CRANFIELD INSTITUTE OF TECHNOLOGY

COLLEGE OF AERONAUTICS

DEPARTMENT OF AEROSPACE TECHNOLOGY

PhD THESIS

Academic Year 1989-1991

H M PASARIBU

An Approach To Configuration Design Synthesis of Subsonic Transport Aircraft Using Artificial Intelligence Techniques

Supervisor : Dr J P Fielding

December 1991

This thesis is submitted in partial fulfilment of the requirements for the Degree of Doctor of Philosophy.

ProQuest Number: 10832381

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

ProQuest 10832381

Published by ProQuest LLC (2019). Copyright of the Dissertation is held by Cranfield University.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

to
my mother,

many women do noble things
but you surpass them all.
Proverbs 31:29

ABSTRACT

This thesis outlines a computer system developed to tackle the configuration design synthesis of subsonic transport aircraft. The system provides an interactive design environment which combines the reasoning process of the aircraft configuration definition and the numerical analyses that lie behind the reasoning.

The system applies Artificial Intelligence (AI) techniques to the configuration analysis. This gives the system the capability to reason with the configuration choices.

The system is built in modules, with each module is clearly separated to deal with a complete analysis in a specific aspect of design. The integration of the modules is done through an intelligent interface and a common database. The interface also incorporates AI techniques in deciding the sequence of execution of the modules and the processing of the input/output data for a particular module.

The interface permits the sharing of information among the modules and supports modularity and flexibility of the system for future development. Since each module is independent, it can be easily modified or replaced without disturbing the balance of the system. An extensive library of application programs is included in the engineering analysis module which enable detailed analysis to be performed.

The system is equipped with a simplified database management and a special purpose graphics module which is extensively used for the presentation of the output.

The system has been validated and tested. It can handle both propeller driven and jet engined aircraft. An example of the case studies is presented.

The improvement of the system for future development is also considered. These include the extension of the knowledge base for dealing with problems in other modules, the enhancement of the application programs in the engineering module, and a possible interface with an established Computer Aided Drafting (CAD) system.

CONTENTS

ABSTRACT i
CONTENTS ii
FIGURES ix
NOTATION xi

CHAPTER 1 INTRODUCTION 1

1.1 AIRCRAFT DESIGN PROCESS 1
1.2 COMPUTER SYSTEMS FOR AIRCRAFT DESIGN 4
1.3 APPLICATIONS OF ARTIFICIAL INTELLIGENCE IN AIRCRAFT DESIGN 6
1.4 THE AIRCRAFT CONFIGURATION DESIGN SYSTEM FOR SUBSONIC TRANSPORT AIRCRAFT 9

CHAPTER 2 KNOWLEDGE-BASED AIRCRAFT DESIGN SYSTEM 12

2.1 INTRODUCTION 12
2.2 THE MODEL OF AIRCRAFT DESIGN PROCESS 12
2.2.1 PARAMETRIC ANALYSIS 16
2.2.2 CONFIGURATION ANALYSIS 16
2.2.3 FUSELAGE DESIGN 17
2.2.4 WING DESIGN 17
2.2.5 ENGINE SELECTION 17
2.2.6 WEIGHT AND BALANCE 18
2.2.7 AERODYNAMIC ANALYSIS 18
2.2.8 FLIGHT ANALYSIS 18
2.2.9 LANDING GEAR DESIGN 19
2.2.10 PERFORMANCE ANALYSIS 19
2.2.11 STABILITY AND CONTROL ANALYSIS 19
2.2.12 COST ESTIMATION 19
2.2.13 DESIGN OPTIMISATION 20
2.3 IMPLEMENTATION OF THE KNOWLEDGE-BASED AIRCRAFT DESIGN SYSTEM 20
2.3.1 SELECTION OF COMPUTER LANGUAGE 20
2.3.2 SYSTEM ARCHITECTURE 22

CHAPTER 3 KNOWLEDGE BASE OF AIRCRAFT DESIGN 25

3.1 SOME ASPECTS OF DESIGN KNOWLEDGE 25

3.2 CONFIGURATION DESIGN KNOWLEDGE 26
3.2.1 FUSELAGE LAYOUT 27
3.2.2 WING POSITION 29
3.2.3 ENGINE CONFIGURATION 30
3.2.3.1 Propulsion Type 31
3.2.3.2 Engine Type 31
3.2.3.3 Engine Position 31
3.2.4 TAIL GEOMETRY AND ARRANGEMENT 32
3.2.4.1 Tail Type 33
3.2.4.2 Tail Layout 35
3.2.5 LANDING GEAR CONFIGURATION 37
3.2.5.1 Landing Gear Layout 37
3.2.5.2 Landing Gear Position 38
3.2.6 FLAP TYPE 39
3.3 KNOWLEDGE REPRESENTATION 39
3.3.1 THE CHART OF DESIGN TASKS 40
3.3.2 PRIMITIVE PROPOSITIONS 41
3.3.3 KNOWLEDGE SOURCE 42
3.3.4 RULESETS 42

CHAPTER 4 INFERENCE PROCEDURE 44

4.1 INTRODUCTION 44
4.2 THE STATUS RECORDER 46
4.2.1 THE QUEUE OF DESIGN TASKS 46
4.2.2 THE DATABASE ELEMENTS OF THE STATUS RECORDER 47
4.3 IMPLEMENTATION INFERENCE MECHANISM 48
4.3.1 SELECTING A TASK TO EXECUTE 48
4.3.2 EVALUATING A TASK 48
4.3.3 SOLVING TASK INTERACTION PROBLEMS 49
4.3.4 EXPLAINING THE REASONING PROCESS 49

CHAPTER 5 INTERFACING WITH ENGINEERING MODULES 50

5.1 SYSTEM ARCHITECTURE 50
5.2 DEVELOPMENT AND MODIFICATION OF ENGINEERING MODULES 51
5.2.1 OPDOT 53
5.2.2 FUSELAGE 54
5.2.3 WEIGHT ESTIMATION 55
5.2.4 DRAG ESTIMATION 55
5.2.5 ACCOST (Aircraft Cost Program) 56
5.2.6 FLTANAL (Flight Analysis Program) 56
5.2.7 MITCHELL (Aircraft Stability And Control) 57

5.2.8 ENGINE DATABASE 57
5.2.9 PARAMETRIC ANALYSIS 58
5.3 IMPLEMENTATION OF THE ENGINEERING MODULES 60
5.3.1 SYSTEM CONTROLLER 60
5.3.2 AIRCRAFT DATABASE 61
5.3.3 PROGRAM INTERFACE 62

CHAPTER 6 CASE STUDIES 64

6.1 INTRODUCTION 64
6.2 DESIGN REQUIREMENTS FOR THE X-91 AIRCRAFT 65
6.3 DESIGN STUDY FOR THE X-91 AIRCRAFT 66
6.3.1 PARAMETRIC ANALYSIS FOR INITIAL SIZING 66
6.3.2 CONFIGURATION ANALYSIS 68
6.3.3 FUSELAGE DESIGN 75
6.3.4 WING DESIGN 76
6.3.5 ENGINE SELECTION 76
6.3.6 WEIGHT ESTIMATION 78
6.3.7 AERODYNAMIC ANALYSIS 78
6.3.8 FLIGHT ANALYSIS 78
6.3.9 COST ESTIMATION 79
6.4 SUMMARY 80

CHAPTER 7 DISCUSSIONS 81

CHAPTER 8 CONCLUSIONS AND RECOMMENDATIONS 84

ACKNOWLEDGMENTS 85

REFERENCES 86

APPENDICES

APPENDIX A DESCRIPTION AND APPRAISAL OF EXISTING AIRCRAFT DESIGN COMPUTER PROGRAMS 94

A.1 INTRODUCTION 94
A.2 CONVENTIONAL DESIGN SYNTHESIS PROGRAMS 95
A.2.1 BIZJET 96
A.2.2 GATEP (General Aviation Twin Engines, Propeller Driven Aircraft) 96

A.2.3 CASTOR (Commuter Aircraft Synthesis and Trajectory Optimisation Routine) 97
A.2.4 MVO (Multi Variate Optimisation) PROGRAM 97
A.2.5 ADAS (Aircraft Design and Analysis System) 99
A.2.6 CAPDA (Computer Aided Preliminary Design of Aircraft) 100
A.2.7 CPDS (Computerised Preliminary Design Synthesis) 101
A.2.8 ACSYNT (Aircraft Synthesis) 102
A.2.9 CONCLUSIONS 102
A.3 THE USE OF ARTIFICIAL INTELLIGENCE TECHNIQUES IN AIRCRAFT DESIGN SYNTHESIS PROGRAMS 104
A.3.1 PASS (Program for Aircraft Synthesis Studies) 104
A.3.2 RUBBER AIRPLANE 105
A.3.3 DSIDES (Decision Support In the Design of Engineering Systems) 106
A.3.4 ADROIT (Aircraft Design by Regulation Of Independent Tasks) 107
A.3.5 ACES (Aircraft Configuration Expert Systems) 107
A.3.6 CONCLUSIONS 108

APPENDIX B DESCRIPTION AND APPRAISAL OF KNOWLEDGE-BASED TOOLS, LANGUAGE AND TECHNIQUES 112

B.1 KNOWLEDGE-BASED SYSTEMS 112
B.2 KNOWLEDGE ACQUISITION AND REPRESENTATION 114
B.2.1 KNOWLEDGE ACQUISITION 114
B.2.2 KNOWLEDGE REPRESENTATION 114
B.2.2.1 Production Rule 115
B.2.2.2 Predicate Logic 116
B.2.2.3 Structured Objects 117
B.3 LANGUAGES AND TOOLS FOR BUILDING KNOWLEDGE-BASED SYSTEMS 118
B.3.1 FORTRAN 119
B.3.2 PROLOG 119
B.3.3 SMALLTALK 120
B.3.4 C AND C++ 121
B.4 SOME USEFUL TECHNIQUES 122
B.4.1 CONSTRAINT PROPAGATION 123
B.4.2 OBJECT-ORIENTED PROGRAMMING 124
B.4.3 TREE AND GRAPH IN PROBLEM SOLVING 124

APPENDIX C DETAILS OF KNOWLEDGE REPRESENTATION 127

C.1 INTRODUCTION 127

C.2 REPRESENTATION OF KNOWLEDGE BASE RULES 128
C.2.1 STRUCTURE KNOWLEDGE SOURCE 128
C.2.1 STRUCTURE RULESETS 132
C.3 THE KNOWLEDGE BASE FILES 133

APPENDIX D DETAILS OF INFERENCE ENGINE 135

D.1 INTRODUCTION 135
D.2 THE STATUS RECORDER 136
D.2.1 THE QUEUE OF DESIGN TASKS 136
D.2.2 DATABASE ELEMENTS OF THE STATUS RECORDER 137
D.3 THE INFERENCING PROCEDURE 139
D.3.1 SELECTING A TASK TO EXECUTE 139
D.3.2 EVALUATING A TASK 139
D.3.3 SOLVING TASK INTERACTION PROBLEMS 140
D.3.4 EXPLAINING THE REASONING PROCESS 141

APPENDIX E DEVELOPMENT OF THE SYSTEM USER INTERFACE 143

E.1 INTRODUCTION 143
E.1.1 MOUSE 143
E.1.2 VISUAL OBJECTS IN SUNVIEW 144
E.1.2.1 Panel Buttons 144
E.1.2.2 Panel Items 145
E.1.2.3 Menu 146
E.1.2.4 Scrollbar 146
E.1.2.5 Alert 146
E.1.2.6 Pointer 146
E.2 SYSTEM USER INTERFACE 148
E.2.1 AIRCRAFT DESIGN WINDOW 148
E.2.1.1 The Main Control Panel 150
E.2.1.2 Module Interface Panel 151
E.2.1.2.1 Editing Mode 152
E.2.1.2.2 Execution Mode 155
E.2.1.3 Help Control Panel 156
E.2.2 AIRCRAFT DATABASE SYSTEM WINDOW 158
E.2.2.1 Database Control Panel 158
E.2.2.2 Data Display Panel 160
E.2.2.3 Database Help Panel 161
E.2.3 CONFIGURATION ANALYSIS WINDOW 162
E.2.3.1 Display Sub-window 163
E.2.3.2 Configuration Control Panel 163
E.2.4 TEXT WINDOW 165
E.3 THE AIRCRAFT DATABASE 167

APPENDIX F DETAILS OF APPLICATION PROGRAMS IN THE ENGINEERING MODULES 168

F.1 INTRODUCTION 168
F.2 OPDOT (Optimum Preliminary Design Of Transport Aircraft) 168
F.2.1 OPTIMISATION CODE 168
F.2.2 EVALUATION OF THE UNAUGMENTED PERFORMANCE INDEX 169
F.2.3 EVALUATION OF DESIGN CONSTRAINTS 171
F.2.4 MODIFICATION IN VERSION 2.0 172
F.2.5 PROGRAM IN USE 173
F.3 FUSELAGE 176
F.4 WEIGHT ESTIMATION 177
F.5 DELTA (Drag Estimation Program) 180
F.6 ACCOST (Aircraft Cost Estimation) 181
F.6.1 PROGRAM DESCRIPTION 181
F.6.2 PROGRAM IN USE 185
F.6.2.1 Executing the Program 185
F.6.2.2 Input Data 185
F.7 FLTANAL (Flight Analysis Program) 188
F.7.1 PROGRAM DESCRIPTION 188
F.7.2 PROGRAM IN USE 189
F.7.2.1 Executing the Program 190
F.7.2.2 Input Data 190
F.8 MITCHELL (Aircraft Stability and Control Analysis) 192
F.8.1 INTRODUCTION 192
F.8.2 PROGRAM DESCRIPTION 193
F.8.3 PROGRAM IN USE 194
F.8.3.1 Input Data in Editing Mode 194
F.8.3.2 Execution Mode 196
F.8.3.3 Executing the Program 197

APPENDIX G DETAILS OF PROGRAM OUTPUT 207

G.1 INTRODUCTION 207
G.2 FUSELAGE 207
G.2.1 INPUT DATA 207
G.2.2 OUTPUT DATA 207
G.3 WEIGHT ESTIMATION 212
G.3.1 INPUT DATA 212
G.3.2 OUTPUT DATA 212
G.4 AERODYNAMIC ANALYSIS 214
G.4.1 INPUT DATA 214
G.4.2 OUTPUT DATA 214
G.5 FLIGHT ANALYSIS 218

G.5.1 INPUT DATA 218
G.5.2 OUTPUT DATA 218
G.6 COST ESTIMATION 224
G.6.1 INPUT DATA 224
G.6.2 OUTPUT DATA 225
G.7 CONFIGURATION ANALYSIS 230
G.7.1 EDITING MODE 230
G.7.2 EXECUTION MODE 230

FIGURES

Figure 1.1	Aircraft design cycle	1
1.2	Aircraft design phases	2
2.1	A typical design procedure of transport aircraft	14
2.2	The structure of aircraft design modules	15
2.3	System architecture	23
4.1	The strategy employed in the inference engine	46
5.1	Interface with engineering modules	52
5.2	First-order sizing method	59
6.1	Wing loading variation in parametric analysis	68
6.2	Execution of the configuration analysis module	70
6.3	The presentation of the output of the fuselage design module	77
A.1	The knowledge-based system as a design assistant	109
A.2	The knowledge-based system as an intelligent front end	110
B.1	Components of a knowledge-based system	113
B.2	The tree structure	125
B.3	The graph structure	126
C.1	C/C++ representation of the structure Knowledge Source	129
C.2	C/C++ representation of the structure Problem Source	131
C.3	C/C++ representation of the structures Ruleset and Problem Rules	131
D.1	C/C++ representation of the structure Queue	137
D.2	C/C++ representation of the structure Database	138
E.1	Three-buttoned mouse	144
E.2	Selecting an item from a menu in aircraft design window	147
E.3	The aircraft design window in execution mode	149
E.4	The aircraft design window in editing mode	153
E.5	The aircraft design window in execution mode for the parametric analysis module	157
E.6	The help facility in aircraft design window	159
E.7	The aircraft configuration window	164
F.1	General optimisation scheme in OPDOT program	198
F.2	Flow diagram for evaluating the unaugmented performance index	199

F.3	Flight plan used in OPDOT and FLTANAL programs	200
F.4	Climb procedure in subroutine CLIMB used in OPDOT and FLTANAL programs	201
F.5	The fuselage design procedure	202
F.6	ACCOST program flow diagram	204
F.7	FLTANAL program flow diagram	205
F.8	Descent procedure in subroutine DESCENT of FLTANAL program	206

FIGURES

NOTATION

a	Speed of sound
AR	Aspect ratio of wing
c_i	Actual value of the i^{th} constraint
C	Constants used in weight estimation
C_D	Aircraft drag coefficient
C_{Do}	Aircraft zero-lift drag coefficient
$C_D(1)$	Development cost for engineering hour
$C_D(2)$	Development cost for development support
$C_D(3)$	Development cost for flight test operation
$C_D(4)$	Development cost for tooling hour
$C_D(5)$	Development cost for manufacturing labour hour
$C_D(6)$	Development cost for quality control hour
$C_D(7)$	Development cost for manufacturing material
C_{EF}	Cost escalation factor
C_L	Aircraft lift coefficient
C_{LB*}	Economical cruise lift coefficient
C_{OPSHR}	Aircraft operating cost per hour ($)
$C_P(1)$	Production cost for engineering hour
$C_P(2)$	Production cost for development support
$C_P(3)$	Production cost for flight test operation
$C_P(4)$	Production cost for tooling hour
$C_P(5)$	Production cost for manufacturing labour hour
$C_P(6)$	Production cost for quality control hour
$C_P(7)$	Production cost for manufacturing material
D_{en}	Engine diameter
D_f	Maximum diameter of fuselage cross section
D_{REH}	Dollar rate for engineering hour
D_{RMH}	Dollar rate for manufacturing hour
D_{RQH}	Dollar rate for quality control hour
D_{RTH}	Dollar rate for tooling hour
f	Unaugmented performance index in optimisation scheme
F	Augmented performance index in the optimisation scheme
F_{DIF}	Cost factor to account for the level of complexity of the aircraft technology
F_{FTOH}	Cost factor to account for the flight test overhead cost of the aircraft
F_{OBS}	Cost factor to account for the level of observability the aircraft
H_{ec}	Altitude at the end of cruise flight
H_f	Maximum height of fuselage cross section

g_i	Constraint penalty function
K	Penalty function gains
K_θ	Feed back gains in pitch
L_{en}	Length of engine
L_f	Length of fuselage
L/D	Lift over drag ratio
M	Mach number
N_{oe}	Number of engines
P	Design parameters
Q_D	Number of aircraft used for testing during development phase of the aircraft
Q_P	Number of aircraft produced during production/service phase of the aircraft
Q_{Pl}	Total number of aircraft produced for the entire period of the aircraft project, $Q_D + Q_P$
R	Aircraft range
$R_{DESCENT}$	Distance covered during descent flight
S	Wing area
$S_{\ell i}$	Lower boundary of the i^{th} constraint
S_{ui}	Upper boundary of the i^{th} constraint
S_{to}	Take-off distance
T	Thrust of engine
T_{PFT}	Total flight test hour flown by the manufacture before delivering each aircraft to customers
T_{ref}	Gross thrust of reference engine
T_{req}	Required engine thrust
T_{to}	Gross thrust of engine
T/W	Thrust to weight ratio of aircraft
U_o	Aircraft steady state flight speed
V_{cr}	Cruise speed of aircraft
V_D	Aircraft descent speed
V_s	Aircraft stall speed
V_S	Aircraft diving speed
W_{cargo}	Weight of aircraft
W_{ci}	Component weight of the aircraft
W_{en}	Weight of engines

W_f	Weight of aircraft fuselage
W_{fixed}	Fixed weight of the aircraft
W_{oe}	Empty weight of aircraft
W_{pax}	Weight of passengers
W_{pl}	Payload weight of aircraft
W_{to}	Take-off weight of aircraft
W_{tu}	Weight of tail unit
W_w	Weight of aircraft wing
W/S	Aircraft wing loading
$\mathbf{X}$	Design variables
X_i	The i^{th} independent variable
V_{li}	Lower boundary of the i^{th} independent variable
V_{ui}	Upper boundary of the i^{th} independent variable
Z_i	The i^{th} transformed variable
α	Aircraft angle of attack
γ	Gradient angle during climb or descent
σ	Relative air pressure

Subscript :

max	Maximum condition
to	take-off condition
ref	reference condition
req	required condition

CHAPTER 1
INTRODUCTION

1.1 AIRCRAFT DESIGN PROCESS

Aircraft design [1,2] is a distinct activity that applies different disciplines of aeronautical engineering, i.e aerodynamics, structures, controls and propulsion, to define the geometry and performance of an aircraft. Unlike those disciplines which are analytical, aircraft design is a practical process which combines the analytical procedures used to determine what should be designed and how the design should be modified to better meet the requirement, and the synthesis of each individual discipline to form the actual layout of the aircraft.

Like most design activities, aircraft design is an iterative process, as depicted in Fig. 1.1. Requirements come from different sources, i.e. regulations which are set by the airworthiness authority, operators, and the technical capability of the manufacturing company. The performance requirements are set by both operators and the company based on design trade studies. Several design concepts are then developed to satisfy the requirements. This requires design analysis to investigate the feasibility and the practicality of the concepts in meeting the requirements. This analysis often leads to new concepts and technologies, which can initiate a completely new design effort.

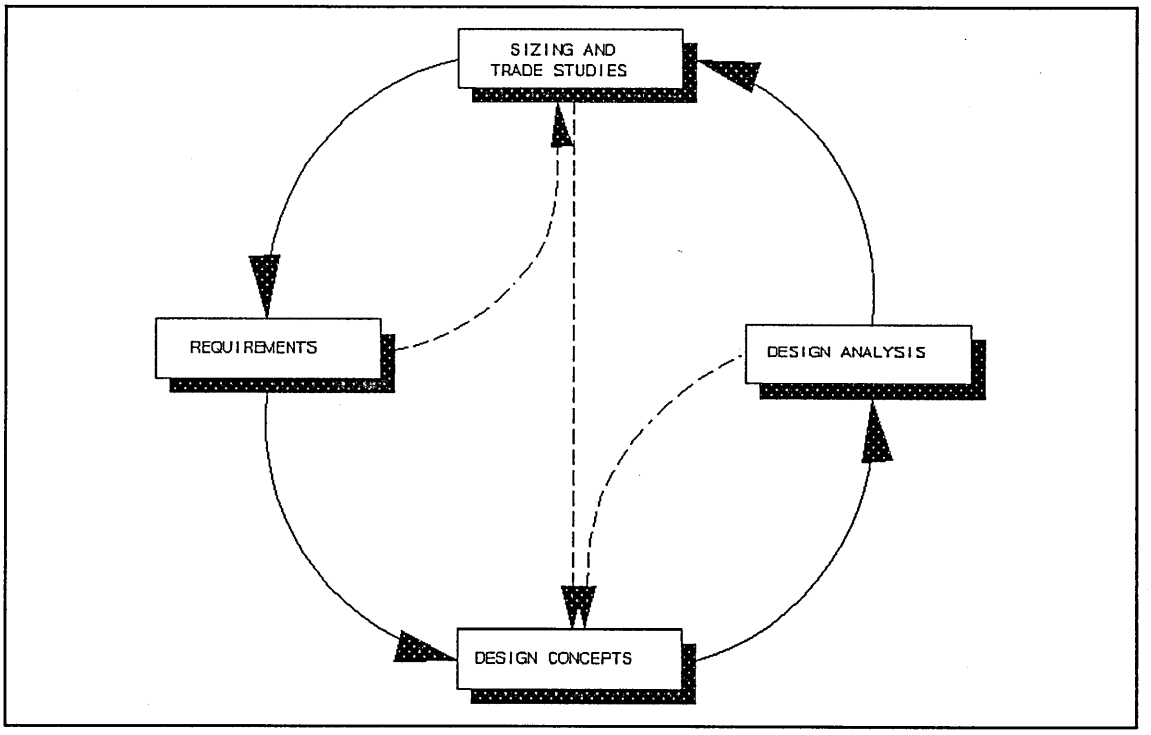


Fig. 1.1 Aircraft design cycle [1]

The aircraft design process can be divided into three major phases, namely conceptual design, preliminary design and detail design, as shown in Fig. 1.2.

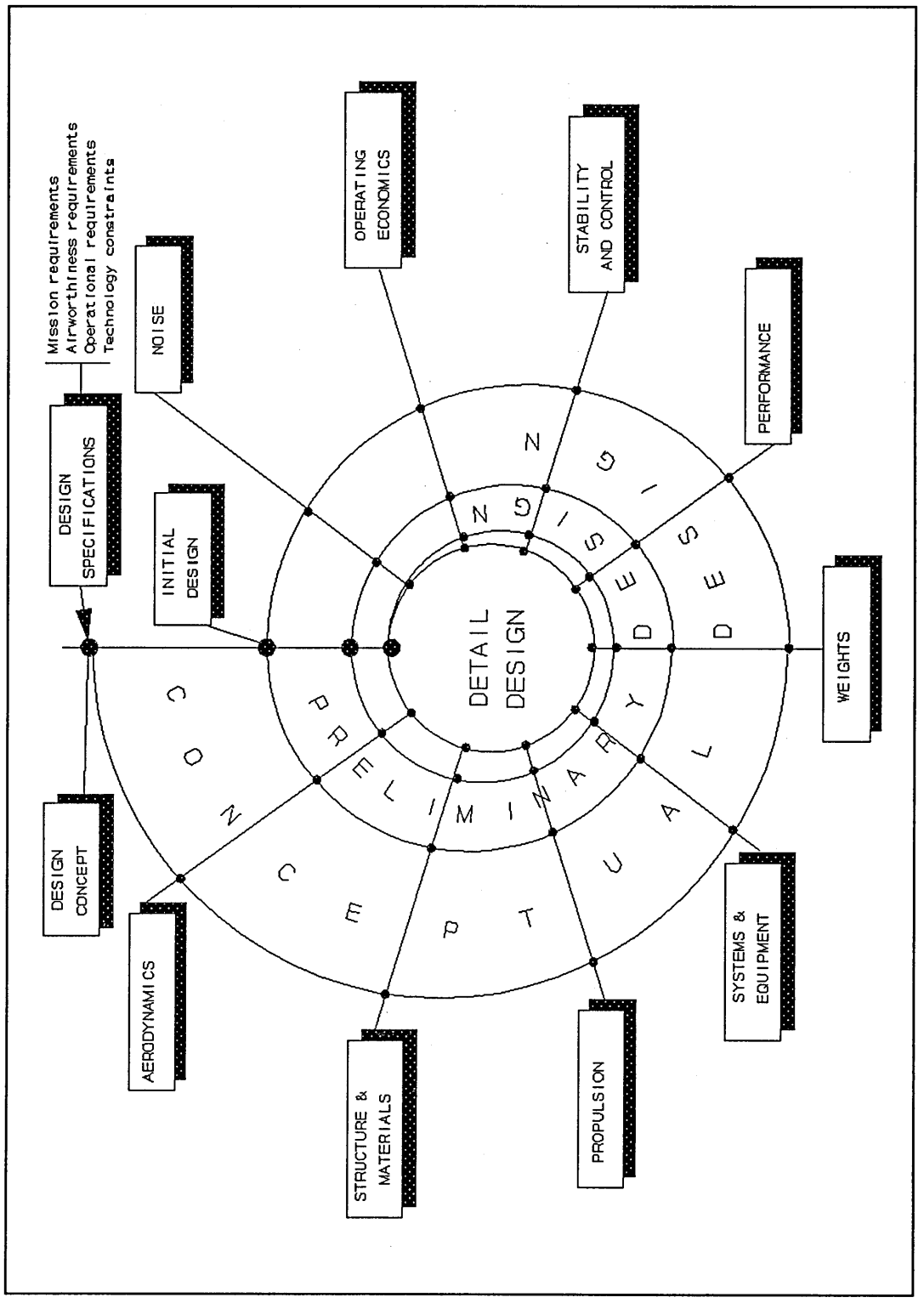


Fig. 1.2 Aircraft design phases [64]

The conceptual design phase deals with the initial configuration arrangement, size and weight, and performance. The objectives are to investigate the viability of the aircraft project and to obtain the first impression of its most important characteristics by conducting parametric study and requirement analysis. In some cases, some investigations are also carried out to gather information in order to decide whether the concept will be technically and economically feasible.

As shown in Fig 1.2, the analysis involves various disciplines in increasing detail. At first only simple analysis can be done to investigate various concepts and the effects of various parameters on the design. As more data is accumulated during the course of the project, more rigorous analyses can be performed. In many cases a few competing designs can be carried out in parallel.

The diagram shows that aircraft design is all about compromise, trying to find the best solution satisfying different requirements set by various disciplines. This will eventually result in the primary selection of the conceptual shape and functional characteristics of the major components of the aircraft, e.g. fuselage, wing, tail surfaces, engine installation and landing gear, and their position relative to each other.

The preliminary design phase begins when major changes are over. The configuration[1] layout of the major components can be expected to be fixed although minor revisions are still allowable especially in the areas which do not affect the whole design in great extent. Components that have not been assessed in great detail in the conceptual design phase need to be specified and evaluated.

The level of analysis involved becomes more detailed not only because most of the characteristics of major components have been well defined but also because minor components are beginning to be taken into consideration. In many cases, these analyses are carried out in different departments, each is responsible for one area of design. This aspect of design gives rise to a few interesting ideas. Organisation of data and communication between various departments play a major part in design activities.

During this period, some forms of testing may be required at certain stage of the process to validate the design, especially in areas related to the aerodynamics of the aircraft.

At some point in the process, the configuration is decided to be *frozen* when even minor changes cannot be tolerated. For this reason, the first two phases of the design process, from the start of the conceptual design up to the decision to freeze the configuration, is called *Configuration Development* phase.

[1] The expression *configuration* as used in this chapter refers to the general layout, the external shape, dimensions and other relevant characteristics. It is not intended to indicate the aircraft (flight) configuration such as the deflection angle of flaps, position of landing gear at certain flight conditions.

The ultimate objective during this phase is to prepare the company for the detail design stage, also called full-scale development.

During the detailed design phase, small structural parts that make the major components are designed and analyzed. The production of the components begins with the manufacture of the small parts and continues with the assembly of the components as the design analysis is completed. Some aspects of production design, e.g. ease of manufacture may come into play in the overall concept of the detail design, even though some allowance has been made for these aspects in the preliminary design.

During the detail design phase, structural and system testing gathers momentum. Fatigue testing of the complete airframe is carried out together with the validation test of various aircraft systems leading to the initial flight test of the first complete aircraft.

1.2 COMPUTER SYSTEMS FOR AIRCRAFT DESIGN

Before the invention of computers, research and design in the aircraft industry were conducted through limited analytical derivations and experimentations. Design studies, trade studies and performance evaluations were restricted to a small number of options that could be calculated by a finite number of people. Countless hours of research were dedicated to developing parametric databases that could be applied to new aircraft designs.

Since their invention, computers have changed the way design problems are solved. In the early days, they were mainly used for executing stand-alone programs. These programs were mainly discipline-specific and separately developed by specialists in different departments in the aircraft industry.

As computers and numerical methods advanced, increasingly more difficult problems could be tackled. Engineers and scientists in the aircraft industry became able to simulate the actual physics of a problem by solving the governing mathematical equations. The capability of analytical programs was greatly enhanced, yet they were still tailored specifically to solving discipline-specific problems.

The introduction of computer graphics capabilities in the 1960s initiated a trend towards integrating the stand-alone analytical programs into a complete computerised design system. The application of such a system in an aircraft design project has proved to be invaluable to the improvement of the quality of the design. It also reduces the design cycle time of converging to a refined design, thereby increasing the design capacity in that more design studies can be performed.

A number of these integrated computerised design systems have been developed by different teams over many decades. A survey made by the author over a number of existing systems shows the potential benefits of the systems as stated earlier. An extract of this survey is reported in section A.2 of Appendix A, which gives detail descriptions

of each system.

The survey shows that an effective computerised conceptual and preliminary design programs should have the following characteristics :

- Modular
  All functions to be performed are clearly separated into program modules and can be assembled in various sequences of computations depending upon the problem.

- Integrated
  All individual modules should be linked together to form an integrated system in which data can be transferred to and from different modules.

  A control is required to exercise the individual modules and to transfer the output for additional computations. It should be possible to carry out direct analytical computations for each individual module in a particular discipline and to develop a scaling information for sensitivity studies. These scaling relationship are usually simplified by omitting parameters of secondary importance.

  Interfacing among the modules becomes a major problem area. In some applications, an executive program is used to carry out the control over the execution of various modules.

- Multi-level
  The program should be capable of performing analysis at different levels of detail which is required in different phases of the design process.

- Sensitivity Analysis
  This will enable the program to carry out assessment of critical areas in a design. The sensitivity study defines the feasibility of a particular design to meet a specified requirement and performance.

- Optimisation and Trade-off Studies
  The program should provide a method which enables the user to select the best concept based upon a specified performance criterion.

Many of these features have been implemented in the existing aircraft design systems to a certain degree of success. In addition to these features, future systems need to include the following characteristics :

- Transparency
  The user should be able to retain full monitoring capabilities of what the computer is doing and extract explanation if necessary. The main drawback of the existing systems is that the user sometimes does not understand the performance of the system. Some of the actions executed are out of the control of the user and it is very difficult to get hold of the whole process of the execution.

- Extensibility
  A primary feature of any new design system should be unprecedented extensibility in both the set of variables used by the program and the set of analytical routines. The representation of the geometric and mathematical model of the aircraft should be able to be updated continuously in conjunction with real life development of the design process. This particular requirement is especially important if the system is to be used to assess new technologies. The extension beyond the empirical knowledge of existing aircraft should be made possible. This is necessary to enable the system to be used for experiment with new concepts.

  To support this requirement, such a system normally incorporates a high degree of modularity to reduce complexity. This is done by separating the interface and database management functions from the computational routines [3].

- Flexibility
  Flexibility covers the way the design process is carried out. The sequence of various design tasks and the logic behind it should be able to be altered according to the problem involved and the data available.

  This flexibility should reflect the iterative nature of the aircraft design process. In the early stage not much information is available to carry out in-depth analysis of the design. During the conceptual design phase the design teams need to perform parametric studies to gather data which help support design decisions at a later stage.

  This flexibility is difficult to incorporate using conventional programming languages, such as Fortran, which is used in most of current programs. Although several attempts have been made on existing programs to include this feature, they are not as successful as with the other features. As conventional programming languages are more structured and procedural, the flexibility of the programs is greatly limited. Each computational path should be encoded into the system, which in Fortran by using IF-THEN-ELSE and GOTO statements which is a tedious and inefficient way of programming. Different procedures would then require different routines which add to the amount of computer memory required. This would increase the computing time considerably since the computer has to search through the routines to find the appropriate path before executing it. Furthermore the response would tend to be slower which overrides the usefulness of the feature.

1.3 APPLICATIONS OF ARTIFICIAL INTELLIGENCE IN AIRCRAFT DESIGN

Realising the problems listed in section 1.2, several attempts have been made recently to explore different ways of programming. Several systems such as PASS [4], Rubber Airplane [5], DSIDES [6,7], ADROIT [8,9], ACES [10] and Light-plane [11] attempted to investigate the use of current development in programming languages and programming

techniques in the field of Artificial Intelligence (AI) and Knowledge Based Systems (KBS) in aircraft design applications. A thorough discussion on these systems is reported in section A.3 of Appendix A. Appendix B presents a detailed discussion on several programming languages and techniques potentially suitable for AI and KBS applications.

All these attempts are aimed towards simulating the design procedure in the manner similar to a manually performed design. The approach emphasizes an interactive design in which the computer is used to help the designers make judgment and carry out design tasks.

The investigation reported in section A.3 of Appendix A leads to the following conclusions :

- There is a need for an efficient tool that deals with the configuration definition phase of aircraft design process :

  This need is reflected in all types of aircraft, but an application in commercial transport aircraft will be feasible due to several reasons :

  - the configuration choices in the design of commercial transport aircraft are more limited, and
  - the design knowledge is much easier to be acquired.

  The knowledge acquired in this process should be able to represent the real interaction between various configuration elements and help the designer decide possible aircraft configurations for certain design requirements.

  Most of the current systems are geared towards the sizing definition of the design process and less emphasis is laid on the configuration definition stage. The reasons are numerous, some of which are :

  1. The process involved in the sizing process and detailed design contains more mathematical computations, hence it is more structured and procedural which suits most of the conventional computer languages. The procedures are easier to be encoded in a program.

  2. There is great difficulty in acquiring and compiling the knowledge surrounding the configuration decisions. The interactions between various configuration decisions are complex and it is difficult to separate them into independent tasks. Some of the decisions taken in the process are based on *hard* information which can be derived from scientific principles and mathematical equations, but most of them are based on *soft* information, based on designer's judgment and experience.

     In many cases there are many different solutions that can be applied to a

particular configuration decision as many design teams take different approaches. There are many measures of merit for judging the level of acceptance of a design decision and all measures may not be equally important. One classical example is the comparison between AVRO Vulcan B-1 bomber and the Handley Page Victor or the Vickers Valiant. Although these aircraft were designed for basically the same specifications, the final layouts were completely different. It can be concluded that various solutions are possible, each with its own particular merits. The same could be said for the selection of the engine position in the design of the Boeing 737 aircraft in which two configurations were studied at the early stage: engines mounted in pods underneath the wing or on each side of the rear fuselage. It was difficult to support a particular layout and the final choice could only be made when the overall designs had been made for each engine position with the same specifications.

The hard information which consists mainly of sets of mathematical equations is easier to be represented in a computer program by using procedures or subroutines. In most cases there are many numerical routines available that can be applied to solve the problem. On the other hand, soft information contains causal relationships of symbolic statements. It is *heuristic* and cannot be defined easily as procedures as it is with hard information. This creates problems in most conventional computer languages in representing the information into computer codes.

Recent development in computer languages and programming techniques, however, has made it possible to deal with this problem using *Artificial Intelligence (AI)* techniques [12,13]. AI techniques offer system programmers the capability of symbolic computation to handle the soft information involved in the reasoning process.

Many attempts have been made to develop computer systems called *Expert Systems* which try to emulate the skills of human experts in dealing with problems in a narrow domain. The systems, which are also commonly called *Knowledge Based Systems (KBS)*, make use of expertise knowledge and reasoning to produce advice which could justify their conclusions and possibly could handle uncertainties in the domain problems.

Apart from the systems mentioned earlier, there are a number of research effort towards cultivating the benefit that the Knowledge-Based Systems could offer in engineering applications. Some of the examples are as an intelligent front end for computational aerodynamics [14], computer aided design [15], control engineering [16] and machining process [17].

In aircraft design applications the knowledge represents the theoretical or analytical background which support design decisions, and heuristic knowledge which consists of rules of thumb based on past designs and practical interrelationships among various disciplines.

Computer languages suited for these AI or KBS applications have built-in capability

for manipulating this heuristic knowledge into a stylized form that a computer program can use to help solve difficult problem. Some are also designed with techniques of inferencing through the compilation of design knowledge in search for the appropriate knowledge or routines.

- There is a need to include more rigorous engineering analyses :

This need is necessary in order to reasonably increase the accuracy of the programs. On the current development of the existing aircraft design systems mentioned earlier the emphasis has been on redesigning and restructuring the design procedures so that Artificial Intelligence techniques can be used. This entails simplifying the engineering modules so that they can be easily handled by the knowledge-based modules which do the reasoning process; thereby reducing the usefulness of the program in a practical design.

Since there are many analytical programs already available, it is reasonable to incorporate these programs into a new system. These programs have proved to be invaluable in performing design analyses with high level of accuracy. Therefore, the effort of meeting this need should be directed towards devising a system that could incorporate these programs and integrate them with a Knowledge-Based system to facilitate the reasoning process.

1.4 THE AIRCRAFT CONFIGURATION DESIGN SYSTEM FOR SUBSONIC TRANSPORT AIRCRAFT

This study leads to a project to develop a computer system for configuration design synthesis of subsonic transport aircraft. The new system should therefore be able to handle not only the symbolic computations involved in the configuration definition but also the numerical computation extensively used in other activities of design. The application of the Knowledge Based System techniques will complement the design system with the reasoning process that lies behind the numerical computations.

This reasoning ability should reflect the process of design, not only in the set of variables and tasks involved but also in the manner in which the design is carried out. The presentation of the reasoning process will certainly help the designer to understand the process of design as to why certain design decisions are concluded and what the main design parameters are in aircraft design activities.

One of the situations which frequently faced by designers is that only little information is available in the early stage of design project. This information mainly concerns the design specifications and operational requirements of the new aircraft. In such circumstances, it is expected of the designer to use his own intuition or past experience to draw conclusions based on limited information. The decisions, however tentative and

with low level of confidence, should then enable the designer to carry on with the design tasks and perform analytical studies to accumulate data.

As the design data base is built up during the course of the design study, some of the decisions taken at that early stage may need to be reviewed. Some of these analyses may lead to completely different conclusions in the light of more complete information.

The evidence from past aircraft projects shows that about 85% of the total project cost of an aircraft is actually related to the decisions taken in the conceptual and preliminary design phases [18]. This shows the importance of the design decisions in determining the success of the aircraft.

It is then required of the designer a more versatile ability of synthesis and analysis which enable him/her to construct a valuable assessment in the design problem encountered. This ability may reflect in :

- The ability to know when there is enough information to make decisions,
- The ability to suspend problem solving activity in a particular design task when information is not at hand,
- The ability to move between design tasks and to resume work in a suspended task as information becomes available
- The ability to combine information from various tasks

This ability requires a sound comprehension of the whole design process, not only in identifying the tasks but also the main parameters which participate in each task. The iterative nature of the aircraft design may require of a designer an understanding of the underlying aspect of design in which the design tasks are inter-related. This inter-relationships is so complex that no single task can actually be tackled individually without considering the others.

Therefore, this project involves the following tasks :

- To identify all the different modules involved in the design process of subsonic transport aircraft. This activity may entail supplying a suitable computer program for application in each module and developing an interface to integrate various modules.

- To identify design activities involved in configuration definition of aircraft. This activity should reflect the hierarchical interrelationships between various design steps.

- To compile the knowledge surrounding the configuration design of aircraft. This requires a comprehensive study on how this knowledge should be represented in a computer program in order to ensure an efficient use in solving a particular problem.

- To develop a prototype of the system which captures all the ideas accumulated in this course of study. The aim of this prototyping is to clarify the characteristic operation of the system by constructing a version that can be exercised. Prototyping is especially important if the user's requirements for a system are not well understood.

A prototype eases the discussion between the programmers and the users by allowing their discussions to focus on real behaviour in exercising the prototype.

For these reasons, the emphasis in prototyping was put on :

- programming efficiency :
  The aim is to build the prototype as quickly as possible so that the system can be assessed and validated.
- experimenting with representational and algorithmic ideas :
  Emphasis is laid on structuring the design knowledge for easy use and investigating different ways of calling external routines.
- largely non-numeric processing :
  Work is concentrated mainly on the configuration analysis in which the processing is largely non-numeric. This aspect of design will be the main part of the proposed system in which the Knowledge-Based System techniques can be directly applicable.
- implementation of inference mechanisms and interface with engineering modules
- implementation of knowledge representation formalism.

The next few chapters will explain these ideas in more detail. Chapter 2 gives an overview of the overall system. The compilation of design knowledge and its representation in a computer program will be tackled in Chapter 3. Chapter 4 discusses how this knowledge is used to help solve a particular problem in a configuration design of aircraft. Chapter 5 addresses the implementation of various engineering modules and how these modules are integrated into the system. An example of how the system can be used in an aircraft design project is given in Chapter 6. Detailed explanation of the content of these chapters is shown in the relevant appendices.

