than spatial trajectories. A suitable method is presented and evaluated.

Thesis Supervisor: Wallace E. Vander Velde
Title: Professor of Aeronautics and Astronautics

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Professor W.E. Vander Velde for the opportunity to work on this project. I am especially grateful for the patient and insightful way in which he has given direction to my efforts.

The System Sciences Group at the Charles Stark Draper Laboratory has been an indispensable source of information about the Mission Planner and I am particularly indebted to Milt Adams, Bob Beaton and Jim Harrison for their many helpful suggestions.

I would also like to thank the friends at M.I.T. and elsewhere who have provided me with encouragement and support during the preparation of this thesis.

Table of Contents

Table of Contents (Continued)

List of Figures

Glossary of Symbols

Vectors and matrices are represented by lower and upper case boldface characters respectively.

a_{nc}	normal acceleration command
a_n	normal acceleration
a_0	normal acceleration at start of segment
ϕ_c	bank angle command
ϕ	bank angle
ϕ_0	bank angle at start of segment
x, y, z	components of aircraft position in reference frame
v_x, v_y, v_z	components of aircraft velocity in reference frame
a_x, a_y, a_z	components of aircraft acceleration in reference frame
τ_a	time constant of normal acceleration channel in F.C.S.
$\tau\phi$	time constant of bank angle channel in F.C.S.
g	acceleration due to gravity
$\mathbf{c_a}$	normal acceleration command coefficient vector
$\mathbf{c\phi}$	bank angle command coefficient vector
T_i	i th Chebyshev polynomial
t	time
λ	normalized time
t_i	time at start of segment
t_f	time at end of segment
$T_h(x,y)$	horizontal threat function
$T_v(z)$	vertical threat function
T(x,y,z)	composite threat function
J	risk function
s	distance along trajectory
d	vector of constrained variables
k	vector of constraint levels
p	vector of penalty functions
P	diagonal matrix of penalty functions

$\boldsymbol{\psi}$	vector of constraint violations
$\mathbf{M}$	matrix of constraint derivatives
J_a	augmented risk function
L	Lagrangian function
$F(\mathbf{x})$	scalar function of n–vector $\mathbf{x}$
$\mathbf{p_k}$	k th linear search direction
α_k	linear search parameter
$\mathbf{g_k}$	gradient of $F(\mathbf{x})$ at $\mathbf{x} = \mathbf{x_k}$
$\mathbf{G_k}$	Hessian of $F(\mathbf{x})$ at $\mathbf{x} = \mathbf{x_k}$
$\mathbf{H_k}$	inverse Hessian
$\mathbf{Q_k}$	Quasi–Newton update for $\mathbf{H_k}$
k_{alt}	proportionality constant in vertical threat function T_v
z_{ref}	desired altitude
Γ_i	intensity of i th threat source in T_h
φ_f	forward difference operator
φ_c	central difference operator
h	derivative step length
$T_f(h)$	truncation error in forward derivative using step length h
$C(\varphi_f,h)$	cancellation error in forward derivative using step length h
ϵ_a	bound on error in computed function values
E_t	normalized truncation error
E_c	normalized cancellation error
a_{bound}	upper bound on peak normal acceleration

CHAPTER 1

INTRODUCTION

An increasingly important issue in the design of future aerospace vehicle systems will be the need for autonomous or semi–autonomous operation under adverse environmental or mission–imposed conditions. ***Automatic Mission Planning Systems*** are therefore being developed which will formulate optimal strategies for the accomplishment of specific goals using prior knowledge of vehicle characteristics and a continuously updated database containing information about objectives and threats.

This research has its basis in the characteristics and requirements of a Mission Planning System which is under development at the Charles Stark Draper Laboratory. The organization of the vehicle–independent portion of the planning function is illustrated in Figure 1.1. At the highest level is a ***Goalpoint Planner*** which uses knowledge about overall mission objectives to generate a far–term plan consisting of an ordered sequence of goals and their geographic locations along with associated constraints on time and energy. The second level may be characterized as a ***High–Level Trajectory Planner*** because it identifies maximum survivability flight paths between goals along with the basic aircraft mode sequences and subsystem actions that are necessary for execution of the desired flight path. At this level, the information upon which the plan is based will consist primarily of data relating to major threat concentrations, operational facilities and weather systems which have a direct bearing on the basic feasibility of the mission. Flight paths are specified by waypoints whose disposition is such that major threat concentrations are avoided. The primary output from the high level planner is therefore a sequence of waypoints which have been determined to be consistent with successful completion of the mission. They are located as close

together as possible, but not so close that there is any doubt about the physical ability of the vehicle to follow them exactly while remaining within its dynamic envelope. It is not the function of these waypoints to specify a detailed trajectory; merely to ensure that intermediate goals are met and that large–scale threat concentrations are avoided. Nor is it necessary for the aircraft to fly through them exactly; it will normally be possible to achieve the mission goals if the trajectory passes within some non–zero ***Capture Radius*** of each waypoint. Each capture radius is therefore a measure of how important it is to get close to the associated waypoint.

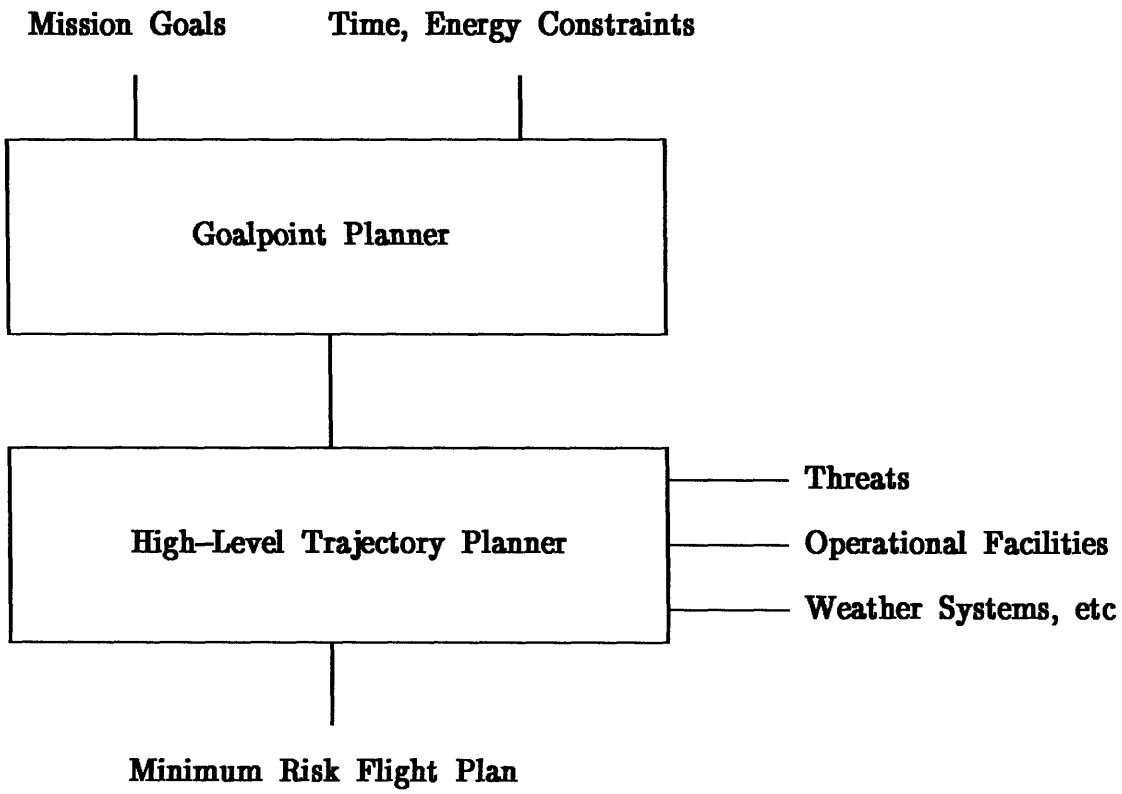


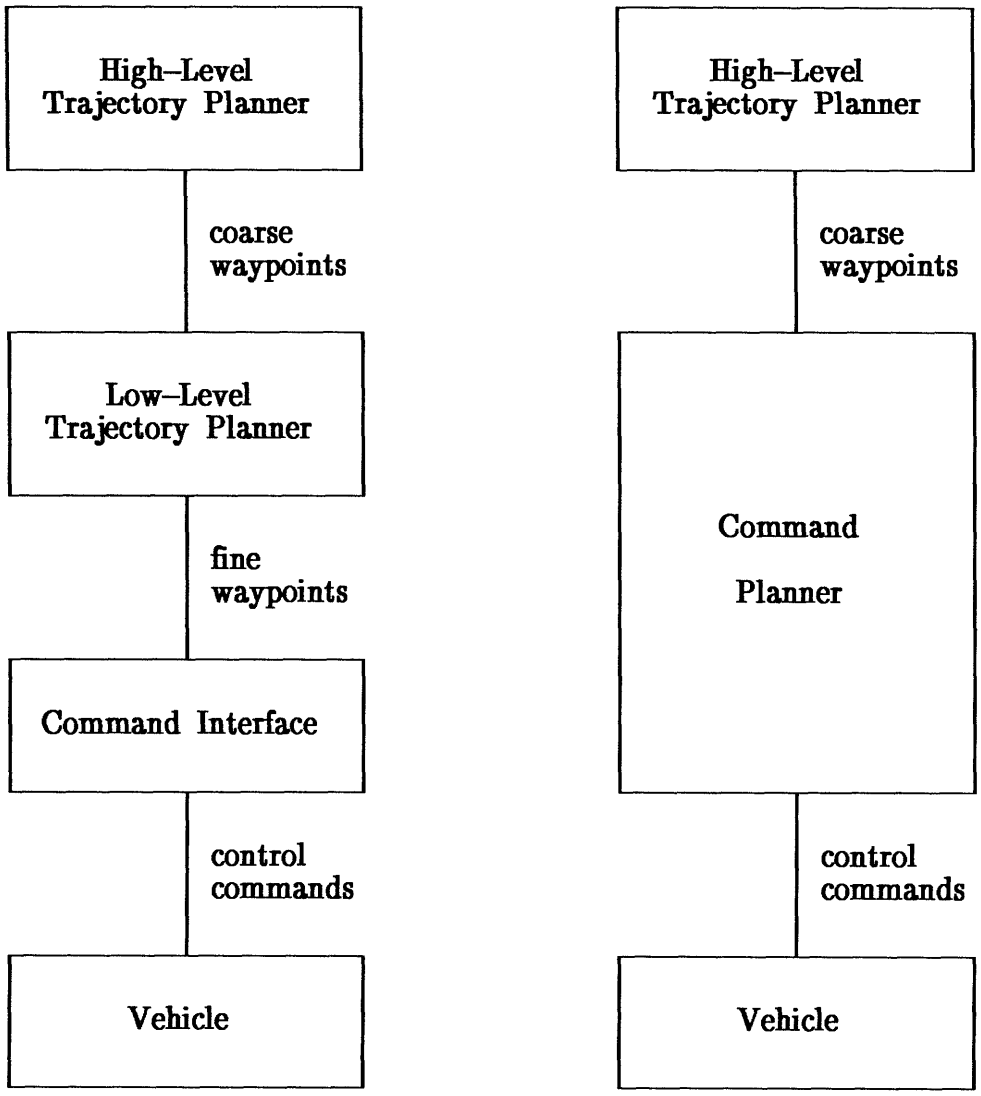
Figure 1.1 Mission Planning System

In the related research of Alexander[1], it was assumed that a third vehicle–independent level within the hierarchy called the ***Low–Level Trajectory Planner*** was capable of constructing a minimum–risk trajectory from the high–level waypoints in the form of a detailed sequence of waypoints with a separation in the order of 100 meters. (Such detailed planning is predicated on the existence of a correspondingly detailed threat database.) For autonomous operation, execution of the desired trajectory calls for communication between the Planner and the vehicle control system. Specifying trajectories with waypoints is natural from the point of view of the planning function, but the control system requires continuous commands. Alexander has shown how the Planner/Control System interaction can be regarded as an interfacing task and has designed a ***Command Interface*** which will generate appropriate commands by a curve fitting process in conjunction with a tracking loop. The operation of the Command Interface and its relationship with the low–level trajectory planner is illustrated in Figure 1.2 (a).

Implicit in the interfacing approach is the assumption that the trajectories supplied by the low–level planner are consistent with the dynamic limitations of the vehicle. Consistency alone can be assured by imposing *a posteriori* constraints such as curvature limits on the unconstrained minimum–risk trajectory but it is entirely possible that a better solution could be found if more complete knowledge of the dynamics of the vehicle and its control system had been incorporated in the search process from the outset. If dynamical equations relating the controls to the vehicle state are available, and if a suitable space of smooth control functions can be spanned by a finite number of parameters, and if a suitable ***Risk Function*** is defined at every point in this space then it is possible to search for the required commands directly by minimizing the risk function with respect to the parameters. This approach has the advantage that the minimization process no longer ignores the vehicle–planner interaction because a model of the dynamics is embedded in the process of evaluating the risk function. By the same token, a separate Command Interface is no longer necessary because the planner outputs are now inherently compatible with the control system. This integration of the planning and interfacing functions is achieved at the expense of increased complexity in the cost function. It is relatively simple to establish a physically meaningful measure of the risk associated with a given trajectory in space because the threat distributions which govern risk are themselves spatial in character. However, if the risk associated with a given set of control system

commands is required, it is necessary first to propagate the commands through the dynamical equations to determine the associated trajectory before risk can be evaluated as before.

The purpose of the present research is to investigate how the techniques of constrained non–linear optimization can be used to design a ***Command Planner*** which will implement the same function as the Low Level Trajectory Planner and the Command Interface combined without going through the intermediate step of defining trajectories explicitly. The structure is illustrated in Figure 1.2 (b). Since the dynamics of the vehicle and its control system are necessarily embedded in the process of evaluating the risk function, this approach provides a framework within which the role of vehicle limitations in the selection of optimal commands can be explicitly recognized from the outset.