CHAPTER 2
A KNOWLEDGE-BASED AIRCRAFT DESIGN SYSTEM

2.1 INTRODUCTION

Chapter 1 discussed in great detail the problems of the present aircraft design systems which do not reflect the reasoning process behind the computational routines. This makes the systems difficult to understand since only limited explanation can be extracted for design decisions made.

The advance of the computer technology has unleashed the analytical power of problem solving activity in design and engineering applications. The complexity of the problems that could be handled by computers has greatly increased. The numerical techniques used to achieve solutions require of engineers not only a sound background of mathematics, but also a good experience of computer programming.

The danger of this development is that engineers are increasingly involved in the mathematical methods of the programs, and less concerned with the underlying physics of the problem and the reasoning process behind the problem solving activity. Since the understanding of the reasoning is as important as the analytical procedures used to derive conclusions, the development of a system with the reasoning capability will be greatly welcome.

This leads to a study project to develop an intelligent system that combines the computational routines and the reasoning capability in aircraft design process. The system proposed in this thesis utilises an interactive design environment which makes use of some Artificial Intelligence (AI) and Knowledge-Based Systems (KBS) techniques for the symbolic computation required in dealing with reasoning. The system should reflect the process of design, not only in the set of design variables and tasks involved but also in the manner in which the design is carried out. This is important for novice designers to learn about the design activities.

2.2 A MODEL OF THE AIRCRAFT DESIGN PROCESS

For the purpose of specifying design tasks in the design process of a civil transport aircraft, a process model needs to be established. This involves effort in determining the participating sub-systems, the possible interactions among the sub-systems and the participating disciplines [19]. These sub-system interactions may be reflected not only in the common parameters shared among the sub-systems but also in the sequence in which they are solved. The sub-systems need to be ordered into a hierarchical structure to form a multi-level problem solving approach to the configuration design process [3].

To use this approach, the complex aircraft design process needs to be decomposed into loosely coupled sub-systems in order to identify its hierarchical structure. The design process is then described as a set of sub-systems, where the completion of the problem solving activity in one sub-system may be dependent on the execution of the others. Each sub-system, referred to as *design modules*, can be decomposed further into a number of inter-related tasks to form a hierarchical graph of problem solving activities in that module.

Fig 2.1 depicts a typical design procedure of civil transport aircraft. This diagram shows the major design modules that need to be included into the model, some of which represent the design process of major aircraft components. The flow diagram of the activities in the engineering analysis is shown in Fig. 2.2.

Currently, 13 modules are identified, and these are :

- Parametric Analysis
- Configuration Analysis
- Fuselage Design
- Wing Design
- Engine Selection
- Weight and Balance
- Aerodynamic Analysis
- Flight Analysis
- Landing Gear Design
- Performance Analysis
- Stability and Control Analysis
- Cost Estimation
- Design Optimisation

The modules that lie outside the box described by the dashed line in Fig. 2.2 are currently not included in the system. The system, however, allows a possibility for incorporating them.

Apart from these 13 modules, two extra modules are required to support the design activities at various levels. These modules are multi-purpose in that they can be utilised for any other module and lie separately from the rest. The two modules are :

- Database Management Module
- Graphic Module

The database management module contains the compilation of data of existing aircraft. The data are structured in a way that enables easy retrieval of data. The objective of this database is to supply data for comparison or consultation if the appropriate value is unknown for the aircraft being designed. The database should also include engine-related data. This should cover the geometry, weight and performance chart of each engine. Ideally the whole spectrum of aircraft engines need to be included to enable a

proper assessment of engines before selecting the best available one.

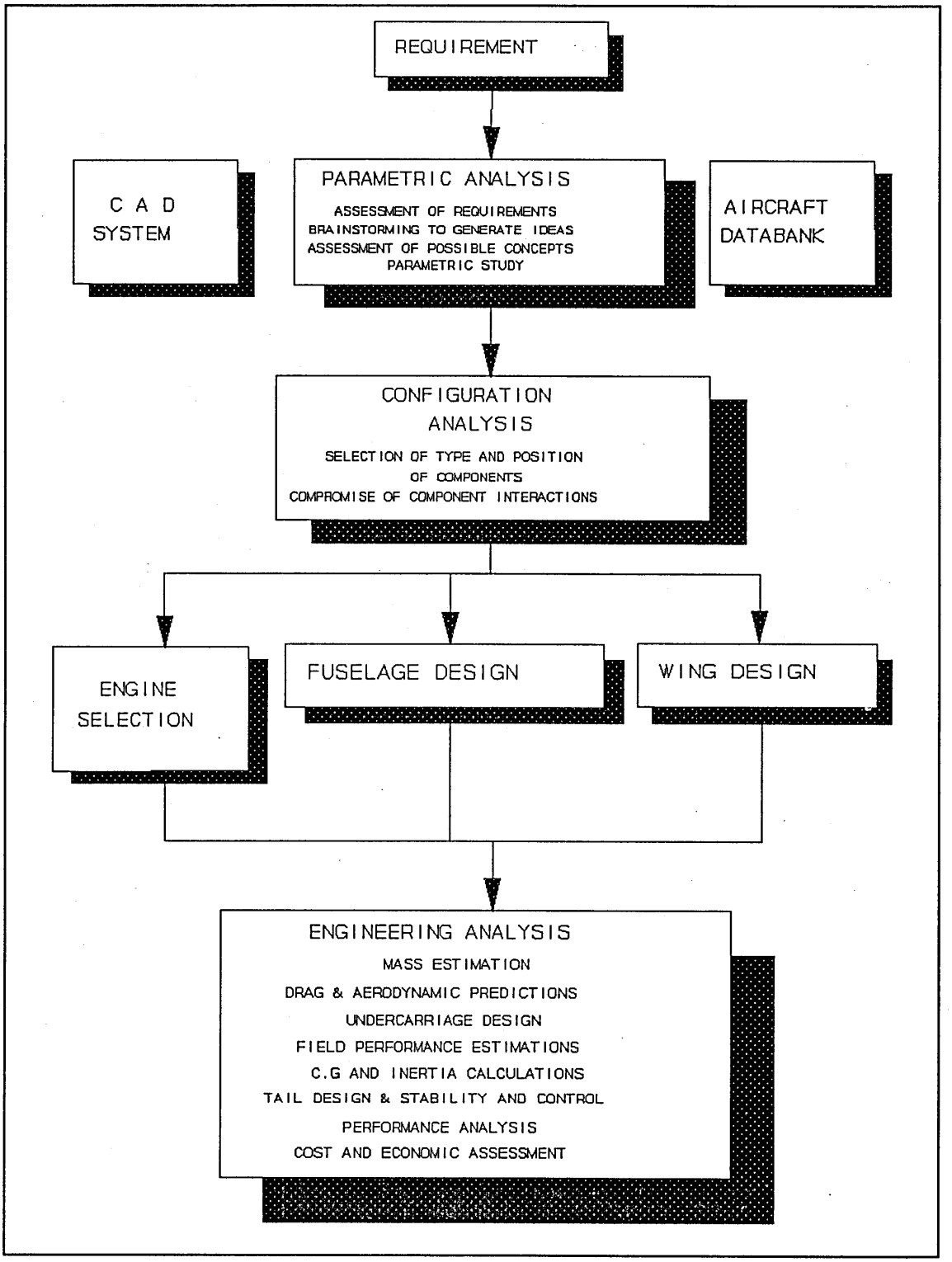


Fig. 2.1 A typical design procedure of transport aircraft

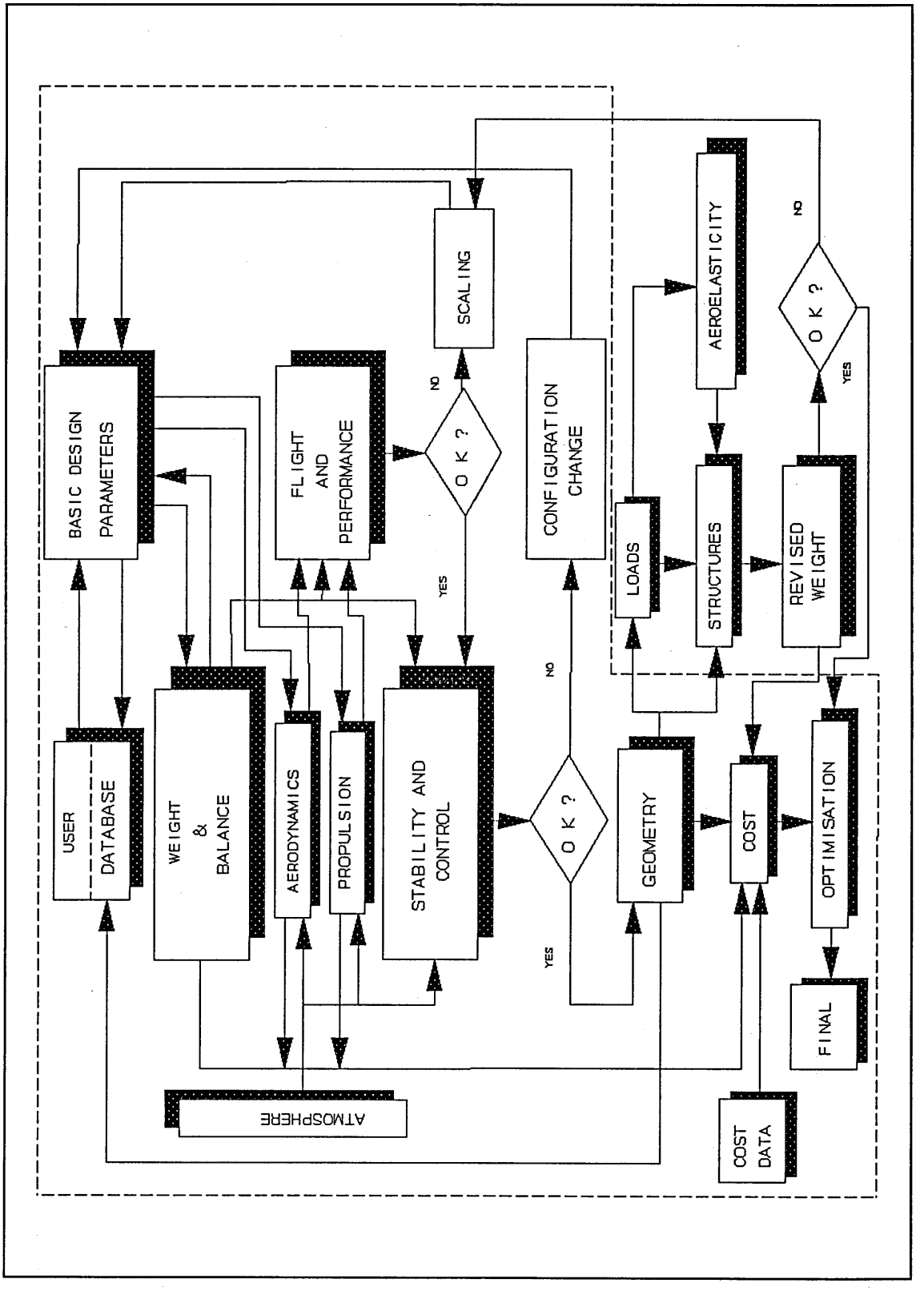


Fig. 2.2 The structure of aircraft design modules

The graphic module would enhance the presentation of the input/output of the all the application modules. This will enable the presentation of the geometry of the aircraft in the form of drawings and the performance characteristics in the form of graphs. Ideally this module should utilise an established Computer Aided Drawing (CAD) system which provides a wide range of graphical representation capabilities. This however would require an extensive interface to link the CAD system with the application modules.

2.2.1 PARAMETRIC ANALYSIS

The parametric analysis module concerns variations of aircraft parameters to investigate their effects on the design. This parametric analysis is more than just a parametric variation for performance study. The objectives of this module are :

- To identify the specific design requirements that are critical to the design. The module will enable the designer to assess the requirements which are too severe or too ambitious for the present state of technology,
- To build a compilation of data for the specific aircraft being designed, and
- To obtain a combination of parameters which yields the best design compromise.

A complete evaluation of the effect of the design parameters on the complete aircraft is virtually equivalent to carrying as many design studies as there are parametric variations. At this stage of design, only a simplified analysis can be performed and many details may therefore remain undiscussed.

Despite this, parametric analysis allows the designers to identify the most critical aspects of design and to shape the design around those aspects. This early analysis will ultimately result in an acceptable initial design of which the major characteristics are tentatively defined. This will provide an adequate background on which the designers can base their decisions.

2.2.2 CONFIGURATION ANALYSIS

The selection of the general layout of a new aircraft should be based on a proper analysis of the design specifications. It should discuss the operational requirements of the aircraft and translates them into a suitable positioning of the major components in relation to each other. In other words, the aircraft configuration should be selected as such to meet its operational requirements.

The main objective of this module is to address the primary selection of various configurations which leads to the initial configuration of the aircraft. This requires an extensive synthetic exercise whose arguments and background information need to be addressed in detail. This exercise will construct the design choices which will govern the analytical and synthetic effort in the following modules. They serve as a basis for

more detailed design efforts.

One of the problems encountered in this module is that no clear-cut design procedure can be followed. The heuristic nature of problem solving activities involved makes it a suitable design problem area in which the Knowledge-Based System techniques can be applied. The implementation of the Knowledge-Based System techniques, therefore should be directed towards solving problems in this module. The knowledge base will capture the design knowledge applicable for the selection of the configuration layout of an aircraft. This design knowledge is explained in Chapter 3. The problem solving strategy involved is described in Chapter 4.

2.2.3 FUSELAGE DESIGN

The fuselage design module addresses not only the determination of its external shape and geometry but also the internal layout of the sub-systems it houses. For civil transport aircraft, the fuselage has an important place in its revenue-earning capability. Traditionally, the fuselage becomes the only part of the aircraft where the fare paying passengers and cargo can be accommodated. It also becomes the only suitable place where the flight crew can be stationed. Avionics and various aircraft systems are also placed inside the fuselage.

From the structural point of view, the fuselage is the main structural component on which other components are mounted. It becomes the load carrying structure, through which the loads from other parts, i.e. wing, tail surfaces and landing gear, are transferred. For this reason, the position and geometry of other components in many cases are influenced by the shape and geometry of the fuselage. Because of this, the fuselage design module is placed high up in the design hierarchy.

2.2.4 WING DESIGN

The wing design module addresses the process of determining the wing geometry. The wing provides the lift-generating capability of an aircraft and therefore as far as the flying performance is concerned it is the heart of an aircraft. The wing characteristics affect the cruise speed, take-off and landing performances, stall speed, handling quality and the overall aerodynamic efficiency during all phases of flight.

The module should discuss the selection of the airfoil type and the choice of wing planform shape to obtain the optimum lift distribution at various flight conditions. Some structural aspects, fuel tank volume and the overall aerodynamics need to be considered.

2.2.5 ENGINE SELECTION

The engine selection module investigates the selection of a suitable engine type for an

aircraft to perform its mission. This addresses the consideration of the flight envelopes of the aircraft, fuel consumption, empty weight of the engine, its first cost, noise, vibration level, reliability and maintenance aspects.

It also performs an assessment of the choice of number of engines in great detail as the selection of engine is frequently dictated by the availability of the right size of engine itself. During this selection, the engine out performance and control need to be taken into consideration.

2.2.6 WEIGHT AND BALANCE

The estimation of the weight of a conceptual aircraft is a critical part of the design process. The choice of aircraft layout, geometry and detailed configuration affects weight. Weight prediction is necessary not only to make an assessment of design qualities, but also to set a goal for the structural and system departments.

The initial weight estimation must be able to reflect the sensitivities due to changes in layout and geometry in order to provide accurate results. This is necessary if the aircraft configuration needs to be optimised for minimum weight without extra development costs.

The module should provide weight estimation methods for different levels with a reasonable accuracy. Early in the design it also desirable to do a rough centre of gravity estimate. This is probably based on a crude statistical approach. Positioning of the wing and tail surfaces need to be considered to get an acceptable range of centre of gravity location during the flight and for different passenger and cargo layouts.

2.2.7 AERODYNAMIC ANALYSIS

This module will enable the designer to perform aerodynamic analysis of the aircraft. The layout and the geometry of the aircraft have been relatively fixed and defined to a great detail to allow an accurate analysis to be done. Accurate estimation of the aircraft aerodynamic characteristics is crucial to ensure that the specified performance requirements are met by the aircraft in production. The lift and drag characteristics are critical for cruising flight. A small discrepancy between the estimate and the actual drag coefficient of the aircraft may result in a big difference in the range performance.

2.2.8 FLIGHT ANALYSIS

In this module the designers are not only able to estimate the total fuel weight for the design mission but also to investigate the variations of mission that the aircraft could fly. Variations of trip distances can be analyzed and the derivation of the payload range diagram can be accomplished.

2.2.9 LANDING GEAR DESIGN

This module addresses the installation and geometry definition of the landing gear and its sub-systems. The design of landing gear frequently poses a problem to designers due to its limited use in the actual operation of the aircraft. Its main use is only during take off and landing, however it contributes to about 3.5% of the total weight and 2% of drag penalty during the flight phases of which it is not useful. It must be placed in the correct down position for take-off and landing, ensuring that it gives stability during roll and ground manoeuvre. It must be strong enough to absorb the impact load during landing touch down, but small and light to reduce performance penalty during other flight operations in which the landing gear is not being used. The retraction of landing gear into the aircraft, either into fuselage or wing, should be made with minimum modification in structure and the external surface of the aircraft.

2.2.10 PERFORMANCE ANALYSIS

In this design module the designers estimate various performance characteristics of the aircraft. This allows them to perform analysis on the climb flight to estimate the rate of climb, service ceiling and climb gradient during take-off climb. It also addresses the prediction of low speed and field performances of the aircraft for the intended missions. In many cases it is necessary to estimate the field performance for different airfield elevations and temperatures to make sure that the aircraft is capable to fly the mission between two points of different conditions.

2.2.11 STABILITY AND CONTROL ANALYSIS

An understanding of the important stability and control design parameters is necessary in order to make sure that the aircraft is stable during all phases of flight. If the stability of the aircraft is disturbed, the aircraft is expected to be able to generate forces that can return the aircraft to a stable flight state.

In this module, the geometry and layout of the tail and control surfaces are considered. The size of the surfaces should allow an effective generation of the control forces. This design modules should allow the analysis of equations of motion in longitudinal and lateral modes to be performed. It should also discuss control in case of engine failure and in low speed flight.

2.2.12 AIRCRAFT COST ANALYSIS

This module provides an accurate cost estimate which is probably the ultimate goal of aircraft design activity. It covers the estimation of aircraft development and production costs which, with an allowance for profit and bank charges, determine the purchase cost of the aircraft. This activity also involves the prediction of the operation and

maintenance costs and the potential economic of the aircraft during its service. These frequently dictate the financial success of the aircraft project.

Aircraft cost is probably the best criterion, other than aircraft performance, to select the best design. While there will be some differences in technical credibility, data substantiation and intrinsic design qualities, the final selection will often hinge on cost.

2.2.13 DESIGN OPTIMISATION

The design optimisation module should enable the designer to select the best design after a more rigorous analysis. The module should provide ways of varying the specified design variables in order to reach an optimum design based on a given criteria. In the light of a detailed analysis, a constrained optimum can be achieved, taking into account all the design and operational constraints pertinent to the specific design. These constraints may include geometric constraints and performance constraints. The geometric constraints limit the particular dimensions of the aircraft to avoid infeasible geometry which may be imposed by the technological capability. The performance constraints are imposed by the performance requirements of the aircraft in order to ensure that the aircraft can operate safely. These constraints cover the mission requirements and handling quality design criteria.

The module should enable the designer to perform multi-variable optimisation. The optimisation technique employed should be able to provide ways of changing the variables in order to obtain an efficient search for the optimum design point. The analytical methods should ideally cater for different configurations of aircraft to identify the sensitivity of design to the configuration selected.

The design process in itself is a design optimisation activity which addresses a more global aspect of design. This module is however required to quickly consider the aspects of design with variations of several design parameters. The position of this module in the design activities is therefore very much dependent upon the level of the analysis. The optimisation program can therefore be placed after the parametric analysis when several important parameters of the aircraft have been tentatively defined.

2.3 IMPLEMENTATION OF THE KNOWLEDGE BASED AIRCRAFT DESIGN SYSTEM

2.3.1 SELECTION OF THE COMPUTER LANGUAGE

The Aircraft Configuration Design System is implemented in C/C++ computer language. This language was selected based on several factors :

- Expressive power for knowledge based application
- Ability to handle complex mathematical problems
- Extensibility in the long run.

The expressive power means that the language should be able to represent knowledge in a structure that is easy to understand. It should be easy to structure the knowledge in the form that is suitable for inferencing. The language should ideally have some built-in capability in handling several aspects of knowledge based programming, i.e assertion of database, pattern matching and unification, tracing mechanisms and list processing. Several languages, such as Prolog [20], Smalltalk [21] and LISP, do have these aspects which make them suitable for AI and KBS applications. In the early stages of development of this system, Turbo-Prolog [22] and Smalltalk V [23] had been used to experiment with some representational and inferencing ideas. Using these languages, however, led to several difficulties in interfacing with engineering routines as reported in section B.3 of Appendix B.

Since aircraft design involves an extensive engineering analysis, it is required of the language to be able to provide means of solving complex mathematical problems. The selected language needs to have extensive mathematical functions and a structure that enables the programmers to easily develop procedures for this application. In this area Fortran and C are strong candidates since they are more procedural and structured. Fortran in particular is supported with a large library of functions which are easy to use and has proved capable of solving difficult calculations. This is reflected in the fact that Fortran has been widely used for engineering applications.

Extensibility is a virtue because of the nature of the problems tackled in aircraft design process. The size of the problem is enormous, and to deal with this task need a powerful language that could handle large programs. In this area C/C++ has an advantage, and it is increasingly popular for many applications.

Ideally it is easier to build the prototype using Knowledge-Based System tools which are commercially available. These tools have everything required for Knowledge Based applications. They have built-in procedures for building knowledge bases and are equipped with some general problem solving procedures suitable for inference engines. This makes prototyping easier and faster. Some of them are designed with a version of procedural languages that can be used for extensive numerical computations. The representation models, however, tend to be restricted to those supported in the tools.

Several virtues of the C/C++ language that contribute to the selection are :

- C/C++ is a hybrid language, that can be used in both procedural and object-oriented programming [24]. The object oriented concept is a powerful quality in C++ that enables the programmers to structure the program using user defined classifications of objects [25]. This class structure can be used to define new data types, other than the traditional data types used in many languages, i.e integers, floating point numbers, characters and strings, and logical type. The beauty of

this way of programming is that the new data types can be reused repeatedly in different parts of the program just like the traditional data types. The operations on the newly defined data types can be specified accordingly to suit applications. This will ensure a consistent application of the new classes of objects.

- C/C++ is a middle level language that retains the functionality of the assembler while providing most of the capabilities of the high level language such as Fortran and Prolog. The low level functionality of C/C++ offers the programmers the access to machine level, to operating systems, with considerable ease. Calling external routines and executing operating system commands can be performed without too much difficulty. This will be a powerful facility in areas such as interfacing.

- C/C++ retains almost the same format across many machines which makes the program easier to transport from one machine to another.

2.3.2 SYSTEM ARCHITECTURE

The system architecture for the proposed aircraft design system is shown in Fig. 2.3. There are three main building blocks in the system, they are :

- The System Controller
- The Knowledge-Based Module
- The Engineering Modules

The system controller acts as the interface between the user and the system. It accepts the input from the user and translates it into commands understood by the system and then executes the commands. It displays results requested by the user in a specified format, either in text or graphs. The system controller also becomes the interface between the knowledge-based part of the system and the engineering modules. It has its own planning strategy in moving along the design process from one module to the other. It decides which module to execute by consulting the status recorder. The details of the system controller are described in Appendix E.

This system architecture, in which the knowledge-based module and the engineering modules are kept separate, is chosen to enhance flexibility and extensibility as explained in Chapter 1 and to enable new engineering modules developed in the future to be easily incorporated.

The most fundamental task is to interact with these engineering modules. An interface to these engineering modules is required to enable easy transfer of data. This is done by having a common database for the whole system. Any data required for the execution of a specific module will be retrieved from this database. In the other way around, new data obtained as output of the module can be added into the database. This approach is applied in order to make sure that all modules will work on the same value

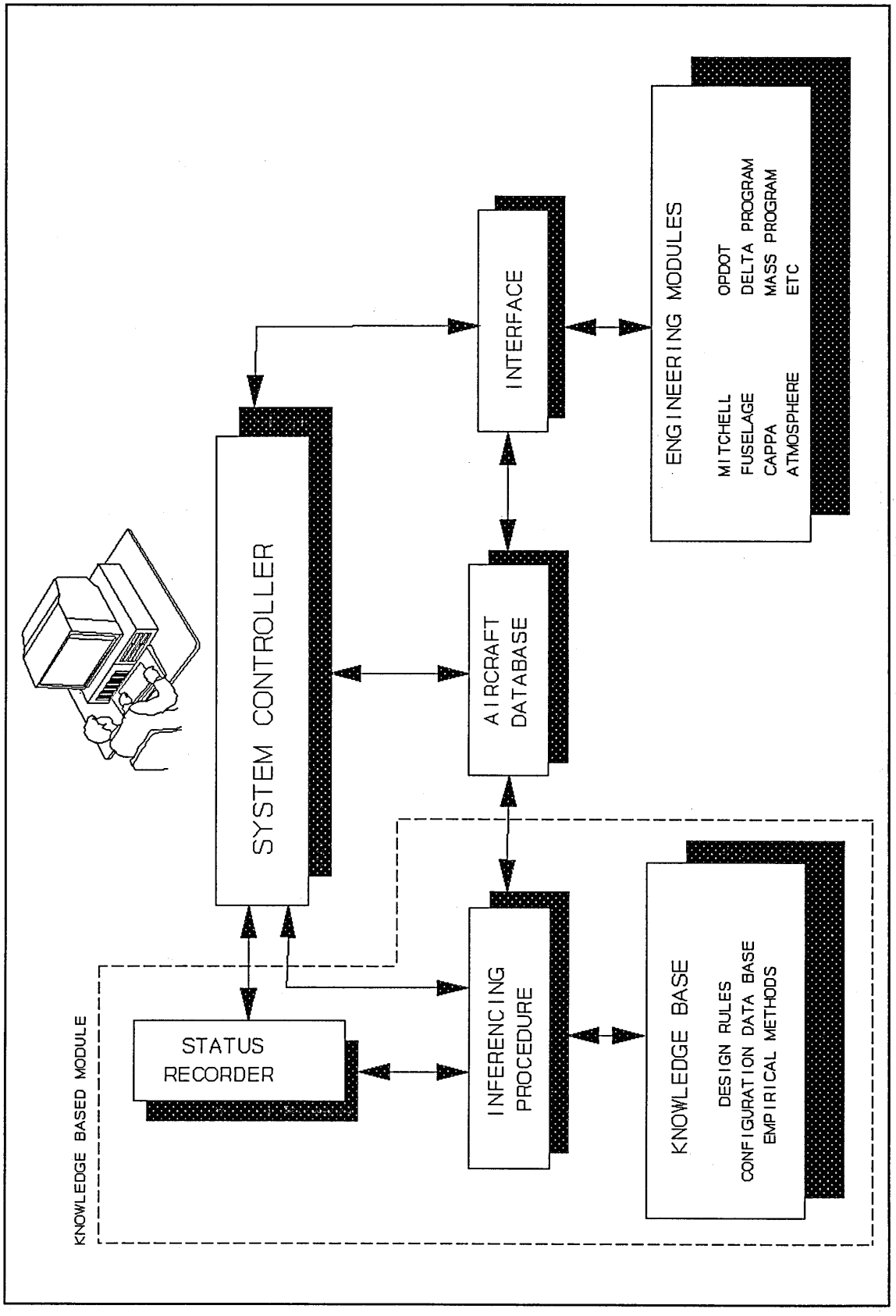


Fig. 2.3 System architecture

of data. Details of how these engineering modules are integrated in the system are explained in Chapter 5.

As explained earlier, the knowledge-based module mainly deals with the configuration analysis, while the engineering modules cater for the rest. The knowledge-based module consists of three parts :

- Inferencing Procedure, commonly called the Inference Engine,
- Knowledge Base,
- Status Recorder.

The Inference Engine performs the following functions :

- To decide which task or design step needs to be executed at any one time
- To search for the appropriate problem solving strategy for the task in question
- To solve interrelationship problems occurring during the execution as a consequence of the decision taken in the previous tasks. This will cross-examine the consistency of the decisions with previous solutions.
- To justify the decision or advice given by the system.

The details of these inferencing procedures are discussed in Chapter 4.

The knowledge base contains records of all information about the configuration design process. It consists of the following :

- Record of all the tasks involved in the process and the priorities among the tasks, i.e the sequence in which tasks are executed.
- The main part of the knowledge base is the set of rules containing the possible solutions for each task, the conditions that need to be satisfied for the solutions and the consequences of each solution/decision. The rules may also hold other rules as the conditions.
- Factual knowledge of aircraft design.

The knowledge base is described in the following chapter.

The status recorder serves as the working space of the inference procedure. It records the potential actions awaiting execution which correspond to the knowledge base rules that seem relevant to decisions taken previously. It records all the decisions given by the inference procedure and the consequences of the decisions for future examination.

CHAPTER 3
KNOWLEDGE BASE OF AIRCRAFT DESIGN

3.1 SOME ASPECTS OF DESIGN KNOWLEDGE

In order to solve design problems, a designer requires a large body of knowledge. The knowledge comes in many different forms. The most common form is the theoretical background of the disciplines that support the analyses. This may be in the form of engineering principles and mathematical equations. Designers have all of this type of knowledge at their disposal and must use them accordingly to construct solutions. These principles and equations can be ordered into sequences to form procedures that can be used as tools to do the job.

In aircraft design application, this knowledge may be applied in different ways. Several examples are given in the following :

- Geometric Constraints :
  It may be used in the form of physical relationships between different components. There are many geometric constraints that define the dimension and shape of certain components and can be translated into mathematical equations. One example is the aspect ratio of a wing which, by definition, is the ratio between the wing span to the power of two and the wing planform area :

$$AR = b^2 / S \qquad 3.1$$

  This simple equation obviously restricts the magnitude of all the three participating variables. A designer would know that if two of the variables are known the other will be defined. This equation is used frequently in design areas in which the wing span or aspect ratio is involved.

  Another form of this geometric constraint is related to spacial relationships. One famous principle is that the outside has to be bigger than the inside. For example, the fuselage volume needs to be larger than the space occupied by the passengers, seats and cargo accommodated within the fuselage. The wing area and thickness ratio selected should be able to accommodate fuel and fuel system, stowed landing gear if it is mounted on wing, and flap and aileron control system.

  Another principle concerns the layout of components, which governs the dimensions and distances between components. For example, propeller engines should be placed at a distance of more than half of the propeller diameter from both fuselage and the ground to avoid any contact.

- Scientific Evidence
  Another form of knowledge is scientific evidence. The wing contributes to most of

the lift of an aircraft. An aircraft needs tail or control surfaces to provide stability and control.

This scientific knowledge also comes as physical phenomena which affect the configuration of the aircraft. This form of knowledge can be translated into causal relationships in the form of action - consequence relationships. A wing with finite span suffers from tip vortices that reduces lift and effective angle of attack at the tip area. This results in an increase in drag. Thus a wing with a high aspect ratio will have a less loss of lift and increase in drag as compared to a low aspect ratio wing of equal area. At high angle of attack the air flow separates from the wing which results in a sudden loss of lift. This also leaves a turbulent wake behind the wing which, if it impinges on the tail surfaces, will reduces the effectiveness of the tail in providing stability and control. This will affect the position of the tail surfaces relative to the wing.

What distinguishes good designers from the rest is not only the amount of knowledge they have about the subject, but also how much they know about using the knowledge in solving problems. This concerns skill. In other words, they need expertise knowledge about the subject. The scientific and engineering principles are basically the same and accessible to many people, but expertise knowledge is specific to the designer.

This expertise knowledge is acquired through experience and can be classified into several types :

- Commonsense :
  This may appear as assumptions which subject to certain conditions :
  ASSUME wing position is low wing

  The assumptions may also comes as tentative decisions due to incomplete or partial information :
  IF aircraft type is commercial transport
  ASSUME engine type is turbofan

- Metaknowledge
  Metaknowledge refers to knowledge about reasoning. It deals with how to apply knowledge in a sequence that can produce solutions. This may involve setting priorities for goals if there are many conflicting goals.

  Metaknowledge can be used to explain the reasoning process.

3.2 CONFIGURATION DESIGN KNOWLEDGE

The activity in the configuration analysis is aimed towards making decisions for the general arrangement of the aircraft. These decisions involve the selection of a particular type or layout for each major component of the aircraft. Several decisions are identified

in this process:

- Fuselage Layout
- Wing Position
- Propulsion Type
- Engine Type
- Engine Position
- Tail Type
- Tail Layout
- Landing Gear Type
- Landing Gear Layout
- Landing Gear Position
- Flap Type

Each of these decisions represents a design task in the configuration analysis. The interaction between various tasks is shown in Fig. 3.1 which depicts the hierarchical structure of the tasks. The following section explains the activity involved in each task and the list of choices to be made.

3.2.1 FUSELAGE LAYOUT

Fuselage layout addresses the selection of the fuselage internal and external arrangement for commercial transport aircraft. The traditional choice is whether to arrange the passenger seats with one or two aisles. Recently having a double deck passenger cabin becomes a popular option for large aircraft. This layout was originated from the Boeing 747 which remains the only choice for long and crowded routes. For a number of years, many studies have been made to investigate the possibility of a twin fuselage [26], in which two identical fuselages are arranged side by side and joined by the wing and or tail surfaces. The primary deciding factor in selecting any layout is obviously the number of passengers to be accommodated. When the number of passengers allows several possibilities, other factors may come into play. These include the operational, aerodynamic efficiency and structural integrity of the aircraft.

Single aisle fuselage

A single aisle fuselage is the only possible choice for an aircraft with less than 80 passengers in a single class. For a number of passengers of up to 250, this layout is still possible if the length of the fuselage does not interfere with the operational quality of the aircraft, especially in take-off and landing.

Double aisle fuselage

A double aisle fuselage becomes an obvious choice for aircraft with more than 250. passengers. With passengers as little a number as 180, this layout is still competitive if the aerodynamic and handling quality requirement can be satisfied, since it will result in a squat body with little room for shifting the tail surfaces longitudinally for stability. The prospect of extending the fuselage for future development may give an extra advantage for this layout.

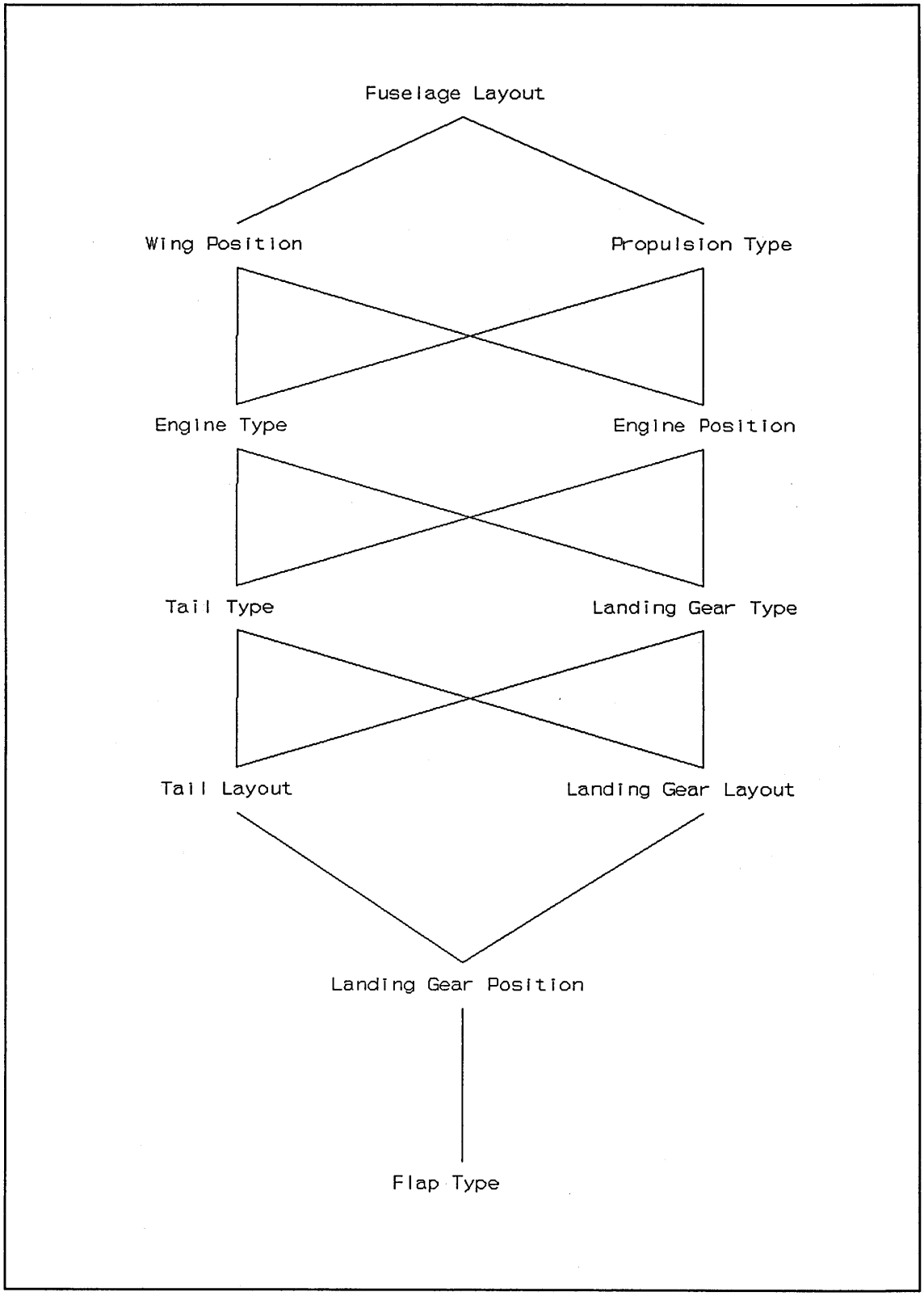


Fig. 3.1 A graph of configuration design steps

Double deck fuselage

Double deck is a reasonable choice only for aircraft with double aisle fuselage. The diameter of fuselage is large enough to allow room for a single or twin aisle cabin on top or below the main deck without any major structural complexity. For aircraft with passengers less than 450, the decision to opt for this layout is strongly influenced by the market, since having an extra cabin deck may result in reduced capacity for cargo. The second deck normally shares the space allocated for cargo containers.

Twin fuselage

The twin fuselage may be competitive if the aerodynamics can be used to maximum benefit. The two fuselages may give to the centre wing the end-plate effect which increases the span efficiency, but the interference on the joint may reduce this benefit. Furthermore, the total wetted area of the fuselage may be more than that of a single fuselage. The main advantage of having two fuselages is that the wing bending relief. The wing bending moment at the wing-fuselage joint can be greatly reduced since the span of the outboard wing can be smaller even though the actual span is much larger. However, the position of the flight deck may pose a problem to the designer.

3.2.2 WING POSITION

Wing vertical location with respect to fuselage is generally set by the real-world environment in which the aircraft will operate. There are three possible positions for the wing : high, low and mid wing.

High wing

In this configuration the wing is placed high on fuselage, either on top of the fuselage or with the upper surface of the fuselage cut to allow the wing box structure passing through. There are several considerations that lead to this configuration :

- For military transport aircraft, the high wing will allow loading and unloading of cargo without special ground handling equipment.
- For low speed aircraft, in which external strut is normally used to reduce the wing weight. Since the strut joins the wing on the under surface, the air flow on the upper surface which provides most of the lift can be free from disturbance.
- For aircraft designed with short take off and landing (STOL) capability, a high wing will allows room for a very large flaps required for a high lift coefficient.
- If the aircraft is designed to operate on semi or unprepared airfield, a high wing will protect the engines and propellers from flying debris or rock, assuming that the engines are mounted on the wing.
- For large aircraft with the propeller engines mounted on wing, this configuration becomes a popular choice due to its inherent advantage of having large ground clearance. The propeller diameter is normally large and the high position of the wing will give enough clearance.

Low Wing

There are several considerations that lead to this layout :

- For high-speed commercial transport aircraft, in which the loading and unloading of payload is not a problem, since they normally operate out of established airports with proper equipment.
- For large transport aircraft, the large diameter of fuselage will give enough space for the wing to pass through the fuselage structure without obstructing passenger cabin.
- Low wing configuration gives a good ditching and wheel-up landing capability.

Mid Wing

Mid wing configuration offers the maximum aerodynamic benefit. Several situations that lead to this layout are :

- If the aircraft is designed for high speed performance in which the minimum drag in high speed flight if of paramount importance. This is only possible provided that the fuselage is largely cylindrical and the diameter is large enough to allow the wing to pass through without disturbing the passenger cabin.
- For small transport aircraft, for example business aircraft, this layout is possible if the volume for useful load is small in relation to the fuselage internal volume.

3.2.3 ENGINE CONFIGURATION

The engine configuration addresses the selection of engine type and installation. The size of the engine is assumed approximately the same as the total thrust or power obtained from the parametric analysis, given the number of engines assumed in that process.

In general, there are two means of propulsion of commercial aircraft. These form the two classifications of aircraft : propeller driven and jet propelled aircraft. The selection of the engine type and its position depend on the type of propulsion used. In the case of a propeller driven aircraft, for example, the design of the propeller becomes an influential factor. This normally restricts the choices of position of the engines.

There are three decisions need to be made :

- Propulsion Type
- Engine Type
- Engine Position.

In general, the following factors need to be taken into consideration :

- Operating flight envelope which includes the operating speed, cruise altitude and the engine rating
- Fuel consumption
- Engine weight and dimension
- First and maintenance costs
- Noise and vibration level.

3.2.3.1 Propulsion Type

The selection of the propulsion type depends strongly on the maximum operational speed of the aircraft.

- Propeller driven aircraft operates at maximum efficiency at Mach numbers between 0.2 and 0.55, but is still quite efficient up to Mach 0.65, while jet engines are good at speeds higher than 0.7.
- At the speed range where propeller and jet engines are possible, the fuel consumption, installation, weight and cost may be taken into consideration.
- For commercial aircraft, there is a strong tendency to opt for jet engines at speed higher then 0.65 due to better passenger appeal, lower noise and vibration level.

Propeller Engines

- This type of propulsion is the obvious choice if the maximum operation Mach number is less than 0.55.
- For longer range, propeller engines are still a good choice for Mach number of up to 0.65 due to its lower fuel consumption.
- For a higher Mach number up to 0.8, a possibility of using propfan engine with high speed advanced propellers may be considered to opt for this type. Initial cost of development, fuel price, noise and vibration level associated with the high speed rotation of the propeller need to be kept minimum to have a chance for this type.

Jet Engines

- In general this type of propulsion is good for Mach number greater than 0.65.
- For small aircraft, jet engines are good even at low Mach number, around 0.55, with the advantage of eliminating large propellers.
- For Mach numbers in the region of 0.8 up to 3.0, the jet engines is the only choice of propulsion type.

3.2.3.2 Engine Type

- The choice is between turboprops, turbofans, propfans and turbojets.
- If propeller engines are used, turboprops are good for Mach number less than 0.55.
- Otherwise, unless the initial cost, vibration and noise level can be minimised, turbofan engines are to be used up to Mach 0.95.
- For Mach number close to and beyond supersonic (M = 0.95-3.0), only turbojet can provide enough thrust.
- Otherwise, if fuel cost is high, propfans are used.