(a) Command Interface

(b) Command Planner

Comparison of Interfacing and Planning Approaches
to Generation of Flight Control Commands

Figure 1.2

CHAPTER 2

PROBLEM FORMULATION

2.1 Preliminary Assumptions

Mission Planning Systems have a variety of potential applications and many of the higher–level planning tasks can be performed without detailed knowledge of the vehicle characteristics. However, as the planner/vehicle interface is approached from these higher levels, the dynamical behavior of the vehicle becomes an increasingly important consideration. This is particularly true in the case of a high–speed vehicle where a significant distance may be covered in the time required to execute a typical maneuver. The present research is therefore based on the characteristics of a high performance military aircraft. To this end, the following specific assumptions are made:

- Nominal Speed: $250\ ms^{-1}$
- Initial Altitude: 200 m
- Maximum Normal Acceleration: 4 g

These conditions resemble those encountered when operating at low level in a Terrain Following/Terrain Avoidance mode.

2.2 Information Required By Command Planner

Real–time measurements of the position, velocity and acceleration of the vehicle are essential for implementation of the command planning task and it is assumed that they are provided by on–board sensors.

With the vehicle characteristics defined in Section 2.1, the minimum turning radius is about 1500 meters and so a nominal separation between the waypoints supplied by the High–Level Trajectory Planner in the order of 5000 meters is appropriate. Since planning takes place in real time and since there is a limit to the look–ahead capability of the system, it is assumed that only the next two waypoints are available to the Command Planner at a given instant, together with their respective capture radii.

An important high–level constraint which impacts the Command Planner directly is the requirement that the mission should be accomplished using a limited amount of fuel. It is assumed that the High–Level Planner is capable of using this overall restriction to derive a *time constraint* for each inter–waypoint segment using knowledge about engine characteristics and the operational environment.

To summarize, the information which is available to the Command Planner is as follows:

- Present Position, Velocity, Acceleration, time
- Coordinates of next two waypoints to be captured
- Capture Radius for each waypoint
- Approximate time available for each inter–waypoint segment

2.3 Flight Control System and Vehicle Dynamics

The command variables chosen for the Flight Control System are Normal Acceleration (a_{nc}) and Bank Angle (ϕ_c). When operating in TF/TA mode at speeds close to Mach 1, the normal practice is to maintain a constant power setting and so thrust is not considered to be an active control in this thesis. The reference directions for the aircraft body frame are taken to be Forward, Right and Down as depicted in Figure 2.3.1.

Motion can be controlled in three dimensions using forces normal to the

velocity vector. The sideslip angle and the angle of attack are neglected for simplicity so that the Forward body axis coincides with the direction of motion. The Flight Control System is modeled as a first order lag between each of the command variables a_{nc}, ϕ_c and the corresponding physical quantities a_n, ϕ as depicted in Figure 2.3.2.

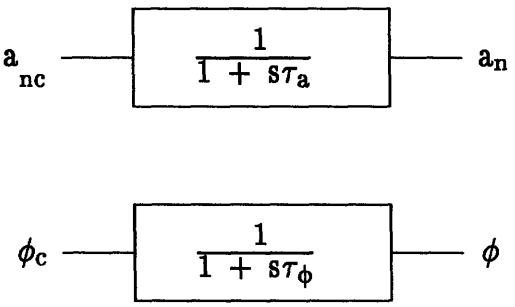


Figure 2.3.2 Flight Control System Model

Motion is referred to a flat earth reference frame whose z axis points vertically downwards and whose x and y axes lie in the horizontal plane. The relationship between the Normal Acceleration in the body frame and this reference frame is given by the following Euler Transformation:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = a_n \begin{bmatrix} \cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi \\ \cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi \\ \cos\phi \cos\theta \end{bmatrix} \qquad 2.3.1$$

where:

ϕ, θ, ψ = bank, pitch, yaw angles

a_x, a_y, a_z = acceleration components in reference frame

Combining the Flight Control System model with the Euler Transformation leads to the following state equations for the vehicle and its control system:

$$\dot{x} = v_x \tag{2.3.2}$$

$$\dot{y} = v_y \tag{2.3.3}$$

$$\dot{z} = v_z \tag{2.3.4}$$

$$\dot{v}_x = a_n (\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi) \tag{2.3.5}$$

$$\dot{v}_y = a_n (\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi) \tag{2.3.6}$$

$$\dot{v}_z = a_n \cos \phi \cos \theta + g \tag{2.3.7}$$

$$\dot{a}_n = \frac{1}{\tau_a}(a_{nc} - a_n) \tag{2.3.8}$$

$$\dot{\phi} = \frac{1}{\tau_\phi}(\phi_c - \phi) \tag{2.3.9}$$

where:

x, y, z = position in reference coordinates

v_x, v_y, v_z = velocity in reference coordinates

τ_a = F.C.S. time constant for normal acceleration channel

τ_ϕ = F.C.S. time constant for bank angle channel

g = Acceleration Due to Gravity

2.4 Representation of Command Functions

In order to be able to search for the commands which lead to a minimum risk trajectory, it is necessary to restrict the infinite set of all possible command functions to one which can be spanned by a finite number of parameters. This can be done by specifying the structure of each command as a function of time which has n degrees of freedom by virtue of its dependence on n parameters:

$$a_{nc} = a_{nc}(\mathbf{c_a}, t) \qquad \mathbf{c_a} \in \mathbb{R}^n \tag{2.4.1}$$

$$\phi_c = \phi_c(\mathbf{c_\phi}, t) \qquad \mathbf{c_\phi} \in \mathbb{R}^n \tag{2.4.2}$$

A useful way to structure the command functions is as a weighted sum of n basis functions which are orthogonal on the interval $[t_i,t_f]$ where t_i and t_f are the initial and final times associated with a particular trajectory segment. The Chebyshev Polynomials[2] defined by the following recursion are orthogonal on the interval [–1, 1]:

$$T_0(\lambda) = 1 \qquad 2.4.3$$

$$T_1(\lambda) = \lambda \qquad 2.4.4$$

$$T_{i+1}(\lambda) = 2\,\lambda\,T_i(\lambda) - T_{i-1}(\lambda) \qquad i \geq 1 \qquad 2.4.5$$

A basis which is orthogonal on $[t_i, t_f]$ is then obtained using the following change of variable:

$$\lambda = 2\,\frac{t - t_i}{t_f - t_i} - 1 \qquad 2.4.6$$

It was decided to use a 4th order representation (i.e. n=5) so that all possible commands can be represented as follows:

$$a_{nc}(\mathbf{c_a},\lambda) = c_{a1} + c_{a2}\,T_1(\lambda) + c_{a3}\,T_2(\lambda) + c_{a4}\,T_3(\lambda) + c_{a5}\,T_4(\lambda) \qquad 2.4.7$$

$$\phi_c(\mathbf{c_\phi},\lambda) = c_{\phi1} + c_{\phi2}\,T_1(\lambda) + c_{\phi3}\,T_2(\lambda)\; c_{\phi4}\,T_3(\lambda) + c_{\phi5}\,T_4(\lambda) \qquad 2.4.8$$

where:

$$\mathbf{c_a} = [\; c_{a1}\; c_{a2}\; c_{a3}\; c_{a4}\; c_{a5}\;]^T \qquad 2.4.9$$

$$\mathbf{c_\phi} = [\; c_{\phi1}\; c_{\phi2}\; c_{\phi3}\; c_{\phi4}\; c_{\phi5}\;]^T \qquad 2.4.10$$

The command functions must be continuous at the start of a given segment and so it will be necessary for them to take prescribed values a_0 and ϕ_0 at $t=t_i$. Using Equation 2.4.6–2.4.8 to evaluate a_{nc} and ϕ_c at $t=t_i$ indicates that the command coefficients are subject to the following constraints:

$$a_0 = c_{a1} - c_{a2} + c_{a3} - c_{a4} + c_{a5} \qquad 2.4.11$$

$$\phi_0 = c_{\phi 1} - c_{\phi 2} + c_{\phi 3} - c_{\phi 4} + c_{\phi 5} \qquad 2.4.12$$

Since any selected element from either $\mathbf{c_a}$ or $\mathbf{c_\phi}$ can now be expressed in terms of the other four, it is clear that imposition of a prescribed initial value on each command eliminates one of its degrees of freedom so that the dimension of the space of all admissible commands is 8 (rather than 10).

2.5 Threat Distribution and Risk Function

It is assumed that a non–negative three dimensional function T(x,y,z) is defined at every point in the vicinity of the aircraft's present position. The value of T(x,y,z) is a measure of the threat to which the aircraft is exposed when its position is (x,y,z). The ***Risk*** J of an arbitrary trajectory ζ is defined as the integral of the threat function with respect to distance s along its length:

$$J = \int_{\zeta} T(x,y,z)\, ds \qquad 2.5.1$$

The optimal trajectory which is sought has the property that it minimizes J subject to certain constraints which will be specified later. Unconstrained minimization of J always leads to the trivial result $J^*=0$ and so the trajectory must always be constrained to have non–zero length to ensure a physically meaningful problem.

For a given set of command functions defined by $\mathbf{c_a}$ and $\mathbf{c_\phi}$, the corresponding trajectory can be found by integrating the dynamics given in Equations 2.3.2–2.3.9 and it is therefore possible to express the position histories formally as follows:

$$x = x(\mathbf{c_a}, \mathbf{c_\phi}, t) \qquad 2.5.2$$

$$y = y(\mathbf{c_a}, \mathbf{c_\phi}, t) \qquad 2.5.3$$

$$z = z(\mathbf{c_a}, \mathbf{c_\phi}, t) \qquad 2.5.4$$

Further simplification can be effected by noting that from Equations 2.4.11 and 2.4.12, only four of the five elements in each of the vectors $\mathbf{c_a}$ and $\mathbf{c_\phi}$ are independent and that it is therefore possible to write:

$$x = x(\mathbf{c},t) \tag{2.5.5}$$

$$y = y(\mathbf{c},t) \tag{2.5.6}$$

$$z = z(\mathbf{c},t) \tag{2.5.7}$$

where:

$$\mathbf{c} = \left[\; c_{a2} \; c_{a3} \; c_{a4} \; c_{a5} \; c_{\phi 2} \; c_{\phi 3} \; c_{\phi 4} \; c_{\phi 5} \; \right]^T \tag{2.5.8}$$

This means that the risk associated with a particular trajectory segment beginning at $t=t_i$ and ending at $t=t_f$ can be written as follows:

$$J(\mathbf{c}) = \int_{t_i}^{t_f} T \left[x(\mathbf{c},t), y(\mathbf{c},t), z(\mathbf{c},t) \right] \frac{ds}{dt} \, dt \tag{2.5.9}$$

where:

$$\frac{ds}{dt} = \left[\left[\frac{dx}{dt} \right]^2 + \left[\frac{dy}{dt} \right]^2 + \left[\frac{dz}{dt} \right]^2 \right]^{\frac{1}{2}} \tag{2.5.10}$$

It is therefore possible to search for desirable commands by minimizing J with respect to the vector **c** which contains the free command coefficients.

2.6 Problem Statement

Under the conditions described in Section 2.2, the core problem to be solved by the Command Planner at any given time is to generate a segment of the complete command history that will cause the aircraft to capture the next two waypoints ahead of its present position while minimizing the risk as defined in Section 2.5.

A typical scenario is illustrated in Figure 2.6.1.

Since both the waypoints and the threat environment are subject to change as the mission proceeds, it is desirable to update the solution to this problem as rapidly as possible. Each time an update becomes available, the unused portion of the preceding segment is discarded in favor of the new one and continuity is ensured by using the measured position,velocity and acceleration of the aircraft as initial conditions. The question of how to allow for the computational delay between the start of the update when the initial conditions are valid and the point at which the commands become available is not addressed here in detail. The initial conditions used in the minimization will be predicted values obtained by extrapolating from the measurements which are available when the calculation begins and so it is necessary that the processing delay should be small in relation to the dominant time constant of the vehicle. In this thesis, it is assumed that the environment does not change appreciably in the time taken to cover one inter–waypoint distance (approximately 20 seconds at 250 m/s) and so it will be sufficient to compute a new solution each time the aircraft passes a new waypoint. Thus, each command segment will normally be executed only up to the point where the trajectory has its point of closest approach with respect to the first waypoint. A new solution for the next two–waypoint segment will then be obtained which will be executed as far as the next waypoint – and so on. This process is illustrated in Figure 2.6.2. The fact that the next two waypoints are always available tends to reduce the need for unnecessarily sharp turns.

In Section 2.2 it was pointed out that each waypoint pair has an associated time constraint which derives ultimately from the fuel allocation for the entire mission. It is reasonable to assume that the fuel allocation will be conservative so that it is not necessary to insist on precise compliance with the time constraint for each segment. Nor is it desirable to do so because there will frequently be situations where the pressing need to avoid threats should be allowed to override the need to comply with a nominal time constraint in the short term. Accordingly, the time constraint is used to fix the duration of each 2–waypoint segment but the time to reach the point of closest approach with respect to the first waypoint is allowed to vary freely depending on the needs of the risk minimization process. This approach provides for a limited amount of freedom in the inter–waypoint timing while still causing the time constraint to be loosely enforced.

The core problem may be stated in words as follows:

Given initial values for position, velocity, normal acceleration and bank angle, find the normal acceleration and bank angle command histories that will cause the aircraft to pass within specified capture radii corresponding to each of the next two waypoints with minimum risk within a prescribed time and subject to the constraint that the Normal Acceleration shall not exceed a predetermined maximum value.