3.2.3.3 Engine Position

The selection of engine position may pose the most difficult problem in configuration design due to conflicting requirements. Two layouts are the most popular : wing and fuselage mounted engines. Recently, the possibility of tail mounted engines has been

investigated. There are many factors need to be considered :

- Empty weight
- Foreign object ingestion
- Maintenance
- Aircraft systems : fuel, anti-icing, air-conditioning system
- Noise level
- Performance
- Handling qualities

Wing Mounted Engines

Some of the considerations which may apply are :

- If there are more than three engines, assume the engines are mounted on the wing.
- Mounting the engines on the wing will require lighter supporting structure (pylons) due to the nature of the loading and reduce wing weight due to its mass relief effect on the bending moment on the inboard wing.
- If the engines are large, there is a strong tendency to place the engines on the wing for balance and to minimise weight.
- In general, wing mounted engines will slightly increase drag and reduce the drag-divergence Mach number due to interference effects of the pylons.
- For propeller engines there is a strong tendency to place the engines on the wing to provide enough fuselage clearance.

Fuselage Mounted Engines

Some of the considerations which may apply are :

- The fuselage mounted engines may result in large centre of gravity travel which may restrict the loading flexibility of the passengers/cargo.
- Mounting propeller engines on the fuselage will require long supporting pylons which in turn increase bending moment on the root of the pylons. This would require extra strengthening of the pylons and fuselage structure.

3.2.4 TAIL GEOMETRY AND ARRANGEMENT

The main function of the tail surfaces is to provide trim, control and stability of the aircraft both longitudinally and laterally. Trim refers to the generation of a lift force to balance other moment produced by the aircraft. If the aircraft is disturbed from its stable flight condition, the tail surfaces should be able to generate adequate aerodynamic forces to counter the perturbation and return the aircraft to its original state (pitch, yaw, roll, velocity, etc.) While it is possible to design a stable aircraft without the tail surfaces, such a design usually results in a penalty in some areas, e.g. a compromised wing airfoil shape, excessive wing area or sweep, and a narrow range of centre of gravity.

The tail surfaces must also be arranged to provide sufficient control power at all critical conditions. For longitudinal control, these conditions typically include nose-wheel lift-off, low speed flight with flaps down, stall and post stall recovery, and transonic

manoeuvring. For lateral control, critical conditions include engine-out flight at low speed, maximum roll rate and spin recovery. The control action is normally provided by the moving surfaces, e.g. elevators, all-moving stabilizers, and rudders.

For these reasons, the arrangement of the position and size of the tail surfaces depend strongly on the layout and geometry of other major structural components. Because the location of the tail surfaces, their effectiveness is influenced by the wing and the operation of the engines. The way in which the tail surfaces are mounted on the fuselage affects the structural arrangement of both the tail and the fuselage.

There are two important configuration decisions which need to be solved in relation to the geometry and arrangement of the tail surfaces :

- Tail Type
- Tail Layout.

Both decisions are related but for the purpose of simplifying the choices they are separated. Tail type generally concerns the arrangement of the tail surfaces with respect to the fuselage. Tail layout addresses the arrangement of the tail surfaces relative to each other.

3.2.4.1 Tail Type

There are two principal types of the tail configuration which are commonly used for subsonic aircraft :

- Single fin layout
  A single vertical tail with the stabilizer (horizontal tail) mounted either on the fuselage or on the vertical tail itself.

- Twin fin layout
  In this configuration, two vertical tails are mounted on the fuselage. This arrangement may include a single horizontal tail or both the vertical and horizontal tails are combined to form the so-called 'V' or butterfly tail arrangement.

Fig. 3.2 shows some of the possible arrangements.

Several conditions that lead to either configuration are :
- For commercial transport, assume single fin are normally used.
- If the twin fuselage layout is used, twin fin layout is the obvious choice.
- For most aircraft designs, the single fin layout usually provides adequate stability and control at the lightest weight.
- Twin fins may be considered if a single fin layout results in a large fin size. A considerable height of the fin may cause an undesirable rolling moment due to the deflection of the rudder as well as hangar and maintenance problems.

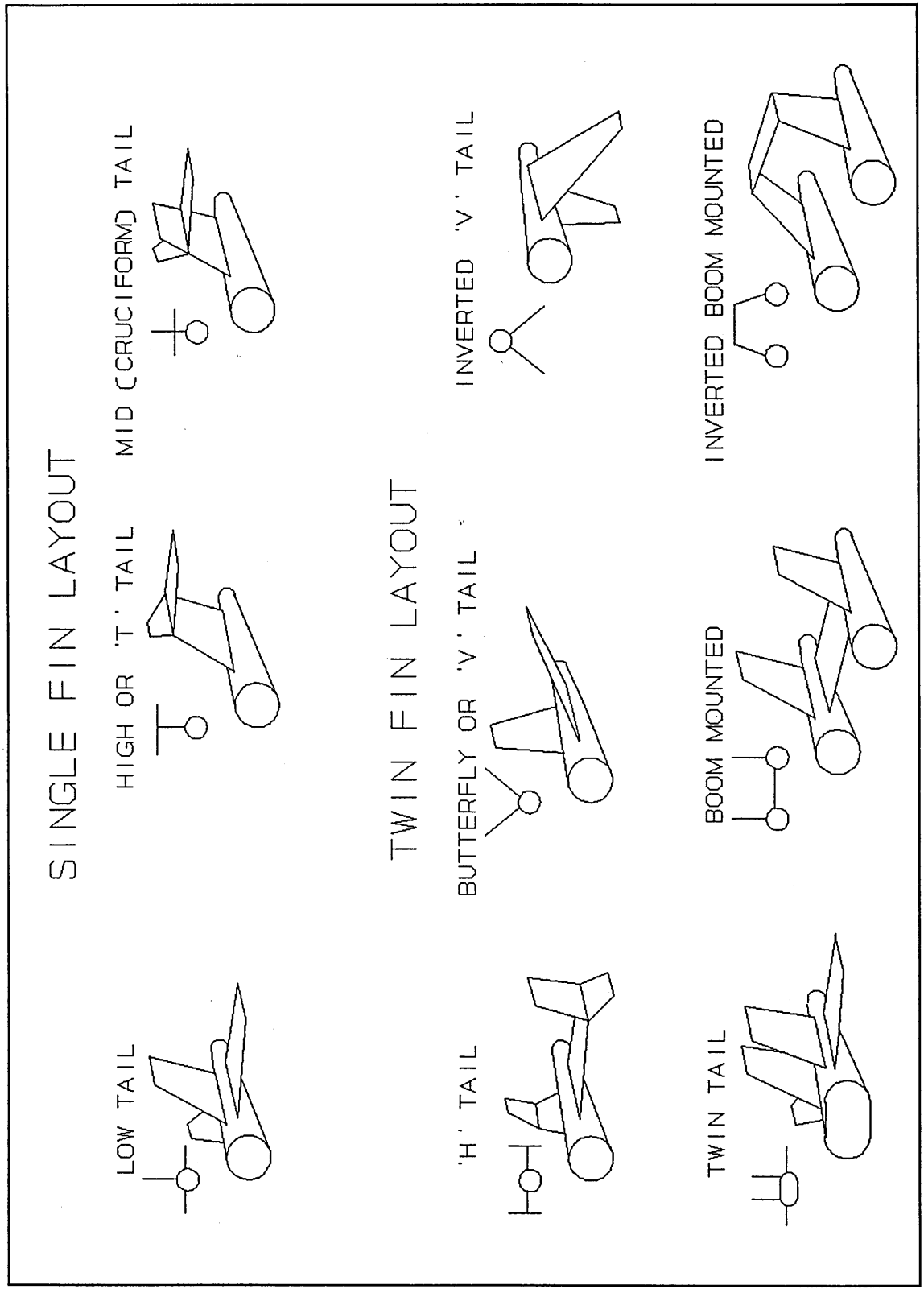


Fig. 3.2 Several configurations of the tail surfaces

- Twin fins may be considered for a multi-engined aircraft to position the vertical tail and rudders outboard for engine-out control.
- Twin fin configuration is normally good for spin recovery. In a single-fin layout, the separated flow, which comes off the horizontal tail during a spin, may impinge on the vertical tail. This may cause the vertical tail to be covered with turbulent flow. which makes the rudder ineffective.

3.2.4.2 Tail Layout

For the single fin layout, there are three positions of horizontal tail mounting on the vertical tail : low tail, high tail and mid (cruciform) tail. For the twin fin layout, the choices are more open, i.e 'H' tail, 'V' tail, inverted 'V' tail, twin tail, boom mounted and inverted boom mounted, as shown in Fig. 3.2.

Low Tail
For most aircraft designs, the low tail configuration, in which both the horizontal and the vertical tails are mounted on the fuselage roughly perpendicular one to another, is normally adequate for stability and control purposes. The associated empennage weight is smallest of all the configuration since both tails can be directly supported by the fuselage structure.

High or 'T' Tail
The high tail is also widely used. The horizontal tail is mounted on the tip of the vertical tail. This position requires extra strengthening of the vertical tail, which adds to the overall weight. However, it provides an end-plate effect to the vertical tail which increases its span efficiency, thereby reducing the size of the vertical tail. It also increases the tail arm if the fin is swept, thereby reducing the size of the horizontal tail. The main problem associated with this configuration is post stall conditions.

This configuration is normally chosen if the horizontal tail needs to be placed outside the wing wake and engine wash, which happens in the case of a high wing or fuselage-mounted engines.

Mid (Cruciform) Tail
The mid tail is a compromise between high and low tail, the horizontal tail is placed at around the mid span of the vertical tail. There are several reasons for selecting this layout :

- To lift the horizontal tail to avoid the proximity of a jet exhaust of the engines.
- To take the horizontal tail out of prime wing downwash region.
- To expose the lower part of the rudder to undisturbed air during high angle of attack conditions and spins.
- For a small aircraft, to lift the horizontal tail off the ground during take-off especially if a STOL capability is required.

'H' Tail

There are a number of reason to select for this layout :

- It is mainly used to position the vertical tail in undisturbed airflow during high angle of attack.
- To position the rudders in the propwash on a multi-engined aircraft in order to enhance engine-out control. The aircraft normally requires a large fin to counter the yawing moment due to the asymmetric thrust. The total area can be reduced with two fins with the advantage of the end-plate effects on the horizontal tail.

'V' Tail

The V tail configuration is intended to reduce the total wetted area of the tail surfaces. It offers a reduction in the interference drag but with some penalty in control actuation complexity. Since the moving surfaces has to serve both as rudders and elevators, the deflection of the moving surfaces may lead to an undesirable roll-yaw coupling.

Inverted 'V' Tail

The inverted V tail avoids the problem of adverse roll-yaw coupling. But it may cause a problem in providing sufficient ground clearance during lift-off.

Boom-mounted Tail

The boom-mounted tail layout can offer the same advantages claimed for the 'H' tail to a certain extent. The interference drag associated with the tail-boom junction may be reduced to minimum since the joint can be made almost perpendicular. The main problem is the mounting of the booms themselves. If the tail arm, i.e the longitudinal distance between the wing and the tail, is small they can be mounted on the wing. For a long tail arm, this obviously cannot be done since the associated drag and weight penalty will be too large. For a twin-fuselage layout, this problem can be neglected since the fuselage is already strong, thereby leaving out the necessity for extra supporting structure.

Inverted Boom-mounted Tail

The inverted boom mounted tail offers the same advantage as the boom-mounted layout. The horizontal tail may add extra stiffness to the two vertical tail.

3.2.5 LANDING GEAR CONFIGURATION

The arrangement of the landing gear often poses the most trouble for the internal layout of an aircraft with retractable landing gear. When the landing gear is down, it must be placed at the correct position for take-off and landing. Its retraction and stowage inside the airframe must be made without causing major structural discontinuities, obliterating the fuel tank or necessitating the requirement for excessive fairings that may disturb the contour of the external surface of the aircraft.

The general considerations in the arrangement of the landing gear are :

- The landing gear must be able to absorb the landing impact and taxying loads and to

transmit them to the airframe. The loads include :
- vertical loads mainly caused by non-zero touch down rates and taxying over rough surfaces,
- longitudinal loads primarily caused by braking, and rolling friction roll,
- lateral loads associated with cross-wind taxying and ground manoeuvring.

- It must provide adequate ground clearance for other parts of the aircraft during taxying, take-off and landing.
- No instability must occur during ground manoeuvring, maximum braking effort, cross-wind landing and high speed taxying.
- Retraction and the stowage of the landing gear.
- Maintenance cost.

Two major decisions are involved in this activity :
- Landing gear layout
- Landing gear position

3.2.5.1 Landing Gear Layout

Although there are many landing gear arrangements that have been used, most of them were for very special purposes, only three layouts are usually in practice today:
- Tail-wheel landing gear
- Nose-wheel tricycle
- Bicycle gear.

In general, these layouts have two main gears with differences in the arrangement of the third support.

Tail-wheel Landing Gear

The two main gears are located in front of the centre of gravity of the aircraft with the third support, the tail wheel, positioned further aft of the centre of gravity. There are a few reasons for selecting this layout :
- It is usually the lightest and simplest to design.
- During braking the aircraft tends to pitch nose down, increasing the vertical load on the main gear, thereby reducing the possibility of skidding.

Despite these, the tail wheel type has become obsolete due to several drawbacks :
- Hard braking tend to cause overturning. The drag forces due to braking act in front of the centre of gravity, and hence are destabilising in yaw. This increases the tendency of the aircraft to swing.
- In a two-point landing the resulting pitching moment is nose up. This tends to increase the incidence and hence the lift, which may cause the aircraft to bounce.
- Inconvenient loading and unloading of passengers and cargo due to the inclined position of the fuselage.

Nose-wheel Tricycle

There are several reasons for selecting this layout, mainly the opposite of the tail-wheel layout :

- The nose wheel acts as a prop to prevent overturning during braking.
- Easy loading of passengers and cargo since the fuselage is roughly level when the aircraft is on the ground.
- In a two-point landing the nose down pitch helps shed lift and prevent bouncing, although lift dumping may still be required.
- The initial take-off attitude has a low drag.
- Good visibility for pilot.

Bicycle Landing Gear

The two main wheels are arranged in the aircraft's plane of symmetry with the front and rear wheels share the landing load almost equally. Auxiliary outriggers are required to provide lateral stability. The use of the bicycle landing gear can be justified for several reasons :

- The main wheels are located nearly equidistant fore and aft of the centre of gravity, thereby leaving considerable space close to centre of gravity clear of obstruction. This can be very useful for payload.
- The wheels can be stowed in the fuselage without interrupting the wing structure.

The main problems with this layout are :

- The design of outriggers may pose structural problems. Extra care is required to keep the size and weight to minimum.
- The aircraft landing attitude must be carefully controlled.
- Large elevator power is required to lift the nose up during take-off.

3.2.5.2 Landing Gear Position

This decision concerns the mounting and the stowage of the landing gear on the airframe. Two configurations are possible :

- Wing mounted landing gear
- Fuselage mounted landing gear

This decision is very much dependent on the lateral position of the main gear which is set by the requirement of good stability and control during take-off rotation, landing touch-down and ground manoeuvring. This position is normally decided after the elevation of the aircraft above the ground has been set for take-off and landing. The landing gear legs should be sufficiently long to allow any combination of pitch angle and roll angle which may occur in operational use without the risk of any other parts of the aircraft touching the ground. In the case of the retractable landing gear, the lateral disposition is further complicated by the requirement for stowage.

Some of the considerations applied in the selection are :

- For aircraft with cruising speed of more than 150 kts., the landing gear should be

retractable to reduce drag.

- In most cases, the wing mounted landing gear is assumed to give sufficient track for stability. For slower aircraft a fuselage mounted landing gear is possible only if the lateral disposition for stability is sufficient without necessitating the use of very long landing gear legs.
- Virtually all low-wing, civilian jet transports retract the main gear into the wing-fuselage junction.
- If the main gears are so far separated that this stowage is impossible, a wing stowage should be attempted only if the size of the landing gear is small enough to be fitted nicely into the wing box. Otherwise, extra landing gear nacelles may be required.

3.2.6 FLAP TYPE

During a low speed flight, the required lift coefficient is so large that a normal wing cannot generate it. To prevent the aircraft from entering an unacceptable flight speed during take-off, approach and landing, high-lift devices are required as a means of increasing wing lift. This increase is obtained by the high-lift devices mainly through the following ways :

- increased airfoil camber,
- boundary layer control resulting from improved pressure distributions along the wing chord line,
- an increase of the wing effective area by extending the chord when the high lift devices are deflected.

There are a number of feasible high lift devices system available, although the choices are often limited by the associated drag penalty, mechanical complexity, development and maintenance cost, and structural weight. The types of high-lift devices included in the analysis are :

- Plain flaps
- Split flaps
- Single-slotted flaps
- Double-slotted flaps
- Fowler flaps
- Double-slotted flaps with slats
- Triple-slotted flaps with slats

3.3 REPRESENTATION OF KNOWLEDGE

The most fundamental characteristics of a Knowledge-Based System is the database of knowledge. Since the amount of knowledge is so crucial to the success of the system, the question of how the knowledge is to be represented is critical to the success of the system.

The representation of knowledge means formulating it into a data structure suited for

a particular application [12]. The formalism should be linked with the way the knowledge is used. It should suit the mechanisms with which the knowledge is inferred and applied in solving a particular problem. The aim is to obtain an efficient way of applying the knowledge.

One of the important things that need to be considered in knowledge representation is the organisation of knowledge. This is because of the large quantity of knowledge that the system uses. Furthermore, it is important to separate the knowledge database from the program part to allow the knowledge base to grow and change. As the number of rules in the database increases, it becomes increasingly more difficult to add new rules without interfering with existing ones.

There are a few types of knowledge representation formalism considered for the Aircraft Configuration Design System.

- Production rules
- Predicate logic
- Structured objects.

Production rules representation is probably the most conventional way of formulating knowledge. The knowledge is translated into IF-THEN rules. Predicate logic formulates knowledge in terms of relationship. This type of representation is powerful in representing factual knowledge. Structured objects formulates knowledge in a hierarchical network of object. This structure represents the relationship between various objects, concepts and events in the domain. Section B.3.2 of Appendix B gives a detail description of each formulation.

The selection of the knowledge representation in this application is largely influenced by the choice of C/C++ as the computer language used to develop the system. Production rules representation is best in Fortran and LISP, whilst predicate logic is the basic concept in Prolog. While C/C++ can also do the job, it is considered to be the best in representing knowledge in structured objects.

The knowledge in this system is structured by combining the rule-based representation and the object-oriented concept. The object-oriented concept is applied to define structures of objects, called classes, which are used to store rules. The following sections briefly explain several of these structures and how they can be used to represent descriptive and prescriptive knowledge. The detailed implementation is reported in Appendix C.

3.3.1 THE CHART OF DESIGN TASKS

The arrangement of design tasks into a chart as depicted in Fig. 3.1 is intended to help organise the knowledge into chunks that are specific to a design task. This graph will form the agenda, i.e the list of possible tasks the inference engine will pursue as explained in Chapter 4.

The significance of the agenda mechanisms is that it allows several independent tasks to communicate with each other [13]. This is done by adding their evidence that supports (or opposes) the choice of a particular task to the list of justifications for that task.

3.3.2 PRIMITIVE PROPOSITIONS

The knowledge is stored in three different structures : primitive propositions, knowledge source and rulesets. These structures combine with one another to form more complex structures.

The low level structure is the primitive proposition which contains descriptive knowledge. These primitive propositions include facts or the initial conditions of a problem, the intermediate hypothesis drawn in the course of processing by the inference engine and the final solutions that the system knows. The structure, called class Association, basically takes the form of an attribute and its associated value:

attribute <----> value

This is similar to Prolog's representation of facts using predicates which describes the relationship and its arguments which represent the objects involved in the relationship. For example :

wing position <----> high wing

can be translated as "wing position is high wing", or

cruise altitude <----> high

which means "cruise altitude is high".

It can be seen that the attribute and the value only hold a string of characters or words. This is due to the way the matching pattern is done as explained in Chapter 4. It is easier to match a pattern consisting of strings of characters rather than numerical values. This however does not prevent the system from using numerical values since the numerical values can be translated into symbolic terms that are meaningful to the system. This translation is carried out by employing a range of numerical values for each attribute with each interval within two values corresponds to one symbolic term. For example, in the case of the cruise altitude :

0 - 15000 ft	--> low
15000 - 25000 ft	--> medium
25000 - 35000 ft	--> high
> 35000	--> very high

The propositions are stored in the database whose contents can be modified through

assertions and denials. When a proposition is asserted, it is added into the database. Denials remove the corresponding proposition from the database.

3.3.3 KNOWLEDGE SOURCE

A knowledge source is a structure that records the corresponding evidence that may support a decision. Each piece of evidence corresponds to an attribute of a proposition and has a rating representing the significance of that attribute in the decision making. An attribute which has a strong influence in the decision making is given a high rating.

A knowledge source may have other knowledge sources as its evidence to form a chain of rules. At least one knowledge source should be available for one design task.

For example, suppose in deciding the propulsion type of an aircraft the following factors need to be considered :

- Operating Mach number
- Operating cruise altitude
- Aircraft type.

To simplify the explanation, other factors that may contribute to this decision are not considered. The knowledge source would record this knowledge as :

[propulsion type] <---- [operating Mach number], [operating cruise altitude], [aircraft type]

in which each items in the angle bracket represents an attribute of a proposition.

3.3.4 RULESETS

Associated with each knowledge source is a set of rulesets. Rulesets group the primitive propositions into meaningful chunks of rules. The primitive propositions in a ruleset are ordered in the same sequence as the sequence of their attributes in the corresponding knowledge source.

For the previous example, suppose there is a rule as follows :

IF operating Mach number is greater than 0.65,
and aircraft type is commercial transport
THEN use jet engines

This is recorded in the ruleset as :

[jet engines] <--- [greater than 0.65], [void], [commercial transport]

The [void] proposition means that the proposition does not have any effect on the outcome of the rule.

Suppose there is another rule that says,

IF	operating Mach number is less than 0.55 and operating cruise altitude is lower than 25000
THEN	use propeller engines.

This is translated into a ruleset as follows :

[propeller engines] <--- [less than 0.55], [lower than 25000], [void].

The knowledge source and the rulesets record prescriptive and procedural knowledge. This prescriptive knowledge captures commonsense and metaknowledge and shows how to solve a problem, given some facts.

When a design task is executed, the corresponding source is activated. The inference engine examines such evidence by processing the attribute and applying its knowledge either to create a new hypothesis for that attribute and write it in the database or modifying an existing one. The execution of an individual knowledge source is a sequential process as suggested by the order of the evidence and does not stop until all the evidence is examined.

The detail of this implementation is given in Appendix C.

CHAPTER 4
THE INFERENCE ENGINE

4.1 INTRODUCTION

The main functions of the inferencing engine is to apply knowledge in the database in searching for solutions in a particular problem. The inference engine utilises a particular problem solving strategy which would be suitable to solve the problem in question.

There are a number of problem solving strategy which could be applied, some of them are :

- Data driven strategy

  This strategy applies the available data, and looks for combinations of data which may suggest a few solutions, test the solutions against a few criteria to come up with the best possible answer to the problem. This strategy is commonly called *forward-chaining* search since the search starts from the beginning point, i.e the data, and progresses forward to the end point, i.e the goal.

- Goal directed strategy

  This strategy applies an inverse approach to the earlier strategy, in which the inference mechanism suggests a few solutions that are of interest and that may or may not be true. It then tests the conditions of each solution against the available data if any solution is valid and selects the best, if there are more that one possible solutions. This strategy is commonly called *backward-chaining* search due to its backward directed approach. The search starts from the goal and moves backward to the data.

- Generate and test strategy

  This strategy is a hybrid between the previous two strategies. It identifies the factors that most determine the solutions, looks for these primary factors in the database of known variables to obtain the promising partial solutions, test the partial solutions against the secondary data to select the solutions.

The strategy selected for solving a particular problem depends on the type of the problem involved. The data driven strategy and the generate-and-test strategy are best applied in design applications. The goal-directed strategy is best applied in diagnosis and planning.

The aim of developing the inference engine must be to ensure the most efficient strategy for pursuing the solutions of the problems in question. To do this the strategy applied must be tied up with how the knowledge is structured in the database.

The inference engine of the system performs several functions :

- To decide which design step to execute :
  In design applications, the inference engine must also reflect the manner in which the design is carried out. This is translated into the sequence in which the design steps are executed. One of the situations that frequently happens in an aircraft design project is that information is not always readily available to make a sensible decision. In this case it is necessary to suspend the decision until enough information is accumulated. The inference engine should therefore be able to direct the system to select a design step to execute based on the information available at any stage of the design process.

- To provide a problem solving strategy for each design step :
  In executing a selected design step, the inference engine decides what strategy to follow in solving the problem at that step. It searches for the rules that apply to the design step. It then tries to process the conditions of each rule by either testing them against the information available in the status recorder or by asking the user for information. It binds the variables in the conditions to one of the intermediate solutions. At the same time the inference engine records all the conclusions of each rule in the status recorder, as intermediate results.

- To maintain consistency in the solutions :
  The inference engine should also be able to handle the inter-relationship problem which often occurs in the design process. A decision in the one step may influence the decision in the other step. In most case it is difficult to define a design step as a completely separate activity from the others as addressed in Appendix C.

  Finding a consistent decision that agrees with all the conditions requires a good plan. To do this, the inference engine checks if there is a conflict between the solutions suggested. It cross-examines the solutions and possible problems due to task interactions created during the execution of previous tasks. It then tries to solve this problem either by updating the solution or by introducing a correction.

- To provide an explanation facility :
  The inference engine explains the action of the system to the user. To do this the inference engine uses a few general types of question-answering plans. This requires the inference engine to trace back along the problem solving activity to find the conclusion in question. It then searches the knowledge base for the rules that supported the conclusion and translates the rules into English for presentation for the user.

The strategy employed by the inference engine is depicted in Fig. 4.1. The following sections will discuss each of these functions in more detail. The implementation of the functions is also given. To help clarify the explanation of these functions, a description of the status recorder is presented first. A detail implementation of the status recorder is explained later.

THE INFERENCE ENGINE

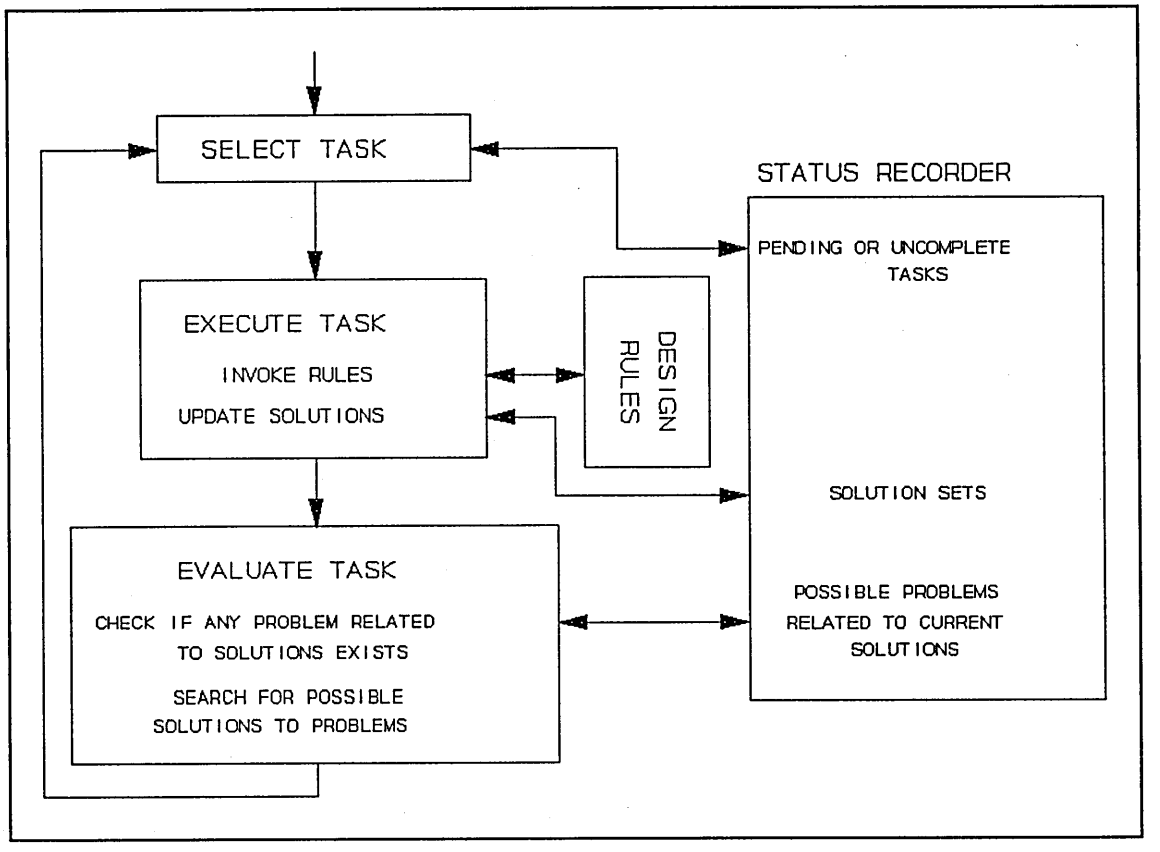


Fig. 4.1 The strategy employed in the inference engine

4.2 THE STATUS RECORDER

The status recorder works together with the inference engine. It can be viewed as the memory space for the inference engine to work on. Its main functions are :

- To record the control of actions awaiting execution
- To record all the conclusions generated during the processing of knowledge based rules.

4.2.1 THE QUEUE OF DESIGN TASKS

The queue is a list of design tasks which are ready for execution. It has the characteristics of last-in first-out (LIFO) container, in which the last item added into the queue will be the first one to be removed from the queue when the call is made.

There are two different queues used in the system, they are :

THE INFERENCE ENGINE

- Design Module Queue
- Design Step Queue

Both queues have the same form and perform identical actions. The difference between the two is only the items that are placed within them. The design module queue contains the list of design modules. It is consulted by the system controller in determining the design modules to activate. The design step queue is used only in the configuration analysis module. It contains the list of design steps for configuration analysis. It is consulted by the inference procedure in deciding the design step to solve.

Since both queues are identical, any performance corresponding to one queue can be similarly applied to the other. Because of this, to avoid repetition, the following description can be applied to either of the two only with different member objects.

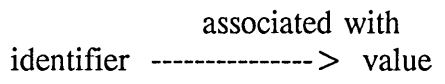
When the system is loaded or a new design effort is initialized, each queue is filled with design modules or steps from the highest level of the hierarchical tree of actions.

The design step next on the line in the queue, i.e. at the top of the stack, will be recalled as the next step to execute, called the active step, hence removing it from the queue. If the user decides to suspend the active step, it will be pushed back on to the queue and the next step on the line will taken out as the new active step.

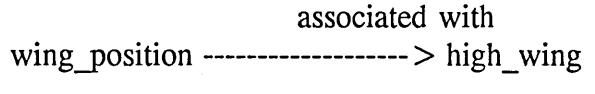
A design step which has been successfully completed will be removed from the queue. Its independent steps, i.e the steps placed immediately below in the action tree, and are directly related to it, will be added onto the queue one at a time.

4.2.2 THE DATABASE ELEMENTS OF THE STATUS RECORDER

The database element of the status recorder is used to store conclusions and intermediate hypotheses constructed during the inferencing of knowledge and problem solving activity. Each conclusion or hypothesis is stored in a specially defined structure of objects called Association. Structure Association is defined as objects which has a pair of an identifier and its value, like the following :



For example, if the decision for wing position is a high wing, this decision is stored in the database as follows :



Each association is unique, i.e there will be no more than one association which has the same identifier and value. However, an identifier may have several values, each of

which is stored in a separate association.

Several associations are linked together to form a class called Dictionary.

The dictionaries used in the system are of three different forms :
- Configuration database
- User database
- Intermediate database

The configuration database is a dictionary which consists of associations of aircraft configuration. The associations it holds are those whose identifier represents the configuration design step.

The user database is a dictionary which comprises of associations of user input. It may hold a numerical value as well as symbolic value.

The intermediate database is a dictionary whose members are associations which holds the intermediate conclusions or hypothesis as a result of the processing of rules.

4.3 IMPLEMENTATION OF THE INFERENCE MECHANISM

4.3.1 SELECTING A TASK TO EXECUTE

First the inference engine forms an agenda of design tasks and stores this agenda in the queue element of the status recorder. This is done by consulting the hierarchical tree of actions to identify the priority of various tasks. The tasks placed at the highest level are pushed into the queue to form the list of actions which are ready for execution. It then retrieves the task placed on the top of the queue to execute and remove it from the queue.

This queue forms an agenda of design actions for the inference procedure to execute. It includes a list of design actions ready for execution.

The user however has the authority to control the execution. He/she can override this agenda by suspending the current task suggested by the inference engine. If this is the case, the inference engine will retrieve the next task from the queue, i.e the item that is on the top of the queue, and push the suspended task back into the queue.

4.3.2 EVALUATING A TASK

The next step is to evaluate the selected task. The inference engine searches the knowledge base for the design rules applicable to the task. It validates every condition of the rules and records the conclusion of each rule in the intermediate database of the

status recorder. When all the appropriate rules have been considered, the inference engine then tries to match the set of conclusions in the database with the pattern of the solutions sets.

To find the best solution, the inference engine applies some kind of weighting factors based on specified performance criteria to each conclusion and calculates the maximum values accumulated for each possible solutions. The solution with the highest value is chosen as the final solution. The new solution is then added into the configuration database of the status recorder.

4.3.3 SOLVING TASK INTERACTION PROBLEMS

In maintaining the consistency of the solutions placed in the status recorder in the light of new evidence obtained during the execution of a specific task, the inference engine consults the database if there is a consequence of the step decision in the database. Then it tests whether the consequence conflicts with any prior decision in the configuration database. If so, the inference engine will place the step whose decision is conflicting into the queue and delete all the decisions associated with its dependent step from the database.

The inference engine then needs to re-evaluate the step in the light of new evidence. In some cases, the inference could suggest the second best solution for the step and test it still conflicts with the consequences.

4.3.4 EXPLAINING THE REASONING PROCESS

In explaining the reasoning process that leads to a conclusion, the inference engine needs to trace back along the database element of the status recorder for that conclusion. First it trace each knowledge rule that relates to the step and the conclusion. Each rule corresponds to an intermediate solution placed in the intermediate database. The inference engine collects all these intermediate solutions one by one in the order that it finds them. It then translates the solution according to the rule associated with it, into English.

In explaining the solution, the inference engine identifies the identifier and the value of the solution association. It then match the identifier and the value with the keywords in the explanation file in order to locate the appropriate explanation text for the solution. When they both match, the procedure then reads the text until the end of the explanation and displays the text to the user.

Details of the implementation is given in Appendix D.

CHAPTER 5
INTERFACING WITH ENGINEERING MODULES

5.1 SYSTEM ARCHITECTURE

Any computerised aircraft design system has to be supported by programs to calculate various different aspects of design. Some of these programs are currently available in different forms and levels of accuracy, each able to solve a specific problem.

The approach implemented in this study is based on several considerations :

1. There are various computer programs available at different stages, most of which are developed in FORTRAN. Each of these programs is tailored for dealing with a particular aircraft design problem and they are all independent, even though they can be related one to another to a certain extent. Most of these programs have been extensively used in aircraft design projects and have shown their potential significance.

   The simplest approach in building an integrated design system is to incorporate these programs into the system. There is no reason to reinvent the wheel by developing new programs to specifically suit the new system. The main problem is probably the difference in the format of the input/output between various programs.

2. The design process is an on-going process and spans a number of years. It applies up-to-date technologies to maximum benefit with minimum cost. As technology progresses and computing capability improves, different resources are available for enhancing design analysis. Any design system should be able to capture these technologies and incorporate them into the system.

3. In developing new programs, it is easier to build many small programs, each designed specifically to solve a particular design problem than to develop a multi-purpose program which handles a much broader, and hence more complicated, subject. The problem with this idea is that most problems require basically the same input data and hence much effort will be wasted in creating the input data for different uses. Furthermore, some of the problems may require several common routines; hence increasing the necessity for duplicating the same routines. These drawbacks are however outweighed by the benefit of having to deal with smaller problems which are obviously easier to handle. More over, duplicating common routines may not be necessary in the programming stage if all the common routines can be built into libraries of compiled (object) files, which can be incorporated in any program at link-time.

The approach taken in this study is then to develop a system which enables the user to include any existing programs while at the same time to keep the system open to newly

developed programs/routines. This approach employs an intelligent interface which enables shorter iteration cycle times. Coupling with a knowledge-based system this approach will be able to consider some of the downstream issues quite early.

To achieve these aims, there is a need to develop :

1. An intelligent system controller which decides which program to run at any one stage of the execution. This system controller should be made flexible in that it can deal with any number of executable programs in any order.

   Design integration is a means of bringing together powerful design and analytical tools [27,28]. It couples separate computer-based tools by employing an intelligence interface to accomplish the task of extracting and transforming information from and to the analysis programs.

2. A database management module which controls the transfer of data to and from different modules/programs. This database system deals with input/output of each program/module. It records all the data relevant to the design, which can be retrieved when necessary.

3. A program interface which primarily controls the input/output of the programs. It therefore works together with the database management module in retrieving data from the database for input and storing data from output to the database. This will eventually change the way the application program is executed, since data preparation is now taken over by the interface. Ideally each application program should have its own interface, complete with detailed record of its input and output variables. This information is vital to enable cross-reference between application programs.

The architecture of the system is shown in Fig. 5.1.

5.2 DEVELOPMENT AND MODIFICATION OF ENGINEERING MODULES

There are several modules/programs developed or modified for use in the system, they are as follows,

1. OPDOT
2. FUSELAGE
3. WEIGHT
4. DRAG
5. ACCOST
6. FLTANAL
7. MITCHELL
8. ENGINE DATABASE

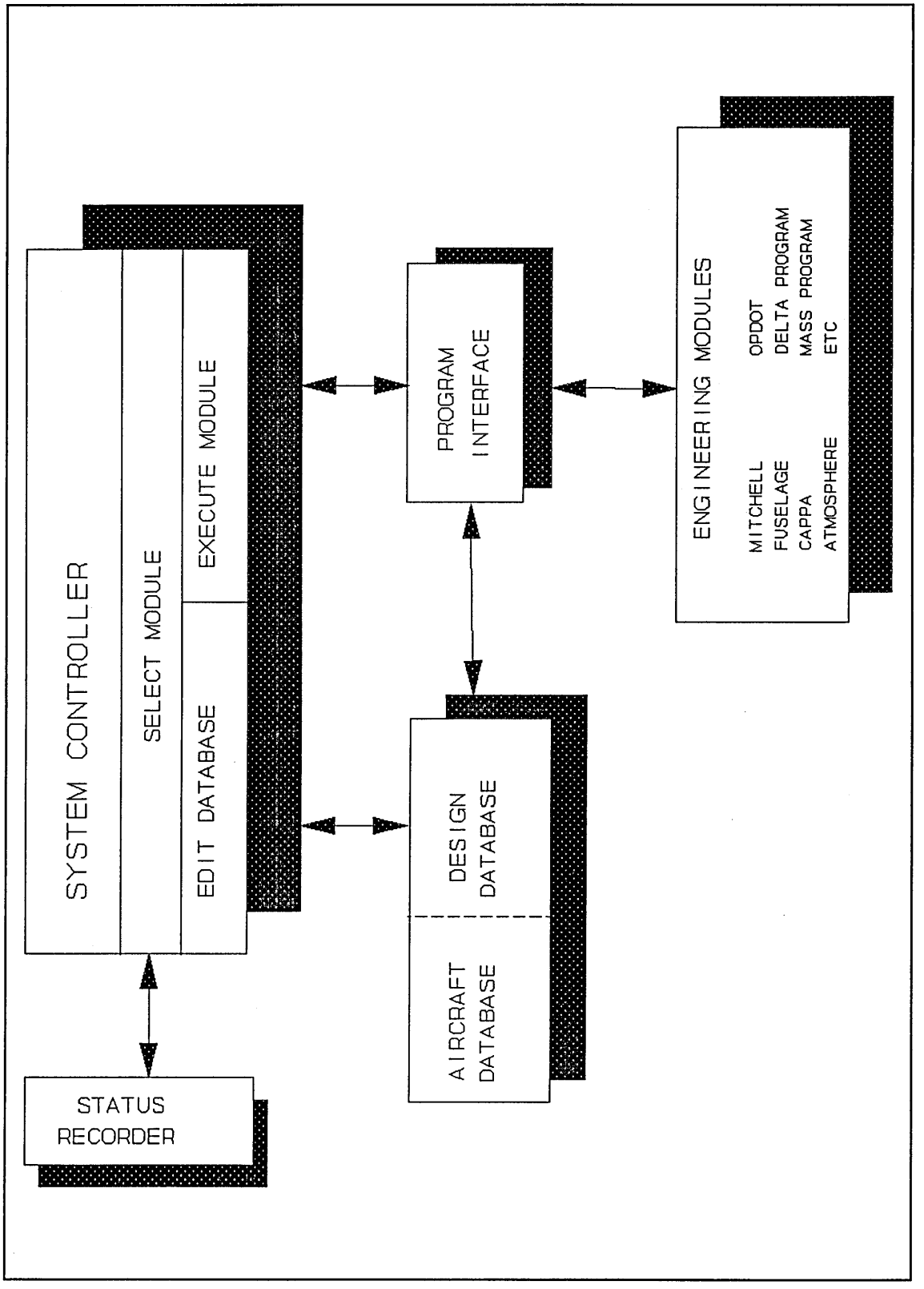


Fig. 5.1 Interface with engineering modules

INTERFACING WITH ENGINEERING MODULES

9. PARAMETRIC STUDY

Most of the modifications are required either to enhance the capability of the module to deal with different concepts of design or to update the module with a more recent empirical values. The development of new programs is necessary to make the system complete in its analysis, even though only to the first degree. The following section describes the modification made in each program.

5.2.1 OPDOT

OPDOT [29,30] is a computer program originally developed by NASA for optimisation of a jet powered transport aircraft. It aims primarily to provide an optimisation routine that can be applied to an aircraft design problem with an example specifically tailored for a 200 passenger Boeing 727 type aircraft.

The modifications undertaken were mainly to provide a more general method of calculation to enable different types or size of aircraft to be analyzed [30]. These modifications are mainly as follows,

1. To modify the subroutines calculating the component weights of the aircraft including the fuel weight, which is dependent upon the cruise /climb condition and airfield performances.
2. To modify the subroutines calculating the aerodynamic performance, i.e. the drag build up, induced and interference drag characteristics including Mach number, wave drag factors, lift characteristics in cruise and approach, trimmed lift calculation and stability and control derivatives calculation.

   Most of these modifications were required to enable the system to deal with different aircraft configurations such as conventional tail aft [31], canard wing [32,33,34] and three-surface configurations [35,36] as well as different engine locations [37].

3. To update subroutines calculating the aircraft costs including first cost estimates, direct and indirect operating costs. These were necessary as the old routines become obsolete for the new generation of aircraft in that the accuracy of the estimation is very low.

Section F.2 of Appendix F explains most of these modifications to a greater degree and gives a description of the program.

This program utilizes some of library routines which for the College VAX computers are replaced with the NAG subroutines. For the College Sun workstations, in which the prototype of the design system was developed, there was no suitable routines for similar applications. Although several attempts were made to replace the routines by writing new subroutines, the performance of the system degrades quite dramatically. For this reason it was decided to leave the program out from application with the system until

suitable routines are available.