Mathematically, the problem may be stated in the following form:

Minimize:

$$J(\mathbf{c}) = \int_{t_i}^{t_f} T\,[x(\mathbf{c},t),y(\mathbf{c},t),z(\mathbf{c},t)]\,\frac{ds}{dt}\,dt \qquad 2.6.1$$

Subject to:

$$d_1 - k_1 \leq 0 \qquad 2.6.2$$

$$d_2 - k_2 \leq 0 \qquad 2.6.3$$

$$d_3 - k_3 \leq 0 \qquad 2.6.4$$

where:

t_i = initial time for current segment

t_f = final time for current segment (based on fuel allocation)

d_1 = minimum miss distance relative to waypoint 1

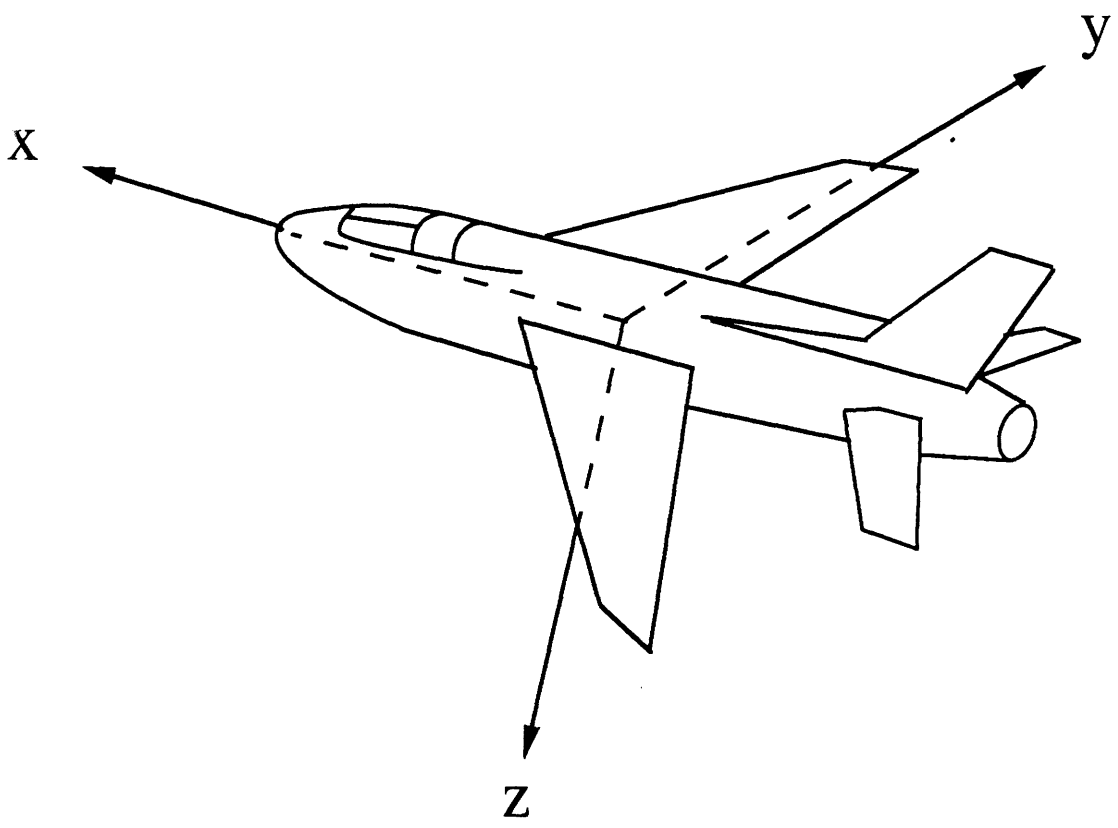
d_2 = miss distance relative to waypoint 2 at $t=t_f$

d_3 = upper bound on absolute value of maximum acceleration

k_1 = capture radius for waypoint 1

k_2 = capture radius for waypoint 2

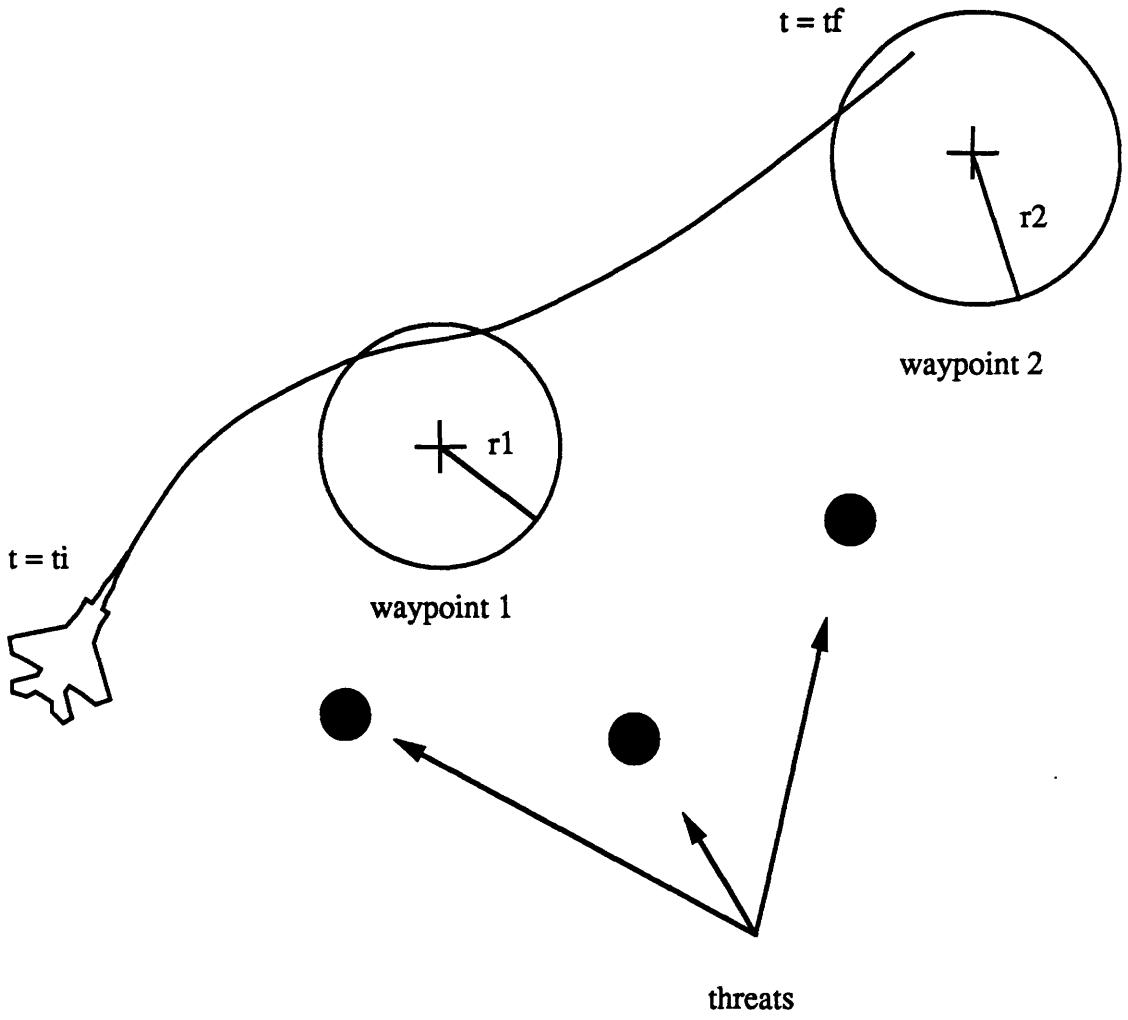
k_3 = limit on normal acceleration



Aircraft Reference Frame

Reference directions are Forward (x), Right (y) and Down (z)

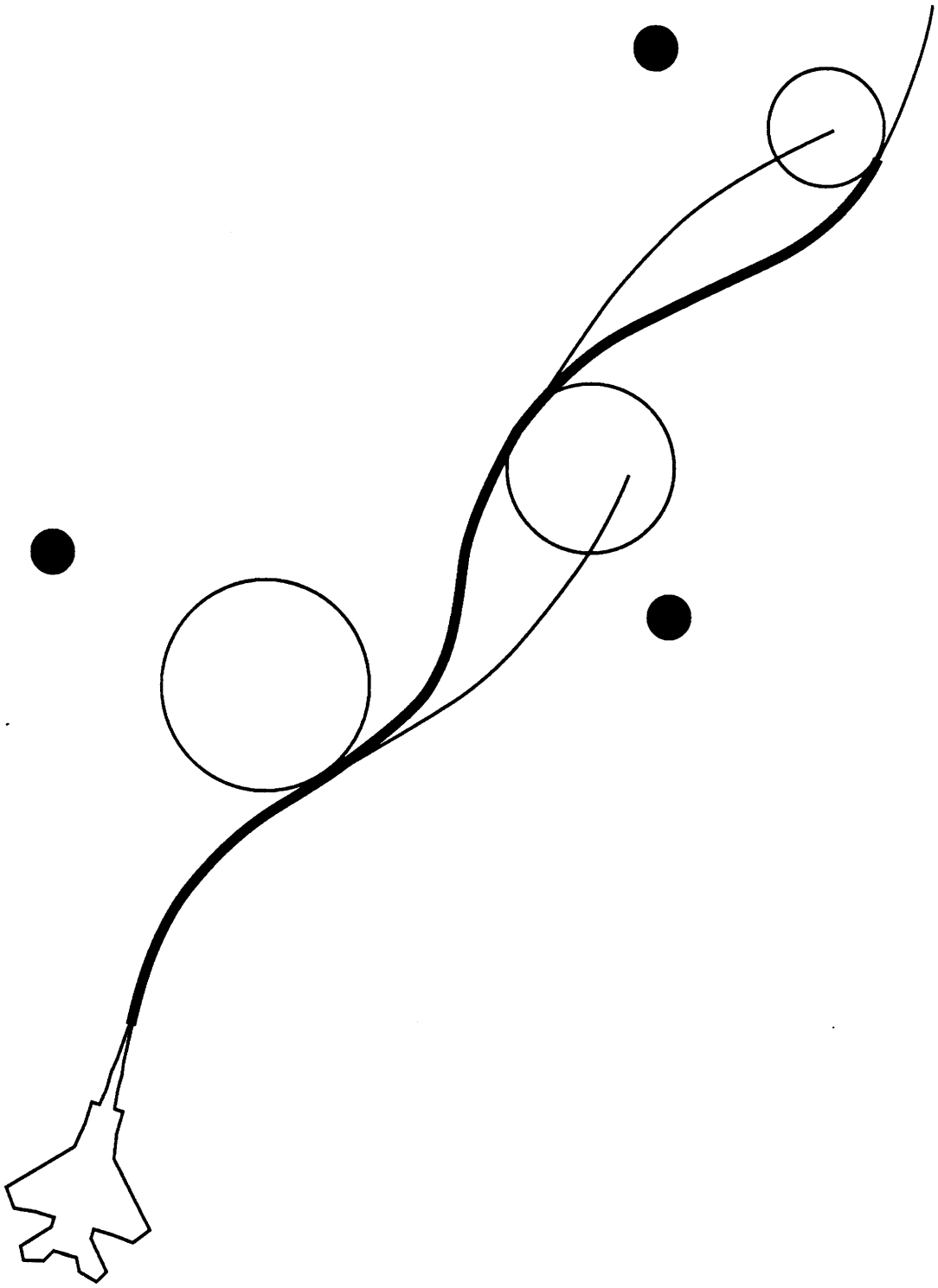
Figure 2.3.1



Typical Two Waypoint Trajectory Segment

Algorithm searches for commands which cause aircraft to capture specified waypoints while minimizing threat exposure and satisfying acceleration constraint.

Figure 2.6.1



Construction of Extended Trajectories

Figure 2.6.2

CHAPTER 3

PENALTY FUNCTION ALGORITHM

With reference to the problem statement in Section 2.6 (Equations 2.6.1–2.6.4), unconstrained minimization of J with respect to **c** does not lead to a useful solution because it embodies no recognition of the need to satisfy the constraints. To remedy this, the cost function is augmented with quadratic terms which penalize the constrained quantities d_1, d_2 and d_3:

$$J_a(\mathbf{c}) = \int_{t_i}^{t_f} T\,[x(\mathbf{c},t),y(\mathbf{c},t),z(\mathbf{c},t)]\,\frac{ds}{dt}\,dt + P_1\,d_1^2 + P_2\,d_2^2 + P_3\,d_3^2 \quad 3.1.1$$

By choosing the *penalty functions* $P_1 - P_3$ appropriately, it is possible to penalize constraint violations to a sufficient extent that the unconstrained minimizing argument of J_a is the same as the minimum of J with the constraints enforced. The procedure for constrained minimization using this approach is as follows:

```
assign small values to P1, P2, P3
while (any constraint violated)
        Minimize Ja without constraints
        increase penalties for violated constraints
end while
```

Selection of an unconstrained minimization method is dealt with in Section 3.2. Assume for the moment that a suitable algorithm is available for this purpose. The procedure used to update the penalty functions must be chosen with care to ensure efficiency. Correct values for the Penalty Functions can always be found by making sufficiently small corrections at each stage but this is likely to lead to very slow convergence. On the other hand, if the penalties become unnecessarily large, the minimum of the augmented cost function becomes difficult to locate because it tends to lie at the bottom of a steeply–sided n–dimensional ravine. The procedure to be described strikes a balance between these extremes by building up an estimate of how rapidly the constraint violations are changing in response to changes in each of the penalty functions and using this information to determine appropriate penalty updates after each unconstrained minimization. Equation 3.1.1 can be written more compactly as follows:

$$J_a(\mathbf{c}) = \int_{t_i}^{t_f} T\,[x(\mathbf{c},t), y(\mathbf{c},t), z(\mathbf{c},t)]\,\frac{ds}{dt}\,dt + \mathbf{d}^T\,\mathbf{P}\,\mathbf{d} \qquad 3.1.2$$

where:

$$\mathbf{d} = [\,d_1\ d_2\ d_3\,]^T$$

$$\mathbf{P} = \mathrm{diag}\,(\mathbf{p})$$

$$\mathbf{p} = [\,P_1\ P_2\ P_3\,]^T$$

The motivation for the following development is that each time J_a is re–minimized with a new value of **p**, the change in **p** and the consequent change in **d** together contain information about the derivative of **d** with respect to **p**. To first order, the incremental relationship between **d** and **p** can be written as:

$$\Delta\mathbf{d} = \mathbf{M}\,\Delta\mathbf{p} \qquad 3.1.3$$

where:

$$\mathbf{M} = \begin{bmatrix} \frac{\partial d_1}{\partial p_1} & \frac{\partial d_1}{\partial p_2} & \frac{\partial d_1}{\partial p_3} \\ \frac{\partial d_2}{\partial p_1} & \frac{\partial d_2}{\partial p_2} & \frac{\partial d_2}{\partial p_3} \\ \frac{\partial d_3}{\partial p_1} & \frac{\partial d_3}{\partial p_2} & \frac{\partial d_3}{\partial p_3} \end{bmatrix} \qquad 3.1.4$$

Suppose that the kth unconstrained minimization has just been performed and that an estimate $\mathbf{M_k}$ is available. If $\Delta\mathbf{d_k}$ and $\Delta\mathbf{p_k}$ are the changes in **d** and **p** which occurred during the preceding minimization then it is necessary that an updated estimate $\mathbf{M_{k+1}}$ of **M** should satisfy:

$$\Delta\mathbf{d_k} = \mathbf{M_{k+1}}\,\Delta\mathbf{p_k} \qquad 3.1.5$$

where

$$\Delta\mathbf{d_k} = \mathbf{d_k} - \mathbf{d_{k-1}} \qquad 3.1.6$$

$$\Delta\mathbf{p_k} = \mathbf{p_k} - \mathbf{p_{k-1}} \qquad 3.1.7$$

This can be re–written as:

$$\Delta\mathbf{d_k} = (\,\mathbf{M_k} + \Delta\mathbf{M_k}\,)\,\Delta\mathbf{p_k} \qquad 3.1.8$$

Or equivalently as:

$$\Delta\mathbf{M_k}\,\Delta\mathbf{p_k} = \Delta\mathbf{e_k} \qquad 3.1.9$$

where:

$$\Delta\mathbf{e_k} = \Delta\mathbf{d_k} - \mathbf{M_k}\,\Delta\mathbf{p_k} \qquad 3.1.10$$

Thus, $\Delta\mathbf{e_k}$ can be regarded as the error between the change in d that actually resulted from changing **p** by $\Delta\mathbf{p_k}$ and the change that would have been predicted using the best **M** estimate that was available before doing the kth minimization. The value of $\Delta\mathbf{M_k}$ which minimizes the sum of the squares its elements while satisfying the constraint imposed by Equation 3.1.9 is obtained by solving

the following problem:

$$\text{minimize:} \quad \text{tr}\,(\Delta M_k \Delta M_k^T) \tag{3.1.11}$$

$$\text{subject to:} \quad \Delta M_k \Delta p_k = \Delta e_k \tag{3.1.12}$$

The Lagrangian function is:

$$L = \text{tr}\,(\Delta M_k \Delta M_k^T) + \lambda^T (\Delta M_k \Delta p_k - \Delta e_k) \tag{3.1.13}$$

$$= \text{tr}\,(\Delta M_k \Delta M_k^T) + \text{tr}\,[\lambda^T (\Delta M_k \Delta p_k - \Delta e_k)] \tag{3.1.14}$$

Differentiating with respect to ΔM_k:

$$\frac{dL}{d\Delta M_k} = 2\Delta M_k + \lambda\, \Delta p_k^T \tag{3.1.15}$$

The minimizing value of ΔM_k is therefore given by:

$$\Delta M_k = -\frac{1}{2} \lambda\, \Delta p_k^T \tag{3.1.16}$$

Substituting for ΔM_k in the constraint equation gives:

$$\lambda = -\frac{2}{\Delta p_k^T \Delta p_k} \Delta e_k \tag{3.1.17}$$

Substituting for λ in Equation 3.1.16 gives the required ΔM_k:

$$\Delta M_k = \frac{\Delta e_k\, \Delta p_k^T}{\Delta p_k^T \Delta p_k} \tag{3.1.18}$$

To summarize, the equations that are used to update **M** after each minimization are:

$$\mathbf{M}_{k+1} = \mathbf{M}_k + \Delta\mathbf{M}_k \qquad 3.1.19$$

$$\Delta\mathbf{M}_k = \frac{\Delta\mathbf{e}_k \, \Delta\mathbf{p}_k^T}{\Delta\mathbf{p}_k^T \Delta\mathbf{p}_k} \qquad 3.1.20$$

$$\Delta\mathbf{e}_k = \Delta\mathbf{d}_k - \mathbf{M}_k \, \Delta\mathbf{p}_k \qquad 3.1.21$$

The following recursive update for $\mathbf{M}_k^{-1}$ will also be useful and arises from parallel reasoning:

$$\mathbf{M}_{k+1}^{-1} = \mathbf{M}_k^{-1} + \Delta\mathbf{M}_k^{-1} \qquad 3.1.22$$

$$\Delta\mathbf{M}_k^{-1} = \frac{\Delta\mathbf{e}_k \, \Delta\mathbf{d}_k^T}{\Delta\mathbf{d}_k^T \Delta\mathbf{d}_k} \qquad 3.1.23$$

$$\Delta\mathbf{e}_k = \Delta\mathbf{p}_k - \mathbf{M}_k^{-1} \Delta\mathbf{d}_k \qquad 3.1.24$$

$\mathbf{M}_{k+1}^{-1}$ can be used to determine how $\mathbf{p}_k$ should be changed so as to reduce any constraint violation in the next cycle. Define a *constraint violation vector* $\boldsymbol{\psi}_k$ as follows:

$$\boldsymbol{\psi}_k = \mathbf{d}_k - \mathbf{k} \qquad 3.1.25$$

where **k** is a vector containing the constraint levels to be imposed. If all three elements in $\boldsymbol{\psi}_k$ are positive then all three constraints are violated and the desired change in **d** to be achieved by changing **p** is simply:

$$\Delta\mathbf{d}_k = -\boldsymbol{\psi}_k \qquad 3.1.26$$

To first order, the change in $\mathbf{p}_k$ that will be necessary in the next minimization to eliminate this violation is given by:

$$\Delta p_{k+1} = - M_{k+1}^{-1} \phi_k \qquad 3.1.27$$

Since Δp_{k+1} is chosen on the basis of constraint violations, it will be underdetermined when one or more of the constraints is satisfied. In this case a vector $\Delta \tilde{d}_k$ representing the desired changes in the elements of d_k corresponding to the violated constraints is constructed and a corresponding partition $\tilde{M}_{k+1}$ is formed so that the constraints to be satisfied by Δp_{k+1} can be expressed in the following form:

$$\Delta \tilde{d}_k = \tilde{M}_{k+1} \Delta p_{k+1} \qquad 3.1.28$$

A unique value for Δp_{k+1} is obtained by choosing the minimum norm solution of 3.1.28. This is reasonable in view of the fact that large changes in p_{k+1} are undesirable. Specifically, Δp_{k+1} is found as the solution to the following constrained minimization problem:

minimize: $\frac{1}{2} \Delta p_{k+1}^T \Delta p_{k+1}$ 3.1.29

subject to: $\Delta \tilde{d}_k = \tilde{M}_{k+1} \Delta p_{k+1}$ 3.1.30

The Lagrangian function is:

$$L = \frac{1}{2} \Delta p_{k+1}^T \Delta p_{k+1} + \lambda^T (\tilde{M}_{k+1} \Delta p_{k+1} - \Delta \tilde{d}_k) \qquad 3.1.31$$

Differentiating with respect to Δp_{k+1}:

$$\frac{dL}{d\Delta p_{k+1}} = \Delta p_{k+1}^T + \lambda^T \tilde{M}_{k+1} \qquad 3.1.32$$

For stationarity,

$$\Delta p_{k+1} = - \tilde{M}_{k+1}^T \lambda \qquad 3.1.33$$

Substituting for Δp_{k+1} in Equation 3.1.30 gives:

$$\Delta \tilde{d}_k = - \tilde{M}_{k+1} \tilde{M}_{k+1}^T \lambda \qquad 3.1.34$$

Solving for λ and substituting in Equation 3.1.30 gives:

$$\Delta p_{k+1} = \tilde{M}_{k+1}^T (\tilde{M}_{k+1} \tilde{M}_{k+1}^T)^{-1} \Delta \tilde{d}_k \qquad 3.1.35$$

The desired change $\Delta \tilde{d}_k$ is given by:

$$\Delta \tilde{d}_k = - \tilde{\psi}_k \qquad 3.1.36$$

where $\tilde{\psi}_k$ is a vector containing the constraint violations which are positive. The required penalty update in this case is therefore:

$$\Delta p_{k+1} = - \tilde{M}_{k+1}^T (\tilde{M}_{k+1} \tilde{M}_{k+1}^T)^{-1} \tilde{\psi}_k \qquad 3.1.37$$

Notice that 3.1.37 reduces to 3.1.27 when all three constraints are violated so that $\tilde{M}_{k+1} = M_{k+1}$ and $\tilde{\psi}_k = \psi_k$. Having determined the penalty increment, the penalty vector to be used in the next minimization is found using:

$$p_{k+1} = p_k + \Delta p_{k+1} \qquad 3.1.38$$

To the extent that the first order approximation is valid, this penalty vector will cause the violated constraints to be reduced to zero after the next minimization. However, since the non–violated constraints were ignored in the derivation of the penalty update, there is no guarantee that they will continue to be satisfied after application of Δp_{k+1}. Rather than performing an entire minimization to

determine whether this is the case, $\mathbf{M}_{k+1}$ can be used to obtain the following prediction of the constraint variables:

$$\mathbf{d}_{k+1}' = \mathbf{d}_k + \mathbf{M}_{k+1}\,\Delta\mathbf{p}_{k+1} \qquad 3.1.39$$

If any element in $\mathbf{d}_{k+1}'-\mathbf{k}$ is positive it is added to the corresponding element in $\tilde{\psi}_k$ and $\Delta\mathbf{p}_{k+1}$ is recomputed before calculating $\mathbf{p}_{k+1}$. This process is repeated until all elements in $\mathbf{d}_{k+1}'$ are less than or equal to their limits.

CHAPTER 4

UNCONSTRAINED MINIMIZATION

In Chapter 2, it was shown how parameters defining a desirable set of commands can be found as the solution of a problem in constrained non–linear optimization. The penalty function approach presented in Chapter 3 allows such a problem to be solved by performing a sequence of unconstrained minimizations. A variety of methods is available for minimization without constraints[3][4][5] and this chapter deals with the selection of a suitable one for the present purpose.

4.1 Descent Methods

The more efficient and well–proven algorithms for unconstrained minimization of an arbitrary function of n variables $F(\mathbf{x})$ rely on an iterative process comprising a series of univariate minimizations conducted along a specific set of *search directions*. Thus, if the current estimate of the minimum at the start of the kth iteration is $\mathbf{x_k}$, the next estimate can be expressed in the form:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \alpha_k \, \mathbf{p_k} \qquad 4.1.1$$

where:

$\mathbf{p_k}$ = kth search direction

$\alpha_k = \arg \min_{\alpha} F(\mathbf{x_k} + \alpha \, \mathbf{p_k})$

In practice, it is found that algorithms which satisfy the following *descent condition* have good convergence characteristics:

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k) \qquad \forall\, k \geq 0 \qquad 4.1.2$$

Assuming that F is continuously differentiable, an explicit requirement on $\mathbf{p}_k$ which guarantees that F can be reduced at the kth iteration can be found by considering the first order Taylor expansion of $F(\mathbf{x}_k)$ about $\mathbf{x}_k$:

$$F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \approx F(\mathbf{x}_k) + \alpha_k \mathbf{g}_k^T \mathbf{p}_k \qquad 4.1.3$$

where:

$$\mathbf{g}_k = \left.\frac{\partial F}{\partial \mathbf{x}}\right|_{\mathbf{x}_k} \qquad 4.1.4$$

The descent condition will always be met for sufficiently small α_k if $\mathbf{p}_k$ is chosen so that:

$$\mathbf{g}_k^T \mathbf{p}_k < 0 \qquad \forall\, k \geq 0 \qquad 4.1.5$$

When $\mathbf{p}_k$ satisfies this condition, it is said to be a *descent direction* and the corresponding algorithm will have the descent property. The *steepest descent* algorithm is obtained by choosing:

$$\mathbf{p}_k = -\mathbf{g}_k \qquad 4.1.6$$

This rationale is intuitively appealing because $-\mathbf{g}_k$ is the direction along which $F(\mathbf{x})$ has its greatest rate of decrease and because calculation of $\mathbf{p}_k$ is straightforward. However, the algorithm is characterized by very slow convergence and therefore has little practical utility. It is mentioned primarily because a single "steepest descent" step is generally used as the starting point for the more powerful Quasi–Newton methods described in the following section.

4.2 Quasi Newton Methods

The essential basis for the steepest descent method is a linear approximation to the objective function in the vicinity of the minimum. If the second derivative or *Hessian* matrix of $F(\mathbf{x})$ is available (or can be estimated) in addition to the gradient, it is possible to develop more rapidly convergent techniques. The second order Taylor expansion of $F(\mathbf{x})$ about $\mathbf{x_k}$ is:

$$F(\mathbf{x_k} + \mathbf{p_k}) \approx F(\mathbf{x_k}) + \mathbf{g_k}^T \mathbf{p_k} + \frac{1}{2} \mathbf{p_k}^T \mathbf{G_k} \mathbf{p_k} \qquad 4.2.1$$

In the neighborhood of a strong minimum, Equation 4.2.1 can be regarded as a quadratic model of the objective function and $\mathbf{G_k}$ will be positive definite. The direction and magnitude of the step from $\mathbf{x_k}$ to the minimum of the model function is obtained by solving the following system for $\mathbf{p_k}$:

$$\mathbf{G_k} \mathbf{p_k} = - \mathbf{g_k} \qquad 4.2.2$$

When $\mathbf{p_k}$ is determined using Equation 4.2.2, the corresponding minimization technique is referred to as *Newton's Method.* A distinguishing feature of this technique is that the minimum of a quadratic objective function with a positive definite Hessian can be located in one step from an arbitrary starting point. Even when the objective function is non–quadratic, the Newton step will still be productive if $\mathbf{x_k}$ is in the neighborhood of a strong minimum. This is because the quantity $\mathbf{g_k}^T \mathbf{p_k}$ in the descent test (Equation 4.1.5) is intrinsically negative when the Hessian is positive definite. Moreover, it can be shown that the rate of convergence of Newton's Method is always second order when the Hessian is positive definite at the minimum. Newton's Method is therefore very desirable in situations where the Hessian is readily available but it is often prohibitively expensive to compute the Hessian and so various *Quasi–Newton* algorithms have been developed whose performance approaches that of Newton's Method without the requirement for explicit calculation of the Hessian matrix.