5.2.2 FUSELAGE

The fuselage program is used for aircraft fuselage sizing. It was originally developed by the author in Turbo-Prolog for IBM PCs and compatibles under DOS operating system [38]. It was later rewritten in FORTRAN for the College VAX and Sun workstations for use in the system.

Several considerations are taken into the program :

- Simple structural design giving an efficient use of space and flexible layout. This assumes a fuselage with has a mid section with constant cross section to give the possibility of extending the length of fuselage if necessary.
- The cabin should provide easy and unobstructed access to aircraft services.
- The cabin should provide sufficient level of comfort. This is translated into the width of seats, aisles, longitudinal distance between seats, adequate galley and toilet facilities.
- There should be adequate number and size of emergency exits for the maximum number of passengers that can be accommodated. This should comply with the airworthiness regulations.
- The fuselage should provide adequate room for the required amount of cargo. The cargo normally occupies the space underneath the floor, if this is possible. A standard container type is used to minimise costs.
- The fuselage should have a good aerodynamic shape and provide adequate crew vision and tail clearance for take-off and landing.

The program works from the inside to the outside. It works on the principle that the outside has to be bigger than the inside, in that the fuselage should be able to house all the items that needs to be placed in it. It first determines the internal size of the passenger cabin for a particular cabin layout specified by the user.

Having set the inside diameter of the cabin, the available cargo volume underneath the floor is examined to see if any standard cargo containers can be fitted.

Once the internal layout of the fuselage has been determined, the outside dimension can be calculated. The external dimensions of the fuselage can be estimated by adding the allowance for the width of the fuselage frame to the internal dimensions. The length of the fuselage can be calculated as the total length of the passenger and cargo section, flight deck, unpressurised nose and tail section of the fuselage.

The output of the program can be obtained as a text file and graphs. A detailed explanation is presented in section F.3 of Appendix F.

5.2.3 WEIGHT ESTIMATION

In order to support analysis at any level of design, weight estimation methods should be supplied with different levels of detail. These methods should not only give a reasonable accuracy at any level but should also reflect the sensitivity to variations of the main parameters.

Weight estimation during the conceptual and preliminary design phases is mainly statistical. The simplest estimation method is the crude statistical technique used for first estimation of the take-off and empty weights from performance requirements. This method is normally adequate for determining the initial size of the aircraft.

The fuel weight for cruise is calculated from the Breguet range equation. The total fuel weight is estimated from this cruise fuel weight fraction with additional allowance for fuel burned during take-off, climb, descent and landing phases. Extra fuel weight is added to account for the reserve mission. This allowance is normally depending upon the range of the aircraft.

For a more elaborate method, the empty weight of the aircraft is broken down into smaller weight units. Simple estimation methods are used to calculate the weights of each major structural component, the powerplant unit, the aircraft equipment and systems. These methods, although admittedly crude, are adequate to represent the effect of the aircraft geometry and configuration variation on the overall weight of the aircraft.

The details of the equations used for this weight estimation is given in section F.4 of Appendix F.

5.2.4 DRAG ESTIMATION

Accurate estimation of the aerodynamic characteristics of a new aircraft plays a crucial role in ensuring that the potential performance estimates are met by the aircraft in production. The lift and drag relationships mainly govern the cruise efficiency of the aircraft.

At the first stage, a simple drag estimation method is normally adequate for initial analysis. However, as the design progresses, a more accurate method is necessary not only to reflect the influence of dimensions of all components but also the interference between various components.

The program employed in this system utilise a drag build up method in which the drag of each component is calculated by using flat plate analogy and is accounted for the total parasite drag. The lift dependent drag is also calculated. The program, called DELTA, was originally developed for NASA by a team from Lockheed [39]. A version of the program was written based on the original report for the College of Aeronautics [40,41]. The program was later updated to include drag increment due to landing gear and

flap configurations [42].

A description of the program is given in section F.5 of Appendix F.

5.2.5 ACCOST (AIRCRAFT COST PROGRAM)

Aircraft cost is probably the best criterion, other than aircraft performance, in selecting the best design. Unfortunately, aircraft cost estimation is not as straight forward as performance estimation since there is large discrepancy between methods used. Each manufacturing company has different way to divide the cost into cost elements, and there are no definite methods widely used.

Aircraft cost estimation is largely statistical and in the final analysis the cost of a new aircraft is predicted based on the actual costs of existing aircraft. However, it is difficult to determine how much an existing aircraft really did cost in terms that are meaningful to the next aircraft. Part of the problem is that the aircraft unit cost is largely influenced by production quantity and rates. The more aircraft produced, the more the manufacturer learns and the cheaper the next aircraft can be produced. Furthermore, the development costs can be spread over more sales. For this reason, it is difficult to compare costs of a new aircraft just entering production and an existing aircraft already produced in hundreds.

Despite these problems, the aircraft cost estimation program, ACCOST, was developed by the author to enable the designer to perform cost analysis of a new aircraft [43]. The core of this program is obtained from the OPDOT program described earlier. The program can be used to estimate the following costs :

- The first cost of the aircraft based upon a specified production quantity and rates. The purchase price of the aircraft is set to recover the costs of research, development, test and production during the expected production time span of the aircraft [1], with some allowance for profit and bank charges.
- The operations and maintenance costs of the aircraft during its service life including Direct Operating Costs (DOC) and Indirect Operating Costs (IOC). The methods, which are applicable to aircraft of medium to large size, are based on a cost analysis of the American Airlines [44].
- Airline revenue, which primarily comes from the passenger fare. The ticket price is approximately proportional to the trip distance, but there is a tremendous variation between ticket for different passenger classes. The program assumes that about 1/8 of the total passenger are first class passengers.

5.2.6 FLTANAL (FLIGHT ANALYSIS PROGRAM)

The flight analysis program, FLTANAL [44], was developed by the author to estimate the fuel weight required of an aircraft of a given empty weight and payload to perform

its specified mission. The main core of the program is taken from the OPDOT program. The program follows a model of the flight plan of the aircraft during its intended mission and calculate the fuel used during each phase of flight. The flight plan consists of several flight phases :

- Engine start, taxi and take-off
- Climb and accelerate to initial cruise flight condition
- Cruise at the specified cruise condition
- Descent for landing
- Fly the reserve mission which is performed as an extension of the cruise flight for a given distance and a holding flight for a specified duration.

5.2.7 MITCHELL (Aircraft Stability And Control Analysis Program)

The MITCHELL program was originally developed at the Royal Aircraft Establishment to enable the designer to make a rapid prediction of the stability and control characteristics of aircraft with distinct bodies, wing and empennage using only limited geometric information [46]. The program which was rewritten in the College of Aeronautics for use in the VAX computers [47] can be used to estimate the longitudinal stick fixed modes, to calculate some of the limiting speed for low speed flights for a given weight and flight conditions, and to predict the responses to control for handling qualities. The program has been extensively used in many design project studies and has proved sufficiently accurate for many applications.

Due to unavailability of a suitable subroutine calculating the first order differential equations, the version of the program installed in this system is not capable to predict the responses to control for handling qualities. Despite this, the program is adequate to perform analysis for several stability and control qualitites.

5.2.8 ENGINE DATABASE

An extensive engine database is required to support analysis in the selection of engines. For the purpose of calculating the fuel weight, cruising speed, take-off and landing performance and rate of climb, the actual performance chart of the selected engines is required. This performance chart includes the available thrust and fuel consumption for different engine ratings, speed and altitude.

The data base also covers the engine weight and geometric data. These data are required for the calculation of weight and balance, drag and stability derivatives. If the right size of engines are not available for a particular application, it is possible to up rate or down rate the existing engines to the required size. For this purpose, the geometry and weight of the engines need to be adjusted to suit the required size. The following equations are derived from Ref. [1] and used in this case :

$$D_{en} = D_{en_{ref}} \left[\frac{T_{req}}{T_{ref}} \right]^{C_1}$$

$$L_{en} = L_{en_{ref}} \left[\frac{T_{req}}{T_{ref}} \right]^{C_2} \qquad 5.1$$

$$W_{en} = W_{en_{ref}} \left[\frac{T_{req}}{T_{ref}} \right]^{C_3}$$

where T, D, L and W represent the gross thrust, diameter, length and weight of the engines, ref and req denote the reference and the required values. These equations can only be used with reasonable results for T_{req}/T_{ref} of about 0.7 - 1.4. The coefficients C_1, C_2, C_3 are specific to the engine specification, but in general the following values can be taken [1] :

For turbofan and turbojet engines :

$C_1 = 0.4$
$C_2 = 0.5$
$C_3 = 1.1$

For turboprop engines :

$C_1 = 0.12$
$C_2 = 3.73$
$C_3 = 0.803$

5.2.9 PARAMETRIC ANALYSIS

The parametric analysis is a design investigation which is based on general estimation methods and sizing procedures. It performs variations of design parameters in order to study their effects on the overall designs.

The simplified sizing method employed in this module evolves around the iteration of the take-off weight calculation. While admittedly crude (with accuracy of about 10%), the methods addresses all the essential features of most sophisticated designs.

The take-off weight can be found iteratively from the following equation [1],

$$W_{to} = \frac{W_{pax} + W_{cargo}}{1 - \left(\frac{W_{oe}}{W_{to}}\right) - \left(\frac{W_f}{W_{to}}\right)} \qquad 5.2$$

where

W_{to} = take-off weight

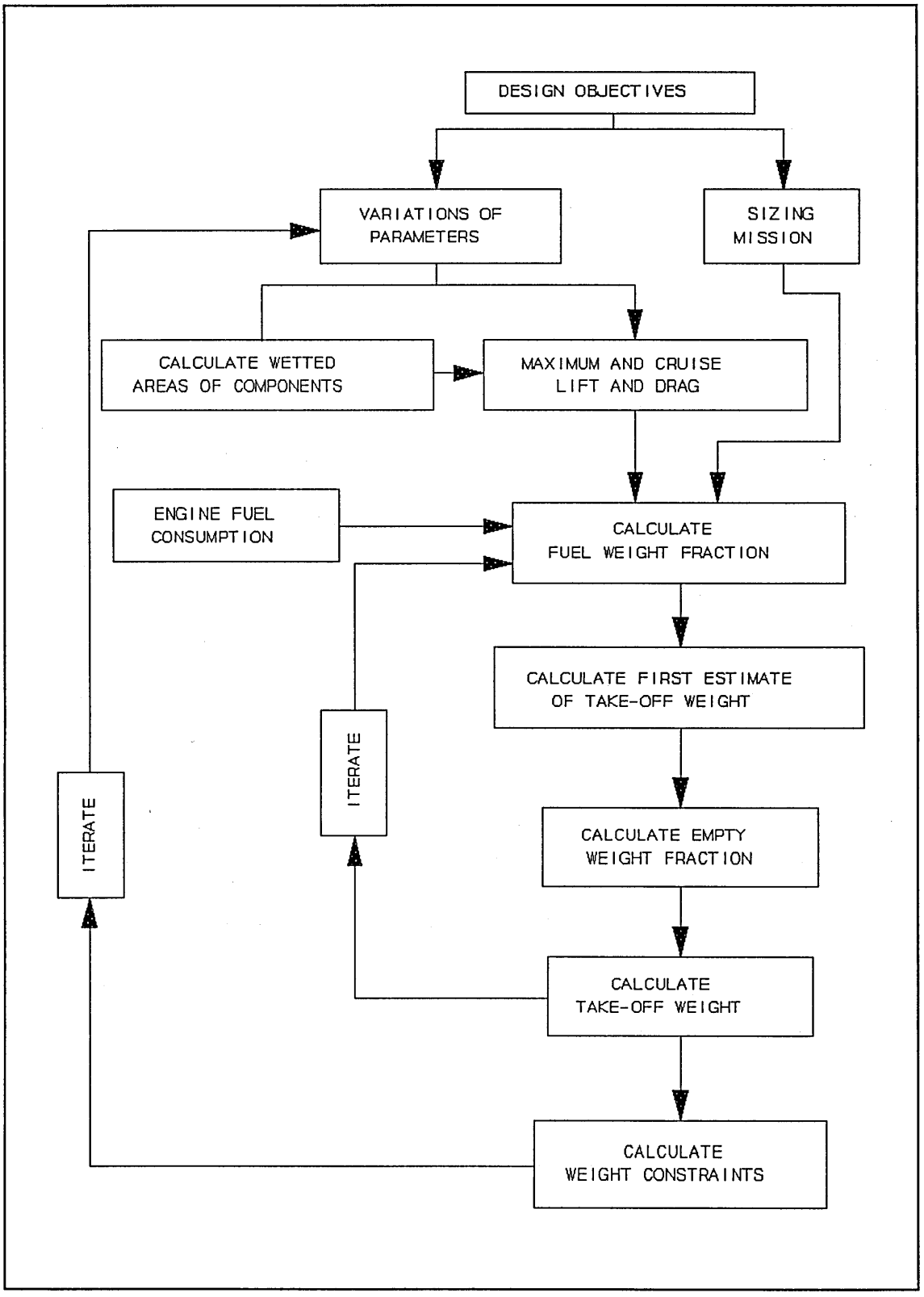


Fig 5.2 First-order sizing method

INTERFACING WITH ENGINEERING MODULES

W_{pax} = weight of passengers including luggage
W_{cargo} = weight of cargo
W_{oe} = empty weight of the aircraft
W_f = fuel weight

The payload weight, i.e the sum of the weights of passengers and cargo, is normally specified as a design requirement. The empty weight fraction, (W_{oe}/W_{to}), is a function of airplane geometry and layout. The fuel weight fraction, (W_f/W_{to}), is a function of not only the geometry and layout of the aircraft, which is reflected in the lift and drag coefficients, but also the performance requirements and engine type.

The methods for calculating these fractions are based on the methods suggested by Raymer [1], Torenbeek [2,48] and Loftin [49]. Several limitations on weight and engine size are imposed upon the design. These limitations are set by design requirements for cruise, take-off, landing, maximum rate of climb and service ceiling [50,51,52,53]. A structural limitation on the wing aspect ratio is also imposed by the aeroelastic bending and torsion stiffness of the wing. The first order sizing procedure is depicted in Fig 5.2.

5.3 IMPLEMENTATION OF THE ENGINEERING MODULES

5.3.1 SYSTEM CONTROLLER

The main functions of the system controller is to decide which design module to execute and to control the execution of the module. The system controller employs a plan of actions based on the design model explained in Chapter 2. The hierarchical graph of design modules is translated into a queue stored in the status recorder explained in Chapter 4. Each time the activity in a module is completed the next module in the queue is loaded.

The list of modules and the corresponding programs is stored in a text file. When loaded the system will consult this file so as to identify the available modules. The order in which the modules and the appropriate programs are listed in this file will be reflected in the order in which the module can be executed. This order however is not strictly binding, since it can be overridden by the user. This will eventually update the plan of actions suggested by the system controller.

The system controller consults the pool of the program library to identify the suitable program for the problem in question. This program library file records information about all the programs available for use. This information covers :

- the name of the program (the executable file),
- the design module : problem area associated with the program,
- the directory in which the program is stored
- the input/output format of the program. This includes all the data type names of the variables required for input/output. In most cases, this information is stored in a

separate input/output file which is consulted before the selected program is executed.

The system controller has two modes of operation :

- Editing Mode
  The editing mode of the system is designed to carry out two functions :
  - to enable access to the database of aircraft being designed.
    The editing of data can only be performed while the system is on this mode.
  - to load a specific design module to work on.
    The selection of design modules can only be done while the system is on this mode.

- Execution Mode
  The execution mode enables the designer to control the execution of a specific design module. It activates the program interface which allows the designer to select variables to analyze (for example, to specify design cases, to define the range of parametric variation), to display results when the execution is completed, either in a form of text or graphs.

  However, neither the selection of design module nor the editing of data can be done in this mode. This is to ensure that the work in any module is done with the correct set of data and to protect the database from 'illegal' update.

The system controller is designed to enable easy transfer from one design module to another. Once the system is informed that the activity in a specific module has been completed, it will prompt the designer if any data update is required. In any case the mode is automatically changed into the editing mode to allow the editing to be made and the next module on the queue is loaded.

5.3.2 AIRCRAFT DATABASE

The data base management module performs two independent functions :

- To store data of existing aircraft which serve as a reference to the designers.
  This database is required to supply data for the purpose of comparison and consultation in a particular design activity. In the beginning of a new design project, it is always desirable to review past designs in order to learn about the competitors. This study of comparison is aimed toward specifying a reasonable design requirement for the new design. Another particular area in which this database becomes handy is when a data for a particular parameter of design needs to be guessed. A reasonable value can be obtained from past designs of similar requirements. This consultation may probably not solve the problem, but at least will initiate the calculation.

  For these purposes, an extensive database is required, not only in the quantity of data recorded but also in the number of existing aircraft included in the database. The

record should cover all data relevant to design, not only geometry but also other aspects, i.e. weight, performance, aerodynamics and systems.

This database management module should be able to perform all the basic functions of a database system, i.e :

- data retrieval : to recall a specific data of a particular aircraft,
- data editing : to update a specific data of a particular aircraft and store it permanently for future use,
- data extension : to add new data to the existing set of data,
- database extension : to include new aircraft into the database.

- To store data related to the aircraft being designed.
  This database is only relevant to a new aircraft project. The whole design activities can actually be seen as a process of database build-up. Each activity involves data processing, in which existing data are used to create new data either by analysis or synthesis. As the design progresses, the data are accumulated. With this the management of data becomes a crucial aspect of design, in which the organisation of data has to be as such that each design department responsible for a particular design activity can have easy access to it. More important is that all design activities should have an agreement of which data to use for each design parameter.

  The purpose of this database is therefore to ensure that all the design analyses are carried out with the same set of data. Any data obtained as output of any analysis which is considered to be final will replace the data previously stored in the database so that it can be used in other analyses that apply that particular data.

5.3.3 PROGRAM INTERFACE

The main functions of the program interface are :

- to translate data from the aircraft database into a format that is acceptable to the program selected. This is done by consulting the program library file to identify the set of parameters required for that particular program, retrieving the data for each parameter from the database, and then writing them into an input data file for the appropriate program.
- to control the execution of the particular program. Sometimes several case studies need to be performed in the program. The program interface will provide the facility to specify the design cases. This also provides the access to the result of the execution, in which output can be presented in text or graphs.

The basic procedure taken by the interface to execute an application program is as follows :

- Create the input data file for the application program by consulting the input/ouput data file,

- Change the system directory into the application directory as specified in the program library file,
- Execute the application program with the specifed input file
- Redirect output to a file specified by the system controller.

The detailed implementation of the system controller and program interface is given in Appendix E.

CHAPTER 6
CASE STUDIES

6.1 INTRODUCTION

For the purpose of demonstrating the capability of the system in dealing with the aircraft conceptual design problem, case studies are performed. These will show how the system can be used to carry out design tasks, addressing several aspects of design as it progresses. Ideally it is preferable to select a new type of aircraft to work on in order to test the validity of the conclusions. This is however difficult to perform since there will be no real means of comparing the result. Selecting a completely new type of aircraft will inevitably lead to a situation in which two unknowns exist, one relating to the aircraft specification since no real data is available and the other to the conclusions.

For this reason, it was decided to select a type of aircraft which is comparatively new but of which more information can be obtained about the design characteristics. The aim of these case studies were therefore not only to evaluate the sensibility of the advice with its line of reasoning, but also to demonstrate how the system moves along the design activities to formulate the design.

There are a few aircraft which could be used as reference for this case study :

- Airbus A340
- Boeing 777
- DASA/CATIC MPC 75

The Airbus A340 is a very long range aircraft with capacity of 295-440 seats. The aircraft has just been rolled out from the production line and ready for flight testing. The aircraft is powered by four CFM-56 turbofan engines, mounted in pods under the wing. It has a low wing configuration with the tailplane mounted on fuselage. The aircraft is equipped with the fly-by-wire concept which will help fly the aircraft within its operating envelope.

The Boeing 777 is a completely new aircraft and is currently in the design stage. It is expected to fly medium to long range routes to fill up the gap between the Boeing 767 size aircraft and the top of the range Boeing 747. It has capacity of about 350 passengers with range of more than 8000 km. The aircraft is equipped with two turbofan engines, a decision which is considered to be optimistic for such a long range.

The MPC 75 is also a new aircraft which is still in design stage. The aircraft was originally designed to be equipped with two propfan engines mounted one each on each side of the rear fuselage. The design was later changed to two wing-mounted turbofan engines for commercial reasons. It is expected to carry 70-80 passengers for a design

range of 1500 nm (about 2800 km.).

Due to the difficulties to compile data on the Boeing 777 and MPC 75 aircraft, the Airbus A340 was selected. The design requirement for the case aircraft will be taken similar to that of the Airbus A340. To make reference easier, the case aircraft is designated as X-91 aircraft.

6.2 DESIGN REQUIREMENTS FOR THE X-91 AIRCRAFT

The design requirement for the X-91 is taken similar to the A340-300 [54] as follows :

- The aircraft is expected to accommodate 295 passengers in 3 class layout, with the arrangement as follows :
  18 first class passengers at seat pitch of 1.52 m, six seats abreast,
  81 business class passengers at seat pitch of 0.96 m, seven seats abreast,
  196 economy class passengers at seat pitch of 0.86 m, eight seats abreast.

  An alternative layout is available for 335 passengers in 2 cabin classes : 30 first class passengers at seat pitch of 1.02 m, and 305 passengers in the economy class, with seat pitch of 0.84-0.86 m.

  The aircraft should be able to carry 375 passengers in tourist class with 8 seats abreast. Alternatively, 440 seat capacity will be available in a 9 seat abreast layout.

  The aircraft should be able to accommodate LD-1 and LD-3 containers underneath the floor.

- Design range of the aircraft should be 13805 km with payload of 295 passengers including baggage and diversion distance of 370 km (230 nm.). The range for 295 passengers including luggage and 20.3 ton of cargo should be 10175 km.

- The aircraft is expected to fly with cruising Mach number of 0.8. The maximum operating Mach number would be 0.86.

- The aircraft will cruise initially at 9.15 km with maximum ceiling of 9.635 km. with one engine inoperative.

- The aircraft should be able to operate from airfields with take-off distance of 2915 m to 10.5 m (35 ft) height clearance at sea level, ISA conditions. The landing distance from 15 m (50 ft) height will be 2214 m. (Due to unavailability of data, this landing distance requirement has been similar to that of McDonnell Douglas MD-11 aircraft which has maximum landing weight of almost the same magnitude to that of A340-300).

- The aircraft should have a maximum rate of climb of 16.0 m/s at sea level with all

engines operating.

6.3 DESIGN STUDY FOR X-91 AIRCRAFT

The following describes the synthesis of aircraft configuration and the analysis of various aspects of design for the X-91 aircraft using the Aircraft Configuration Design System. Unfortunately no hard copy can be reproduced to show the step by step procedure that is performed during the execution since there is no printers connected to the Sun computers. Nevertheless, a limited number of photographs are presented to clarify some of the explanations.

To start the design for the new aircraft, the name of the aircraft in the system panel (by default, it is set to 'Boeing-727') was replaced with the new name, X-91. First it was required to initialise the database for the new aircraft. This was done by loading the data file with the newly specified name, i.e X-91, into the system. Obviously since this was a new name, no data file existed. The system displayed a message window :

No aircraft on record with that name.
Do you want to start a new record with that name ?
Yes, Continue No, Cancel

which was confirmed by selecting the 'Yes, Continue' option. The system then initialised the database for the X-91 with all numerical data were assigned to zero. The system then displayed another message window :

Have you saved previous edits ?
Continue, discard edits Cancel

The 'Continue, discard edits' was selected since the prior data file, the Boeing-727, had been saved. This action loaded the newly created data file, the X-91 data, into the system to replace the Boeing-727 data.

6.3.1 PARAMETRIC ANALYSIS FOR INITIAL SIZING

As the first step it was suggested by the system to perform a simple analysis to obtain the major dimensions of the aircraft. This is carried out in the Parametric Analysis module of the system. To do this a few more data are required in addition to the performance requirements specified in section 6.2. This includes :

Aspect ratio of the wing = 9.65
Thickness ratio of the wing = 0.12

The aspect ratio is the same as the value for A340. The thickness ratio of the wing was obtained from the average value in the database of the existing aircraft. These X-91 data

were entered into the system in the editing mode, which is the default mode when the system is loaded.

Firstly, a wing loading variation was performed to obtain a rough estimate of the size of the aircraft. This was done by changing the mode into the execution mode by selecting the 'Evaluate Aircraft' button. The wing loading was varied between 500 to 800 kg/m^2. Even though the graphs showing the total aircraft weight for different performance requirements can be produced in the system, only the take-off weight variation can be presented in Fig. 6.1. The weight is restricted by two constraints : the landing distance and the cruise conditions, which decided the selection of the best design.

Secondly, the aspect ratio was varied with constant wing area to find the best condition. The aspect ratio was varied between 7-12. The best condition suggested by the system was 10.0, slightly off the actual value for the A340.

The best design point suggested by the system is :

	X-91	A340-300
Wing loading W/S (kg/m^2)	= 680.0	
Take off weight W_{to} (kg)	= 272175.2	255000.0
Empty weight W_{oe} (kg)	= 130970.5	126061.0
Thrust to weight ratio $(T/W)_{to}$	= 0.2053	
Wing area (m^2)	= 383.20	363.10

Drag equation :

$$C_D = 0.0148 + 0.0451\ C_L^2$$

Having decided the main parameters of the design, it was decided to continue the process. This is done by switching back to the Editing Mode. When leaving the Parametric Analysis module the system pops out a message window :

Have you finished with module : Parametric Analysis ?
Do you wish to continue with other module ?
Yes, Continue No

The 'Yes, Continue' option was selected to inform that the execution needs to be continued to the next module. Then the system displayed another message window :

Do you want to update the database with the output ?
Yes No

The 'Yes' option was selected to update the database and to store the output into the database. Having done this, the system then loaded the next module on the queue and changed the mode into the editing mode.

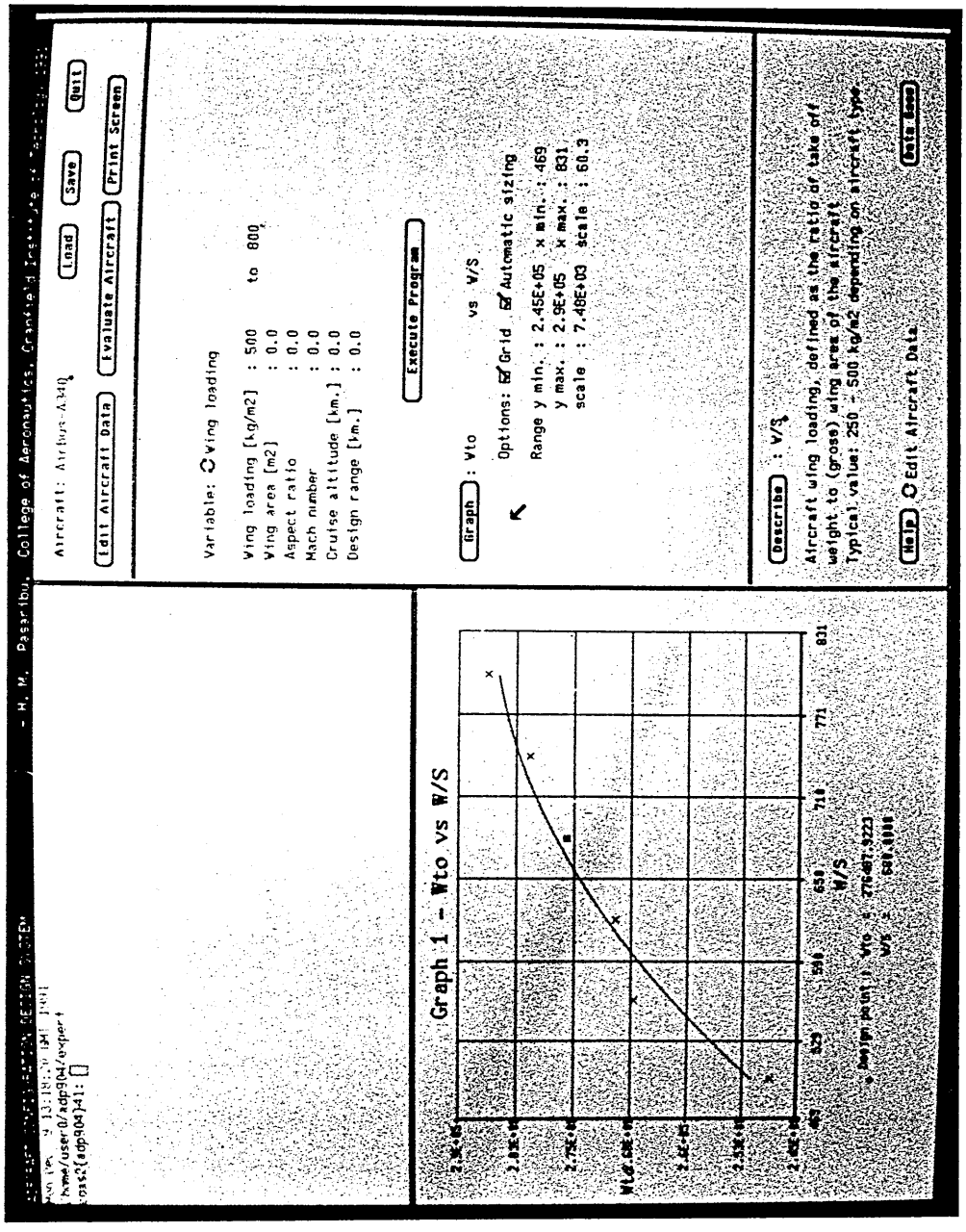


Fig. 6.1 Wing loading variation in parametric analysis

6.3.2 CONFIGURATION ANALYSIS

The next module suggested by the system was the configuration analysis. While in the editing mode, a few input data could be updated. The current values, which were obtained from the database of the X-91 aircraft, were displayed on the screen. Since no data need changing, it was decided to start the execution by switching into execution mode. The Configuration Analysis window popped out once the execution was triggered.

The first step suggested by the system was *Fuselage Layout*. An explanation about this step can be obtained by activating the 'Help' button on the bottom panel of the window. To start the analysis, the 'Execute Step' button was activated. The system displayed the *execution menu* window with the following options :

About to execute step : Fuselage Layout
Help Continue Suspend Step

Again an explanation can be requested through the 'Help' button. This is in fact the same as the previous help facility which gives information about the active design step. To start the inference, select the 'Continue' button. The system displayed a menu :

Type of payload :
Passenger Cargo/freighter

The 'Passenger' option was selected, to inform the system that the X-91 aircraft is a passenger transport aircraft. And then the following *question menu* was displayed :

It is possible to install a double_deck layout.
Is this acceptable ?
Yes No

The answer was *no*. However, the A340 is offered with a option of having a passenger cabin at the lower deck, underneath the main cabin floor, which can accommodate up to 50 passengers in five seat abreast layout. This will obviously take up some space previously allocated for the cargo containers. This possibility is also be offered for the X-91 aircraft as shown in the reasoning.

When the system completed the analysis, the following *control menu* was displayed as shown in Fig. 6.2 as follow :

Design step : Fuselage Layout completed
Show Result Explain Result Continue

When the 'Show Result' option was pressed, the following information was presented in a pop-up window :

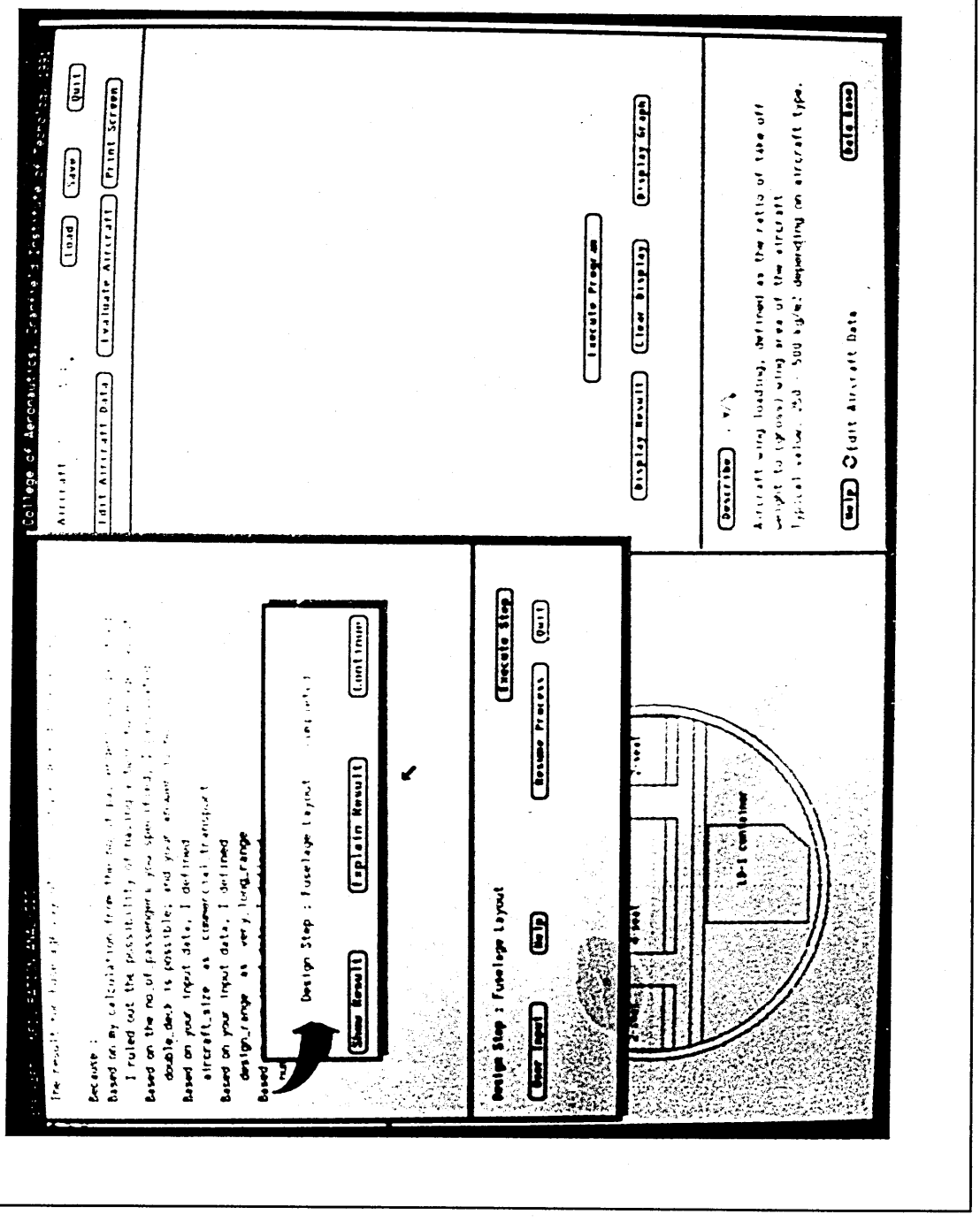


Fig. 6.2 Execution of the configuration analysis module

FUSELAGE LAYOUT
SINGLE BODY DOUBLE AISLE
A passenger seating layout with double aisles is normally required for large passenger aircraft with a single fuselage. With two aisles, the seats can be arranged so that the passenger is no further than one seat away from the aisle. This is necessary to increase the level of comfort around the cabin, through ease of movement. Only if triple seats are installed at each side of the cabin, the passengers seating next to the cabin wall will be two seats away from the aisle. This is however compensated by the fact that they have the luxury of sitting next to the windows.

The system again displayed the same *control menu* to give a chance to the user to select other options. When the 'Explain Result' option was selected, the following information was displayed on the panel :

The result for Fuselage Layout is single_deck_double_aisle
Because :
Based on my calculation for the no_of_passengers you specified,
I ruled out the possibility of having a twin fuselage
Based on the no_of_passengers you specified, I calculated
double_deck is possible, and your answer is no
Based on your input data of aircraft_size, I defined
the aircraft as commercial_transport
Based on your input data of design_range, I defined
the aircraft as long_range
Based on your input data of number_of_passengers, I defined
the aircraft as intercontinental_aircraft

Again the *control menu* was displayed. This time it was decided to continue the execution to the next design step. The next step suggested by system was *Wing Position*. An explanation about this design step was given when requested. This was done by selecting the 'Help' option in the *execution menu*. The following information was presented in a pop-up window :

WING POSITION
GENERAL :
Wing vertical location with respect to the fuselage is generally set by the real-world environment in which the aircraft will operate. Although the aerodynamic and structural differences are not without importance, they can only be deciding factors when the choice between high, low and mid wing is not dictated by considerations of maximum operational flexibility.

When executed, the system popped out the question menu :

Is it operating on semi-prepared runway ?
Yes No

The answer to this question was *no*. The result for the wing position was explained as follows :

> The result for Wing Position is low
> Because :
> Your answer to the question
> 'Is it operating on semi-prepared runway ?' is no
> Based on your input data of aircraft_type, I defined
> the aircraft as passenger aircraft

When requested, the following information was given for the 'Show Result' option :

> WING POSITION
> LOW WING :
> Low wing configuration offers several advantages. The landing gear stowage can be easily provided. With a low wing, the trunnion about which the gear is retracted can be attached directly to the wing box which, being strong already, will not require much extra strengthening to absorb the landing load. Since the leg is comparatively short the retracted gear can be stowed easily inside the wing itself, thus eliminating the external blisters usually needed with high wing.
> To provide the adequate engine and propeller ground clearance, the fuselage must be placed farther off the ground than a high wing aircraft. This will require a slightly longer landing gear leg. In most cases, a certain degree of dihedral is required to increase ground clearance, especially for the wing tips.
> Since the fuselage compartment underneath the floor is normally large enough, the wing box can pass through the fuselage to reduce drag penalty. This layout removes the structural complexity associated with high and mid wing configurations.

Once the wing position had been set, the next step to tackle was *Propulsion Type*. Again the execution menu was displayed. An explanation for this step was given when requested as follows :

> PROPULSION TYPE
> GENERAL :
> The selection of propulsion type depends strongly on the operational and performance requirements of the aircraft. There are currently two types of propulsion system available for application in civil aircraft : thrust producing propulsion system (jet engines) and power producing propulsion, in which the power is used to rotate the propellers to provide the required forward thrust (propeller engines).

CASE STUDIES

The aircraft maximum speed is normally the dominant factor for selecting either one of these two types, in which the latter is effective for low speed aircraft since the propeller efficiency decreases as the speed increases.

Having understood the activity involved in this step, the step was executed. After a little while, the system informed that the analysis was completed by displaying the control menu. When requested the conclusion for the propulsion type was explained as follows:

The result for Propulsion Type is jet_engines
Because :
Based on your input data of aircraft_size, I defined
the aircraft as commercial aircraft
Based on your input data, I defined
the cruise altitude is high
The decision for wing_position is low

The next step to deal with was *Engine Type*. Again the inference began when the Continue option in the execution menu was pressed. The system displayed a question window :

Is first cost crucial in design ?
Yes No

The answer was 'Yes' since one of the design criteria was minimum cost. The system asked another :

Is fuel price high ?
Yes No

The answer was 'No'. When the analysis was completed, the result was displayed as follows :

The result for Engine Type is turbofan
Because :
The decision for propulsion_type is jet_engines
The decision for wing_position is low
Based on your input data of aircraft_size, I defined
the aircraft as commercial_aircraft
Based on your input data, I defined
the cruise altitude is high
Your answer to the question
'Is first cost crucial for design ?' is yes
Your answer to the question
'Is fuel price high ?' is no

The explanation given for the Show Result option for this step was :

> ENGINE TYPE
> TURBOFANS :
>
> Like turboprop engines, turbofan engines use a turbine to extract mechanical power from the exhaust gasses. This mechanical power is used to accelerate a larger mass of outside air with a ducted fan of one or several stages. This accelerated air is then split, with part remaining in the engine to further compression and burning, and the remainder being 'by passed' around the engine to exit unburned.
>
> The 'by pass ratio', i.e the mass-flow ratio of the by passed air to the air that goes into the core of the engine, ranges from as high as 6 for most of the current turbofan engines to as low as 0.25 (the so-called 'leaky turbojet').
>
> The specific fuel consumption of turbofan engines is relatively lower than that of turbojet engines, although its thrust diminishes quite rapidly at very high speed (supersonic).

The next step to tackle was *Tail Layout*. An explanation of the step was given on request in the following :

> TAIL LAYOUT
> GENERAL :
>
> The design of the tail surfaces probably depends more on the general arrangement and the detail layout of the aircraft than any other major component. Because of their location, their effectiveness is influenced by the wing and the operation of the engines, particularly in the case of propeller driven aircraft. The way in which the empennage is mounted to fuselage affects the structural layout of the tail surfaces and that of the fuselage.
>
> For most aircraft designs, the conventional single-fin tail usually provides adequate stability and control at the lightest weight, but there are other reasons to opt for a different layout.

The result was explained in the following :

> The result for Tail Layout is low_tail
> Because :
> Based on your input data of aircraft_size. I defined
> the aircraft as commercial_transport
> The decision for engine_type is turbofan
> The decision for wing_position is low

At this stage, the arrangement of the major component had been defined, and it was thought to be adequate to continue the analysis in other modules even though the configuration analysis had not been fully completed. When leaving from the Aircraft

Configuration window, the system prompts the message :

Do you want to quit ?
Confirm Cancel

When the decision was confirmed, the system displayed the following message :

Do you want to save the current design ?
Yes, Save It No, Discard It

It was decided to save the design analysis so that it could resumed for future use. A more detailed example of the execution of this module is given in section G.7 of Appendix G.

6.3.3 FUSELAGE DESIGN

When leaving the execution mode of the Configuration Analysis module, the system asked :

Have you finished with this module ?
Yes No

When the reply was confirmed, the system changed the module to the next module on the queue, i.e Fuselage Design. In the editing mode, the input data is updated. The following data was obtained from the database of existing aircraft and added into the database of the X-91 aircraft :

Cabin height	= 2.08 m
Fuselage frame width	= 0.15 m
Floor depth	= 0.25 m.

At first, the seat dimensions are set the same as the ones used in the Airbus A300 and A310. This however resulted in discrepancies between internal diameters of the first class and the economy class cabin. The system message stated that the largest internal diameter was not determined by the densest class, i.e the economy class but by the first class which terminated the execution of the program prematurely. To correct this, it was suggested to change the seat dimensions for the first class, similar to those of the Fokker 100.

Having done this, the module was re-executed. The consultation can be done through the terminal window on the top left of the Aircraft Design System window :

THE CONTAINER TYPE(S) THAT CAN BE FITTED INTO UNDERFLOOR BELLY COMPARTMENT (IN ORDER OF INCREASING AREA) :
TYPE-727

TYPE-LD2

THE BIGGEST CONTAINER TYPE THAT CAN BE FITTED IS TYPE-LD1

Is this acceptable ? (1=yes 0=no)
1

The fuselage dimension obtained from this module is as follows :

	X-91	A340-300
Cabin width (m)	5.37	
Fuselage diameter (m)	5.67	5.64
Fuselage length (m)	58.29	63.65
Passenger door/ Emergency exit	3 TYPE A	3 TYPE A
	1 TYPE I	1 TYPE I
Maximum number of passengers	360	375
Container type	LD-1	LD-1 / LD-3

The detail of the input and output is presented in Appendix G. The result of this module was added into the data base and replaced some of the data previously stored in the data base. The result can be seen in a text file or graphs as shown in Fig. 6.3. The text output is displayed in a text window when the 'Display Result' button is activated. Selecting the 'Display Graph' button will present the some of output in graphs, in this case the cross section of the fuselage in economy class is shown.