Quasi–Newton methods are based on the idea that an approximation to the curvature of a non–linear function can be built up as the iterations of a descent method proceed using only function and gradient evaluations. In the

particular algorithm to be described, the curvature information is propagated in the form of an estimate of the inverse Hessian matrix **H**. This eliminates the need to solve a system of linear equations to obtain the search vector $\mathbf{p_k}$ so that at any point the required search direction is calculated using:

$$\mathbf{p_k} = - \mathbf{H_k}\, \mathbf{g_k} \qquad 4.2.3$$

Suppose that the current estimate of the inverse Hessian is $\mathbf{H_k}$. After each iteration, $\mathbf{H_k}$ is updated using the following formula:

$$\mathbf{H_{k+1}} = \mathbf{H_k} + \mathbf{Q_k} \qquad 4.2.4$$

$\mathbf{Q_k}$ is a matrix which is to be chosen in such a way that $\mathbf{H_{k+1}}$ contains the same curvature information as $\mathbf{G_k}^{-1}$ in the direction $\mathbf{p_k}$ when the objective function is quadratic. This constraint is enforced by requiring that $\mathbf{H_{k+1}}$ satisfy the following *Quasi–Newton Condition*:

$$\mathbf{H_{k+1}}\, \Delta \mathbf{g_k} = \Delta \mathbf{x_k} \qquad 4.2.5$$

where:

$$\Delta \mathbf{g_k} = \mathbf{g_{k+1}} - \mathbf{g_k}$$

$$\Delta \mathbf{x_k} = \mathbf{x_{k+1}} - \mathbf{x_k}$$

4.3 Broyden–Fletcher–Goldfarb–Shanno (BFGS) Method

The Quasi–Newton Condition does not specify the **H** matrix uniquely and many updating formulae are available. The technique used in this research employs an update formula due to Broyden, Fletcher, Goldfarb and Shanno which can be stated in the following form:

$$\mathbf{H_{k+1}} = \left[\mathbf{I} - \frac{\Delta \mathbf{x_k}\, \Delta \mathbf{g_k}^{\mathbf{T}}}{\Delta \mathbf{x_k}^{\mathbf{T}}\, \Delta \mathbf{g_k}} \right] \mathbf{H_k} \left[\mathbf{I} - \frac{\Delta \mathbf{x_k}\, \Delta \mathbf{g_k}^{\mathbf{T}}}{\Delta \mathbf{x_k}^{\mathbf{T}}\, \Delta \mathbf{g_k}} \right]^{\mathbf{T}} + \frac{\Delta \mathbf{x_k}\, \Delta \mathbf{x_k}^{\mathbf{T}}}{\Delta \mathbf{x_k}^{\mathbf{T}}\, \Delta \mathbf{g_k}} \qquad 4.3.1$$

In recent years, the BFGS formula has gained widespread acceptance as an efficient basis for implementation of the Quasi Newton method. The main reason for this is its relative insensitivity to the accuracy of the line search so that the number of function evaluations needed to make a given amount of progress towards the minimum is generally smaller than that required when using other update schemes.

The algorithm can be started with H_0 set equal to any positive definite matrix. The identity matrix is normally used when there is no prior information about curvature and so the first step will be along the direction of steepest descent (Equation 4.2.3). Exit occurs if either the gradient norm or the difference between two consecutive function evaluations become smaller than small thresholds gtol and ftol respectively. To summarize, the BFGS algorithm used has the following form:

input x_0, H_0, gtol, ftol

do k=0, itmax

$p_k \leftarrow -H_k g_k$

$\alpha_k \leftarrow \arg\min_{\alpha} F(x_k + \alpha p_k)$

$x_{k+1} \leftarrow x_k + \alpha_k p_k$

if $2*|F(x_{k+1}) - F(x_k)| \leq \text{ftol}*(|F(x_{k+1})| + |F(x_k)|)$ **exit**

$g_{k+1} \leftarrow \nabla F(x)\big|_{x=x_{k+1}}$

if $||g_{k+1}|| \leq \text{gtol}$ **exit**

$\Delta x_k \leftarrow x_{k+1} - x_k$

$\Delta g_k \leftarrow g_{k+1} - g_k$

$$H_{k+1} = \left[I - \frac{\Delta x_k \, \Delta g_k^T}{\Delta x_k^T \, \Delta g_k} \right] H_k \left[I - \frac{\Delta x_k \, \Delta g_k^T}{\Delta x_k^T \, \Delta g_k} \right]^T + \frac{\Delta x_k \, \Delta x_k^T}{\Delta x_k^T \, \Delta g_k}$$

end do

4.4 Line Search Algorithm

The efficiency of the procedure used to conduct line searches along the directions $\mathbf{p_k}$ is an important factor in the overall effectiveness of the multivariate minimization process. Although the BFGS method is less demanding than most in terms of univariate search accuracy, the computational expense associated with evaluating the risk function in the present application means that it is particularly important for the chosen line search technique to be as frugal as possible in its use of function evaluations. Methods which use the gradient of the objective function are to be avoided because the evaluation of derivatives calls for multiple function evaluations (see Chapter 5). Parabolic interpolation converges rapidly to a stationary point which has been bracketed in a finite interval, but it makes no distinction between maxima and minima and gives unreliable results if the curvature is low. On the other hand, function comparison methods such as Golden Section Search are slower, but will always converge towards the lowest function value in a given interval. The technique used here is based on an algorithm due to Brent[6][7] which uses a sequence of Golden Section and Parabolic Interpolation steps to bracket the minimum in a progressively smaller interval. At each stage, the algorithm uses retained information about the recent behavior of the function to decide whether Golden Section or Parabolic Interpolation is appropriate for the next step. The search is terminated when the minimum has been bracketed with a fractional precision of 10%.

CHAPTER 5

IMPLEMENTATION

5.1 Evaluation of Risk Function

In order to minimize the risk J(**c**) defined in Equation 2.5.9, it is necessary to specify its dependence upon the elements of the command coefficient vector **c** explicitly. A closed form for J(**c**) is not available, primarily because x(**c**,t), y(**c**,t) and z(**c**,t) arise from propagation of the controls (implied by **c**) through the non–linear vehicle dynamics and are not expressible in terms of elementary functions. It is therefore necessary to evaluate the risk numerically, even when the threat distribution T(x,y,z) is a relatively simple function. In this thesis, the horizontal and vertical variations of the threat function are considered separately so that the threat is resolved as follows:

$$T(x,y,z) = T_h(x,y) + T_v(z)$$

where:

$T_h(x,y)$ = Horizontal Threat Distribution

$T_v(z)$ = Vertical Threat Distribution

It is further assumed that the primary consideration with regard to the vertical component of the aircraft's motion is the need to maintain a safe altitude and that this requirement can be adequately reflected in the following vertical threat distribution:

$$T_v(x,y,z) = k_{alt}\,(z - z_{ref})^2$$

where:

k_{alt} = scale factor

z_{ref} = desired altitude

$T_h(x,y)$ is constructed as the superposition of a number of discrete threat sources on the basis that the contribution of a given source located at (x_i,y_i,z_i) is inversely proportional to its horizontal distance from the aircraft:

$$T(x,y) = \sum_{i=1}^{n} \frac{\Gamma_i}{\sqrt{(x-x_i)^2 + (y-y_i)^2}} \qquad 5.1.1$$

where:

Γ_i = intensity of i th threat

This characterization has merit in cases where the source is truly localized in space and where the threat varies inversely with the length of the displacement vector from the aircraft to the source. It is possible to conceive of a threat whose lethality varies with the direction of the displacement as well as its magnitude so that the contours of equal threat might be ellipses or some quite arbitrary figure rather than circles. Also, the threat data in an implemented system is likely to be given in the form of a table of numerical values. The analytical form given above was adopted purely for convenience and it is not thought that the use of any reasonably well–behaved threat function would compromise the operation of the algorithm as a whole. As suggested by Equation 2.5.9, J is evaluated by treating it as an additional state variable to be integrated numerically alongside Equations 2.3.2–2.3.9. Figure 5.1 illustrates the time evolution of the system dynamics in block diagram form. A fourth order Runge–Kutta method[7] is used to perform the integration.

5.2 Finite Difference Approximation of Gradient

Analytical gradients are not available and finite difference approximations are necessary for the reasons discussed in Section 5.1. Each element in the gradient is a partial derivative of the multivariate risk function, but in order to simplify the notation, a univariate objective function will be considered in the following discussion. Function evaluations are expensive and the following *Forward Difference Formula* is therefore attractive because it requires evaluation at only two points:

$$\varphi_F(f,h) = \frac{\tilde{f}(x+h) - \tilde{f}(x)}{h} \qquad 5.2.1$$

where: $\varphi_F(f,h)$ is the forward difference operator, $\tilde{f}(x+h)$ and $\tilde{f}(x)$ are computed function values and h is the step length. The computed function values are subject to machine–dependent errors which can be represented as follows:

$$\tilde{f}(x) = f(x) + \sigma \qquad 5.2.2$$

$$\tilde{f}(x+h) = f(x+h) + \sigma_h \qquad 5.2.3$$

The forward difference approximation can therefore be written:

$$\varphi_F(f,h) = \frac{f(x+h) - f(x)}{h} + \frac{\sigma_h - \sigma}{h} \qquad 5.2.4$$

In addition to these machine–dependent errors, the forward difference approximation will also differ from the true derivative because it arises from a truncated Taylor series. This can be seen by expanding f(x+h) to second order:

$$f(x+h) = f(x) + h\,f'(x) + \frac{h^2}{2} f''(x) + O(h^3) \qquad 5.2.5$$

Substituting for f(x+h) in Equation 5.2.4 gives the following relationship between φ_F and the true derivative f′(x):

$$\varphi_f(f,h) = f'(x) + T_f(h) + C(\varphi_f,h) \qquad 5.2.6$$

where:

$$T_f(h) = \frac{h}{2} f''(x) + O(h^2) \qquad \textit{truncation error} \qquad 5.2.7$$

$$C(\varphi_f,h) = \frac{\sigma_h - \sigma}{h} \qquad \textit{cancellation error} \qquad 5.2.8$$

It can be seen that the truncation error is an increasing function of h while the cancellation error decreases with h. Moreover, if second and higher order terms are neglected, the total error is bounded by:

$$|\varphi_f(f,h) - f'(x)| \leq \frac{h}{2} |f''(x)| + \frac{2}{h} \epsilon_a \qquad 5.2.9$$

where ϵ_a is a bound on the error in the computed function values in the vicinity of x. In principle, an optimal step length could be found by minimizing Equation 5.2.9 with respect to h. Unfortunately, f′′ is not known and further information is therefore required before the error associated with a particular value of h can be estimated. Such information can be obtained at the expense of an additional function evaluation by considering the following *Central Difference Formula*:

$$\varphi_c(f,h) = \frac{\tilde{f}(x+h) - \tilde{f}(x-h)}{2h} \qquad 5.2.10$$

Introducing the explicit machine–dependent errors σ_h and σ_{-h} and expanding f(x+h) and f(x–h) to third order gives a relationship analogous to Equation 5.2.6:

$$\varphi_c(f,h) = f'(x) + T_c(h) + C(\varphi_c,h) \qquad 5.2.11$$

where:

$$T_c(h) = O(h^2) \qquad \textit{truncation error} \qquad 5.2.12$$

$$C(\varphi_c,h) = \frac{\sigma_h - \sigma_{-h}}{2h} \qquad \textit{cancellation error} \qquad 5.2.13$$

The fact that φ_c is accurate to second order in h while φ_f is accurate to first order suggests the following as a computable estimate of the error in φ_f:

$$\varphi_f(f,h) - \varphi_c(f,h) = T_f(h) + C(\varphi_f,h) - T_c(h) - C(\varphi_c,h) \qquad 5.2.14$$

$$= \frac{h}{2} f''(x) + \frac{\sigma_h - 2\sigma + \sigma_{-h}}{2h}$$

The triangle inequality leads to the following upper bound on the unknown second derivative:

$$\frac{h}{2} |f''(x)| \leq |\varphi_f(f,h) - \varphi_c(f,h)| + \frac{2}{h} \epsilon_a \qquad 5.2.15$$

Substitution in Equation 5.2.9 gives a bound on the total error in the forward approximation:

$$|\varphi_f(f,h) - f'(x)| \leq |\varphi_f(f,h) - \varphi_c(f,h)| + \frac{4}{h} \epsilon_a \qquad 5.2.16$$

Since the objective function will never be zero in the present context, ϵ_a can be related to the relative precision ϵ_r of f(x) (which is known) as follows:

$$\epsilon_a = \epsilon_r |f(x)| \qquad 5.2.17$$

Equation 5.2.17 can be re–written in terms of the relative error in the gradient as follows:

$$\left| \frac{\varphi_f(f,h) - f'(x)}{\varphi_f(f,h)} \right| \leq E_t + E_c \qquad 5.2.18$$

where:

$$E_t = \left| \frac{\varphi_f(f,h) - \varphi_c(f,h)}{\varphi_f(f,h)} \right| \qquad 5.2.19$$

$$E_c = 4\epsilon_r \left| \frac{\tilde{f}(x)}{\tilde{f}(x+h) - \tilde{f}(x)} \right| \qquad 5.2.20$$

E_c is a measure of the cancellation error, and if it is small then E_t will be dominated by the truncation error. Accordingly, the policy adopted in the gradient algorithm is to use the smallest h for which E_c is less than .1%. The corresponding estimate of the total forward difference error was usually sufficiently small, a warning message being generated in the event that it exceeds 1%. Moreover, the value assigned to the derivative is that obtained using the central difference formula so that the actual truncation error will normally be significantly better than that indicated by Equation 5.2.18.