6.3.4 WING DESIGN

This module has not been implemented yet. The intention was to include the ADROIT system [8] as part of the module. This however needs a complete modification of the program, since the program which was written in Turbo-Prolog cannot be ported as it is to the Sun computer. The College Sun microsystem does not have any Prolog compiler. For this reason, the major wing dimensions obtained from the Parametric Analysis were accepted as adequate at this stage.

6.3.5 ENGINE SELECTION

Like the Wing Design module, this has not been implemented. The engine size obtained from Parametric Study was assumed as final at this stage. The number of engines is assumed four, the same as the A340. The size of each engine were calculated as follows,

Gross thrust per engine $= (T/W)_{to}\, W_{to} / N_{oe}$
$= 0.2053 \times 277175.0 / 4 = 14041.0$ kg.

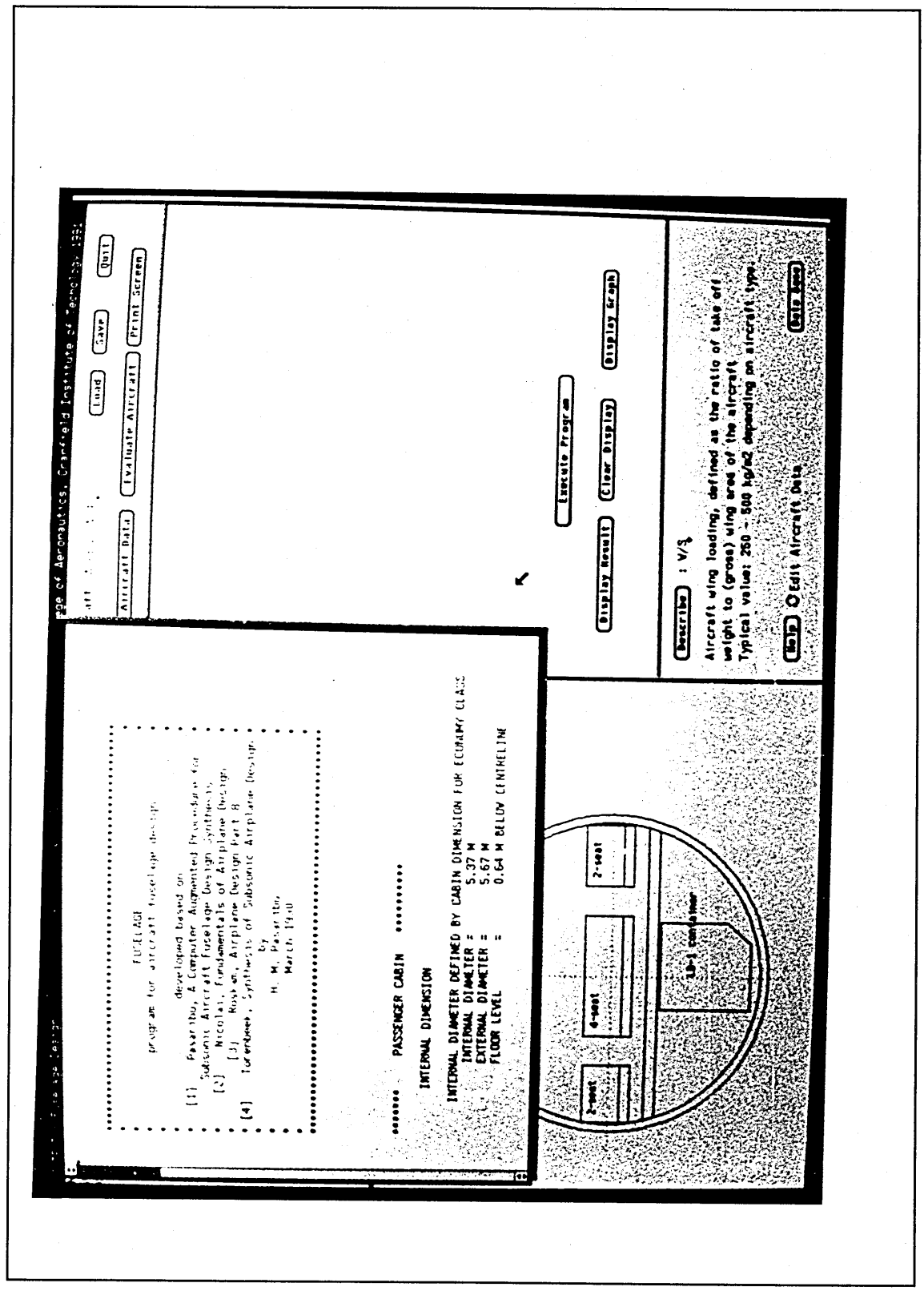


Fig. 6.3 The presentation of the output of the fuselage design module

6.3.6 WEIGHT ESTIMATION

The following result is obtained for the X-91 aircraft from this module :

	X-91	A340-300
Empty weight of the aircraft (kg.)	132064.3	126061.0
Take-off gross weight (kg.)	277013.8	255000.0

The detail of the output is given in Appendix G. The result of this module is again added into the database and replaces some of the data previously calculated.

6.3.7 AERODYNAMIC ANALYSIS

In addition to the data obtained from the database of the X-91 aircraft, the following data is modified in the editing mode :

Type of airfoil : Advanced airfoil

Two flight cases were analyzed, of which the flight conditions were specified in the execution mode :

Case No.	:	1	2
Mach number	:	0.8	0.82
Altitude (km.)	:	9.15	9.15
Aircraft weight (kg.)	:	264000.0	264200.0

At first a conventional type of airfoil was unawarely attempted, which resulted in an error and terminated the execution. The system message stated that the drag divergence Mach number limit was exceeded due to the thickness ratio of the airfoil. Therefore it was decided to use advanced airfoils.

The following drag equation was obtained for the first case :

$$C_D = 0.0166 + 0.0429\, C_L^2$$

and replaced the corresponding data in the database. The detail of the output is given in Appendix G.

6.3.8 FLIGHT ANALYSIS

The Flight Analysis was performed to calculate the flight time and fuel weight for the design payload. In addition to the data retrieved from the data base, the following data was modified in the editing mode :

Number of passenger	= 295
Payload weight (kg.)	= 27639 kg. (295 passenger plus baggage)
Range (km.)	= 13805
Altitude for diversion (km.)	= 7.0
Engine Name	= General Electric CF-6 (default engine)

The result obtained from the execution is as follows :

Take-off weight (kg.)	= 269894.22
Total fuel weight (incl. reserve, kg.)	= 110190.90
Block fuel (kg.)	= 99881.75
Flight time (hour)	= 16.27
Average speed (km/h)	= 848.48
Block time (hour)	= 16.71
Block speed (km/h)	= 826.71

These data were added into the database for use in calculating the direct operating cost. The detail of the result is given in Appendix G.

6.3.9 COST ESTIMATION

This module was suggested to calculate the cost estimate for the X-91 aircraft. Again in addition to the data obtained from the database, the following data were specified in the editing mode :

Active control system	= installed
Centre of gravity control	= installed
Cost per engine	= $ 2 million
Avionics system cost	= $ 5 million
Active control system	= $ 1 million
Fuel price ($/gallon)	= 0.57
Passenger load factor	= 0.65
Annual utility (hour)	= 3200
No of aircraft used for tests	= 6
Rate of production during development phase	= 0.5 aircraft per month
Expected no of aircraft produced	= 700
Rate of production during production phase	= 4 aircraft per month
Interest rate for financial loans	= 10.0 %
Profit margin	= 10.0 %

Most of these data were obtained from the database of existing aircraft. The market for the A340 and A330 project was forecast by the Airbus Industries to be around 1000. Due to the difficulty in identifying the market share between the two aircraft, it was decided to set the number to 700 for the A340. This is almost the same as that of the Airbus A320. Most of these data are required to estimate the production and

development costs, from which the purchase price of the aircraft is calculated.

Since no cost data were available for the X-91, it was decided to calculate all the cost related data in this module. This was done by informing the control in the executing mode to calculate all elements of cost. The following result were obtained :

Purchase price ($)	= 115 million (rounded off)
Direct operating cost ($/hour)	= 6044.20
Maintenance cost ($/hour)	= 545.52
Maintenance burden ($/hour)	= 572.79
Total maintenance cost ($/hour)	= 1118.313

Apart from the purchase price, which is about $ 105 million per aircraft for the A340, the rest of the results cannot be directly compared. This is partly because the data has not been established for the A340 since the aircraft has not been in operation yet. Furthermore the operating costs of an aircraft is very much dependent upon the operators. Nevertheless, this calculation was intended to obtain the first estimate of the various aircraft costs. The detail of the result is presented in Appendix G.

6.4 SUMMARY

The following shows the summary of the results obtained for the X-91 aircraft :

	X-91	A340-300
Wing Position	Low wing	Low wing
Engine Type	4 Turbofans	4 Turbofans
Tail Layout	Low tail	Low tail
Fuselage Diameter (m.)	5.67	5.64
Fuselage Length (m.)	58.29	63.65
Maximum no. of passengers	360	375
Container type(s)	LD-1	LD-1 and LD-3
Passenger door type(s)	3 TYPE A 1 TYPE I	3 TYPE A 1 TYPE I
Take-off weight (kg.)	269894.2	255000.0
Operating empty weight (kg.)	132064.3	126061.0
Aircraft price (US $, 1991)	115 Million	105 Million

CHAPTER 7
DISCUSSIONS

A prototype for the aircraft configuration design system has been developed and implemented. The system incorporates basic features and capabilities which are considered as those required for carrying design tasks in the configuration development phase. It includes a configuration analysis module that allows the user to have access to the line of reasoning behind the selection of the configuration layout of aircraft. This module can be used to generate the configuration of the baseline aircraft based upon the requirements.

The system can be used in the conceptual and preliminary design of subsonic civil transport aircraft, including propeller driven and jet engined aircraft. The case studies shows that the analyses give a good accuracy not only in the reasoning that lead to the configuration choices, but also in the subsequent numerical computations.

Much effort had been given to building an extensive database of design knowledge for the configuration analysis. The representation technique selected, using the structures knowledge source and rulesets, proved adequate in containing and describing knowledge. The advantage of this representation had been shown that it could handle situations where information is incomplete. Furthermore, it allows the extension of the knowledge base with small modifications to the system. These modifications are related to the procedures required for some of the rule predicates as explained in Appendix C. An attempt had been made to completely separate the knowledge base from the system by treating the procedures as stand-alone programs. This idea worked reasonably well for simple problems, but the performance was a bit suspect for large programs. Further work needs to be done to overcome the problem.

The parametric analysis module provided a platform for the system to initiate the design analysis. The variation of the important parameters obviously provided an insight into the sensitivity of the design to the variation which allows the designer to identify the critical aspects of the design.

The idea of separating the Knowledge-Based System module and the engineering modules is a compromise. In one hand, they both can be developed independently to achieve maximum flexibility and extensibility. Since all modules are independent, they can be modified and updated to include new development without disturbing the balance. The integration is done through an intelligent interface which oversees the whole process of execution of each module. On the other hand, operational performance is slightly degraded. Data interpretation and explanation of the performance of engineering modules is not as efficient as if they were integrated together in a single source code. On balance, considering the size of the problem that the system had to deal with, the flexibility and the extensibility of the separated system was preferable.

The interface approach described in Chapter 5 was employed in integrating the individual application modules. This proved to be very efficient in that it does not compromise the independence of any modules and strongly supports modularity. The modules can be developed separately to perform discipline-specific analyses. Since each module is independent, in which it can perform a complete analysis in its own, it can be modified and new programs can be introduced more easily to support future extensibility.

Combined with an extensive common database the interface approach provides a high level of flexibility in which various modules can be ordered into various sequences. The result of one module is input to the others.

At the moment only one program is allocated for each module. Although this is somewhat restricting, it is adequate for the present stage of development. However, the selection of programs used in a particular module is still subject to modification, so is the program itself. The interface can be relaxed to incorporate several programs for each module. These programs can be developed with modularity from the beginning.

Some of the application programs in the engineering modules, although sufficiently accurate at this level, are too simple for representing the activities in the modules. This, in fact, resulted from a compromise of where the emphasis of the research work should be placed. Developing or modifying an extensive program for each module would have required an enormous amount of time and effort. This would definitely have resulted in less time allocated for experimenting with the more important issues, e.g interfacing and devising an efficient problem solving strategy.

Some of the problems stem from the fact that many of the engineering programs were not readily available. Some required a slight modification for use in the system. This was because most of these programs were written for other computer systems, some even developed in computer languages not supported in the College's Sun microsystems on which the prototype was developed.

Providing an easy means of transporting the programs from one computer machine to the other was an important issue. This was another reason of the selection of C/C++ as the basic language for development. C/C++ is very much the same in any machines that not even small modifications are required.

The graphics capability of the system and the database management module are probably the areas in which the system's performance can be greatly enhanced. At the moment the graphics capability is provided by specifically designed routines which utilise the C library functions. A link with an established Computer Aided Drafting (CAD), i.e. the Unigraphics and Ideas, will be an immediate possibility.

The system uses a simplified, special-purpose database management module. Although the system operates reasonably well with this database module, using a professional database management system will surely enhance the interface with other modules in

general and the CAD system in particular. A relational database system should be considered in the future to improve the retrieval and storage of data.

A post processor for each individual engineering module may be required. The purpose of this processor is to analyze the result of the execution and make judgment and advice as to what to do with the result. This can become a major problem area in which the Knowledge-Based System can be applied. The problem of this approach is that the processor becomes too dependent upon the software itself. In other words, each external program included requires its own post-processor facility.

The development of such-post processor may be easier if the program is developed with such an idea from the beginning. It can then be structured in a way that allows this consultation to be carried out efficiently during the execution.

This necessity can probably be reduced to minimum if the processor is designed to tackle the problems in each design module in general, rather than tailored specifically for a particular program. This can be done by compiling a database of possible problems that are encountered in each design module. The database is supplied with all possible solutions for each problem. Each time a problem is encountered the system will search the database for that particular problem and report the corresponding advices for solving the problem. The system can then display all the advices for the user to make proper adjustments before re-executing the program. This technique has been attempted for the fuselage design module alone with promising results. It is however still subject to further refinement.

CHAPTER 8
CONCLUSIONS AND RECOMMENDATIONS

8.1 CONCLUSIONS

An effort has been made to develop a computer system for configuration design synthesis of subsonic transport aircraft. The system, called the Aircraft Configuration Design System, includes the configuration definition phase which is not included in the current computer systems used in aircraft design. The system combines the reasoning process behind the selection of configuration layout of aircraft with the computational aspects of design. It provides the user with the line of reasoning behind its selection of configuration choices.

The system utilises a new technique in representing the configuration design knowledge by using specially defined structures Knowledge Source and Rulesets. These structures, which couple a rule-based knowledge representation and object-oriented modelling, is an efficient way of producing a knowledge representation that could handle procedural and heuristic reasoning. The object-oriented modelling enables the knowledge to be expanded more easily.

The system is supported by engineering analytical modules. These are a library of application programs separated into several design modules. The modules are separated from the Knowledge-Based System module. This idea is a compromise to achieve high modularity and extensibility with minimum loss of performance quality. This was felt to be the correct approach in this case.

The integration between various modules is carried out through an interface program. An interface approach is a powerful means of integrating the design activities. It does not require linking two different source codes and communication of data through external functions. It permits the sharing of information between applications operating in potentially different environments. The application remain discipline-specific and modular, yet they are able to directly contribute to the global goal of design optimisation. The use of interface does not affect application programs. This modularity allows new programs to be introduced and old ones to be modified without disturbing the existing applications.

The use of control files and a common database was found to be a simple and efficient way to communicate data between various modules. It allows easier extension of database which may be required if new programs are added into the system.

These features have been shown as significant improvements in the way computer programs are structured to help aircraft designers in carrying out design tasks. The configuration analysis module included in the system is a valuable addition to the aircraft design activities. The knowledge structures used has provided a new technique

with which design knowledge can be represented and stored in computer codes.

8.2 RECOMMENDATIONS

Some recommendations for future development have been identified and discussed. The following summaries a few areas in which future work can be done to enhance the capability of the system :

- The knowledge base need to be extended to cover design aspects in the engineering modules in order to improve the quality of the consultation.

- Integration with an established Computer Aided System (CAD), e.g the Unigraphics and Ideas which are available in the College need to be investigated. This may entail the use of a professional relational database management system to ease the interfacing.

- Further development and incorporation of the application programs need to done to enhance the capability of the engineering analyses, and to keep up with new technologies.

ACKNOWLEDGMENTS

The author wishes to express his appreciation to many persons who have contributed to the completion of this thesis. I am indebted to my supervisor, Dr. J.P. Fielding, for his patience and advice through out the period of this study project.

Special thanks are due to my colleague and special friend, Shai Macci, for the moral support that has sustained me through the difficult periods and the invaluable time he set aside, out of his normal working hour, for correcting the thesis and taking the photographs. I am also grateful to Dr. Les Oswald, who has given general and technical assistance by setting up the computer network, and to my colleagues, Dr. Jaz Saggu and Lamine Darbouche, for their critical views on the presentation of the thesis.

Finally, I would like to express my gratitude to Sergio Ponzi, Laurent Dala and Arne Egholm, who have made my life at Cranfield such a wonderful experience. Their friendship, perseverance and the moral support they have given me have been invaluable.

REFERENCES

1 RAYMER, D.P. — Aircraft Design : A Conceptual Approach. AIAA Education Series. Washington, D.C. American Institute of Aeronautics and Astronautics, Inc. 1989.

2 TORENBEEK, E. — Synthesis of Subsonic Airplane Design. Delft University Press. 1982.

3 PASARIBU, H.M. — Applying Knowledge Based System Techniques to Aircraft Configuration Design. In: 'Knowledge Based Systems in Engineering' Workshop, The British Computer Society (BCS) Specialist Group in Expert Systems. College of Aeronautics, Cranfield Institute of Technology. 1990.

4 KROO, I.M
TAKAI, M — A Quasi-procedural, Knowledge Based System for Aircraft Design. AIAA, AHS and ASEE, Aircraft Design, Systems and Operations Meeting. Atlanta, GA. September 7-9, 1988. AIAA Paper No. 88-4428, 1988.

5 KOLB, M.A. — A Flexible Computer Aid for Conceptual Design Based on Constraint Propagation and Computer Modelling. AIAA, AHS and ASEE, Aircraft Design, Systems and Operations Meeting. Atlanta, GA. September 7-9, 1988. AIAA Paper No. 88-4427, 1988.

6 MARINAPOULOS, S, et.al. — Compromise : An Effective Approach for Conceptual Aircraft Design. AIAA, AHS and ASEE, Aircraft Design, Systems and Operations Meeting. St. Louis, Missouri. September 14-16, 1987. AIAA Paper No. 87-2965, 1987.

7 MISTREE, F, et. al. — The Design of Aircraft Using the Decision Support Problem Technique. NASA Contractor Report 4314. NASA Scientific and Technical Information Division. 1988.

8 ALSINA, J. — Development of an Aircraft Design Expert System. College of Aeronautics, Cranfield Institute of Technology, England. PhD Thesis, April 1988.

10 ALSINA, J, et. al. Progress Towards An Aircraft Design Expert System. In: Computer Applications in Aircraft Design and Operation; Proceedings of the First Internal Conference on Computer Aided Design, Manufacturing and Operations in the Aeronautics and Space Industries. Paris, France. June 16-18, 1987. Billerica, MA. Computational Mechanics Publications, 1987. pp.229-247.

10 BARGETTO, R., MAZZETTI, B. GARBOLINO, G. Aircraft Configuration Analysis/Synthesis Expert System: A New Approach to Preliminary Sizing of Combat Aircraft. In: Conference of the International Council of Aeronautical Sciences, 16th. ICAS Proceedings 1988. ICAS-88-1.11.2, pp. 1645-1649.

11 YOELI, R. The Use of Artificial Intelligence Methods in the Conceptual Design of Light, And Aerial-Application Aircraft. In: Israel Annual Conference on Aviation and Astronautics, 28th. Tel Aviv and Haifa, Israel. February 19-20, 1986. Collection of Papers, Technion - Israel Institute of Technology, pp.23-28.

12 WINSTON, P.H. Artificial Intelligence. 2nd ed.
Reading, Mass. Addison Wesley, 1984.

13 RICH, E. Artificial Intelligence.
Singapore. McGraw-Hill Publishing Co, 1983.

14 MEHTA, U. Knowledge Based Systems for Computational Aerodynamics and Fluid Dynamics. In : J.S. Kowalik (ed.). Knowledge Based Problem Solving. Engelwood Cliffs, New Jersey. Prentice-Hall. 1988. pp. 183-212.

15 PREMACK, T. A Rule-Based Computer Aided Design System.
NASA Technical Memorandum. NASA TM-86241. Oct. 1986.

16 PANG, G.K.H. MACFARLANE, A.G.J An Expert System Approach to Computer Aided Design of Multivariable System. Lecture Notes in Control and Information Sciences. Vol. 89. Springer Verlag. 1987.

17 JARDINE, T.J. A Machinability Knowledge Based System. In : J.S. Kowalik (ed.). Knowledge Based Problem Solving. Engelwood Cliffs, New Jersey. Prentice-Hall. 1988.

REFERENCES

pp. 257-269.

18 ROSKAM, J. — Aircraft Design : Where Does It Stand ?. American Institute of Aeronautics and Astronautics. Aerospace America, September 1991. pp. 26-29.

19 SOBIESZCZANSKI-SOBIESKI, J., et.al — Aerospace Engineering Design by Systematic Decomposition and Multilevel Optimisation. Washington, D.C. NASA Technical Memorandum TM-85823. June 1984.

20 CLOCKSIN, W.F. MELLISH, C.S. — Programming in Prolog. Springer Verlag, 1981.

21 GOLDBERG, A ROBSON, D. — Smalltalk-80 : The Language and Its Implementation. Menlo Park : Addison-Wesley Publishing Co., 1983.

22 ANONYMOUS. — Turbo-Prolog : User Guide. Version 2.0. Borland International, 1988.

23 ANONYMOUS. — Smalltalk/V : Tutorial and Programming Handbook. Digitalk, Inc. 1986.

24 WIENER, R.S. PINSTON, L.J. — Introduction to Object-Oriented Programming and C++. Reading, Mass. Addison-Wesley. 1988.

25 COX, B.J. — Object-Oriented Programming : an Evolutionary Approach. Reading, Mass. Addison-Wesley, 1986.

26 LANGE, R.H. — A Review of Unconventional Aircraft Design Concepts. In: Congress of the International Council of the Aeronautical Sciences, 15th. ICAS Proceedings 1986. London, England. September 7-12, 1986. ICAS-86-2.2.1. pp.191-200.

27 GEE, J. E. — The Role of Interfaces in Design Integration. AIAA/AHS/ASEE Aircraft Design, Systems and Operations Conference. Paper AIAA-89-2133. Seattle, WA. July 31 - August 2, 1989.

28 ANDERSON, B. L. — First Step Toward Integrating the Design Process. AIAA/AHS/ASEE Aircraft Design, Systems and Operations Conference. Paper AIAA-88-4403. Atlanta, GA. September 7-9, 1988.

29 SLIWA, S.M.
ARBUCKLE, D.P.
OPDOT : A Computer Program for the Optimum Preliminary Design of a Transport Airplane. NASA TM 81857. September 1980.

30 PASARIBU, H.M.
OPDOT version 2.0 : A Computer Program for the Optimum Preliminary Design of Transport Aircraft. College of Aeronautics, Cranfield. User Guide. 1989.

31 POLHAMUS, E.C.
HALLISSY, J.M. jr.
Effects of Airplane Configuration on Static Stability at Subsonic and Transonic Speeds. NACA Research Memorandum RML56A09a, 1956.

32 ANDERSON, S.B.
A Look at Handling Qualities of Canard Configurations. Journal of Guidance, Control and Dynamics, vol. 10. March-April 1987. pp.129-139.

33 ARBUCKLE, D.P.
SLIWA, S.M.
Parametric Study of Critical Constraints for a Canard Configured Medium Range Transport Using Conceptual Design Optimisation. AIAA Atmospheric Flight Mechanics Conference. AIAA-83-2141. 1983.

34 GOLDSTEIN, S.E.
COMBS, C.P.
Trimmed Drag and Maximum Flight Efficiency of Aft-Tail and Canard Configurations. AIAA, Aerospace Science Meeting, 12th. Washington, D.C. Jan. 30- Feb. 1, 1974. AIAA Paper 74-69.

35 ROKHSAZ, K.
SELBERG, B.
Analytical Study of Three-Surface Lifting Systems. SAE Paper 850866. SAE, 1985.

36 KENDALL, E.R.
The Minimum Induced Drag, Longitudinal Trim and Static Longitudinal Stability of Two-Surface and Three-Surface Airplanes. AIAA Paper 84-2164. AIAA, 1984.

37 PUTNAM, L.E.
Effects of Aft-Fuselage-Mounted Nacelles on the Subsonic Longitudinal Aerodynamic Characteristics of a Twin-Turbojet Airplane. NASA TN D-3781.

38 PASARIBU, H.M.
A Computer Augmented Procedure for Fuselage Design Synthesis of Subsonic Transport Aircraft. College of Aeronautics, Cranfield Institute of Technology. MSc Thesis, September 1988.

39 FEAGIN, R.C. et al.
The Delta Method, An Empirical Drag Buildup

Technique. NASA Conference Report CR-151971.

40 FROSST, G.D. Development of a CAD System for Initial Aircraft Design Purposes. College of Aeronautics, Cranfield Institute of Technology. MSc Thesis, 1981

41 HARTLAND, A.J. Dispersed Base Combat Aircraft. College of Aeronautics, Cranfield Institute of Technology. PhD Thesis. August 1989.

42 BLOOMER, J. DELTA Program : User Guide. College of Aeronautics, Cranfield Institute of Technology, 1990.

43 PASARIBU, H.M. ACCOST : A Computer Program for Aircraft Cost Estimation. College of Aeronautics, Cranfield Institute of Technology. April 1990.

44 MADDALON, D.V. Estimating Airline Operating Cost. NASA Technical Memorandum TM 78694, May 1978.

45 PASARIBU, H.M. FLTANAL : A Computer Program for Flight Path Analysis of Subsonic Transport Aircraft. College of Aeronautics, Cranfield Institute of Technology. June 1990.

46 MITCHELL, C.G.B. A Computer Programme to Predict the Stability and Control Characteristics of Subsonic Aircraft. RAE Technical Report 73079. September 1973.

47 THEOPHILOU, M.K. Development of Computer Programs for Initial Project Design of Combat Aircraft. College of Aeronautics, Cranfield Institute of Technology. MSc Thesis. 1983.

48 TORENBEEK, E. The Initial Calculation of Range and Mission Fuel During Conceptual Design. Technische Hogeschool Delft. Afdeling del Luchvaart-en Ruimtevaarttechniek, The Netherlands. Report TH/LR-525. 1987.

49 LOFTIN, L.K. Subsonic Aircraft : Evolution and the Matching of Size to Performance. Washington, D.C. NASA Reference Publication. NASA RP 1060. 1980

50 TORENBEEK, E. Fundamentals of Conceptual Design Optimisation of

Subsonic Aircraft. Department of Aerospace Engineering, Delft University of Technology, The Netherlands. Report LR-292. August, 1980.

51 TORENBEEK, E. On Conceptual Design of Subsonic Transport Aircraft for Cruising Flight Optimised for Different Merit Functions. Delft University of Technology. Report LR-451. 1985.

52 NICOLAI, L.M. Fundamentals of Aircraft Design. Dayton, Ohio, 1976.

53 ROSKAM, J. Airplane Design, Parts I-VIII. Roskam Aviation and Engineering Co, Kansas, 1990.

54 LAMBERT, L (ed.) Jane's All The World's Aircraft. 1991-1992 Edition. Jane's Information Group, 1991.

55 SALGUERO, D.E. A Design Synthesis Program for Business Jet Aircraft. AIAA, Annual Meeting and Technical Display, 12th. Washington, D.C. January 28-30, 1976. AIAA Paper No. 76-216.

56 JENKINSON, L.R.
SIMOS, D. A Computer Program for Assisting in the Preliminary Design of Twin-Engined, Propeller Driven Aircraft. Canadian Aeronautics and Space Journal, vol. 30, no. 3, Sept 1977.

57 SIMOS, D.
JENKINSON, L.R. Optimization of the Conceptual Design and Mission Profiles of Short-Haul Aircraft. Journal of Aircraft, vol. 25, no. 7, pp. 618-624. July 1988.

58 PIGGOT, B.A.M.
TAYLOR, B.E. Application of Numerical Optimisation Techniques to the Preliminary Design of a Transport Aircraft. Royal Aircraft Establishment (RAE) Technical Report 71074, April 1971.

59 SERGHIDES, V.C. Design Synthesis for Canard-Delta Combat Aircraft. College of Aeronautics. Cranfield Institute of Technology. PhD Thesis. 1988.

60 TURNBULL, A. Design Synthesis and Optimisation of Unmanned Aircraft. College of Aeronautics, Cranfield Institute of Technology. PhD Thesis. 1990.

61 KEHAYAS, N. ASTOVL Combat Aircraft Design Synthesis and

Optimization. College of Aeronautics, Cranfield Institute of Technology. 1991.

62 BIL, C. Applications of Computer Aided Engineering to Subsonic Aircraft Design in a University Environment. Proceedings of the 15th Conference of the International Council of Aeronautical Sciences, London, Sept. 7-12, 1986. ICAS-86-3.1.1, pp. 108-118.

63 BIL, C. Development and Application of A Computer-Based System for Conceptual Aircraft Design. Delft, The Netherlands. Delft University Press, 1988.

64 HABERLAND, C., THORBECK, J., FENSKE, W. A Computer Augmented Procedure for Commercial Aircraft Preliminary Design and Optimization. Proceedings of the 14th Conference of the International Council of Aeronautical Sciences, ICAS-84-4.8.1. pp. 943-953. 1984.

65 WALLACE, R.E. A Computerised System for the Preliminary Design of Commercial Airplanes.
AIAA, Aircraft Design, Flight Test, and Operations Meeting, 4th. Los Angeles, CA. Aug. 7-9, 1972. AIAA Paper No. 72-793, 1972.

66 GREGORY, T. J. Computerized Preliminary Design at the Early Stages of Vehicle Definition.
AGARD CP-141, vol. 1, paper 6, 1973.

67 ELIAS, A. Knowledge Engineering of the Aircraft Design Process.
In: J.S. Kowalik (ed.) Knowledge Based Problem Solving. Engelwood Cliffs, New Jersey. Prentice Hall, 1986. pp. 213-256.

68 BRACHMAN, R.J. et.al. What Are Expert Systems ?. In: Hayes-Roth, F, Waterman, D.A. and Lenat, D.B (eds). *Building Expert Systems*. Reading, Mass. Addison-Wesley Publishing Company, 1983. pp. 31-57.

69 HAYES-ROTH, F., et.al. An Overview of Expert Systems. In: Hayes-Roth, F, Waterman, D.A. and Lenat, D.B (eds). *Building Expert Systems*. Reading, Mass. Addison-Wesley Publishing Company, 1983. pp. 1-30.

70 JACKSON, P.
Introduction to Expert Systems.
International Computer Science Series. Wokingham, England. Addison-Wesley Publishing Co. 1986.

71 RINGLAND, G.A.
DUCE, D.A (eds.)
Approaches to Knowledge Representation: An Introduction. Letchworth, Research Studies Press. Knowledge Based and Expert Systems Series. 1988.

72 STROUSTRUP, B.
The C++ Programming Language. Reading, Mass. Addison-Wesley Publishing Co. 1987.

73 GRIFFITHS, M.
PALISSIER, C.
Algorithmic Methods for Artificial Intelligence. London, Kogan Page, Ltd. 1987.

74 ANONYMOUS.
Sun C++ Programmer's Guide : AT&T C++ Language System Release Notes.
Sun Microsystems. Part Number 800-5173-10. Revision A of 22 February, 1991.

75 ANONYMOUS.
SunView Programmer's Guide.
Sun Microsystems. Part Number 800-1783-11. Revision A of 27 March, 1990.

76 ANONYMOUS.
SunView System Programmer's Guide : Pixrect Reference.
Sun Microsystems. Part Number 800-1784-11. Revision A of 27 March, 1990.

77 ANONYMOUS.
Programmer's Language Guide : C Programmer's Guide.
Sun Microsystems. Part Number 800-3844-10. Revision A of 27 March, 1990.

APPENDICES

APPENDIX A
DESCRIPTION AND APPRAISAL OF EXISTING AIRCRAFT DESIGN COMPUTER PROGRAMS

A.1 INTRODUCTION

For the last two decades much attention has been given to developing computerised procedures for aircraft design synthesis and analysis. Such programs offer integrated design tools that provide the user with many choices of design tasks, ranging from a simple design analysis, parametric and trade off studies, sensitivity analysis up to a more complex optimisation of multi design variables.

Many advantages have been claimed for such programs as compared to a manually performed design. The most interesting one is the huge saving in computing time and cost. The growing increase in computer memory and computing capabilities has led to a considerable decrease in time and cost of running such programs. Moreover, the programmers have fewer restrictions to the limitation of program size. Many independent analytical and synthesis routines which used to be executed separately due to the limitation in memory and cpu time can now be incorporated into an integrated design system. Combined with menu-displaying capabilities, the programs now offer a more friendly way to the user to apply such programs in a practical design case. They also provide a greater degree of freedom for selecting various options from the menu as desired in a particular case study without necessarily giving too many commands which were sometimes quite a tedious and annoying part of executing the programs.

The time saving claimed for using such programs in a practical design looks more apparent if one takes into account the man power required in and the development time involved in a manually performed design study of a specific aircraft. The iterative nature of aircraft design process and the large number of design disciplines involved make the process of defining and sizing various aircraft parameters related to geometry, performance, aerodynamics and propulsion system a tedious task. The more parameters to be considered, the longer the time required; and moreover, the more complex the problem would be. The parametric variation and trade off studies would consequently become more difficult, if not impossible, to handle manually.

In many cases, the final design of a particular aircraft evolves from many designs of different configurations by various teams. While the design parameters and characteristics are different, the methods used to evaluate all those different designs are practically the same; and it would be unrealistic if all the calculations have to be manually duplicated and repeated many times by different people. The time involved would be enormous.

The considerable decrease in computing time gained by applying such programs offers

a further advantage. More design parameters can be analyzed in a period of time and trade off studies of different parameters and design characteristics can be performed more quickly. This would be of great importance as a more global optimum design can be achieved. As the computing capability of digital computers grows more and more, the accuracy of the calculation increases considerably. Furthermore, greater number of different designs can be analyzed simultaneously to obtain the best design of all. The designers are no longer restricted by the limitation of the available resources to evaluate many of those designs and are more encouraged to generate more ideas and explore more unconventional concepts of design.

A number of such computerised design tools are available. They range from a much simpler version specifically tailored to the design of a particular type of aircraft (GATEP, BIZJET) to the most sophisticated ones dedicated to various types of aircraft including civil and military (ACSYNT). Many of these programs are dedicated only to the initial sizing and first degree trade off study, while some covers the whole range of the preliminary design stages including configuration development and detailed design (ACSYNT). It has to be admitted, however, that many of the advantages listed earlier are less than claimed because of the simplicity of the engineering analysis involved, especially in the simpler version of the design tools.

The following section will discuss the features and capabilities of several computerised design tools. The discussion covers the assumption and design philosophy implied, the range of applicability and the strengths and weaknesses of each program. The design systems discussed are :

CONVENTIONAL PROGRAMS :

1. BIZJET
2. GATEP
3. CASTOR
4. ADAS
5. CAPDA
6. MVO
7. CPDS
8. ACSYNT

KNOWLEDGE-BASED PROGRAMS :

9. PASS
10. RUBBER AIRPLANE
11. DSIDES
12. ADROIT
13. ACES

A.2 CONVENTIONAL DESIGN SYNTHESIS PROGRAMS

With the word 'conventional', the author means to stress that most of these programs are written in conventional programming languages such as FORTRAN. They also

apply a conventional technique of programming i.e. procedural approach.

A.2.1 BIZJET

Contrary to the sophisticated approach of the existing computerised tools developed in many aircraft manufacturers, BIZJET program [55] employs a simplified design process. As implied in its name, this synthesis program is applicable only for business jet aircraft.

The simplest among the available design tools, the program which was developed from a similar program published by NASA for general aviation is aimed as an educational tool for teaching aircraft design. Six different modules : geometry, aerodynamics, propulsion, performance, weight and economics are included to form an iterative procedure for sizing the aircraft. Without neglecting its accuracy, the prediction methods employed are of a simple form. The relationships between various disciplines, however, are represented up to the second degree of importance to enable the user to realistically conduct a design study and parametric analysis.

This program can only analyze and evaluate a fixed configuration, low wing and fuselage-mounted jet engined aircraft. Although this configuration represents most of the existing business jet aircraft, its application is considerably limited. Furthermore, it prevents the user from generating different ideas and utilising a somewhat "unconventional" configuration. Nevertheless, its application as a complement to the lecture at class rooms for educational purposes cannot be underestimated. At least in a university environment the availability of such an integrated program can be of great importance for students to conduct parametric analysis of multi variables in order to understand the process of design, and to perform a sensitivity analysis of various design characteristics in order to get a better sense of judgment on various design decision and a better "feel" for correct values of various design parameters.

A.2.2 GATEP (General Aviation Twin Engines, Propeller driven aircraft)

As the name suggests, this program [56] is applicable only for twin engined, propeller driven aircraft. Also implied in the name, the program can only analyze small capacity airplanes, although applications to large turboprops of the size of, say, Fokker 50 are also envisaged.

This program which was developed in the Loughborough University of Technology, UK, is powerful in its application in various ways :

- To compare individual designs either conceptual or existing aircraft. This particular study can be useful for an aircraft manufacturer to obtain the best compromise of design or to the customers/airlines who want to conduct an engineering analysis of existing aircraft and select the best type for its own application.
- To conduct a parametric study around a baseline design in order to determine the

sensitivity of the aircraft characteristics to design parameters.
- To assess the impact of advanced technological features by introducing reduction factors in weight and drag terms.

The program is built in modular way in which all the calculation routines are written separately. All the modules are linked together by a master program, through which the user can control the execution.

The flexibility of the program is greatly enhanced by allowing the user to select the method for calculating a specific parameter or overriding several prediction routines when better data are available. However, the restriction of the type and the number engines used greatly limits its application in a wider range.

A.2.3 CASTOR (Commuter Aircraft Synthesis and Trajectory Optimisation Routine)

Realizing the limited capability of the GATEP program, the Loughborough University of Technology moved on to improve the program to include flight profile analysis. CASTOR [57] integrates both preliminary design optimisation in GATEP and flight profile optimisation in SCOPE (Short-haul Commuter Optimum Profile Evaluation) which was previously developed.

Although this program offers a greater capability, analysis is still confined to twin turboprop aircraft of conventional layout (aft tail, wing mounted engines), covering the whole spectrum of turboprop aircraft, from 6 to over 50 seat designs.

The incorporation of a multivariate optimisation (MVO) [58] which was developed at the Royal Aircraft Establishment (RAE) enables the user to conduct an optimisation study of up to 10 design and 12 flight profile parameters simultaneously. This provides the user a greater degree of freedom to conduct various studies of different parameters.

To further enhance the capability of the program, the flight profile analysis considers the three important segments of flight : climb, cruise and descent, each of different variables. Furthermore, a design with a capability of multi-stage mission of shorter ranges can be analyzed. The considerable increase in computing time, however, prevents the user from conducting an analysis of a full flight profile optimised for each single stage of a multi-stage flight.

2.4 MVO (Multi Variate Optimisation Program)

The role of optimisation technique in aircraft design process has been realised for a number of years. It speeds up the process of converging the design variables within the constraints of design requirements. This helps the designer to obtain the best values of aircraft parameters for a set of design criteria.

There are many optimisation programs available for various number of applications, one of which is the MVO [58] which was developed by the Royal Aircraft Establishment in 1971 to specifically cater for aircraft design synthesis. This program was originally written for transport aircraft of conventional design with rear-mounted engines and wings of moderate sweepback, which was then the most common configuration. Later a further work was carried out to include configuration with wing-mounted engines as this configuration became increasingly more popular. However, the application of the program is still limited to high subsonic transport, even though the core of the program i.e the optimisation module, can be applied to other applications by reformulating the mathematical model of the problem to be optimised.

One of the advantages of using MVO is the wider range of selection of design variables and objective functions (i.e the quantities to be optimised). At present around 18 design quantities representing geometry, flight profile, engine and field performance parameters are included in the design variables that form the mathematical model of the design. Several other quantities such as fuselage dimension and payload are treated as parameters which are held constant during the optimisation process.

A number of programs has been devoloped based on MVO. These programs are tailored for specific military aircraft. A few examples are swept, canard fighters [59], unmanned aircraft [60], and advanced short take-off and vertical landing (ASTOVL) [61].

The problem of using MVO is the creation of the mathematical model of the design, i.e. the selection of the set of design variables and parameters, the selection and formulation of the objective functions as a function of design variables, and the selection and treatment of design constraints. The mathematical model must not only accurately represent the real design characteristics but also be simple so that it can still be processed by the optimisation module. As accuracy sometimes implies complexity, much more time may be spent not in the optimisation process but in the computation of the objective function. On the other hand, a simple model may degrade the value and usefulness of such program in practical design. This conflicting requirement calls for an effective optimisation technique such as the MVO in order to obtain an optimum design which reasonably resembles the real design process.

The other problem is the formulation of design objective and constraints as a function of design variables and parameters. As the objective function and constraints can be different for different types of design, the reformulation process should be as easy as possible in order to avoid the need for complete modification of the mathematical model of the problem. The philosophy of the program should be clearly understood in order to aid the designer to decide on the need for such a reformulation and to actually conduct it.

The selection of constraints can impose another problem in that the user must decide which requirements are treated as inequality constraints and which as equalities. This selection can alter the way the optimisation is performed as the equality constraints are always actively involved in the process while the inequalities are only active when they

are violated.

MVO solves a constrained problem by treating equality constraints in one of two ways: formulating some of the design variables involved in the constraints as a functions of the rest of variables and thereby reducing the number of variables, or taking a projection step at the current design point along the tangent plane to the surface of the constraint which obviously violates the constraint and then restoring the design point back to the surface of the constraint. The inequality constraints are treated in either one of the following : introducing the constraints with penalty factors to the objective function, or using the projection-restoration method as in the equality constraints. The first method may not give an improved objective function in the early stage due to the influence of the constraints but as the optimisation cycles continue the objective function becomes more significant. The second method has the advantage that the optimisation process is always applied directly to the objective function rather than the combination of the objective function and penalty functions and that constraints are only considered when there is a possibility of them being violated. However this method tends to slow down the convergence process.

A.2.5 ADAS (Aircraft Design and Analysis System)

Similar to the idea of BIZJET program, the Delft University of Technology, the Netherlands, has developed a computerised design system [62,63]. Unlike the previous programs, ADAS covers the whole spectrum of subsonic transport aircraft, ranging from small turboprops up to large jet transports of the Boeing 747 size. The system was designed to be flexible in tackling various design problems with little modification. The user is expected to supply analytical procedures to be used within the system.

The ADAS main program control the dialogue between the user and the system through a command-oriented language. Through this program, the user can perform various functions : to invoke programs stored in the program library, to plot graphs, to edit database, etc.

The ADAS system, in principle, does not contain any design knowledge. This information is supplied by the user in the form of a FORTRAN program, referred to as the analysis program. This program can be defined for a specific design problem. This obviously requires of the user some basic experience of FORTRAN programming. Some of the commonly used analytical routines, however, have been encoded and stored in the program library. The user can then select some of these routines and incorporate them in the analysis program.

The interface between the ADAS main program with these analysis programs is done through an executive program called ADAP (Aircraft Design and Analysis Program). The ADAP program is a non-interactive program. Once it executes the selected analysis program, the control does not return to the system until the execution is completed. The ADAP program, however, can be executed in a parametric survey or optimisation

mode, varying the selected design parameters within a specified range.