5.3 Acceleration Constraint

The maximum absolute value of the acceleration command must not exceed the predetermined value of 4g. Since each acceleration command segment has the form of a quartic polynomial in time, the peak will always occur either at one of the end–points or at one of the stationary points of which there are at most three. It is therefore a straightforward matter to identify and penalize the peak associated with a given set of command coefficients. However, the constrained minimization technique of Section 3 does not perform well in this situation because the method requires that each constrained quantity should be continuous with respect to changes in the penalty functions. The fact that the peak acceleration can occur at any one of five points means that a small change in one

of the penalty functions can cause a large change in the location and value of the peak as illustrated by Figure 5.3.1. Instead of penalizing the peak directly, it was decided to use a property of the Chebyshev coefficients to establish an upper bound on the peak acceleration which has the required continuity. None of the Chebyshev basis polynomials $T_i(\lambda)$ (see Section 2.4) can have an absolute value greater than unity within the interval in which they are defined (i.e. [–1,1]). This means that the value of the acceleration command can never exceed the sum of the absolute values of the relevant Chebyshev coefficients:

$$a_{nc} \leq a_{bound} \tag{5.3.1}$$

where:

$$a_{bound} = \sum_{i=1}^{4} |c_{ai}| \tag{5.3.2}$$

As expected, the performance of the penalty function algorithm was found to be much better when a_{bound} was penalized rather than the peak value.

5.4 Initializing Command Coefficient Vector

It is appropriate to choose the initial value c_0 of the command coefficient vector with some care at the start of each constrained minimization . A moderate amount of effort expended here to ensure that c_0 is in some sense reasonably close to the minimum will expedite the minimization considerably. The approach adopted is to ignore the threats and the acceleration constraint initially and to search for a command vector c that will minimize the waypoint penalty terms alone in the expression for $J_a(c)$ given in Equation 3.1.1. This can be done relatively quickly and the resulting value for c will be "close" to the minimum in the sense that it will at least satisfy the waypoint constraints.

5.5 Initializing Constraint Derivatives

In order to start the penalty function algorithm described in Chapter 3, an initial estimate of the matrix of constraint derivatives M is required where:

$$\mathbf{M} = \begin{bmatrix} \frac{\partial d_1}{\partial p_1} & \frac{\partial d_1}{\partial p_2} & \frac{\partial d_1}{\partial p_3} \\ \frac{\partial d_2}{\partial p_1} & \frac{\partial d_2}{\partial p_2} & \frac{\partial d_2}{\partial p_3} \\ \frac{\partial d_3}{\partial p_1} & \frac{\partial d_3}{\partial p_2} & \frac{\partial d_3}{\partial p_3} \end{bmatrix} \qquad 5.5.1$$

Initial values for the elements of M are obtained by perturbing each element in the initial penalty vector by a small amount δ_j, j=1,2,3 and using the following forward difference formula:

$$M_{ij} = \frac{d_i(\mathbf{p}_0 + \delta_j \mathbf{u}_j) - d_i(\mathbf{p}_0)}{\delta_j} \qquad 5.5.2$$

where:

M_{ij} = i,j th element in **M**

$d_i(\mathbf{p})$ = value of i th constrained quantity corresponding to penalty vector **p**

$\mathbf{u}_j$ = unit vector along j th direction in n-space

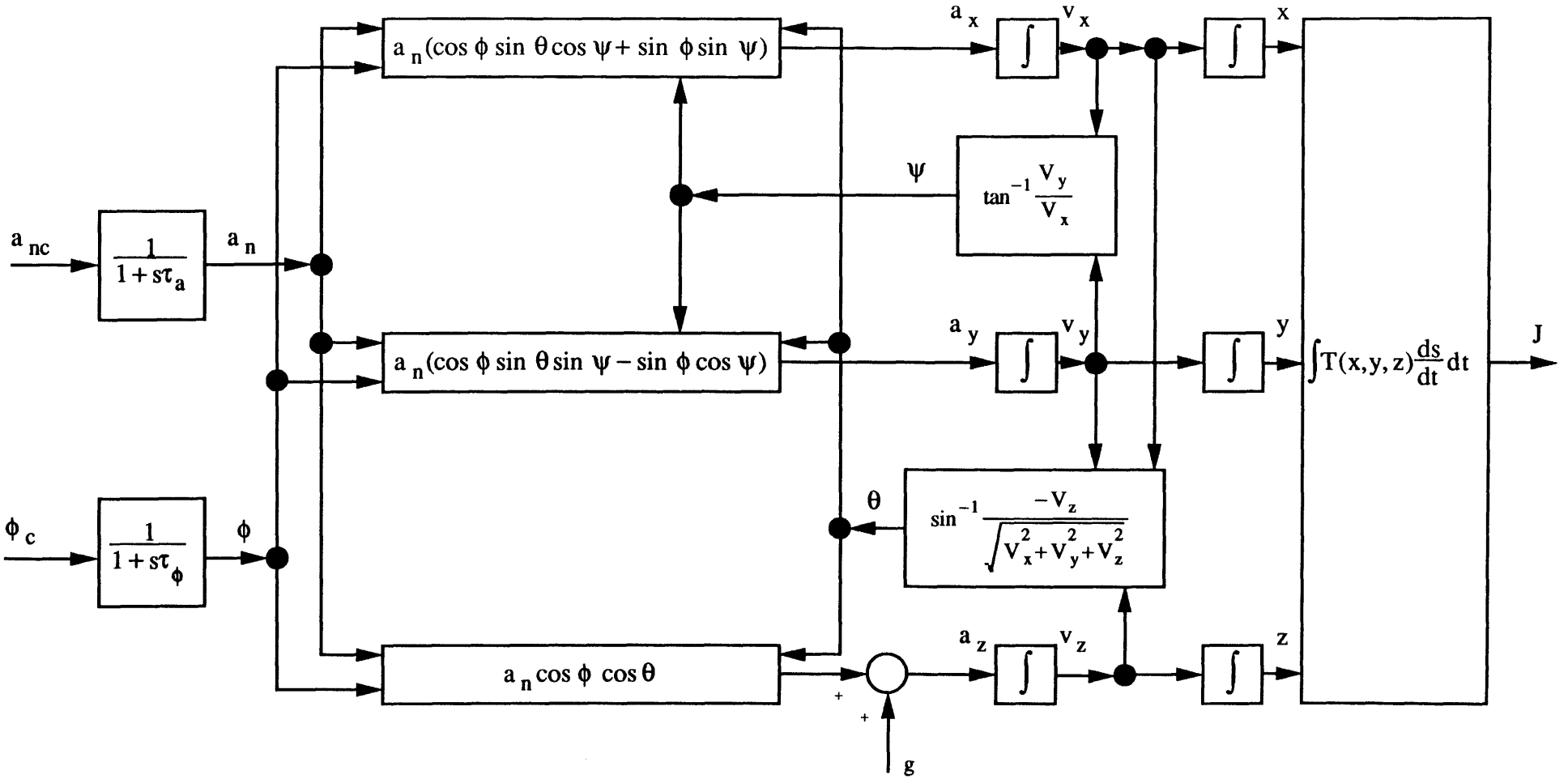
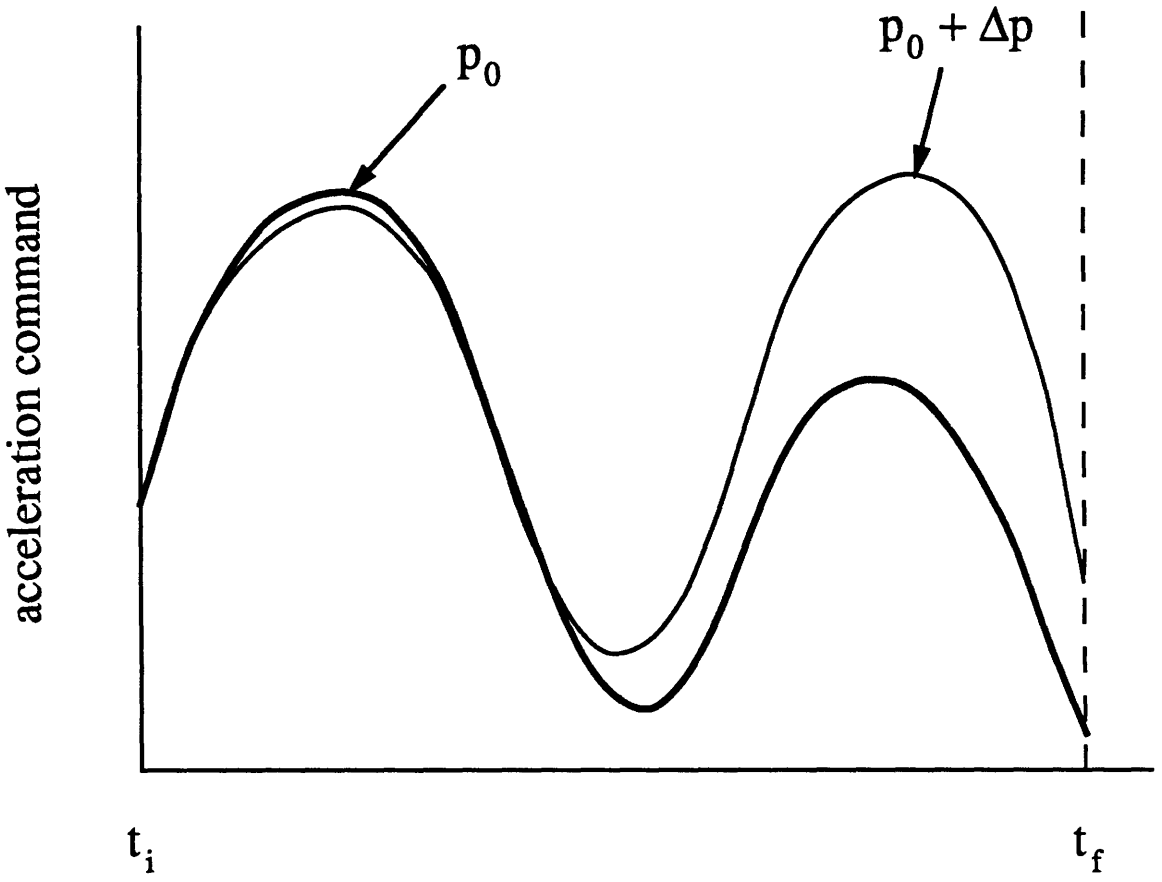


Figure 5.1

Evaluation of Risk Function in Terms of Command Histories



Sensitivity of Peak Acceleration

A small change in one of the penalty functions can cause a large change in both location and value of peak acceleration command.

Figure 5.3.1

CHAPTER 6

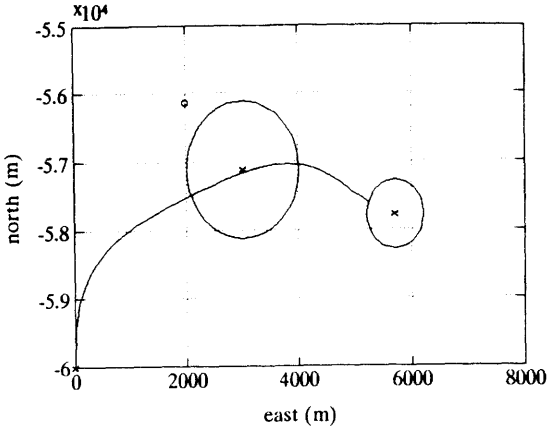
RESULTS

The procedure for finding desirable command functions which has been described in the preceding chapters was implemented and tested using a variety of waypoint and threat geometries chosen to resemble realistic scenarios. The horizontal and vertical threat functions defined in Section 5.1 are used throughout this section. The algorithm is unlikely to experience difficulty with other practically likely distributions although additional processing would be necessary to handle a case where the threat is known only at discrete points or where there are discontinuities in the distribution.

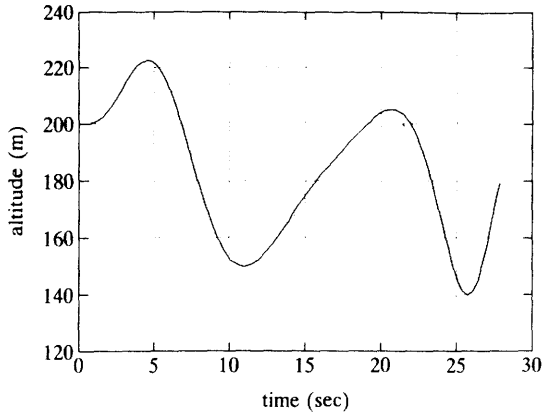
The "core" problem of finding a command segment that will capture the next two waypoints was solved in a variety of test cases. A selection of the resulting trajectories, altitude profiles and command histories is presented in Figures 6.1–6.4. It is emphasized that the altitude profiles result from the use of the very simple vertical threat function discussed in Section 5.1 which penalizes departure from a fixed altitude of 200 meters. The effect on the computed acceleration command and the associated trajectory of progressively reducing the constraint level is illustrated in Figure 6.5. The process by which extended trajectories can be constructed by repeatedly solving the core problem is illustrated in Figures 6.6–6.8. In these examples, a new solution is computed each time the aircraft passes a waypoint (approximately every 20 seconds in real time). More frequent updates would be desirable if sufficient computer duty cycle was available.

The results discussed here were obtained using an IBM compatible personal computer operating at a clock speed of 20 MHz with an 80387 math

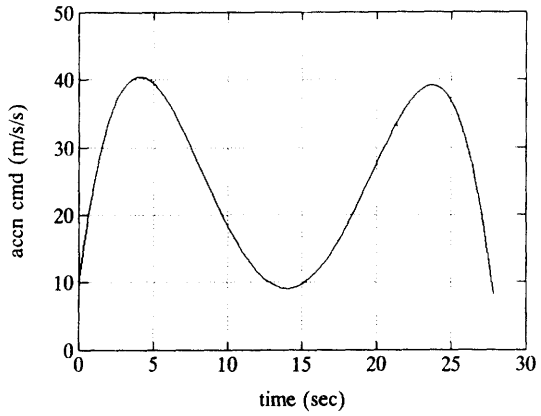
coprocessor. Matlab was used extensively for matrix manipulations, but the BFGS minimization routines were coded in Microsoft Fortran. The speed with which the algorithms could be executed in this environment falls well short of that required for real time operation. Each evaluation of the risk function takes approximately 2 seconds. The gradient is estimated once per BFGS cycle using central differences (see Section 5.2) and this requires a minimum of 16 function evaluations, 20 being more typical when allowance is made for the adaptive step size control process. In addition, each line search calls for approximately 30 function evaluations so that a typical BFGS cycle can be expected to take about 100 seconds. For a given set of penalty functions, BFGS locates the minimum after approximately 15 line searches and each unconstrained minimization in the penalty function algorithm therefore takes about 30 minutes. Convergence to the correct penalty function values was typically found to occur after 6 iterations so that the entire constrained minimization can easily take 3 hours. In addition, initialization of the constraint derivatives using the method described in Section 5.3 requires 3 additional unconstrained minimizations which adds another hour to the total time to compute one segment. To put these figures in perspective, the aircraft will cover each segment in approximately 30 seconds, so it is clear that a dramatic increase in the execution speed would be necessary to permit real–time operation.