ADAS is linked with MEDUSA, a computer aided drafting (CAD) and modelling system with a two-way interface. This enables the system to display the geometry and shape of the design in a form of three-view drawing, and to plot graphs obtained from the analysis.

No design decision is made within the analysis. This approach is deliberately chosen to prevent misinterpretation of the results. Design decision is the responsibility of the user. The system only perform analysis according to the user's command.

A.2.6 CAPDA (Computer Aided Preliminary Design of Aircraft)

CAPDA [64] which was developed by a team of the Berlin Technical University follows similar approach as ADAS to the aircraft design process. Most options that can be carried out in ADAS such as iterative design synthesis, parametric and sensitivity studies, multivariate optimisation and design analysis can also be performed in CAPDA. In many features both programs bear a considerable similarity. They both make extensive use of statistical database of existing aircraft. The database includes geometry, engine, weight and performance data. The graphic display offers not only the three view drawing of the airplane being designed but also various graphs or charts resulted from the parametric and sensitivity analysis. This feature is of great advantage for the users since they can directly review the result and make necessary adjustment to the design afterwards.

Unlike ADAS, however, CAPDA offers a greater flexibility in which the designer is encouraged to shape the design object according to his creativity and demands by means of a set of numerical tools. It also utilises its own CAD system.

One important feature of CAPDA is that its philosophy matches that of a manually performed design, especially the procedure of design tasks, the configuration concept and the selection of various design variables and constraints. This resemblance is however only limited to the sequence of the design procedure. Since the design process is not unique, the logic of the sequence should ideally be able to be altered according to the user's intuition. This important feature however is not contained in CAPDA. The program also requires the user's judgment and experience as many assumptions are contained in the prediction methods selected. Its application is therefore limited to experienced users.

Like ADAS, CAPDA is quite powerful in the later stage of the preliminary design process, in which accurate design analysis and synthesis need to be carried out accurately to provide a reasonable result relevant to the real aircraft. At this design stages many design parameters have been established and many design decisions have been assumed without prior knowledge of the user. The system lacks the configuration design phases where the selection of particular design component and its relative

location and many of these decisions are analyzed and evaluated.

A.2.7 CPDS (Computerised Preliminary Design Synthesis)

CPDS [65] is a very advanced system for a complete design of subsonic and transonic commercial aircraft. The system was developed to support the need for a rigorous engineering analysis in a practical design study in Boeing Commercial Aircraft Company.

The program was built in modules which consist of two design blocks of different complexity and accuracy. The first level enables the user to carry out the design process from scratch where only little information about the design is available. As the design progresses more data are accumulated, which are then used for further analysis. The first level employs a nonlinear problem solving analysis to define the flight mission, design concept and estimates of design parameters. The mathematical model of the aircraft is represented in the form of a set of equations and inequalities.

The first level analysis includes the aerodynamics of the aircraft in all flight conditions, off-design performance analysis, DOC analysis, statistical weight and balance, thermodynamically estimated engine performance and noise investigation. The synthesis results in a schematic two-view drawing of the feasible aircraft and its interior layout, performance charts, and load and balance diagram.

The second level of design employs a more accurate analytical method for component weight determination based on structural and aeroelasticity analysis. The dynamic behaviour of the elastic aircraft, flutter analysis and a more detailed analysis of all design disciplines are also included.

Both levels of design are integrated so that it is possible to return to level I analysis for additional sizing process if some of the analysis in level II are unsatisfactory.

CPDS is unique in the way it is built. It is actually not a single computer program, but rather a set of computerised methodologies by which the user can compile an independent program relevant to their engineering problem. With this concept CPDS is more flexible and open-ended, since it can accept all stand-alone computer programs as long as they have compatible data flow.

The problem with this concept is the difficulty in developing the stand-alone analytical program compatible with the rest of the system. The user must be familiar with the module before he/she can modify or construct program solution logic that controls the compatibility.

A.2.8 ACSYNT (Aircraft Synthesis)

ACSYNT [66] is probably the most sophisticated and complete computerised design synthesis available at the moment. The approach employed in this program emphasizes the purpose of preliminary design studies which is to provide technical and economic feasibility information which is useful for guiding larger efforts during the detailed design phases.

Unlike most of the programs described earlier, which tend to stick to the conventional approach, ACSYNT provides a more imaginative design approach. It enables the user to adequately explore many design approaches suggested during the early definition of the aircraft concepts and to select for further study those concepts that indicate significant cost saving. This feature however requires a vast amount of computer memory and computing time since adequate resources required to investigate those various concepts should be included in the system. The advantage is that many promising concepts which used to be discarded prematurely in the previous design systems due to the unavailability of comparable analytical methods can now be investigated.

Like CPDS, ACSYNT offers multi level design synthesis and analysis. It provides methods of providing information and suggestions for defining adequate levels of resources to accomplish the design objectives. The user can select the design level to be used according to the amount of information available at any design stage. This gives the user the freedom to carry out the design process from an early stage where much design information that controls the later stage of the design is established. It also enables the user to describe criteria for acceptance of early design information.

Another prominent difference between ACSYNT and previous systems is that it also includes several levels of structural analytical methods in addition to aerodynamics and flight profile analysis. This enables the user to perform the whole range of preliminary design process from configuration definition to preliminary structural analysis.

However, this program is likely to require reprogramming an additional code for each new application not incorporated in the original program. The incorporated design methods and their respective input and output are pre-selected and cannot easily modified by the user.

A.2.9 CONCLUSION

Based on the survey of the existing programs, an effective computerised conceptual and preliminary design definition programs should have the following characteristics :

- Modular :
  The functions to be performed are clearly separated and can be assembled to perform various sequences of computation depending on the problem.

- Integration :
  A control is required to exercise the individual modules and to transfer the output for additional computations. It should be possible to carry out direct analytical computations for each individual module in a particular discipline and to develop a scaling information for sensitivity studies. These scaling relationships are usually simplified by omitting parameters of secondary importance.

- Multi level :
  The programs should be capable of providing calculations at different levels of detail.

- Sensitivity analysis :
  This is necessary to carry out assessment of critical areas in the design. The sensitivity defines the feasibility of the design to meet a specified performance requirement.

- Optimisation and trade off studies :
  The program should provide a facility which enable the user to select the best concept or performance.

Many of these features have been incorporated in the existing systems up to a certain degree of success. In addition to these features, the future systems need to include the following characteristics :

- Transparency :
  The user should be able to retain full monitoring capabilities of what the computer is doing and extract an explanation from the computer if necessary.

- Extensibility :
  The representation of the geometric and mathematical model of the aircraft should be able to be updated continuously in conjunction with the real life development of the design process.

- Flexibility :
  Flexibility covers the way the design process is carried out. The sequence of various design tasks and the logic behind it should be able to be altered according to the problem involved and the data available.

All these additional features are difficult to incorporate using conventional programming languages. Although several attempts have been made on the existing programs to include those features, they are not as successful as with the other features. The inclusion of these two features using conventional programming languages tends to require a vast amount of computer memory since each computational path should be encoded into the system. This would increase the computing time considerably since the computer has to search through the routines to find the appropriate path before executing it. Furthermore the response would tend be slower, which overrides the usefulness of the features.

APPENDIX A

A primary feature of any new design system should be unprecedented extensibility in both the set of variables used by the program and the set of analytical routines. This particular requirement is especially important if the system is to be used to assess new technologies.

To support this requirement, such system normally incorporates a high degree of modularity to reduce complexity. This is done by separating the interface and database management functions from the computational routines.

As conventional programming languages are procedural, the flexibility of the program is greatly limited. Different procedures would require additional routines which add to the amount of computer memory required. Furthermore, experience shows that updating an existing program is a rather difficult task, especially if the programmers who update it are different from the ones who originally created it.

A.3 THE USE OF ARTIFICIAL INTELLIGENCE TECHNIQUES IN THE AIRCRAFT DESIGN SYNTHESIS PROGRAMS

The problems facing the programmers to include the transparency and flexibility of the programs due to the limitation of conventional programming languages have led to several attempts to explore the use of Artificial Intelligence (AI) techniques.

The aim is to simulate the design procedures in the manner of a manually performed design. The approach emphasizes an interactive design in which the computer is used to help the designers to make judgment and to carry out design tasks.

The following will discuss several systems that utilise AI techniques in aircraft design synthesis in one way or the other.

A.3.1 PASS (Program for Aircraft Synthesis Studies)

Making use of the advances in computer hardware and new operating system, Kroo and Takai [4] of Stanford University attempted new ideas for program architecture and the application of AI techniques to the aircraft design problem. The basic program is written in FORTRAN 77.

This work focused on the later stage of the preliminary design phase, i.e. the analysis. The system can perform the execution of several analytical modules in the appropriate order to permit the computation of the required variables that have to be computed, warns the user if the requested routine is unable to be executed because of unknown data and prompts the user to enter the required data.

Unlike conventional programming techniques, the program utilises a heuristic approach

rather than a procedural one. The system incorporate a routine that determines the appropriate computational path from the currently specified values to the desired result in a particular problem. An expert advice and warning system which communicates with the knowledge base can be consulted to provide design advice on how a problem might be corrected. This approach provides more flexibility and extensibility than a conventional fixed-input, fixed output procedure. It also permit the user to experiment with new ideas of design without any fear of run-time errors or wrong decisions on any design step since the system can detect such situations and warns the user to take the necessary action.

The drawback of this approach, however, lies on the way this consultation works. The advice and warning database are examined after each routine and if any problem is identified, the system displays a warning and can be requested to provide the corrective action. This may lead to a problem in which the user cannot anticipate the consequences or the probable problem of any decision at the earlier stage. The system works rather by trial and error, which could require more computational effort than necessary and could therefore lengthen the time required to arrive at a desirable design.

The program nevertheless has a clear educational value in its application as it has been used to teach about aircraft design, enabling the students to carry out trade-off and sensitivity studies at a far more comprehensive and sophisticated level than that could be done by hand.

A.3.2 RUBBER AIRPLANE

To increase flexibility of the computer aids for the detailed geometric design, Massachusetts Institute of Technology (MIT) combined object-oriented programming and constraint propagation with component-based database of design knowledge to form a prototype computer aid for conceptual design known as Rubber Airplane [5].

Object-oriented programming is a programming style which emphasises the description of objects (aircraft components in this case) and their behaviour, rather than the delineation of functions, subroutines and calling sequences.

Constraint propagation provides mechanisms for treating declarative statements of mathematical relationships as imperative statements. By interpreting each individual declarative statement as implying an underlying network of (imperative) mathematical relationships amongst all of the included variables, constraint propagation allows the computer to select the appropriate form, based upon the available information. The computer cannot then be confined to a fixed form of input-output but rather flexible that it can take a variable as input or output for each relationship according to whether the variable already has a value or not.

Rubber Airplane is based on Paper Airplane, a research by MIT on the use of constraint propagation to manage the computational aspects of conceptual design [67]. The program

is written in LISP. It allows the user to define a set of arbitrary design variables and design functions (i.e. the mathematical relationships among the design variables) in order to model the problem of interest.

Employing constraint propagation the program will then sequence the application of the design functions and perform whatever design function inversions are necessary to calculate values of the remaining design variables based on the set of initialised variables. This function-modelling approach is used as a base for component-modelling approach in the Rubber Airplane in which each aircraft component is described in its geometric variables and the mathematical relationship between those variables.

Rubber Airplane combines the component-modelling approach with the object-oriented programming by instantiating design components as "objects". Their attributes (i.e. parameters which describe the components) and their constraints (i.e the mathematical relationships between those parameters) are regarded as instance variables of the components with which they are associated.

Rubber Airplane also includes some means of applying constraints between the attributes of multiple components as well as between the attributes of a single component in order to obtain a successful integration between various components.

A simple graphical package is also included for displaying various three dimensional views of the components.

Rubber Airplane is at present stage only useful for preliminary design geometry definition, used for calculating the basic geometry of the concept aircraft and at the same time evaluating its performance, weight and balance and relative locations of the components.

A.3.3 DSIDES (Decision Support In the Design of Engineering Systems)

DSIDES is being developed in the Houston University, Texas as a new approach in engineering system design. The program utilises the Decision Support Problem (DSP) technique in order to improve the efficiency and the effectiveness of the decision making process in aircraft design [6,7]. The principle role of the DSP technique is to provide support for human decision making in design. It helps the designer to make the following type of decision :

- selection
- compromise
- hierarchical
- conditional.

Unlike the other programs which use AI techniques mainly to increase flexibility and transparency, DSIDES applies an intelligence design assistant to the design environment which is not necessarily an aircraft design problem. It helps the designer to identify the

problem, to formulate it within the boundary of the given requirements, to solve and then analyze the solution. It is in fact a tool for decision making in general rather than just a design tool. It emphasises on the problem solving process rather than just on methodology or procedures.

A.3.4 ADROIT (Aircraft Design by Regulation of Independent Tasks)

ADROIT has been developed in the College of Aeronautics, Cranfield Institute of Technology, England as an attempt to incorporate the advantage of expert system techniques in an aircraft design system [8,9]. The scope of the problem is limited to high subsonic civil transport aircraft wing design.

The design utilises three supercritical airfoil sections suitable for high subsonic speed applications, although the inclusion of other types of airfoil is also envisaged. Given the information related to general aircraft concept and fuselage design, the program carries out the design based upon specification and design variables obtained from a parametric study. The design begins with two dimensional analysis of wing profiles and progresses to include wing planform and three dimensional effects including drag rise, aeroelasticity and stall characteristics. Although the wing is mainly designed for cruise conditions, an allowance for take off and landing conditions is taken into account.

The framework utilised in ADROIT allows the user to carry out the design process according to available information. The sequence of the design tasks is flexible it that the design tasks can be performed randomly. The system, however, consults the logic of the design knowledge at any stage that the program can suggest the next design step(s) to be taken. The user can control the execution through the consultation facility.

The drawback of the system, however, lies on the capability of the consultation facility. Even though the program reports to the user what it is attempting to do no explanation can be exerted as to why and how a certain action is performed. The level of engineering analysis involved is also simplified, which renders the system unsuitable for applications in a practical design in industry. This simplified analysis was a deliberate policy, aimed to facilitate the development of this early prototype system.

ADROIT is however still under constant development. A similar work on fuselage design has been successfully completed [38]. A further work to include a conceptual design definition module and parametric analysis is under way.

A.3.5 ACES (Aircraft Configuration Expert System)

ACES [10] is currently under development by a joint team of Aeritalia and CSI Piemonte, Italy to help the designers define a set of good configurations meeting the requirements. It is aimed to aid the designer to make initial choices of each configuration element defined and perform some numerical computations of

aerodynamics, weight and performance in the preliminary design phase of military aircraft.

ACES automatically defines a set of possible configurations, from which, after a preliminary evaluation, the user can select the desirable concepts compliant with a set of requirements. Each of these concepts can then be evaluated in a more advanced trade off and optimisation, refined to obtain the best configuration. The inclusion of the configuration parameters in the design variables of the optimisation make this program different from the other systems previously mentioned. At the present stage, however, ACES can only perform some initial choices of configuration decisions such as engine type, the presence of absence of the afterburner, the type of air intake and by-pass ratio.

Implemented in LISP using expert system shell KEE (Knowledge Engineering Environment) of Intellicorp, ACES claims the advantage of being faster in defining the set of feasible configuration based on a consolidated knowledge database accumulated from previous experience. It can also be used to rapidly evaluate the effect of the requirement changes on the configuration. No information, however, has yet been obtained about how far these claims are validated and justified.

A.3.6 CONCLUSION

Based on this investigation several conclusions can be outlined :

1. There is still a need for an efficient tool that deals with the configuration definition process especially in commercial transport aircraft. The knowledge acquired in this process should be able to represent the real interaction between various configuration elements and help the designer to decide the possible configuration for certain design requirements.

   Most of the current systems are geared towards the sizing definition and less emphasis is on the conceptual configuration definition due to several reasons :

   - The process involved in the sizing process and detailed design is more procedural and structured,
   - The process contains more numerical computations which are easier to be encoded in a computer program,
   - The difficulty in acquiring and compiling the knowledge surrounding the configuration decisions. Some of the knowledge may be 'hard' information which is based on scientific principles but most of them are 'soft', based on designer's judgement and experience. In many cases there are many different solutions that can be applied to a particular configuration decision as many experts take different approaches. There are many measures of merit for judging the level of acceptance of a design decision; and all measures may not be equally important.

2. The need for a more rigorous engineering analysis in order to reasonably increase

the accuracy of the program. On these systems the emphasis has been on redesigning and restructuring the design process so that the expert system can be used. This entails simplifying the engineering modules so that they can be easily handled by the expert system and thereby reducing the usefulness of the program in a practical design.

One solution to this is to apply existing analytical and design modules in the system instead of creating a new module. The advantages of this approach are :

- More effort can be given to generating an effective inference engine and elaborate knowledge database,
- The system can be made more flexible in the long run as the analytical modules can be updated more easily to cover new developments.

There are two ways of implementing this approach :

1. The expert system acts as a design assistant or command spy which provides advice, comments as well as warnings, and watches over the design process (Fig. A.1). The expert system and design and analysis package act as two completely separate packages. The design and analysis package deals with numerical engineering analysis. The user has to use the package in conjunction with the expert system which processes only symbolic manipulation and acts as an interface between the two.

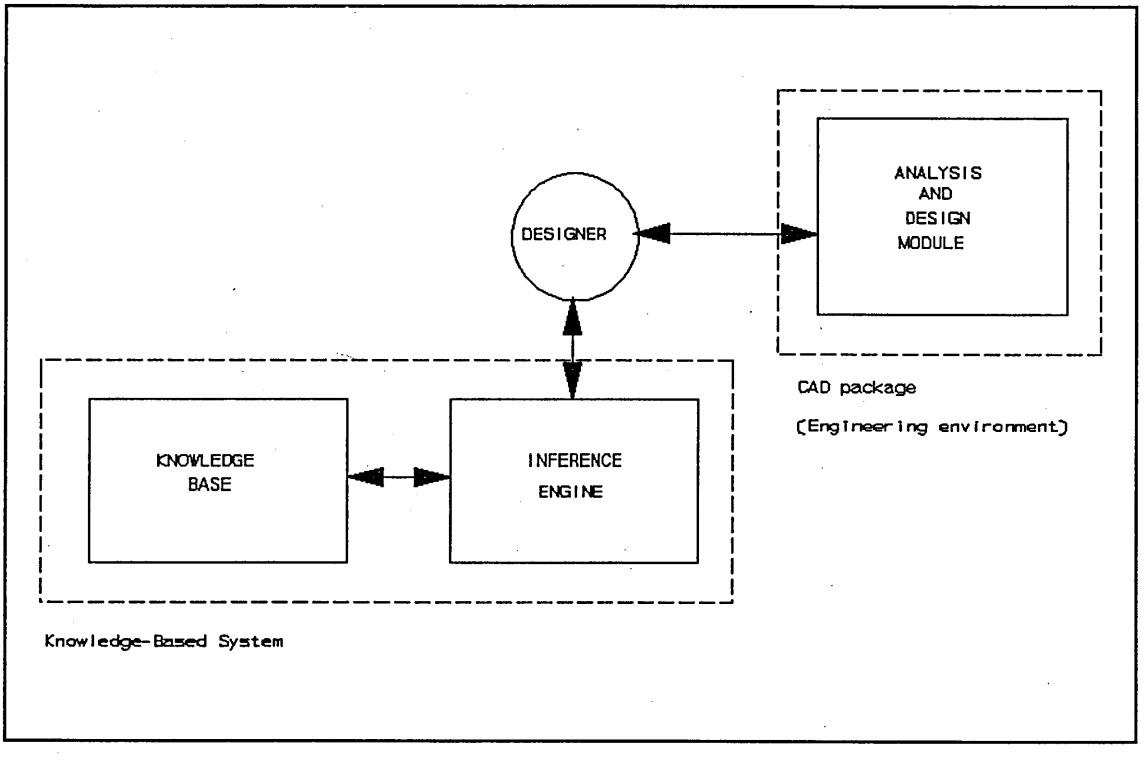


Fig. A.1 The knowledge-based system as design assistant

2. The expert system acts an intelligent front end which controls the entire engineering environment including facilities for graphic and numerical calculations (Fig. A.2). Here the engineering modules become part of the system and may need some modifications in order to share a common interface with the expert system. One way of doing this is to create intermediate files for passing the parameters between the two.

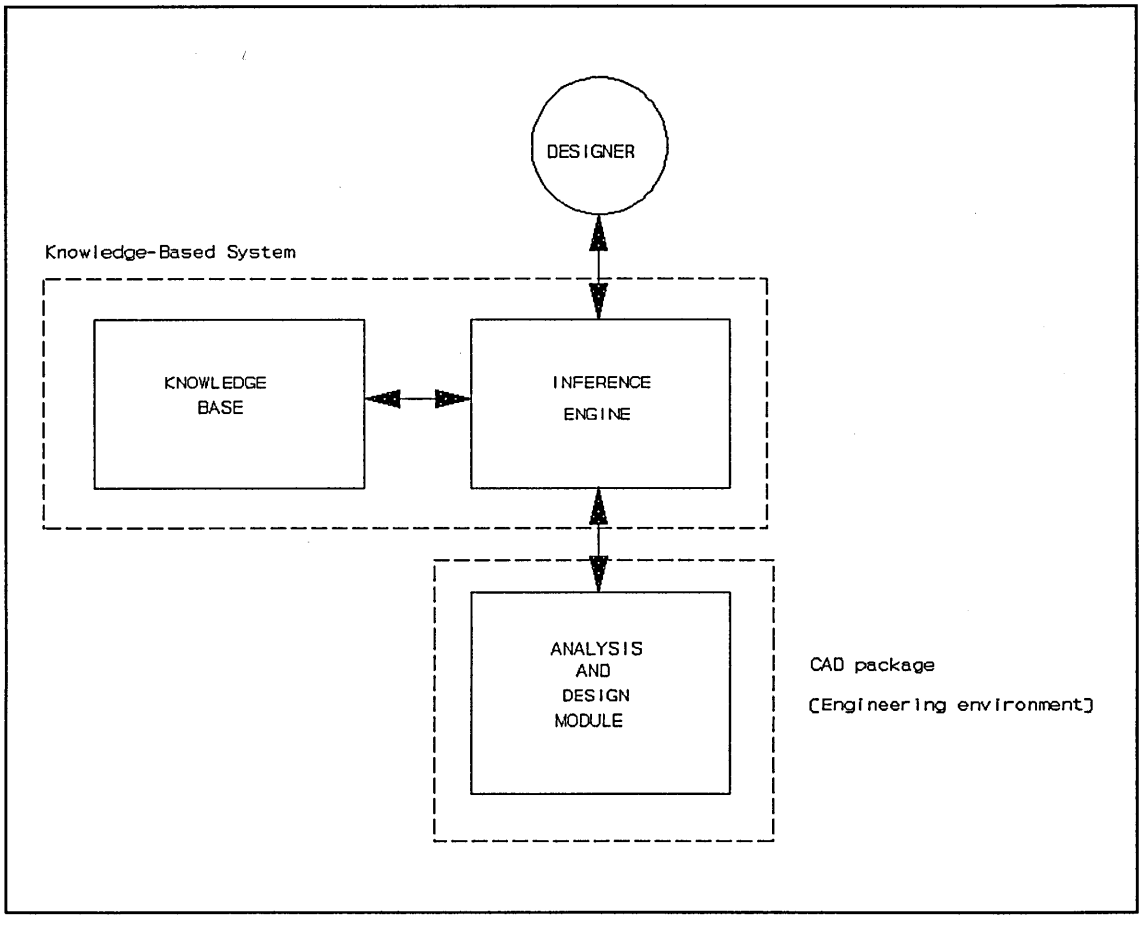


Fig. A.2 The knowledge-based system as an intelligent front end

The first concept looks more flexible than the second in that it suggests that the expert system may be developed more independently from the engineering modules. As the computer languages used in developing most engineering software are different from those used in expert systems, interfaces between the two systems may be easier to implement. The first concept however requires more effort from the user in that he/she has to understand the two systems.

The proposed system utilises the second technique with small modifications. A system controller, which lies between the user and the system, employs the inference engine to select a particular design module to execute. The system controller can then execute

an application program associated to the selected design module through the interface program. The interface program helps the user to prepare data for processing. It also updates the database with the result obtained from the execution.

APPENDIX A

APPENDIX B
DESCRIPTION AND APPRAISAL OF KNOWLEDGE BASED SYSTEM TOOLS, LANGUAGES AND TECHNIQUES

B.1 KNOWLEDGE BASED SYSTEMS

Knowledge based systems (KBS), commonly called expert systems, are computer systems that emulate the skills of an expert in a narrow area of domain using knowledge and reasoning to produce advice justifying its conclusion and (possibly) handling uncertainties.

To be effective, such a system has several independent dimensions [68]:

- Expertise
  The system should provide accurate judgment about the problem and fast performance. This requires a substantial amount of knowledge about the domain expertise which is organised for efficient search.

- Intelligence
  The system should contain general problem solving abilities in the domain. This intelligence depends strongly on the scope of the basic principles of knowledge and the quality of its general purpose reasoning processes. The former concerns about the amount of principles the system knows and the detail in which it knows them. The latter concerns about how the knowledge can be successfully applied to solve the problem.

- Reformulation
  The system should be able to transform a problem in some arbitrary initial form into the form appropriate for processing by the inference mechanisms.

- Abilities requiring reasoning about self
  The system should be able to trace its line of reasoning and understand it as if it were another task requiring expertise.

The structure of a knowledge based system [69] normally comprises of :

- User Interface
- Justifier
- Knowledge Base
- Inference Mechanism

as shown in Fig B.1.

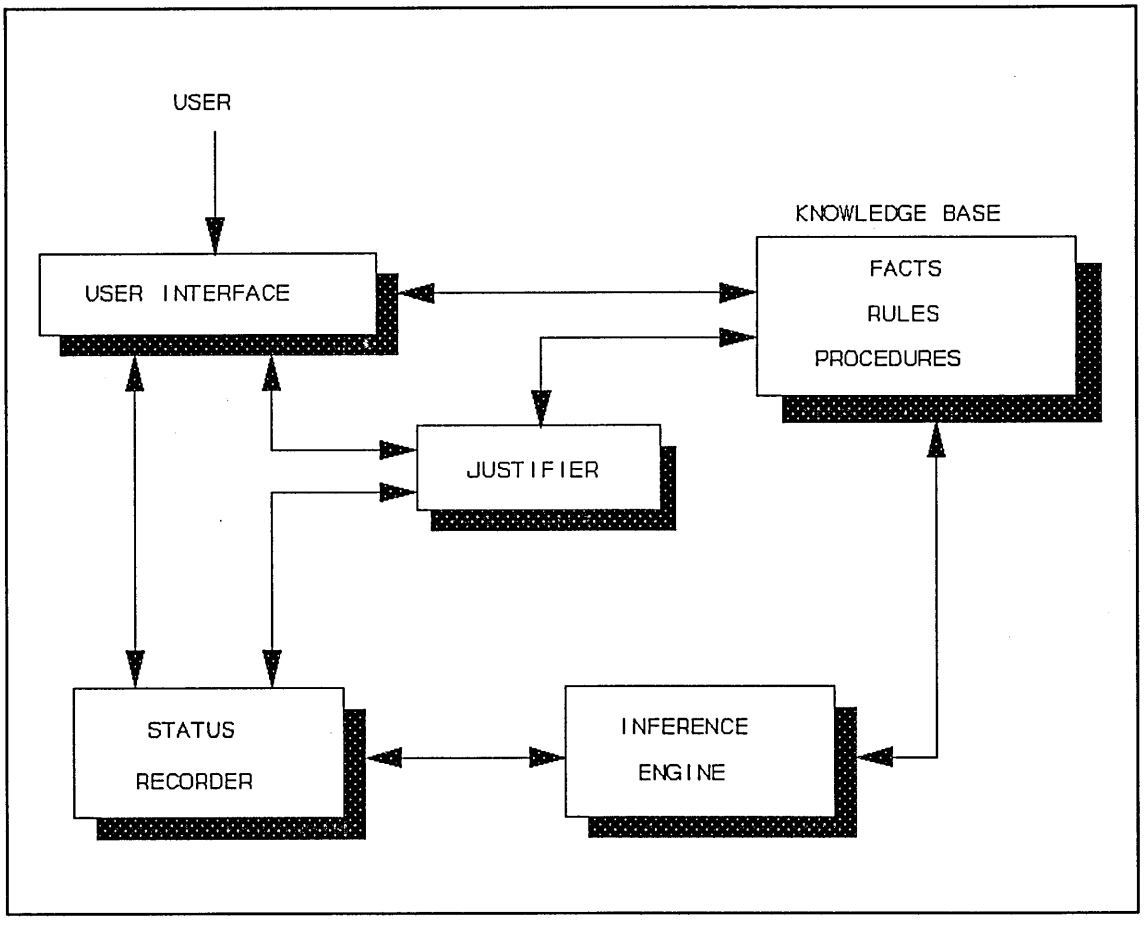


Fig. B.1 Components of a knowledge based system

The user interface can be in the form of a language processor that translates the code that is understood by the system into natural language and vice versa. The justifier explains the action of the system to the user. In general it answers questions about why some conclusion was reached or why certain alternative was rejected.

Knowledge base consists of a data base of facts or assertions about the problem domain and a set of rules or principles. The inference mechanisms control the problem solving activities. It determines which action should be executed and which strategy should be pursued to solve problems. It decides which rules correspond to the problem in question and validates them to obtain solutions.

B.2 KNOWLEDGE ACQUISITION AND REPRESENTATION

B.2.1 KNOWLEDGE ACQUISITION

The fundamental aspect of knowledge based systems is knowledge acquisition and representation. Knowledge refers to true beliefs or facts about the domain expertise.

One of the difficulties of developing a knowledge-based system in design is to acquire knowledge surrounding design decisions. In aircraft design in particular, and in other engineering problems in general, the process is governed by various aspects :

- The number of possible solutions to the problem is numerous
- There are many different approaches to the design depending upon the expert's experience, each approach may have several answers
- There are many criteria of acceptance of any design decisions, each criterion may not be equally important.

In this project the knowledge of aircraft design process was obtained from textbooks (Raymer [1], Torenbeek [2], Nicolai [52], Roskam [53]), the College of Aeronautics' lecture notes and various NASA reports; and interview/consultation with design experts.

The way the experts use their acquired knowledge is different depending upon their experience and the type of the problem. For design problems, facts and knowledge are evaluated to generate solutions compatible to the requirements. The search strategy for solution is therefore driven by the data available. This type of search is normally called *forward-chaining* or *data-driven* search. In other problems, such as diagnosis, the search for solution is best done through *backward-chaining* (*goal-directed*) search, in which solutions (goals) are assumed in the first place and then their premises are evaluated to see if the solutions are valid.

Another way of using knowledge in the quest for solutions is *generate-and-test* search. Here the available data is used to generate tentative solutions which may not be compatible to the requirements and then the intermediate results are evaluated to see the consistency of the solutions.

B.2.2 KNOWLEDGE REPRESENTATION

Knowledge representation means manipulating a large amount of expert, heuristic knowledge into a stylized form that a program can use to help solve difficult problems. It is concerned with the way in which information might be stored in the human brain, and the (possibly analogous) ways in which large bodies of knowledge can conveniently be stored in data structures for the purposes of symbolic computations.

Knowledge representation has been defined as 'a set of syntactic and semantic conventions that make it possible to describe things' [12]. The syntax of a representation

specifies a set of rules for combining symbols and arrangements of symbols to form expressions in the representation language. The semantics of a representation specify how expressions so constructed should be interpreted, i.e. how meaning can be derived from form.

Knowledge representation implies some semantic way of codifying what an expert knows about some domain. It implies organisation, i.e. how knowledge is stored in the database. Knowledge should be accessible and easy to apply via more or less natural mechanisms to make it effective. For this reason the store of knowledge should be addressed by content, rather than by location.

There are many ways of how knowledge can be represented. The type of representation depends on the problem or the subject to which the knowledge is applied. The main criteria for assessing a representation are [70, 71]:

- logical adequacy
- heuristic power
- notational convenience

Logical adequacy means that the formalism should be capable of expressing the knowledge it represents. Heuristic power means that as well as having an expressive language with a well-defined syntax and semantics, there must be some means for using representation so constructed and interpreted to solve problems. Notational convenience is a virtue because most knowledge based systems applications require the encoding of substantial amounts of knowledge, and this task will not be an enviable one if the conventions of the representation language are too complicated. The expressions should be easy to write and to read, and it should be possible to understand their meaning without knowing how the computer will actually interpret them.

Three different formalism have found favour with the knowledge based system's designers :

- production rules
- predicate logic
- structured objects

B.2.1.1 Production Rule

Most early and conventional knowledge-based systems used the familiar production rule representation. The basic building block of the knowledge-based system is IF-THEN rules.

The advantages of using the production rule representation are :

- It allows for a uniform representation of knowledge

- It represents naturally occurring chunks of knowledge
- It allows for incremental growth of the knowledge base. New rules can be easily added.
- It represents knowledge in a declarative manner. Interaction among the rules may lead to unplanned and interesting results.

The disadvantages are as follows :

- It is an inefficient way of representing knowledge and results in little flexibility to handle the research especially if the knowledge base becomes large. Although production rules seem well suited to encoding empirical associations between situation and actions, they appear to be less effective as a means of expressing more subtle forms of knowledge which can be used to reason about the fundamental nature and causes of interesting phenomena
- The expressive power is inadequate for representing concepts and relationship among objects. The fact that representing knowledge as a disorderly and unstructured set of rules has a certain problem which, taken together, probably outweigh the often mentioned advantage that one can easily add another rule to the set and let the conflict resolution sort out when it should fire
- It is difficult to see the consequences of adding a new rule to the system. It may lead to an undesirable interaction and result in the knowledge base containing contradictory and circular rules. Even in the case where the rules do make true categorical statements about the domain, there is no guarantee that the program will perform in the way it is expected.

Production rules typically do forward chaining; i.e. matching data in working memory against the left-hand side of the rules and then executing the right-hand side.

B.2.2.2 Predicate Logic

Predicate logic has gained an increasing recognition among knowledge-based system programmers. Facts are represented by using predicate with arguments. The predicate represents the relationship between its arguments. The application of this approach in *Prolog* (see section B.3.2) proved to be quite successful since the facts can be easily written in predicate logic. The drawback of this approach, however, lies on the fact that every fact must be uniquely defined and hence requiring a large amount of memory. For this reason this type of representation may not be adequate for design problems due to the problem of having to encode a huge amount of predicates. Furthermore, it is suspect to handle uncertainties or incomplete knowledge.

The search for a solution is executed as follows : find all predicates whose arguments are matched, select one of them, and call its conclusions.

B.2.2.3 Structured Objects

Another approach is to represent knowledge by structured objects, such as *semantic networks* and *frames*. The knowledge base is typically a hierarchical network of objects, concepts and events. The frame-like (or node-and-link) structure represents their interrelationships. The use of networks avoids the exhaustive search that may arise in the use of a rule-based representation. It also allows for a very concise, neat and clear way of representing knowledge.

In a frame-based approach, knowledge is represented in functions like tables or questionnaires in which all of the information in a given category is grouped together. A frame is a generic data structure containing any desired number of categories of information called slots, where this information is associated with the subject of the frame. Each frame defines a semi-independent body of knowledge which can be both procedural and declarative. Frames may be linked together to form a hierarchical classification of domain knowledge and allow for inheritance. Each frame can be referred to by its attribute in a manner similar to the concept of a header in a list.

This frame-based representation can be very powerful for many knowledge-based system development tools since it can be dynamically linked to other frames, thus representing the hierarchical knowledge about a problem. It is also popular because it suits the structure of knowledge in general. Much knowledge has a structure, either coming from the structure in the subject domain to be represented and/or the structure imposed for the purpose of dealing successfully with a large amount of knowledge. Other advantages of this representation are as follows :

- The knowledge base of the system is extremely modular. This provides a natural way of representing components of expertise. Also, the knowledge base can be maintained more easily
- Each frame can represent appropriate knowledge as default values
- Each frame is an object which represents an independent body of knowledge
- Once the domain expertise has been organised into frames, it is relatively simple to represent the procedural aspects as rules within the knowledge base. This facility is useful in engineering applications, since procedure-oriented computation can be included quite easily by defining a slot, called *procedure-slot*, which contains procedures that may call other procedures. In this way the program can be organised as sets of nested, recursive subroutines or functions
- Frames can be defined as specialisations of more general frames, leading to a hierarchical classification of the domain knowledge. For example, wing, horizontal tail and canard can be said as member of lifting surfaces, therefore they can have lifting surface as their parent frame
- Frames can be linked together to have inheritance relationships. This is an important aspect for dealing with incomplete knowledge, since a specific frame can inherit all the attributes of its parent frame. Therefore, if no information is available for a slot in a single frame, the information obtained from the parent frame can be assumed. In the previous example, wing, tail and canard have similar

function of generating lift, although for different purposes, and they all share similar geometric parameters

- Uncertainty in the design process calls for flexibility and this can be provided if the frames are semi-independent
- The flexibility of the frames allows the experienced designer to vary the sequence in which the frames are used and therefore provides him/her with a more powerful structure for handling unanticipated types of design problem.

B.3 LANGUAGES AND TOOLS FOR BUILDING KNOWLEDGE BASED SYSTEMS

Knowledge based system tools are computer software that can be used to help develop a KBS application. They may be computer languages, such as Fortran, Prolog, Smalltalk, C, and knowledge based system environments which are widely available. KBS environments are normally equipped with at least one knowledge representation approach and a general problem solving procedure. In a way they are like computer languages with built-in capabilities for developing knowledge base, inference mechanism and user interface for a particular application.

Using KBS environments in developing an application may give an advantageous edge as compared to using languages, since these environments are developed with built-in problem solving capabilities and a wide range of knowledge representation. Therefore the potential user only needs to worry about acquiring knowledge and structuring them in certain types of representation acceptable for the environment. Prototyping is then made much easier.

Using environments, however, has disadvantages, since the representation is normally restricted to the ones that are embedded in the system. Furthermore, the inference of knowledge is quite standard, and may not necessarily be suitable for application in the subject of interest.

Another drawback stems from the fact that most environments have limited numerical computing capability which is a severe limitation in engineering applications. They proved to be powerful for symbolic manipulation, for which they are developed, but may not be suitable to handle procedures or complex numerical problems. Some environments, such as Nexpert Object and Leonardo, which do have these facilities are quite expensive to purchase.

For these reasons, it was decided to build the prototype by using a computer language. A number of languages which are potential for this application are discussed.

B.3.1 FORTRAN

Fortran is the programming language considered by many to be successor to LISP and Prolog for Artificial Intelligence (AI) research. Its advantages include :

- It is very efficient for numerical computation (many AI programming rely heavily on number-crunching techniques)
- AI problems tend to be very poorly structured, meaning that control needs to move frequently from one part of a program to another. Fortran provides a special mechanism for achieving this, the so-called GOTO statement
- Fortran provides a very efficient data structure, the *array*, which is particularly useful if, for example, one wishes to process a collection of English sentences each of which has the same length.

B.3.2 PROLOG

Prolog is a simple but powerful and practical programming language based on the idea of programming in logic [20]. Prolog programs may be viewed as logical clauses and the interpreter as an efficient resolution theorem prover. Prolog may be looked on as an extension of LISP in that it provides as primitives *pattern directed procedure invocation* and *non-determinism* (backtracking). It provides general recursive (tree-like) data structures which are accessed by *pattern-matching* rather than by explicit selector functions. There are no destructive operations on these data structures, but structures may contain empty slots (uninstantiated logical variables) which can be filled in later. There is also an *assertional database* which is used for relatively long-lived or permanent data.

Prolog is a good language for a rule-based system. Its built-in backward-chaining inference and automatic backtracking capability are useful aspects for a rapid search through the knowledge base. The problem with this is that the mechanism is often so rigid and powerful that it makes programmers feel like the program is uncontrollable.

Its similarity to rule-based systems can be unnecessarily limiting. It assumes that the format of rules in the knowledge base be the same as that of Prolog's rules, i.e Horn clauses. It assumes that Prolog's own inference mechanism be used and this is not always necessary for certain applications in which other search strategy may serve better.

Despite these, Prolog is increasingly used in many applications, especially in prototyping. It is powerful for symbolic manipulation, for example in building natural language processors, and proves quite extensible as a programming language in that it can implement other language as well. The procedure for calling external programs is quite straight forward and this facility is a good advantage for many engineering applications.

However, due to its rule-based approach, programs written in Prolog tend to be large in terms of memory size since each rule has to be encoded into the program. A few versions of Prolog, however, have an extensive database management capability, which enables the knowledge base to be written separately from the inference procedure and user interface modules. The linkage is done at run time, in which only a certain chunk of the knowledge base that relates to the problem in question is loaded into memory at any one time. This way of programming will greatly reduce the memory requirement for processing even though the performance will be considerably slower. This however requires that the knowledge to be represented in a tree-like structure to enable the system to identify the part of the knowledge base that needs to be loaded.

The ease of calling external programs seems to be overshadowed by the requirement for large memory space. This is partly because of the way this calling sequence is carried out, in which the part of the program which is currently running when the calling is executed has to be stored in memory stack so that it can be recalled later when the calling sequence is completed. This means some of the memory useable for processing is 'lost' for storage, and only the remaining memory can be used for executing the external programs. This considerably limits the size of the external programs that can be executed. Part of this memory problem stems from the fact that at the College of Aeronautics Prolog compiler is only available in Personal Computer (PC) network in which only 640 Kbytes Random Access Memory (RAM) is available. Even though the RAM is extended to about 900 Kbytes the 256 Kbytes extended memory is not normally accessible for processing. Moreover, implementing routines written in other languages does not look as straight forward as it seemed to be. For these reasons, it was decided not to use Prolog as the basic language for prototyping.

B.3.3 SMALLTALK

Smalltalk is a programming environment developed by members of the Learning Research Group at the Xerox Palo Alto Research Centre. A number of editions of Smalltalk have emerged over the past few years, starting with Smalltalk-72, which appeared around 1976, and culminating with Smalltalk-80 [21]. The fundamental philosophy of Smalltalk is developed from *Simula*, and although the different editions differ in syntax, all share the view of a programming system as a collection of (active) objects, communicating by *passing messages*. In addition, Smalltalk adopted Simula's "class" concept, and has extended and refined it considerably.

The power of the Smalltalk system arises mainly from the modularity enforced by the packaging of declarative and procedural knowledge into individual objects. If the services of an object are required, for instance, to access the value of a certain location in an array, a message is sent to the object which owns the array from the object which is interested in accessing it. The message will specify all pertinent information regarding the request, and will generate a response which should contain the value of the requested location.

Like Prolog, Smalltalk can call external routines quite easily and it can process procedures or rules written in a Prolog version modified to suit Smalltalk syntax. These facilities are quite useful in engineering applications since combining procedures of programs written in numerically-oriented languages such as Fortran with symbolic manipulation of Prolog can be made easily. A representation similar to frames with their hierarchical structure and inheritance can be structured quite easily.

However, this tends to be limited by memory space, especially because of the way of programming in Smalltalk environment which needs large amount of memory. Unlike programming in other languages in which stand-alone programs can be made easily, programming in Smalltalk actually modifies the Smalltalk environment itself.

B.3.4 C AND C++

C is a middle-level computer language because it combines elements of high-level languages such as Prolog and Fortran with the functionalism of the assembler. Among all the computer languages, C holds a unique place - people either love it or detest it to distraction.