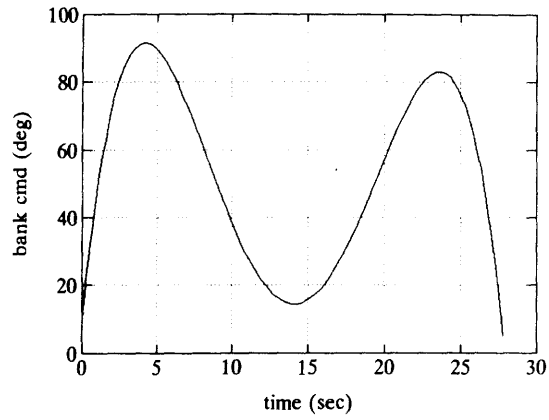
(a) Trajectory



(b) Altitude



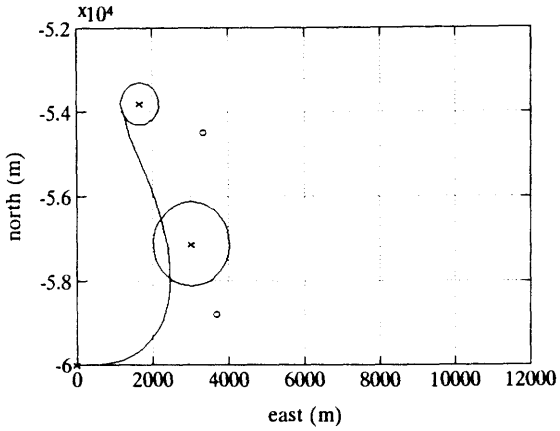
(c) Normal Acceleration Command



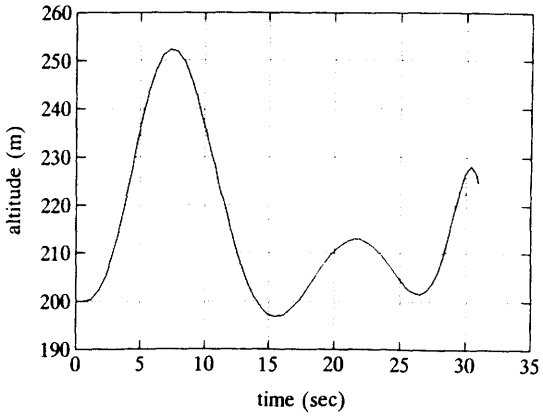
(d) Bank Command

Test Segment No. 1

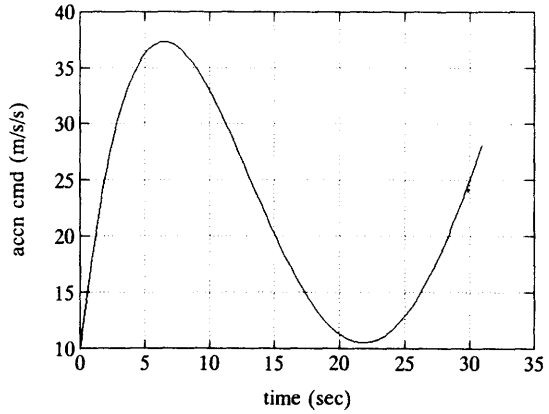
Figure 6.1



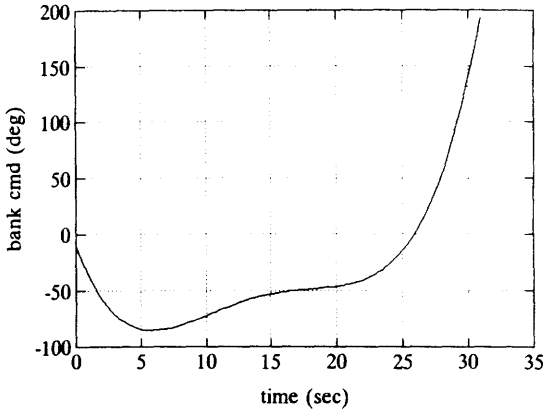
(a) Trajectory



(b) Altitude



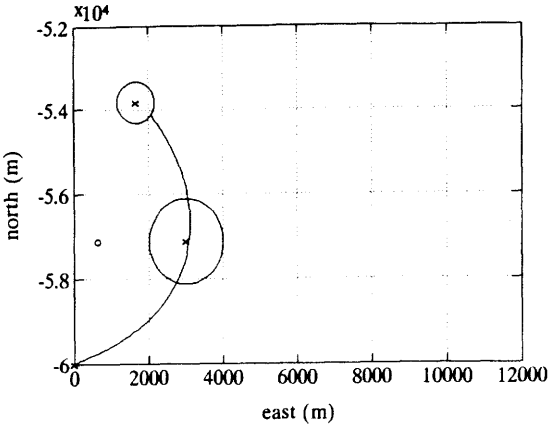
(c) Normal Acceleration Command



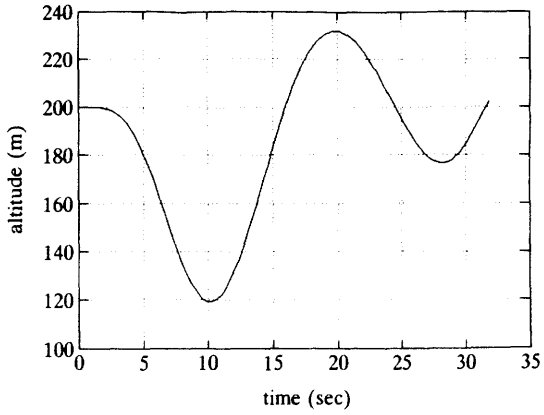
(d) Bank Command

Test Segment No. 2

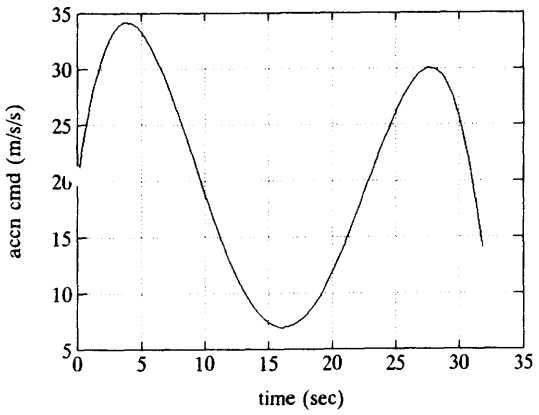
Figure 6.2



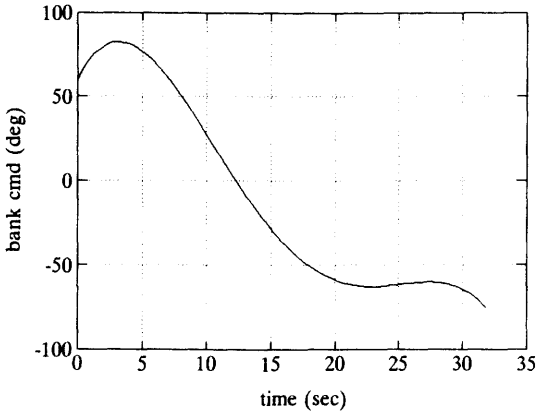
(a) Trajectory



(b) Altitude



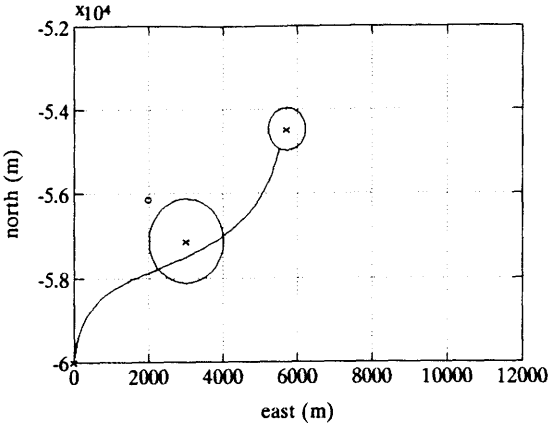
(c) Normal Acceleration Command



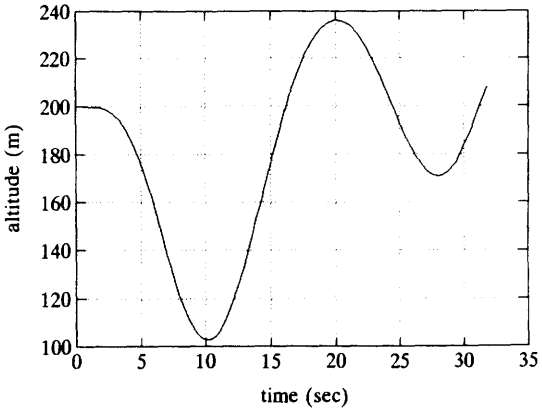
(d) Bank Command

Test Segment No. 3

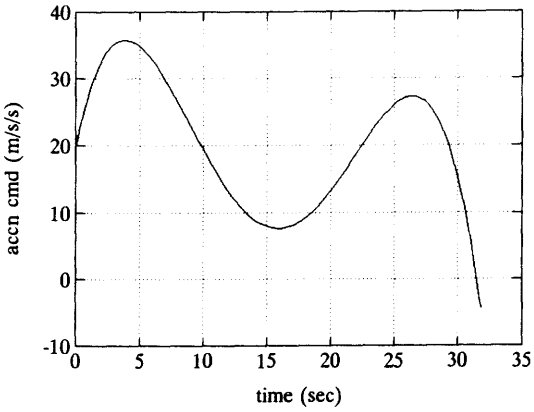
Figure 6.3



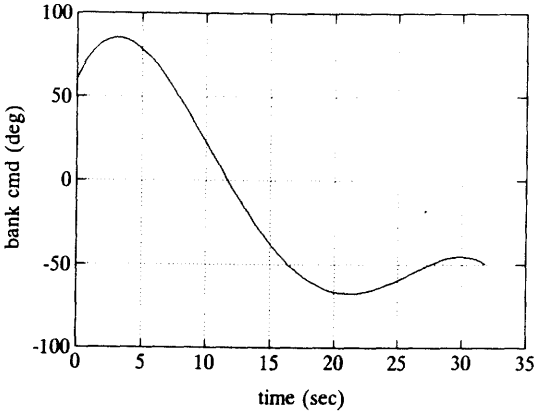
(a) Trajectory



(b) Altitude



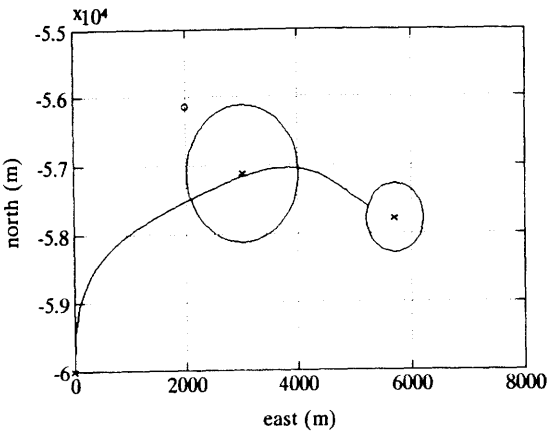
(c) Normal Acceleration Command



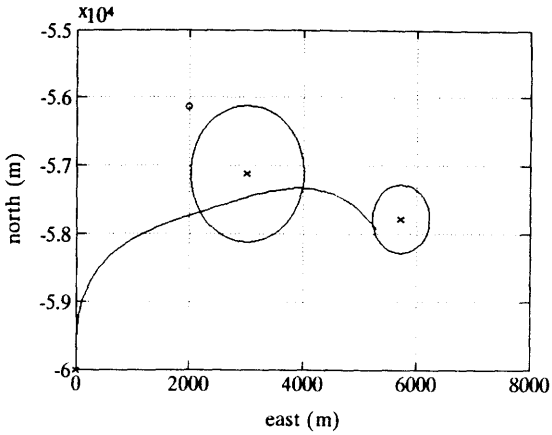
(d) Bank Command

Test Segment No. 4

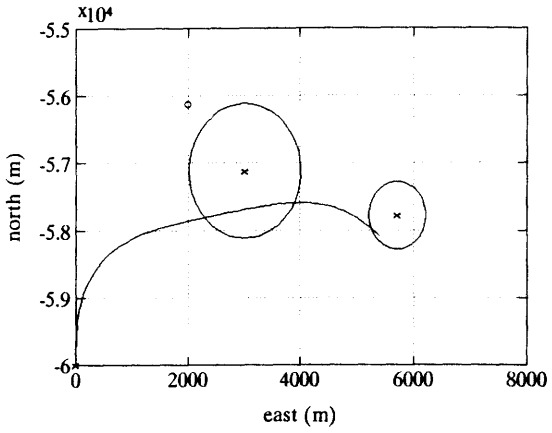
Figure 6.4



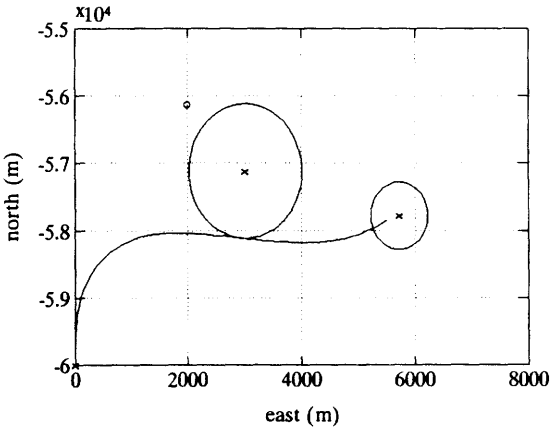
(a) Constraint = 40 ms^{-2}



(b) Constraint = 45 ms^{-2}



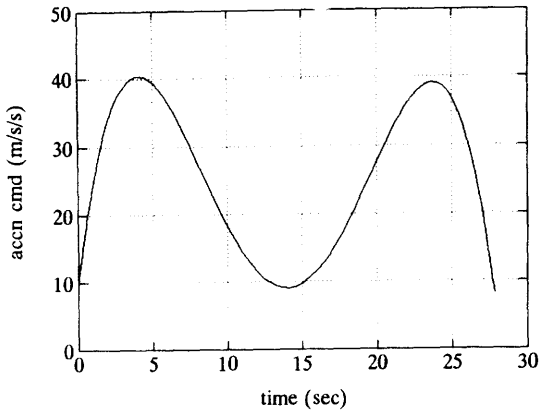
(c) Constraint = 50 ms^{-2}



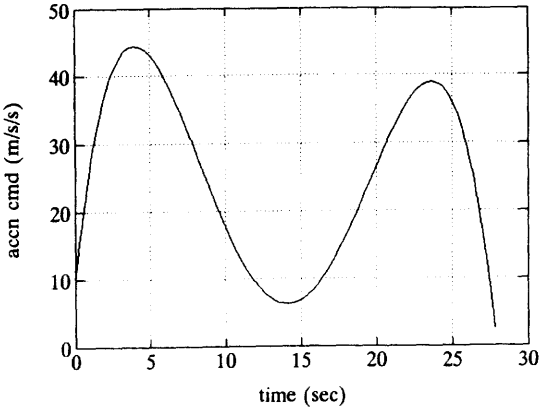
(d) Constraint = 100 ms^{-2}

Effect of Varying Constraint Level

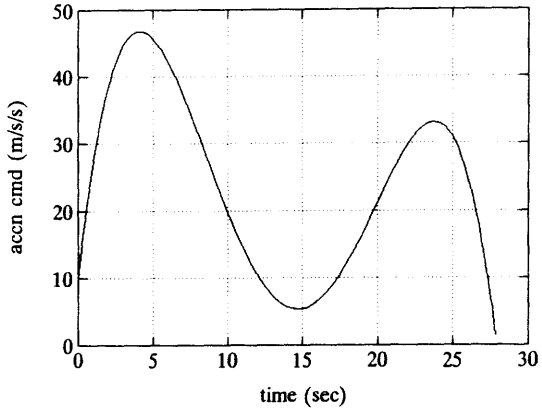
Figure 6.5 (i) Trajectory



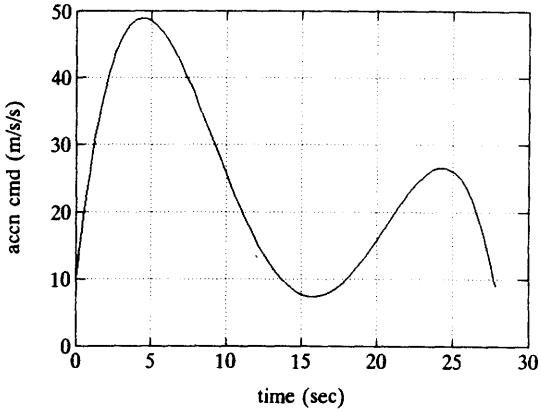
(a) Constraint = 40 ms^{-2}



(b) Constraint = 45 ms^{-2}



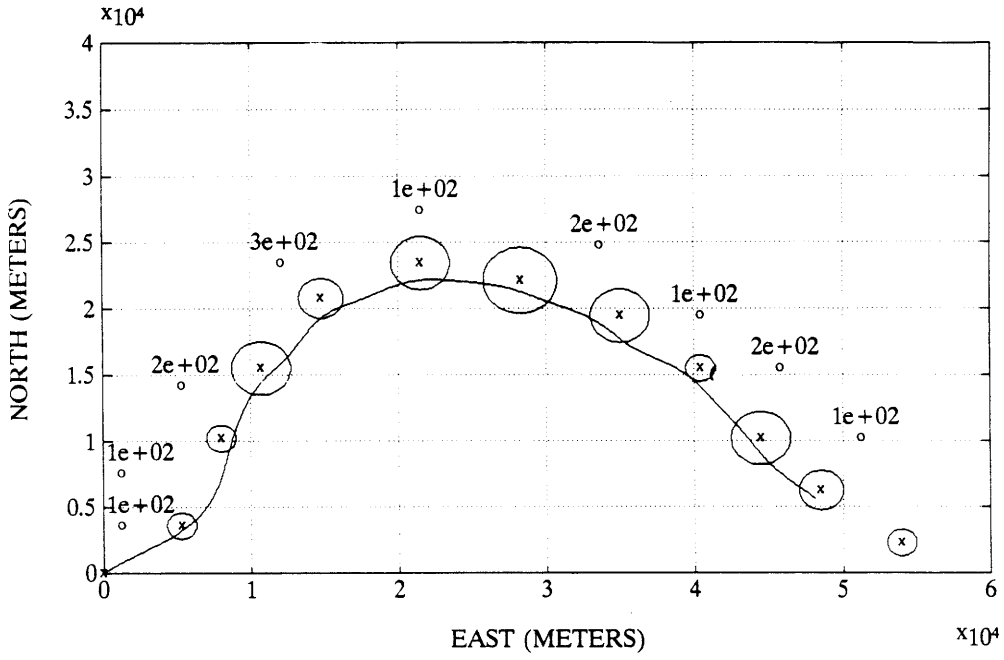
(c) Constraint = 50 ms^{-2}



(d) Constraint = 100 ms^{-2}

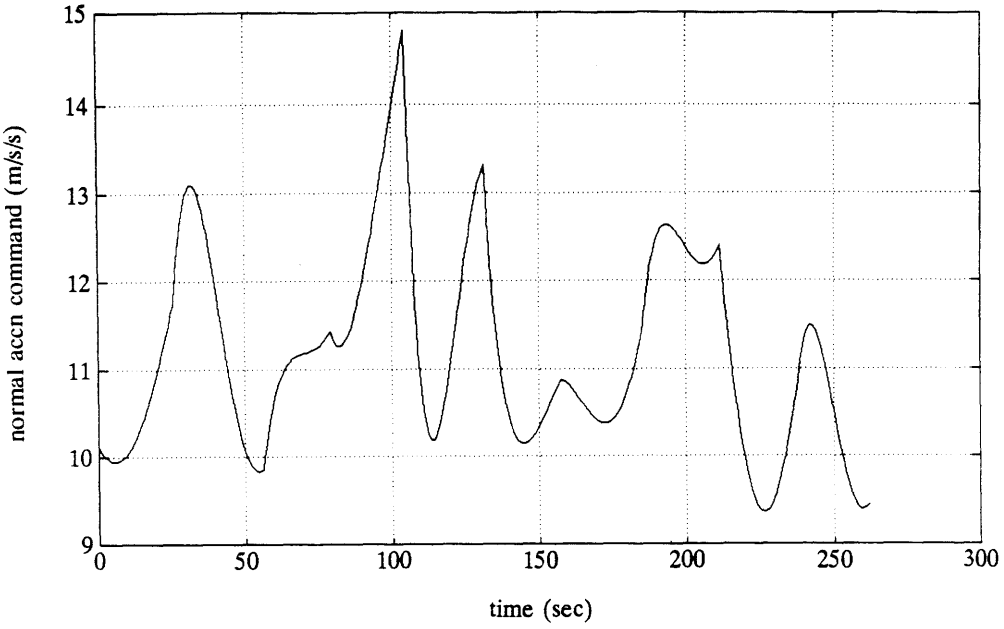
Effect of Varying Constraint Level

Figure 6.5 (ii) Acceleration Command

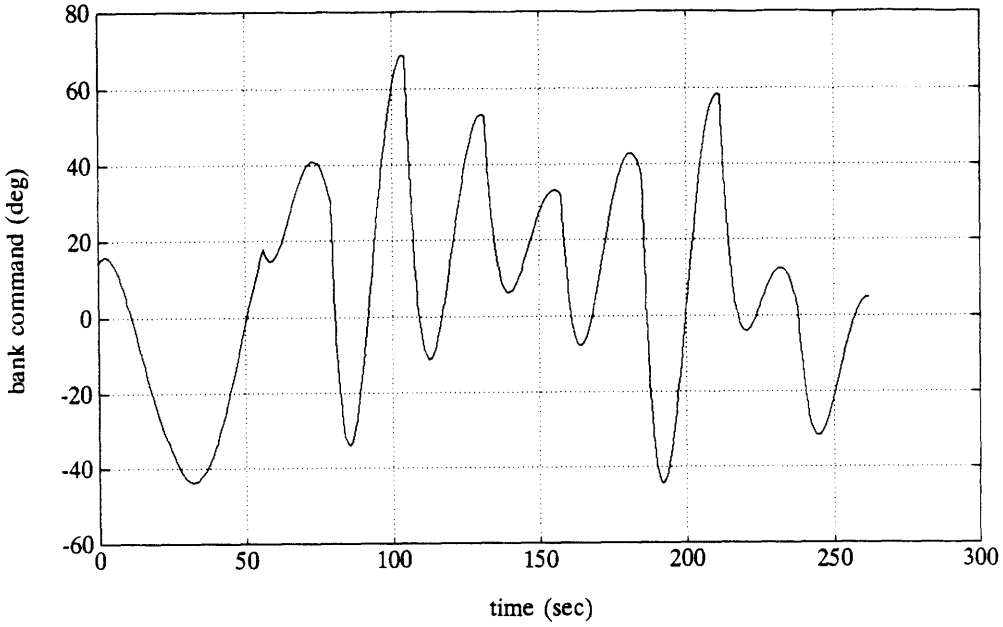


Construction of Extended Command Histories

Figure 6.6 (a)



Normal Acceleration



Bank Angle

Construction of Extended Command Histories

Figure 6.6 (b)

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

A technique by which the process of generating Flight Control System commands can be integrated into the lowest level of a Mission Planning System has been developed within the guidelines imposed by the requirements of a high performance military aircraft. The essential information upon which the choice of commands is based is supplied from higher levels in the planning hierarchy. It includes a sequence of waypoints which defines a desirable trajectory to the extent that this can be done without reference to the capabilities of a specific vehicle. While these waypoints will be chosen so that major threat concentrations are avoided, they will normally be too widely spaced to provide a basis for construction of the best possible trajectory, as determined by the degree to which it minimizes an appropriately defined measure of risk. Further definition of the minimum risk trajectory, and therefore of the required commands, calls for information about the dynamics of the aircraft and its limitations in addition to quantitative knowledge of the threat environment. In this thesis, the required commands are found by minimization of the risk with respect to parameters that span a suitable space of control functions, subject to constraints imposed by the waypoints and vehicle limitations. An algorithm which performs this constrained minimization using a penalty function approach is developed and presented. The method has been shown to perform satisfactorily in a variety of scenarios involving realistic waypoint and threat geometries.

From the outset it was clear that this approach to the generation of commands would be computationally intensive in view of the fact that each

evaluation of the risk function calls for a simulation of the aircraft dynamics. The speed with which the algorithms could be executed on a Personal Computer was much slower than that required for real time operation. The most time consuming tasks were coded in Fortran. Some improvement could be expected from a machine code implementation, but it is clear a substantially faster processor would be required for real–time operation.