One of the strong points of C language is its *portability* - the C programs can be easily transported across different machines without any significant changes. This portability stems from C's use of function libraries for such machine-dependent operations as input/output. C is simple in that there are only a few *keywords* involved. On the other hand, C is richer in operators than most languages.

C offers the programmers access to machine level with a blend of efficiency and elegance, but sometimes the conciseness of the language encourages a cryptic cleverness that hinders maintainability.

C++ is very nearly a proper superset of C language, with a few modest incompatibilities [24,72]. Its enhancements include the ability to define new data types (classes), operations on those types (operators and functions), and a comprehensive set of ways to control operations on these types, including operator overloading, constructors and destructors.

C++ evolved to meet the following goals :

- retain the extremely high machine-efficiency and portability that C is famous for,
- retain compatibility between C++ and C,
- repair long-standing flaws, particularly C's lax treatment of data types. C has long been criticized for its weak type-checking even inside a given function, and no type checking across functions even inside the same file,
- upgrade C in line with modern data-hiding principles.

The advantage of using C/C++ is mainly long-term, especially if maintainability and

extensibility of the system is a dominant factor as requested in Chapter 1. As the system develops, the size of the system is becoming a difficult problem to be dealt with. Although most Fortran compilers are the least likely to face this problem of increasing size as compared to most languages considered, the complexity of programming is a main stumbling block. No matter how powerful the IF-THEN-ELSE and GOTO statements of Fortran are in dealing with problems of different cases, they are still not fast enough in their operation. The computational path of each case has to be encoded into the system and the computer has to search through all the possibilities before executing the path of the case that matches. C's *switch* statement and the Prolog's backtracking are the more efficient facilities in dealing with such problems. Although C/C++ is not as powerful as Fortran in dealing with large programs, it may not have as severe a problem as Prolog and Smalltalk do.

Prolog and Smalltalk may be the better languages in building a knowledge based system application if the time schedule is stringent, since they both have built-in environments that make prototyping much easier. However they both are less powerful in numerical computation which is an important factor in engineering application. On the other hand, C provides the facility which is as powerful as those of the numerically oriented languages like Fortran.

Ideally a computer language should allow as much flexibility to define new data types and to write operations on those data types, in addition to standard data types (integers, floating point, characters and string). This is however not possible in Fortran, C and Prolog. These languages do not allow the programmer to change the meaning of built-in operators like + for a special case of the new data types, so that these operators cannot be applied for that case. On the other hand, Smalltalk and C++ provide this facility.

For these reasons, C/C++ seems the best language to use in developing the proposed system.

B.4 SOME USEFUL TECHNIQUES

The author has developed and tested some simple programs to validate the applicability of many ideas generated during the course of the study project. Many ideas that seem to be working well in a simple application may not necessarily be appropriate in a real problem. Some of these ideas were first implemented in Prolog and Smalltalk, but later converted into C/C++.

In the aircraft design application, it is necessary to put emphasis not only on the speed of the system, but also on the accuracy of the analysis. As mentioned in Chapter 1, there is a strong requirement for a rigorous engineering analysis. These two requirements are however contradictive. In order to increase the speed of the process, the analysis may at times be made as simple as possible.

B.4.1 CONSTRAINT PROPAGATION

One of the characteristics of design problems is that the sequence of computational procedures is not always fixed, but depends on how much information is available at a certain stage of the process. One simple example is the determination of wing geometrical parameter :

$$AR = b^2 / S \qquad \text{B.1}$$

Mathematically this expression can be written as,

$$AR - b^2 / S = 0 \qquad \text{B.2}$$

or

$$f(AR, b, S) = 0 \qquad \text{B.3}$$

in which, if any two of the variables are known, the other variable can be computed. In this case, the expression can be altered accordingly depending upon which data are available.

For a more general problem, the number of variables and expressions involved may be numerous, and if there are more than one unknown variables, it is more difficult to determine which variable should be defined first.

$$\begin{aligned} &f_1 (X_1 , X_2 , \ldots, X_n) = 0 \\ &f_2 (X_1 , X_2 , \ldots, X_n) = 0 \\ &\cdot \\ &\cdot \\ &f_m (X_1 , X_2 , \ldots, X_n) = 0 \end{aligned} \qquad \text{B.4}$$

The problem becomes more complicated if the number of expressions m is less than the number of variables n. In this case some of the variables may have to be assumed first in order to reduce the number of unknown variables.

The principle applied here is to use expressions with the largest number of known variables first and propagate the constraints of unknown variables through out the whole expressions to calculate the rest of variables. This principle is called *constraint propagation* technique.

To effectively apply this technique, a facility should be employed for recording the status of each design step. To determine the next possible steps, the status of all the previous steps should be reviewed. The candidate steps can be suggested based upon the steps which are already established.

B.4.2 OBJECT-ORIENTED PROGRAMMING

The object oriented programming technique is a new way of programming in which data are taken as objects [25]. The objects can perform functions that are assigned to them. The objects are classified according to their attributes in which objects which have the same attributes belong to the same class. The attributes can be of the form of properties or functions that the object performs when activated. All objects of the same class will perform in the same way when a particular function is assigned to them.

The attributes of a class are privately owned by the object members of that class in that they can only be accessed by the object members themselves.

This programming technique enables the data to be structured in a hierarchical way in which classes of objects can be derived from other classes. All objects of one class inherit the attributes of their parent class, while they may have attributes specific to the class they belong to. This concept is powerful in the long run since the library of object classes can be re-used again and again for different applications. Extending and modifying an application is made easier. This however requires a careful consideration on the classification of the objects to allow an effective use of the inheritance.

Another important characteristic of object oriented programming is data hiding in which the detailed implementation of the functions of a class is hidden from the application of the class. This data hiding is powerful for programming security. Since the class functions are inaccessible outside the class, there is no worry about misuse of functions by objects of other class. Furthermore, the properties of a class can only be accessed through member functions, therefore no variables specific to one class can be modified by objects of other classes.

There are a few programming languages that are suitable for this programming technique, some of them are Smalltalk, Object-Oriented Pascal and C++. Smalltalk is a pure object oriented language in which the language was actually developed for specific use in object oriented approach. On the other hand, Object-oriented Pascal and C++ are hybrid languages in that they apply both conventional (procedural) programming technique and object oriented approach.

B.4.3 TREE AND GRAPH IN PROBLEM SOLVING

In solving a specific problem, a human expert normally devices a strategy to follow. The expert need to decompose the problem into smaller units and structure them in a way that would help the expert to consider each unit efficiently. The structure will resemble the relationship among the units and represent the path for problem solving activity.

The most common way to implement any search strategy is tree and graph [13, 73]. Tree, as shown in Fig. B.2, is a structure with an origin, its root. The root is a particular

member of the set of nodes. Each node has its specific value (or set of values) and a set of successors. The successors are themselves the nodes of the tree.

The search strategy on a tree structure can be done by *depth-first* or *breadth-first* traversal. The depth-first traversal visits the successors first before the nodes on the same level of hierarchy. On the other hand, the breadth-first traversal visits the nodes on the same level of the hierarchy first before looking at the next level of successors.

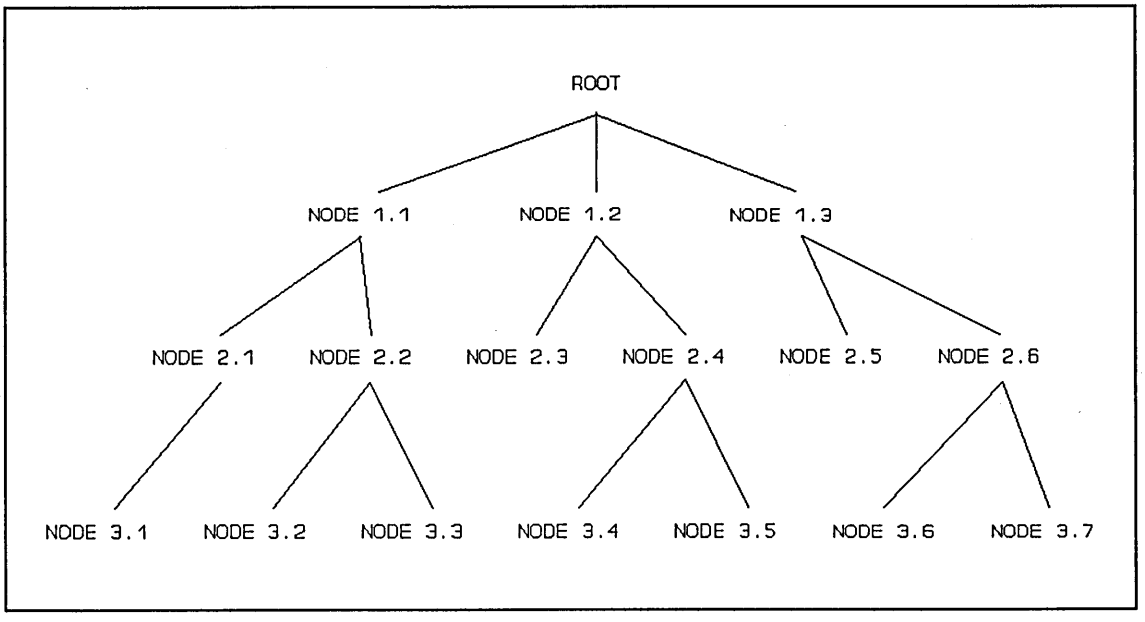


Fig. B.2 The tree structure

The structure of a graph is similar to tree's structure. The only difference is that two nodes in a graph may have common successors, see Fig. B.3. Thus a node can be visited through more than one path.

Graph search procedures are useful for dealing with partially commutative system in which a given set of operations will produce the same result regardless of the order in which they are applied [13]. This frequently happens in aircraft design activities.

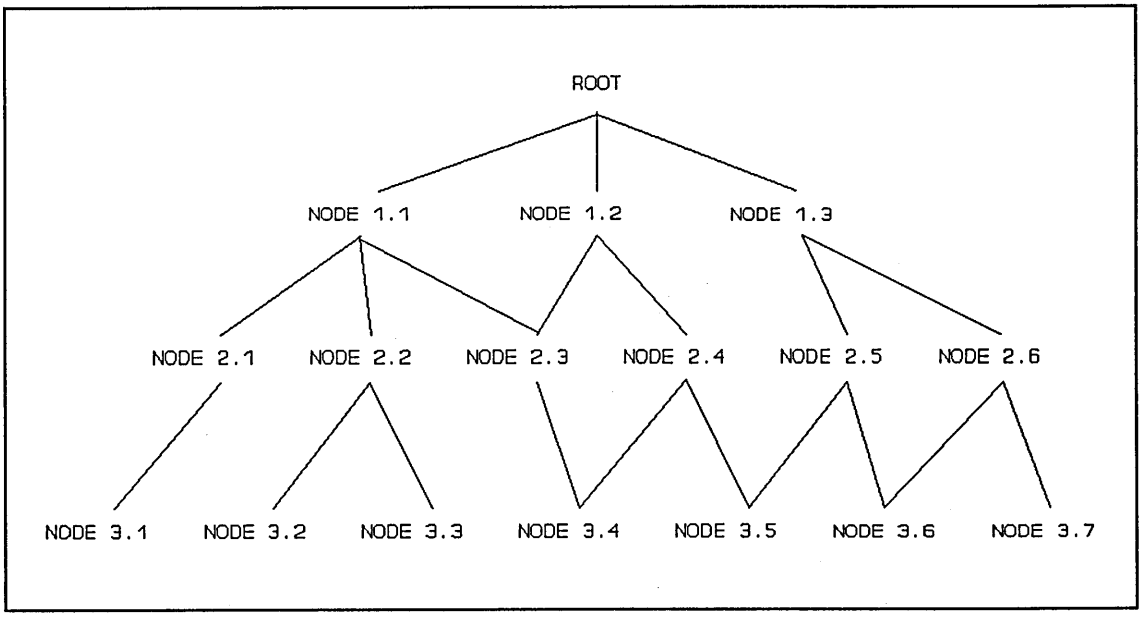


Fig. B.3 The graph structure

APPENDIX B

APPENDIX C
DETAILS OF KNOWLEDGE REPRESENTATION

C.1 INTRODUCTION

The implementation of the rules in the knowledge base is very much dependent on the computer language in which they are programmed. This dependency is influenced by the way the computer language of implementation represents the real world. C/C++ language facilitates a method to define a new types of object to suit applications. This is done by using the structure *struct* and *class* to define new classes of objects in addition to the traditional data types, e.g characters, integer, floating point numbers and logical types.

The IF-THEN rules as used in the production rules representation can be defined as a structure which has a list of conditions and a conclusion.

```
IF      A is B,
        C is D,
        E is F,
        ......
THEN    X is Z.
```

Each of the conditions ("A is B", "C is D", etc.) and the conclusion ("X is Y") are primitive propositions, which can be represented by class Association.

The class Association takes the form of a keyword and its corresponding meaning. The keyword represents the attributes, i.e name of the action or a variable, and the meaning represents the conclusion for the action or the value that the variable holds. For the rule described earlier, the "A is B" proposition can be represented as association of A and B in which A becomes the name of the attribute and B is the value of the attribute in the association. The same can be done with the other propositions in the rule.

The primitive propositions of this kind become the basic elements of the rule. They can be tested against the facts or intermediate hypotheses, which themselves are primitive propositions, in the database to see the validity of each condition of a rule. Any rule whose conditions are met by the propositions in the database will update the database in that its conclusion will be added into the database again in the form of primitive propositions. The recording of these primitive propositions as associations in the database is explained in Appendix D.

C.2 REPRESENTATION OF KNOWLEDGE BASE RULES

A rule in the form of IF - THEN relationship can be translated into C/C++ by using two newly defined structures called Knowledge Source and Rulesets. The format of the rules is reversed to enable easier identification. The rule is then written as follows :

```
THEN    X is Z
IF      A is B,
        C is D,
        E is F,
        .....
```

The attribute of the primitive proposition of the rule's conclusion is used for identification. In this case, X is the attribute, which distinguishes the rule from other rules.

C.2.1 STRUCTURE KNOWLEDGE SOURCE

Structure knowledge source is used to record the attributes of the primitive propositions that participate in a rule.

The structure of a knowledge source is shown in Fig. C.1. It has four attributes : a name, a *head*, a *record length* and the *next pointer*. The *name* represent the name of the attribute of the primitive proposition of the conclusion. An instance of knowledge source can be distinguished from the other instances by its name.

The *head* attribute is a pointer to a structure called *rulePredicate* as shown in Fig. C.1. It represents the list of attributes of the conditions' primitive propositions. A pointer is used here rather than the actual instance of structure rulePredicate so that a variable number of conditions can be accommodated in the list. The pointer in fact refers to the address of the first member of the list, hence the name *head*. The use of pointer is one way of informing the computer that the object the pointer refers to may be a list of objects without the necessity of declaring the size of the list.

The other attribute, the *length of record*, is used to record the number of conditions in the list. This attribute is required for the purpose of traversing the contents of the list. This attribute is specific to any individual instance of structure knowledge source. It is automatically updated when a condition is added into or removed from the list.

The last attribute, the *next pointer*, is a pointer to an instance of structure knowledge source itself. It is a reference to the next instance of knowledge source on record as will be described later.

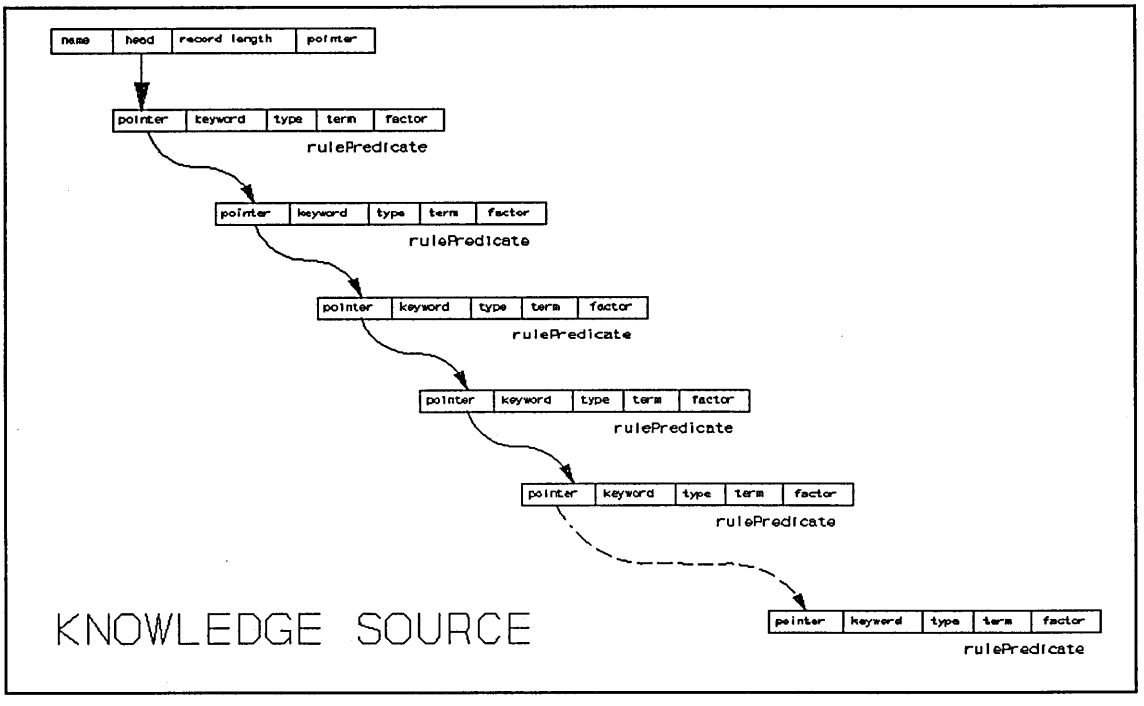


Fig. C.1 C/C++ representation of the structure Knowledge Source

The structure *rulePredicate* is defined to represent each condition of the rule. IT has five attributes : *keyword*, *type*, *term*, *factor* and a *pointer* to an instance of the structure itself, as shown in Fig. C.1. The *keyword* attribute represents the name of the condition, i.e the attribute of the condition's primitive proposition. The attributes *type* and *term* are additional attributes used to store extra information about the condition. The *type* attribute is designed to specify the type of condition. At present, three different types are recognised :

- Rules
- Procedure
- Question
- Input.

A rule predicate of type rule is used to enable inclusion of rules within a rule. The procedure type is used for predicates that consists of computational procedures. It can be used to incorporate numerical calculations that lead to a particular intermediate conclusion. The question and input types are used for conditions that require reply from the user. The question type are for questions that need a positive ("yes") or negative ("no") answer, while the input type is used for any other types of user's reply. It can be used for asking numerical data from the user.

The *term* attribute is used in conjunction with the type attribute. For rules, the *term* attribute specifies whether it deals with a design step or not. This distinction is necessary since the primitive propositions of design step conclusions are stored in a

different database from that of other rules. Therefore this is one way of informing the system which database to call when the rule is activated. If it is a design step, the configuration database will be used, otherwise it will be the intermediate database.

For predicate of type procedure, the term specifies the name of the most important parameter that participates in the numerical calculation. The parameter will be retrieved from the database and used in the calculation. For predicates of type question, the *term* attribute records the question itself. The question is a query which can be answered with a positive or negative reply.

In building the knowledge base, the procedure used in the calculation for a predicate of type procedure needs to be included in the system. This procedure is a C++ function with the following form :

```
void process_equation( char *name, char *value )
{
   if ( !strcmp( name, theIdentifier ) )          // search for the appropriate
                                                  // path
   {
      .....
      .....lines of process
      .....
      strcpy( value, aConclusion);                // assign aConclusion to value
   }
}
```

in which *theIdentifier* is the same as the name of the attribute of the condition's propostions and aConclusion is a terminology that specifies the conclusion of the numerical process.

An attempt had been made to completely separate the system from the knowledge base by treating each procedure as a stand-alone program. This program takes the following form :

```
void main( int argc, char **argv )
{
   FILE *fp;                                      // Pointer to an intermediate
                                                  // file
   .....
   ..... lines of process
   .....
   fp = fopen( "intermediate.file", "w" );        // Open file for writing
   fprintf(fp, "%s %s", argv[1], aConclusion );   // Write attribute and
                                                  // conclusion to the file
   fclose(fp);
}
```

APPENDIX C

in which the *argv[1]* specifies the name of the attribute of the condition. When the system identifies that it deals with a procedural type of a rule's predicate, it retrieves the name of procedure and executes the stand-alone program with the name passed as as argument. The program does the processing of data, selects a conclusion, and write the conclusion to the intermediate file. The system then reads the contents of the file and evaluates the conclusion.

This idea worked reasonably well. The only problem is that, for a large problem, many of the programs require the same analysis, and each time this analytical procedure has to be repeated in each program. This is obviously not efficient from the point of view of programming, which led to the decision to reject this idea.

Therefore, unless this problem is overcome, the previous approach of encoding the procedural knowledge within the system is felt to be adequate.

The *factor* attribute is an integer constant used to specify the level of importance of the condition in determining the outcome of the rule. A condition which has a greater contribution to the conclusion of the rule is give a higher value. This factor is not intended to reflect the probability of the rule being true. It is used to distinguish between the conditions which have a major influence on the conclusion and those which have only minor effects.

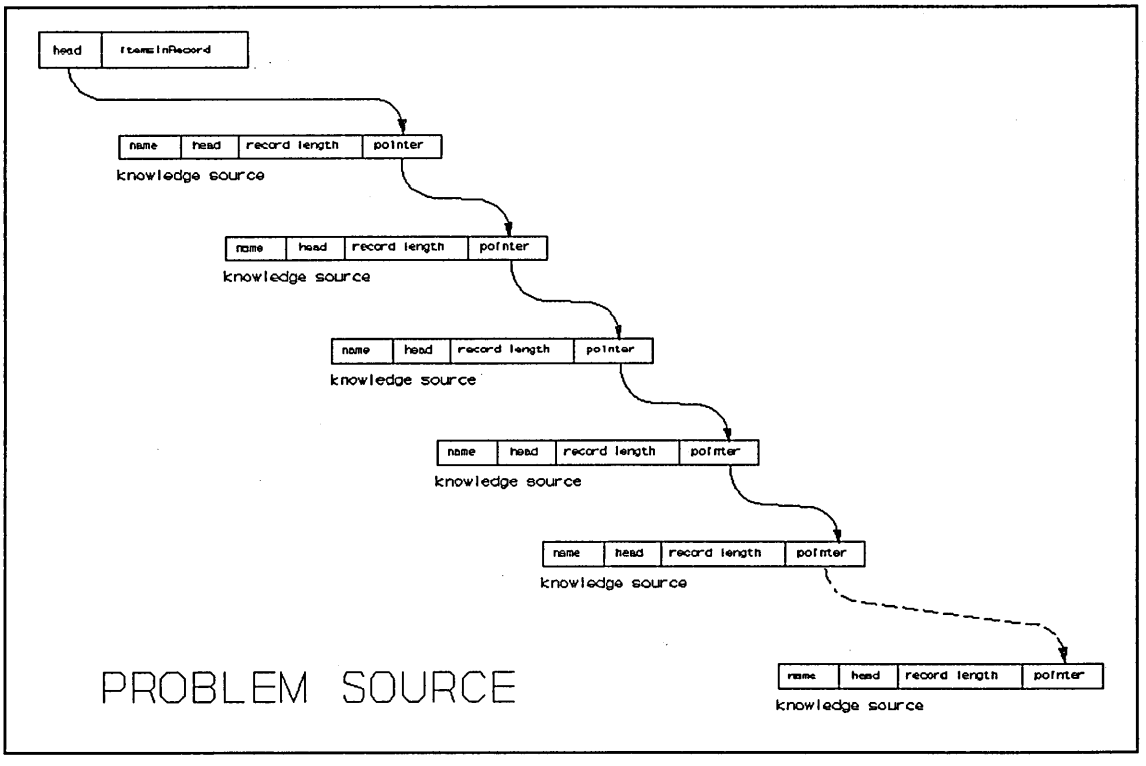


Fig C.2 C/C++ Representation of the structure Problem Source

For the purpose of devising an efficient search procedure, all the knowledge sources are linked together to form a structure called *problemSource*. This structure has two attributes : a *head* and *itemsInRecord*. The *head* is a pointer to the structure Knowledge Source. It is an address on which the first instance of the structure Knowledge Source is recorded. The attribute *itemsInRecord* is a integer variable that records the number of knowledge sources contained in the structure. It is automatically updated each time a new instance of knowledge source is added into the record.

The knowledge sources are linked within the *problemSource* structure by using the *next pointer* attribute of the knowledge source structure as mentioned earlier. Each time a new instance of knowledge source is added into the record, the next pointer of the last instance on record is assigned to the new instance. This is the same as filling the memory address of the new knowledge source into the pointer so that when the pointer is called, this memory address will be given.

C.2.1 STRUCTURE RULESET

Each instance of knowledge source is complemented with a set of *rulesets*. Structure *ruleset* records the values of the primitive propositions of a rule. The order in which the conditions' primitive propositions are recorded in the rulesets is the same as in the knowledge source they associate with. This forms a one to one correspondence between the attributes of the propositions in the knowledge source and the values of the propositions in the rulesets. The set of rulesets associated with a particular instance of knowledge source is grouped into a structure called *problemRules* as shown in Fig. C.3.

Structure rulesets has three attributes : *name*, *pattern-Array* and a *pointer* to an instance of structure ruleset itself. As in the knowledge source, the *name* attribute records the name of the value of the primitive propositions of the conclusion. This attribute specifies the identity that distinguish the instance from other instances of the rulesets that belong to the same *problemRules* structure.

The *pattern-Array* attribute is an array of strings of characters whose members are the values of the primitive propositions of the conditions. The order of the values recorded is the same as the sequence in which its corresponding attributes stored in the appropriate knowledge source. When a knowledge source is activated, its associated problemRules is loaded to find the rules that match. The ruleset within the problemRules is examined one by one by matching the members of its pattern-Array with the value of the propositions in the database. Each time a match is found, the factor of the rulePredicate of the propositions will be added to the total score of the ruleset. The conclusion of the ruleset that scores the highest will be taken as the final conclusion, i.e the value recorded in the *name* attribute of the ruleset.

The *pointer* attribute is the link between one instance of ruleset to the other. This pointer is a reference to the structure ruleset itself which stores the address of the next ruleset on record.

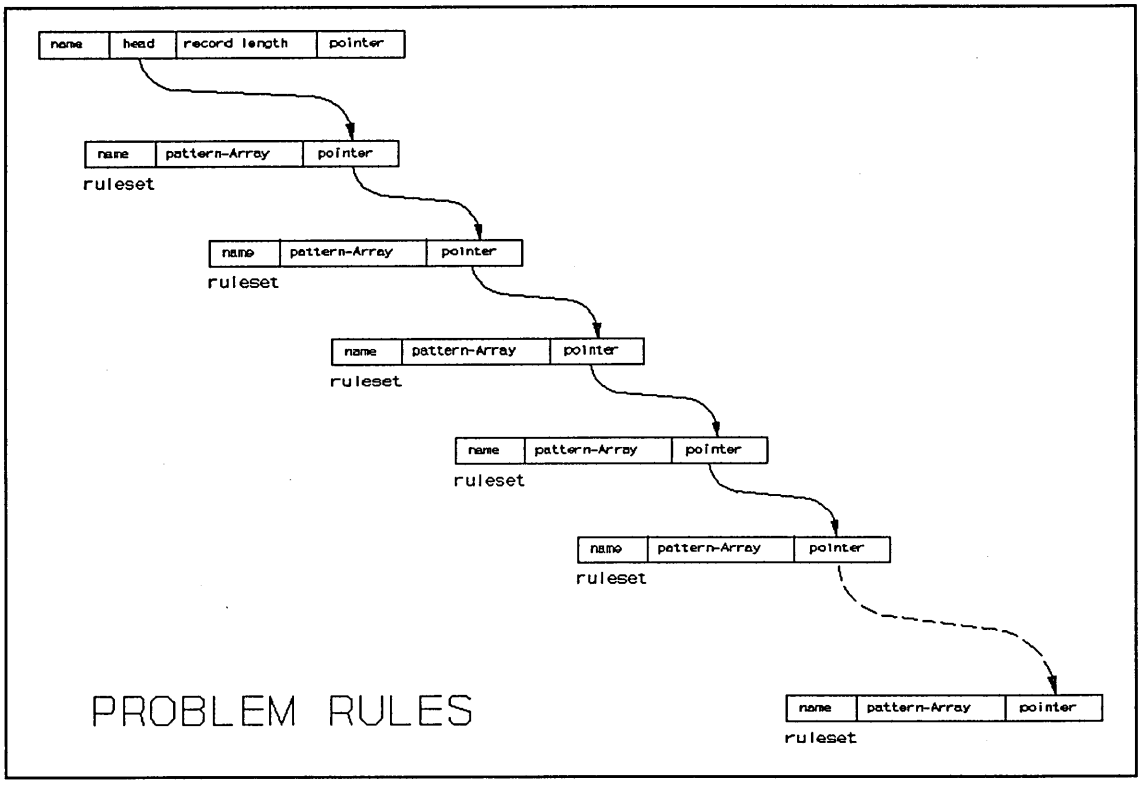


Fig. C.3 C/C++ representation of the structures of Ruleset and Problem Rules

C.3 THE KNOWLEDGE BASE FILES

The configuration design knowledge rules explained in section 3.2 of Chapter 3 is stored in two separate files called '**knowledge.source**' and '**knowledge.rules**'. The file '**knowledge.source**' contains the record of items that will be stored in structure Knowledge Source. The recording of the items takes the following pattern :

.....
name of the attribute of the rule's conclusion
condition's attribute type term factor
.....
end end
.....

The file '**knowledge.rules**' contains the lists of items that will be recorded as rulesets. The recording takes the following pattern :

.....
name of the attibute of the rule's conclusion
value of conclusion list of values of the conditions end
.....

When loaded, the system reads the contents of the two files and stores the corresponding information in the structures Knowledge Source and Rulesets.

The objective of storing the information about the design knowledge in a text file separate from the system is to enable the knowledge base to be extended without any major changes in the system.

APPENDIX C

APPENDIX D
DETAILS OF INFERENCE ENGINE

D.1 INTRODUCTION

The inference engine of the system performs several functions :

- To decide which design step to execute :
  The inference engine forms a strategy in solving the design problem along the hierarchical tree of actions. It consults the status recorder as to what actions are ready for execution. If more than one actions are placed in the queue of the status recorder, the inference engine will take the top items placed in the queue to execute.

  This queue forms an agenda of design actions for the inference procedure to execute. It includes a list of design actions ready for execution.

- To provide a problem solving strategy for each design step :
  In executing a design step selected from the agenda, the inference engine decides what strategy to follow in solving the problem at that step. It searches for the rules that apply to the design step. It then tries to process the conditions of each rule by either testing them against the information available in the status recorder or by asking question to the user for the information. It binds the variables in the conditions to the one of the intermediate solutions. At the same time the inference engine records all the conclusions of each rule in the status recorder as intermediate results.

- To maintain consistency in the solutions :
  The inference engine evaluates if there is a conflict between the solutions suggested. It cross-examines the solutions and possible problems due to task interactions created during the execution of previous tasks. It then tries to solve this problem either by updating the solution or by introducing a correction.

- To provide an explanation facility :
  The inference engine explains the action of the system to the user. To do this the inference engine uses a few general types of question-answering plans. This requires the inference engine to trace back along the status recorder to find the conclusion in question. It then searches the knowledge base for the rules that supported the conclusion. Each rule corresponds to an intermediate result recorded in the status recorder. The inference engine then collects all these intermediate results in logical order as in the knowledge rules and translates them into English for presentation to the user.

The following sections will discuss each of these functions in more detail. The implementation of the functions is also given. To help clarify the explanation of these

functions, the description of the status recorder is presented first. A detail implementation of the status recorder is explained afterwards.

D.2 THE STATUS RECORDER

The status recorder works together with the inference engine. It can be viewed as the memory space for the inference engine to work on. Its main functions are :

- To record the control of actions awaiting execution
- To record all the primitive propositions, i.e the initial conditions or facts, conclusions generated during the processing of knowledge based rules and the final solutions of each design step.

D.2.1 THE QUEUE OF DESIGN TASKS

The status recorder has different ways of storing information to serve each of these functions. The record of the potential actions, i.e the queue of actions that are ready for execution, is stored in a stack. This way of storage utilises the last in, first out principle, in which the last action item pushed to the queue will be the first one to be called back for action. This resembles putting books in a stack. The first book in the stack is placed at the bottom. The book that comes afterwards is placed on top of the last book added to the stack. When removing a book from the stack, the book at the top of the stack will be taken out first. This principle is applied to represent the hierarchical tree of actions.

The queue is represented in C/C++ language using a class or structure definition as shown in Fig. D.1.

The queue element is a C/C++ structure which has two attributes: *data* and *pointer*. The *data* member is represented as a string of characters whose value is assigned to the name of the step. *Pointer* is a special data type which refers to the data type preceding it in its declaration. Pointers are memory addresses used in C/C++ language on which other data types can be allocated. The pointer element is then a data type that points to next element in the list. In other words, it is an address at which the next data is stored.

This representation is chosen so as to enable a sequential data retrieval and storing procedure. The queue is designed as a one-ended list, i.e the member element can only be added at or taken out from one end. In this case, it is structured so that it can be done at the top of the queue as explained earlier.

The class queue has its own specific procedures in performing various actions. These procedures cover ways of adding a new element into the queue, popping out a member from the queue, thereby removing that member from the list, and testing if the queue is empty or if an element is already on the queue.

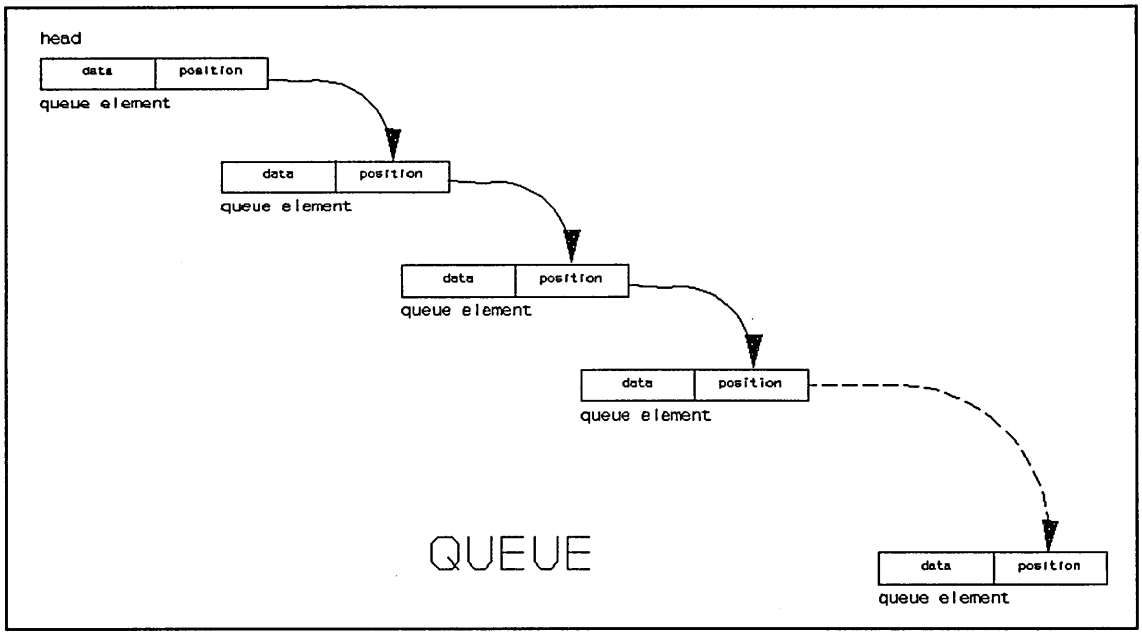


Fig. D.1 C/C++ representation of the structure Queue

D.2.2 THE DATABASE ELEMENTS OF THE STATUS RECORDER

The primitive propositions are recorded in a different way. They are stored as associations in the database. These associations take the form of a *keyword* and its corresponding *meaning*. The *keyword* represents the attributes, i.e name of the action or a variable, and the *meaning* represent the conclusion for the action or the value that the variable holds.

The associations are stored as in a database class called *Dictionary*. A dictionary is a list containing associations linked together in sequence. The sequence reflects how the associations are added into the dictionary. Dictionary is designed as a two-ended list. An association can be added into a dictionary as a new member, either at the beginning or at the end of the list. By default, a new item is added at the end of the list.

Fig. D.2 shows how these associations and dictionary are structured using C/C++ language. Both are represented as classes using C/C++ class definition facility. The association has three attributes : *keyword*, *value* and two *pointers*. Like the pointer in queue, the two pointers are the memory address of an association, one referring to the next association and the other to the previous association.

Dictionary has three attributes : a *head*, a *tail* and an *index*. The *head* is a pointer that refers to the association at the top of the list, while the *tail* is a pointer to the association at the bottom of the list. Both *pointers* are variables which can be changed either by removing or adding items from and into the list. The head changes only when

the addition or the removal takes place at the top of the list. When a new item is added at the top of the list, the item replaces the previous item at the top as head. The next pointer of the new item is assigned to refer to the old head and the previous pointer of the old head is assigned to the new item. In C/C++ this is translated as follows :

```
head -> previous = new_item;
new_item -> next = head;
head = new_item;
```

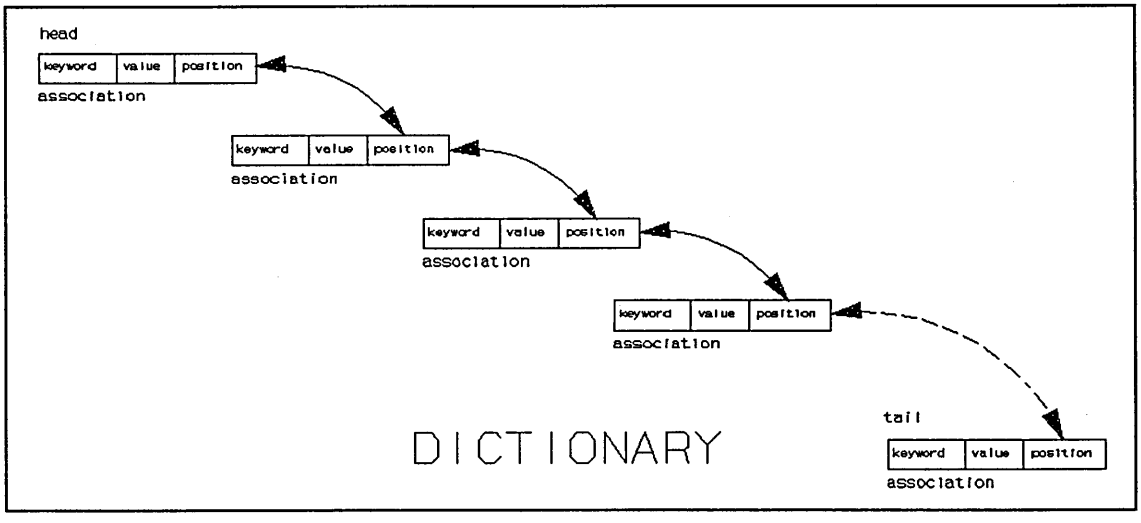


Fig. D.2 C/C++ representation of the structure Database

On the contrary, the tail changes only when these operations take place in the other way around.

The *index* represents the order in which the member association placed in the list, starting from the head to the tail of list. This index is also a variable like the two pointers. But unlike the pointers, it changes whenever the addition or the removal occurs no matter at which end the process happens.

This ordering of the dictionary member is required for efficient search. The inference engine can search for a particular item in the database of a dictionary by moving from one item to the other either from the top or from the bottom of the list. By default, the search starts from the top. Each time the keyword of the item is compared with the item in question. If the keyword matches the item, the procedure will return the item with its corresponding meaning.

D.3 THE INFERENCING PROCEDURE

D.3.1 SELECTING A TASK TO EXECUTE

The general strategy employed in selecting a design task for execution is as follows :

- Search the appropriate queue if any item exists in the queue.

- If the queue is empty, consult the tree of actions and search for the items on the tree that have not been executed. Starting from the top level of the tree,

  1 Recall the item one by one and test if the item is a member of the configuration database, i.e the item has been evaluated before and it has solution in the configuration database. If it does exist in the database, recall the next item at the same level and do the same until all item at the same level is tested. Collect all items that are not in the database and add them into the queue.

  2 If all items are in the configuration database (the queue is still empty), go to the next level and do step 1.

  3 If the queue is empty and all levels in the tree have been searched, inform that all items have been evaluated and return nothing.

- If queue is not empty, call the item on the top of the queue and remove it from the queue. Assign the item as the selected step.

D.3.2 EVALUATING A TASK

The procedure employed in evaluating a task is a recursive process, as follows :

- Search for the knowledge source associated with the task.

- Activate the knowledge source found :
  1 Recall the attributes one at a time

  2 Process attribute :
    - Check the database if the proposition for that attribute exists.
    - If it does exist, go to Step 4.
    - Otherwise do the following :
      If the type is a question :
      - Ask question in a question window.
      - Assign user's reply to the value of the proposition of the attribute.

      If the type is a rule :

- Suspend task by putting it back on the queue.
- Process attribute as if it were another task by activating its knowledge source and to repeat the whole process from the beginning (recursive).
- When finished, assigned solution of the attribute to the proposition and store it in database.
- Resume task by retrieving it back from the queue.

If the type is an input :
- Ask input from user.
- Translate numerical values into a term acceptable to the system.
- Assign the term to the value of the proposition of the attribute.

If the type is a procedure :
- Invoke the appropriate procedure.
- Translate the numerical values of the result into a term acceptable to the system.
- Assign the term to the value of the proposition of the attribute.

3 Assert the proposition into the database.

4 If the attributes have not all been processed, go back to step 1.

- Recall all the rulesets that corresponds to the active knowledge source.
  - For each ruleset, test conditions against the propositions in the database :
    - If the condition is a void proposition, give zero mark (0) to the condition.
    - If the condition matches one of propositions in the database, give positive mark (+1) to the condition.
    - If no match is found in the database, give negative mark (-1).
  - Calculate the total mark for each ruleset by summing up the mark of each condition multiplied by its rating.
- Select the solution with the highest total mark and assert its proposition to the database.

D.3.3 SOLVING TASK INTERACTION PROBLEMS

In maintaining the consistency of the solutions placed in the status recorder in the light of new evidence obtained during the execution of a specific task, the inference engine consults the database if there is a consequence of the step decision in the database. The consequences of a primitive proposition can be obtained by the inference engine by reversing the rules which involve the proposition. The consequences of the a particular primitive proposition are the conclusions of the rules which have that propositions as one of their conditions.

The inference engine searches for the consequences related to the task. It then tests whether the consequence is conflicting with any prior decision in the configuration database. If so, the inference engine will place the step whose decision is conflicting into the queue and delete all decisions associated with its dependent steps, i.e its successors in the graph of actions, from the database.

The inference engine then needs to re-evaluate the steps in the light of the new evidence. In some cases, the inference engine could suggest the second best solution for the step and tests if it still conflicts with the consequences.

D.3.4 EXPLAINING THE REASONING PROCESS

In explaining why a certain conclusion was reached, the inference engine takes the following procedure :

- Find the proposition of the conclusion on the database element of the status recorder.
- Look for the knowledge source for the attribute of the proposition.
- Traverse the knowledge source, processing the attribute of each condition one at a time. For each attribute, do the following :

  1 Find the proposition for the attribute in the database :
    - o If the type is a task, search at the solution database.
    - o Otherwise, look at the intermediate database.