7.2 Recommendations

In this thesis, a very simple model of the aircraft and its control system has been used. A question which merits further attention is that of exactly how accurate this model needs to be. Modeling errors will manifest themselves as discrepancies between the actual trajectory flown by the aircraft in response to a given set of commands and the computed trajectory predicted by the model. Since the computed trajectory is intrinsic to the process by which risk is evaluated, modeling errors will lead to the selection of non–optimal commands. On the other hand, the aircraft will always be subject to external disturbances (wind gusts, shear etc.) and there is little to be gained by using a better model unless the modeling errors are significantly larger than the unavoidable trajectory perturbations due to these effects. Indeed, there is a substantial price to be paid for increasing the modeling accuracy in terms of the increased computational burden associated with evaluating the risk function. Future work should therefore be directed towards establishing a quantitative comparison between trajectory errors due to modeling and those due to disturbances.

A related question is that of how frequently a new command segment can be computed. In this thesis it has been assumed that it is sufficient to do this each time the aircraft passes a waypoint, i.e. approximately every 5000m. Between these updates, there is no protection against disturbances, so it is certainly desirable to update more frequently if there is enough computing power to do so. If this is not possible, attention should be directed towards the introduction of continuous position and velocity feedback to enhance the disturbance rejection capability of the system. This would also suppress the effects of modeling errors and so reduce the required complexity of the model used in the optimization.

The implications of increasing the order of the polynomial form used for the command functions from four to some higher value should be investigated (refer to Section 2.4). Any consequent improvement in performance would have to be weighed against increased complexity in the risk function.

The simple threat function used in this work has a valid physical basis, but its form was chosen in a somewhat arbitrary way. Some consideration should be given to the formulation of a general policy for the construction of threat databases using realistic data which may be incomplete or discontinuous.

An extremely important question is that of how to assess the fundamental reasonability of each solution segment before executing it. One situation which could lead to the generation of an unreasonable solution would arise if the algorithm converged to a non–global minimum. Given the complexity of the risk function, it is probably impossible to test for this condition explicitly but suitable validity tests may reduce its impact on flight safety to an acceptable level, at least in the context of an unmanned vehicle.

Finally, it would be worthwhile to investigate the implications of using the techniques described here in vehicles other than a high performance military aircraft.

REFERENCES

[1] **Alexander, James R.**
Flight Control Interface for a Real Time Trajectory Planning System
Master's Thesis, Massachusetts Institute of Technology 1988

[2] **Fox, L., Parker, I.B.**
Chebyshev Polynomials in Numerical Analysis
Oxford University Press 1968

[3] **Scales, L.E.**
Introduction to Non–Linear Optimization
Springer–Verlag 1985

[4] **Gill, P.E., Murray, W., Wright, M.H**
Practical Optimization
Academic Press 1981

[5] **Polak, E.**
Computational Methods in Optimization
Academic Press 1971

[6] **Brent, R.P.**
Algorithms for Minimization Without Derivatives
Prentice Hall 1973

[7] **Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.**
Numerical Recipes
Cambridge University Press 1986