  2 Translate the proposition into an English-like sentence.

An explanation can be extracted from the system as to what the conclusion of a particular task is all about. To display this information the inference engine takes the following procedure :

- Find the proposition of the solution on the database element of the status recorder.
- Look for the appropriate portion of text in the configuration help file by matching the attribute and the value of the proposition with the two identifiers in the file. When the matched identifiers are located, read the portion of text that follows until the end and display the text in a window.

Note :
The configuration help file is a text file containing the information about all choices of each configuration design task. The information is organised in groups according to the task, as follows :

Task_identifier 1

Choice_identifier 1.1
Lines of text for choice 1.1
end
Choice_identifier 1.2
Lines of text for choice 1.2
end
.
.
.
Choice_identifier 1.n
Lines of text for choice 1.n
end
END
Task_identifier 2
Choice_identifier 2.1
.
.
.

This information is contained in a file called **'configuration.step'**.

APPENDIX D

APPENDIX E
IMPLEMENTATION OF THE SYSTEM USER INTERFACE

E.1 INTRODUCTION

The prototype of the Aircraft Conceptual Design System was developed in a Sun workstation using C++ [74] and SunView (**Sun** Visual / **I**ntegrated **E**nvironment for Workstations) programming environment [75,76]. SunView is a programming tool written in C language [77] and is available for use in Sun computers to develop a program user interface incorporating object oriented concepts. It utilises a graphic based windowing facilities in which many overlapped windows can be created on screen. Each window may contain several non-overlapping windows within its border. Each window is a working space in which various computer operations can be performed.

There are two types of windows : Frames and Sub-windows. Frames are overlapped windows used to bring the window into view. They can have Sub-windows contained within them. They can also have other frames as sub-frames.

Sub-windows are non-overlapped windows on which the operations are actually done. The type of operations performed in a window depends on the type of the window itself. Each window has its own attributes and a set of operations (functions) associated with it. Some of the Sub-window types available in SunView are :

- Panel
- Canvas on which graphs are displayed
- TTY window, a terminal emulator on which the user can enter computer commands to execute
- Text Editor window on which texts are presented

E.1.1 MOUSE

The operations on a window can be activated through a mouse. The mouse is primarily used to move the cursor (symbolised by an arrow pointing North, ▲, or a solid box, ▮) around the windows. The mouse has three buttons whose actions can be defined as required, see Fig E.1 :

- The LEFT MOUSE BUTTON which is normally used to select or activate an operation associated with a panel button,
- The MIDDLE MOUSE BUTTON which can be set to activate certain operations depending on the application,
- The RIGHT MOUSE BUTTON which is normally used to display menu from which a selection is made.

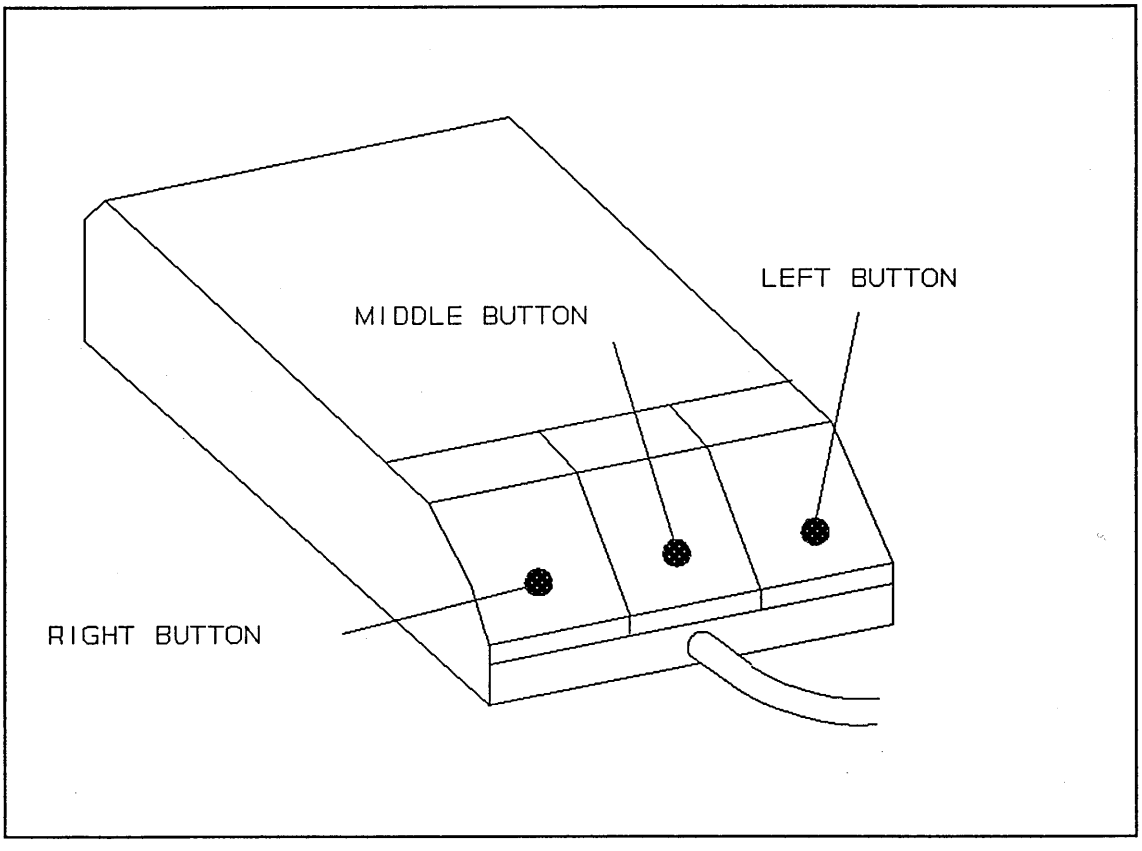


Fig. E.1 Three-buttoned mouse

E.1.2 VISUAL OBJECTS IN SUNVIEW

There are several visual objects used in SunView windows, of which six are extensively used in the System's user interface :

- Panel buttons
- Panel items.
- Menu
- Scrollbar
- Alert
- Pointer

To avoid confusion, the panel buttons are always referred to in lower case letters while the MOUSE BUTTONS are mentioned in upper case.

E.1.2.1 Panel Buttons

Panel buttons are used in Panel window. In the Aircraft Configuration Design System

user interface they are displayed as a **bold** text in a bubble. A panel button normally has an certain action associated with it. This action can be executed when the button is "pressed". To activate the button follow this procedure :

Procedure 1 :

- Move the cursor to the panel button,
- Click the LEFT MOUSE BUTTON to start the action.

E.1.2.2 Panel Items

Panel items are objects which hold a symbolic information. In the Aircraft Design System they are used to display variables and the values they hold. There are two different types of panel items used in the system's user interface :

Panel Choice Item

A panel choice item is displayed as a text with a 'Cycle' symbol, ⟳. It represents a selection from a list of texts. The current value for the item is displayed on the right of the 'Cycle' symbol. The selection can be made with either one of the following procedures :

Procedure 2 :

- Move the cursor to the current choice written on the right of the 'Cycle' symbol, and click the LEFT MOUSE BUTTON to advance to the next choice on the list. Click the LEFT BUTTON several times until the required choice is displayed.
- Alternatively, move the cursor to the current choice. Press and hold the RIGHT MOUSE BUTTON to pop up the menu list. While holding the BUTTON down, drag the MOUSE down until the required item is highlighted and release the BUTTON to select. This procedure is normally taken to quickly select the required item on the list. One example is given in Fig. E.2 in which the choice of the variable for the vertical axis of a graph is chosen from a menu.

Some panel items of this type may activate certain actions associated with the current choice.

Panel Text Item

This type of panel item is used to display variables which have no restrictions on the value it holds. It can hold any value including numerical data. This item is represented in the panel as text with a colon ':', with the value it holds is displayed right after the colon itself. To change the value of the item simply follow this procedure :

Procedure 3:

1. Move the cursor to the current value and click the LEFT MOUSE BUTTON. The arrow cursor will changes into a solid box, ▮ , to inform that the cursor is active on the selected position.
2. Overwrite the value with a new value.

E.1.2.3 Menu

Menu is an object through which a user makes choices and issues commands. By default, menus pop up when the user presses the RIGHT MOUSE BUTTON. Like windows, menus appear on the screen when required, and disappear when they have served their purpose. Menus, however, differ from windows in several ways. A menu only remains on the screen as long as the MOUSE BUTTON remains depressed. On the other hand a window remains on the screen until the user indicates he is done with it or the controlling program explicitly closes it. Menus are less flexible than windows, they are designed to allow the user to choose from among a list of actions.

E.1.2.4 Scrollbar

Scrollbar is an object attached to and displayed within a sub-window through which a user can control which portion of the sub-window's contents are displayed.

E.1.2.5 Alert

Alert is a message displaying object which is designed to attract the user attention on its presence. It is displayed as a box on the screen which informs the user of some condition. It has one or more buttons which the user can push to dismiss the alert or choose a means of continuing. Like menus, alerts disappear as soon as the user pushes a button or otherwise dismisses the alert. Visually they resemble simple panels containing only images, messages and buttons.

In Aircraft Configuration Design System alerts are mainly used for displaying on-line help information. This form of display is selected to enable the user to respond to the information provided. It suspends the execution of the program until the user replies. The actions associated with the panel buttons can be defined for each alert as required.

E.1.2.6 Pointer

Pointer is a visual object indicating the position of the MOUSE on the screen. It is displayed as an arrow pointing to North-West, ↖.

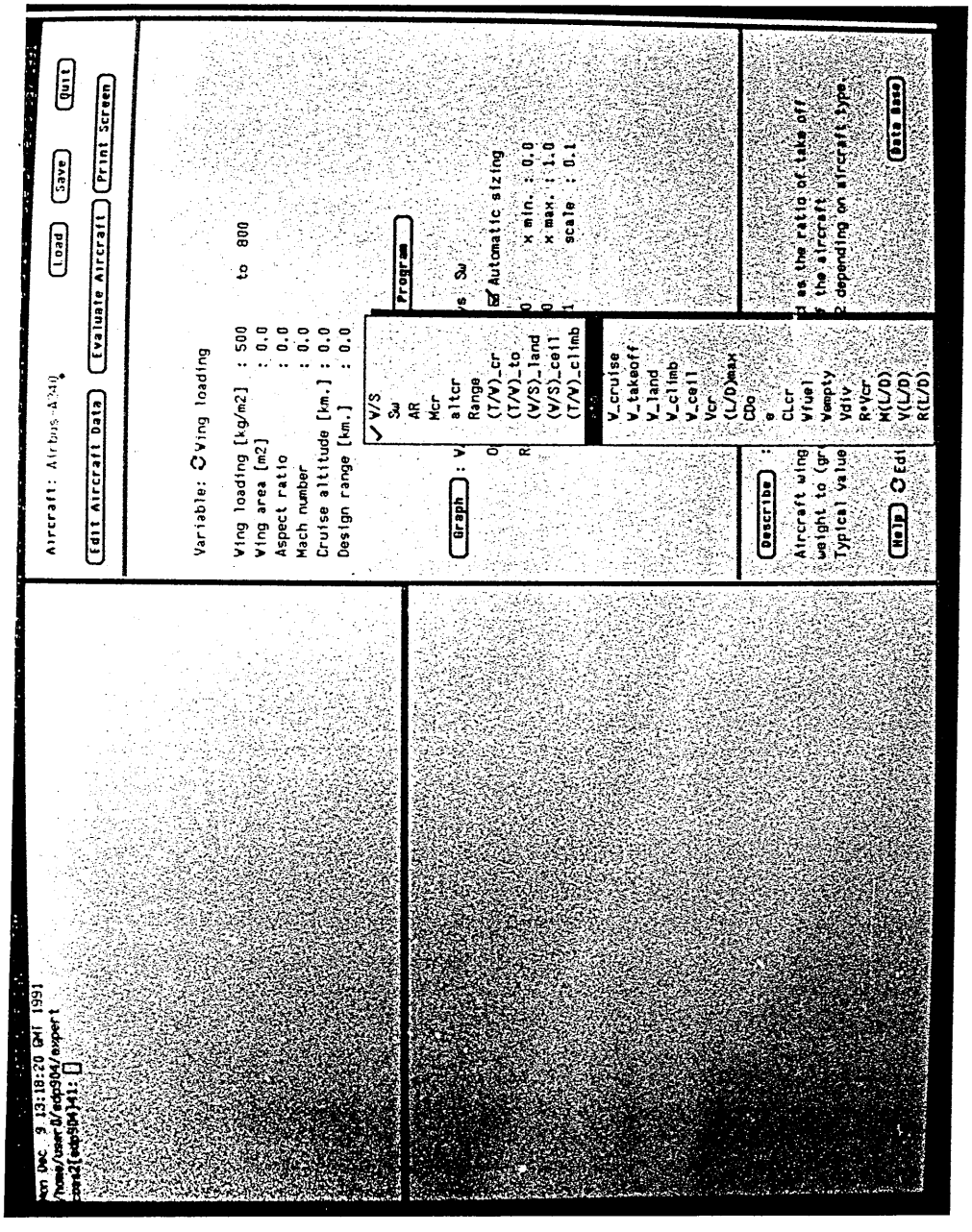


Fig. E.2 Selecting an item from a menu in the aircraft design window

E.2 THE SYSTEM USER INTERFACE

The user interface's window facility is organised as follows :

- Aircraft Design Window
  A base frame which consists of several sub-windows :
  - A TTY sub-window
    A terminal emulator at the top left of the base frame on which the user can execute computer commands. It is used mainly to run external programs.
  - A Graph sub-window
    A canvas below the TTY sub-window within the base frame on which graphs are displayed.
  - A Main Control Panel
    A panel at the top right of the base frame from which the main functions of the program can be activated.
  - A Module Interface Panel
    A panel below the Main Control Panel from which the user can interact with the external programs.
  - A Help Control Panel
    A panel below the Module Interface Control Panel from which the user can extract on-line help.

- Database Management Window
  A sub frame from which data of specific aircraft on the database can be retrieved. The window is equipped with its own on-line help facility.

- Configuration Analysis Window
  A sub frame from which the user can perform the configuration analysis of aircraft design.

- Text Window
  A sub frame on which the result of the execution of a specific program is displayed.

A detailed description will be given in the following.

E.2.1 THE AIRCRAFT DESIGN WINDOW

The aircraft design window becomes the main interface between the user and the system, through which other windows can be activated. The user can interact with the system through three control panels, i.e. the Main Control Panel, the Module Interface Control Panel and the Help Control Panel. The arrangement of the aircraft design window is shown in Fig. E.3.

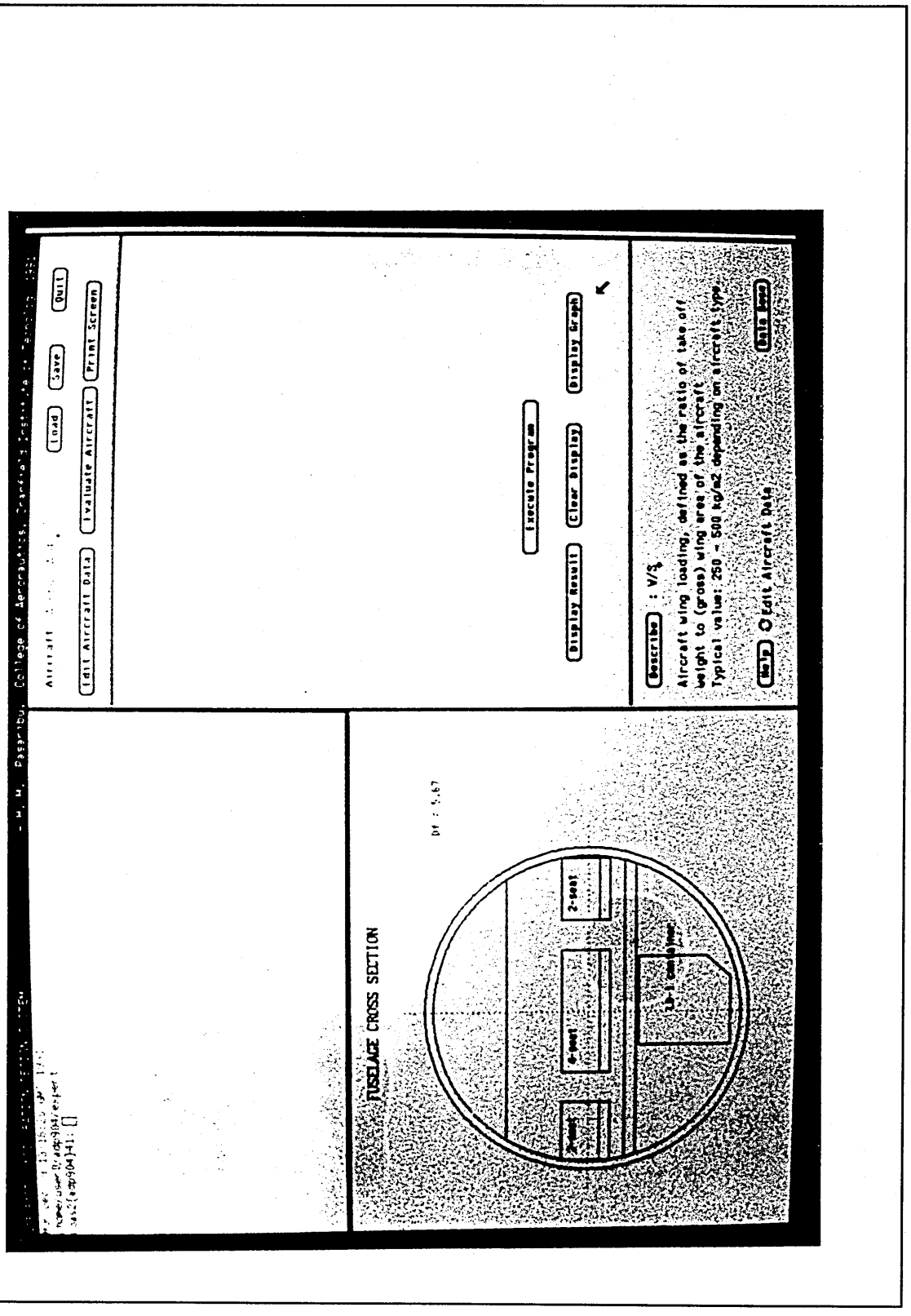


Fig. E.3 The aircraft design window in execution mode

E.2.1.1 The Main Control Panel

The Main Control Panel is located at the top right of the Aircraft Design window. It consists of one panel text item and several panel buttons which will perform the corresponding operation when activated. To activate a specific button, follow *Procedure 1* in section E.1.2.1. The following describes the functions associated with each panel buttons :

Load Aircraft

This button will enable the user to retrieve data of a specific aircraft for editing. The name of the aircraft can be specified on the '**Aircraft :**' panel text item to the left of the button. By default the aircraft name is set to 'Boeing-757' and the data of that aircraft is on display. To change the aircraft's name, place the cursor at the current name and click the LEFT MOUSE BUTTON. Overwrite the current name with the new name. To load the specified aircraft data, position the cursor at the '**Load Aircraft**' button and click the LEFT MOUSE BUTTON. The data will then be displayed on the Module Interface Panel below the Main Control Panel. The data on display are only those associated with the selected component specified on the Module Interface Panel even though the whole data on file are actually retrieved into memory (see 'Selecting A Module :' item on section E.2.1.2.1).

Note :

An appropriate data file should exist in the file directory. This file should have the same name as the name of the aircraft specified in the '**Aircraft :**' item with extension name '.data'. If this file is not available, a new aircraft will be assumed and file 'newaircraft.data' is loaded into the system. If this occurs, the system will display a message to inform the user of the situation. The user can then edit the data to suit the requirement.

Save Aircraft

This will enable the user to save the active data into a file. The data will be stored in a file with extension name '.data'. The name of the file is the same as specified on the '**Aircraft :**' item. To store date in a completely new file, just overwrite the current name on the '**Aircraft :**' item with the new name, and click the LEFT MOUSE BUTTON on the '**Save Aircraft**' button. This is necessary when the user wants to work on new aircraft.

Quit

This enable the user to exit from the system. A system message will be displayed for the user to confirm quitting.

Edit Aircraft Data

This will activate the editing mode which enables the user to change aircraft data on display (see 'Editing Mode' on section E.2.1.2.2). By default the editing mode is activate when the system is loaded.

Execute Aircraft

This will activate the execution mode which enables the user to control the execution of the selected program (see 'Execution Mode' on section E.2.1.2.3).

Print Screen

This function will enable the user to print the current screen layout on the laser printer. Unfortunately, since no laser printer is currently connected to the computer, no action takes place when this function is activated. A message will be displayed to inform the user of the situation.

E.2.1.2 Module Interface Panel

The Module Interface Panel is situated right below the Main Control Panel. The panel is the interface between the user and the supporting programs. The actual layout of the panel depends on the program selected and the active mode of operation. The supporting program module can be selected from the following :

- Parametric Analysis
- Configuration Analysis
- Fuselage Design
- Wing Design
- Engine Selection
- Weight Estimation
- Aerodynamic Analysis
- Flight Analysis
- Landing Gear Design
- Performance Analysis
- Stability and Control Analysis
- Cost Estimation
- Design Optimisation

Note :

The list of computer programs integrated with the system is contained in the Program Library file called '**program.nme**'. This file contains the following information :

- The name of the module in which the program applies
- The name of the executable file of the program
- The file directory in which the program is installed
  of all the programs available.

The order in which the modules are presented is determined by the order in which the modules are listed in the file. A change in the order of the list will be reflected in the order of the modules displayed in the system.

When the system is loaded, it consults this file to identify the available modules. At the moment a maximum of 15 modules can be allocated into the system of which 13 modules are already occupied as listed earlier. The user can add a completely new module by inserting it into the file with all the necessary information. The user however needs to supply the information about the required input data.

The list of parameters required as input for each programs is given in file '**input.dba**'. This file contains the following information :

- The name of the executable file of the program (should be the same as in file '**program.nme**')
- The list of parameters required as input for that particular program. The list is closed with word '**end**' to notify the system that the list is completed.

The order in which the programs are listed in the file is not necessarily the same as that in file '**program.nme**'. However, the list of input parameters should immediately follow the name of the program for which the list applies.

There are two modes of operation available for each module, they are :

- Editing Mode
- Execution Mode

The following will present the description of each mode.

E.2.1.2.1 Editing Mode

This mode is activated by clicking the LEFT MOUSE BUTTON on the '**Edit Aircraft Data**' button on the Main Control Panel. This mode will enable the user to edit data on display. As explained earlier, only the data required for the selected module are displayed. The layout of the Aircraft Design Window in Editing Mode is shown in Fig. E.4.

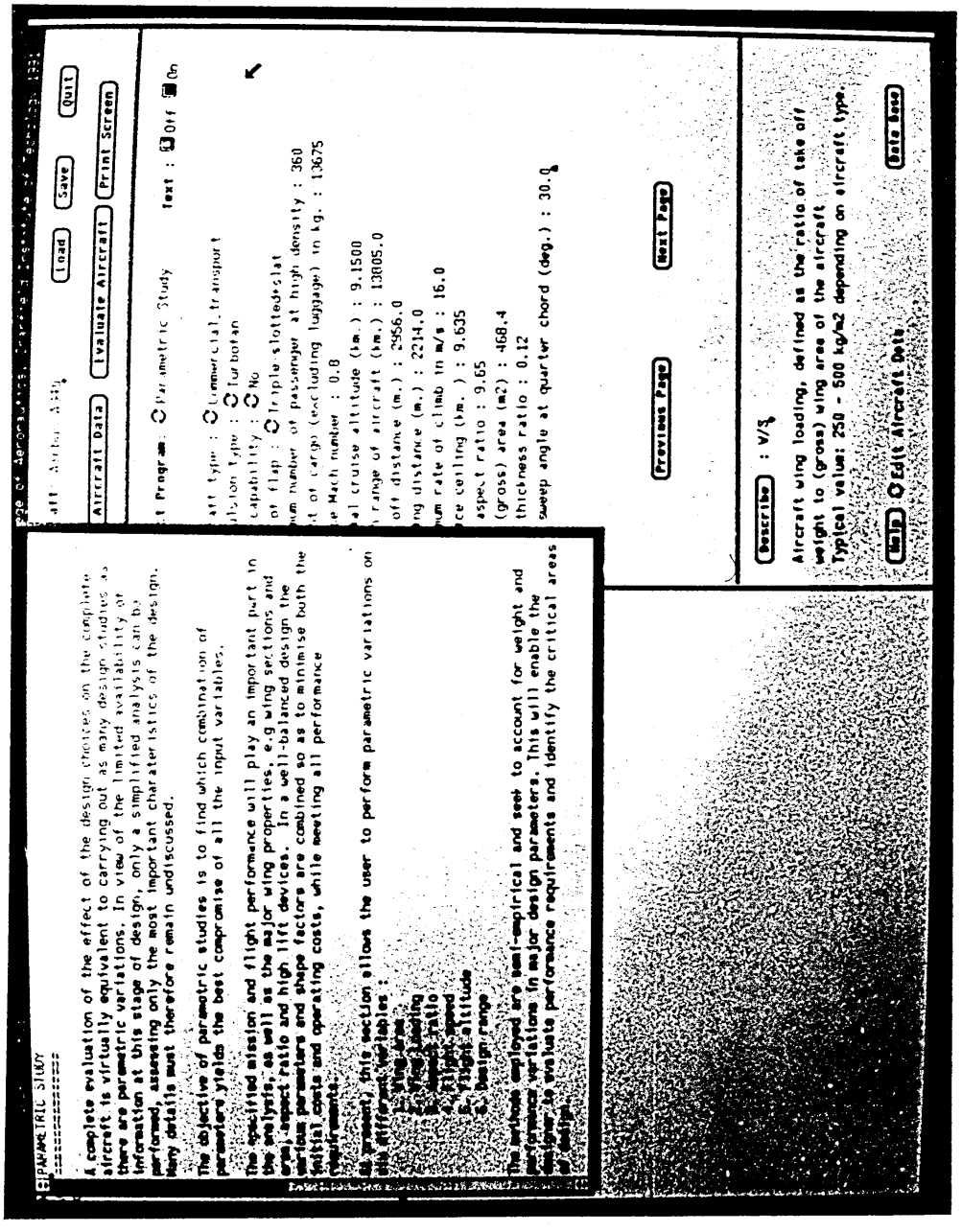


Fig. E.4 The aircraft design window in editing mode

Selecting A Module :

The selection of a module can be done through the **'Select Program :'** item at the top of the Module Interface Panel. This item is a selectable panel item in which the selection can be made by following *Procedure 2* in section E.1.2.2.

Each time the selected module is changed the data displayed on screen will be replaced with the ones correspond to the selected module. This will allow the user to edit only the data required for the particular module he/she is working on. However if the value of any particular data is updated the change will be kept for all modules to ensure that that particular value will used wherever the data is active. This change however will not be recorded into the data file unless the data is saved with the **'Save'** button.

The module can only be selected while the system is in editing mode. This is to prevent the user to execute any module without preparing the correct input data.

Editing Existing Data

The editing mode allows the user to edit data for a particular program module. The list of data applicable to the current module is display right below the **'Select Program :'** item. There are two different types of data :

- Symbolic data
- Numerical data

Symbolic data are non-numeric, represented in the panel as panel choice items. This representation with the 'Cycle' symbol is used to remind the user that the data can be selected from a list of terms acceptable to the system. This type of data can be changed in the same way as *Procedure 2* in section E.1.2.2.

The numerical data are represented as panel text items and can be edited by overwriting the current value with a new value. To do this, follow *Procedure 3* in section E.1.2.2.

In most cases there are more data required for a particular module than those currently on display. If that is the case, the **'Next Page'** and **'Previous Page'** buttons will allow the user to display the remaining data. Clicking the LEFT MOUSE BUTTON while the cursor is on the **'Next Page'** button will allow the user to display the next list of data, while the **'Previous Page'** will reverse the action, i.e. to go back to the previous list of data on display for the selected module. A warning will be given to inform the user if the bottom or top page of data is reached.

On-line Help

An on-line help is available for each module while on editing mode. This on-line help

gives explanation about the selected module. To activate the on-line help, switch the **'Text'** item on by placing the cursor on the box to the left of the 'On' and click the LEFT MOUSE BUTTON. Notice that the inside box will be filled to notify the user that the help mode is on. This will display the help file in a Text Editor window. To find more about the operations on this window see 'Text Editor Window' on section E.2.4. As shown in Fig. E.4, the on-line help for the editing mode is on. The on-line information is displayed in the text window at the top left of the screen.

E.2.1.2.2 Execution Mode

This mode enables the user to control the execution of the selected program module. This mode is activated by clicking the LEFT MOUSE BUTTON on the **'Execute Aircraft'** button (*Procedure 1* on section E.1.2.1).

The actual layout of the Module Interface Panel while it is on execution mode depends very much on the module selected. Two of the examples are shown in Fig. E.3, showing the execution mode for Fuselage Design module, and Fig. E.5 for Parametric Analysis module. In general the user can interact with the execution of the module through the functions of several buttons. Follow *Procedure 1* to activate the buttons.

Execute Program

This will execute the program corresponding to the active module. In general the system will look for the appropriate path of directory in which the program is located as specified by the Program Library file (file **'program.nme'**), translate the input data into a format acceptable to the program, executes the program and write the output for presentation. All these activities can be traced in the TTY sub-window at the top left of the Aircraft Design Window on which the computer operating commands are printed sequentially. Some application programs may run in interactive mode in which user may need to reply for questions being asked by the program. The user input can be entered in the TTY sub-window.

Display Result

This will present the output file of the selected program in the Text Editor sub-window. This enables the user to assess the result of analysis carried out in the program.

Clear Display

This will clear the display of the output file and closes the Text Editor sub-window from screen.

Display Graph

This will display the output of the execution in a graphical form, if any. The graph is presented in the Graph sub-window located at the bottom left of the Aircraft Design Window as shown in Fig. E.5.

E.2.1.3 Help Control Panel

This panel consists of three buttons which will activate the main on-line help functions. The functions are enabled by clicking the LEFT MOUSE BUTTON on the corresponding buttons.

Describe

This gives an explanation for each aircraft parameter used in the system. To request this information on a specific parameter, execute the following procedure :

1. Place the cursor at the current parameter and click the LEFT MOUSE BUTTON.
2. Overwrite the current parameter with the parameter in question, say, Wto.
3. Move the cursor to the **'Describe'** button and click the LEFT MOUSE BUTTON. The available information on the requested parameter is displayed on the panel space just below the **'Describe'** button.

   For parameter Wto, the following will be displayed :
   Maximum take-off weight of the aircraft, which is the sum of payload, empty weight and fuel weight. Unit : kg.

Help

This gives an explanation on the main operations of the Aircraft Configuration Design System. An on-line explanation can be requested for each panel button on display on the Aircraft Design Window. This explanation is presented one at a time as selected through the help item on the right of the **'Help'** button.

To display this explanation, follow this procedure :

1. First select the item on which the explanation is requested. The selection can be done by either one of the following :
   - Place the cursor on the current item and click the LEFT MOUSE BUTTON to advance to the next item in the list. Continue clicking several times until the requested item is displayed.
   - Or, place the cursor on the current item, then click and hold the RIGHT MOUSE BUTTON to bring the list menu on display. While holding the BUTTON down, drag the BUTTON until the requested item is highlighted

and release the BUTTON to select.

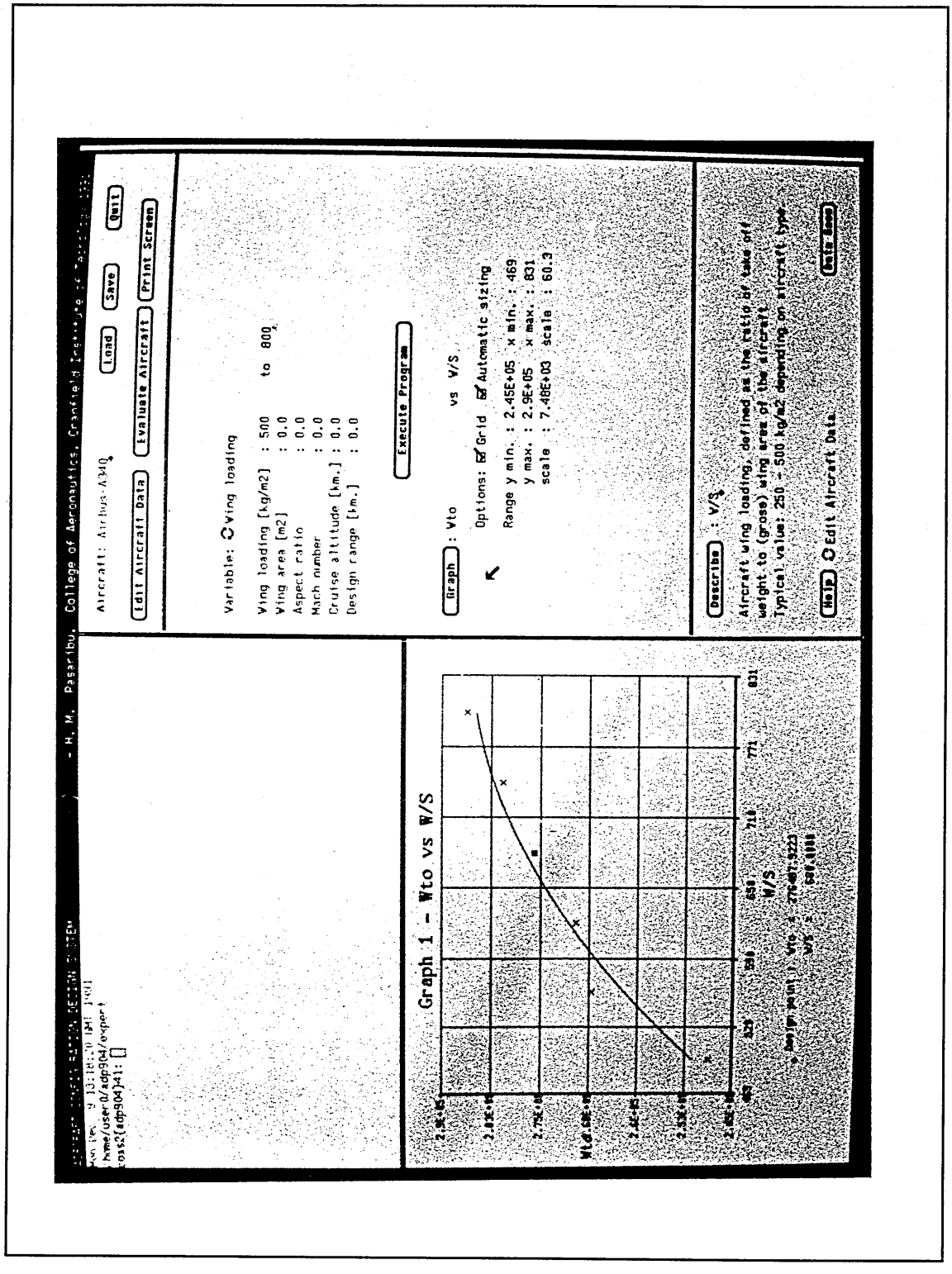


Fig. E.5 The aircraft design window in execution mode for the parametric analysis module

2. Place the cursor on the **'Help'** button and click the LEFT MOUSE BUTTON. The requested explanation will be displayed in a pop-up window.

The information is displayed in an alert window as shown in Fig. E.6.

Database

This will activate the Aircraft Database System window which enable the user to obtain actual data of several existing aircraft in the database. More detail description of this facility is given in section E.2.2.

E.2.2 AIRCRAFT DATABASE SYSTEM WINDOW

The Aircraft Database System is developed to maintain an archive of data of existing aircraft. The data should be organised in such a way that they can be easily retrieved when required. This database system is part of the on-line help and can be used to obtain a comparable value for a specific data which may be needed for a particular application. The system, in its simplest form, supports all the main functions of a database management system, i.e data retrieval, editing, and saving. It also enables the user to build the database by adding new aircraft to the database.

The database system consists of three sub-windows as follows :

- Database Control Panel at the top of the window frame
- Data Display Panel right below the Control Panel
- Database Help Panel at the bottom of the window.

The following sections describe each of the subwindows.

E.2.2.1 Database Control Panel

This Control Panel contains a panel text item and several panel buttons that specify the main functions of the database system, they are :

Aircraft Item

The **'Aircraft :'** item at the top left of the panel is a panel text item and is used to specify the name of aircraft whose data is to retrieved. To specify the name of the aircraft, overwrite the current name with the new name by following *Procedure 3* in section E.1.2.2.

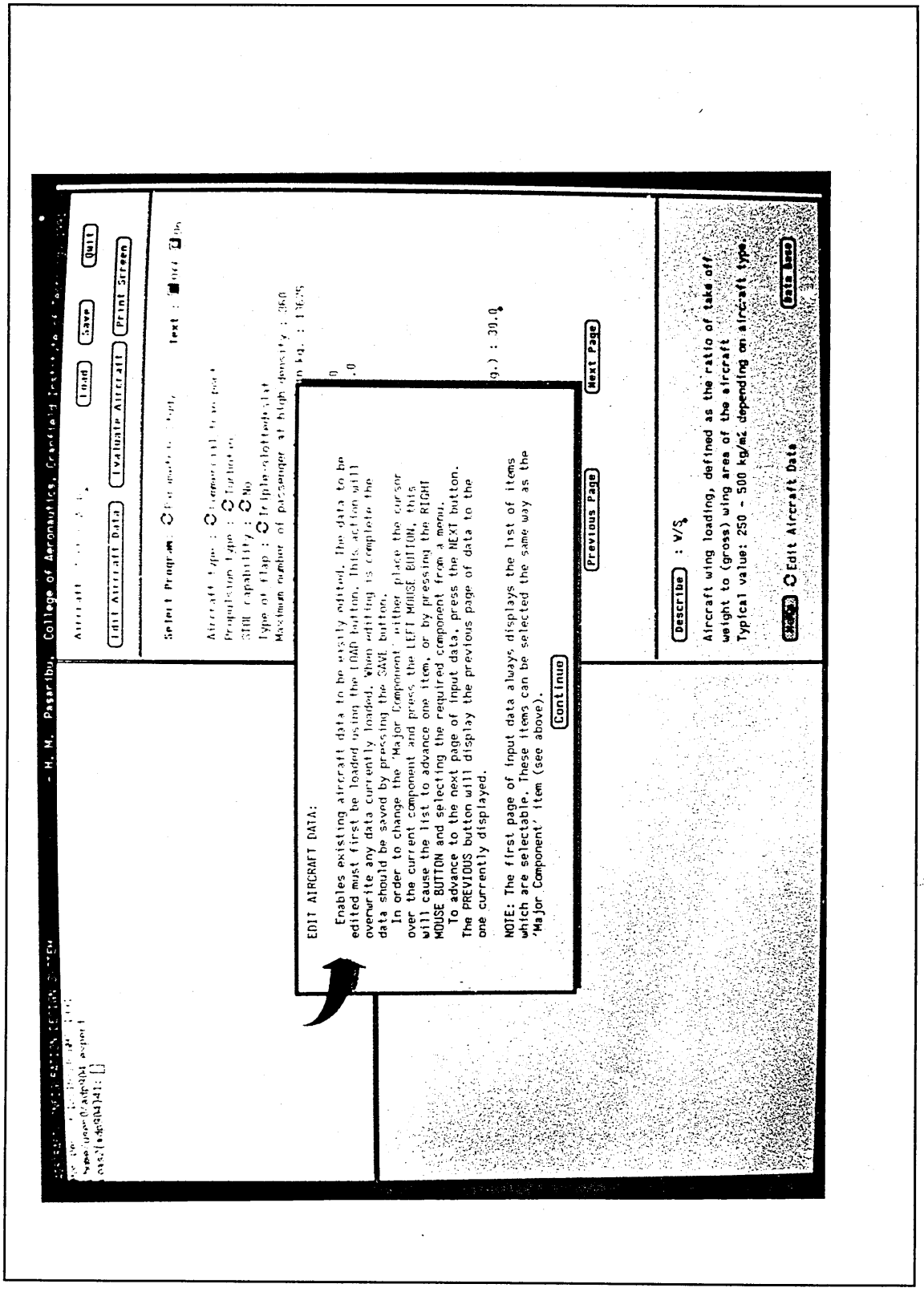


Fig. E.6 The help facility in the aircraft design window

Load Aircraft

This will enable the aircraft data file with the name specified by the 'Aircraft :' item to be retrieved into the system. To do this, follow *Procedure 1* in section E.1.2.1. The aircraft data will be displayed on the Display Panel.

Edit Aircraft Data

This will enable the user to update the data on display. There are two different types of data : symbolic data and numerical data. Each type has a specific way to edit (see 'Editing Existing Data' on section E.2.1.2.1).

To add new aircraft to the database, simply edit all the current data with the new data and save with a new name (see **'Save Aircraft'** in the following).

Save Aircraft

This will enable the user to store the data record into a permanent file which can be retrieved for future use. To store the data on display into a file with the name as specified by the **'Aircraft :'** item, follow *Procedure 1* in section E.1.2.1. To save file with a new name, simply change the current name with the new one before saving it.

Quit

This enable the user to exit from the database system and close the window.

E.2.2.2 Data Display Panel

This panel is used to display data of the selected aircraft. The data is displayed in groups as follows :

- General data
- Fuselage - External
- Fuselage - Internal
- Fuselage - Flight deck
- Passenger Data
- Wing Data
- Horizontal Tail
- Vertical Tail
- Control Surfaces
- Propulsion
- Landing Gear

- Performance
- Weights
- Aerodynamic Data
- Costs

This classification is intended for easy access. The group to which the displayed data belongs is shown in the **'Major Component :'** item at the top of the Display Panel. To display the data associated with a particular group, take either one of the following actions (consult *Procedure 2* in section E.1.2.2) :

- Move the cursor to the current item of the **'Major Component :'** and click the LEFT MOUSE BUTTON once to advance to the next item on the list. Continue clicking several times until the required item appears. In every instance the corresponding data will be displayed on the screen. This will enable the user to scan through the database in search for a particular data.
- Or, position the cursor on the current item of the **'Major Component :'**, then press and hold the RIGHT MOUSE BUTTON to pop up the menu. While holding the MOUSE BUTTON down drag the MOUSE until the required item is highlighted. Release the BUTTON to select the highlighted item. This procedure is normally used to quickly go to the required group without scanning through the whole database.

E.2.2.3 Database Help Panel

The Database Help Panel is used to provide on-line help on two things :

- the operations of the database system
- brief explanations of the symbolic data. Information is available for each valid text value for each type of symbolic data which concerns the configuration of the aircraft.

The information provided is classified in groups as follows :

- Database Functions
- Wing Position
- Propulsion Type
- Engine Type
- Engine Position
- Fuselage Layout
- Lifting Surface Configuration
- Tail Type
- Tail Layout
- Flap Type
- Landing Gear Type
- Landing Gear Position
- Passenger Seats

- Cargo Containers

The active group is displayed on the right of the **'Help'** button. The

To use this help facility, take the following the procedure :

1. Select the group to which the required information belongs, by either one of the following steps :
   - Move the cursor to the current group item on the right of the 'Help' button and click the LEFT MOUSE BUTTON once to advance to the next item on the list. Continue clicking several times until the required item appears. In every instance the corresponding text item will be displayed on the right. This will enable the user to scan through the list in search for a particular group.
   - Or, position the cursor on the current group item on the right of the 'Help' button, then press and hold the RIGHT MOUSE BUTTON to pop up the menu. While holding the MOUSE BUTTON down drag the MOUSE until the required group item is highlighted. Release the BUTTON to select the highlighted item.

   Notice that the text item on the right of the selection changes every time the group is changed.

2. Select the item of the group on which the information is requested, again by either one of the following steps :

   - Move the cursor to the current item on the right of the **'On :'** item and click the LEFT MOUSE BUTTON once to advance to the next item on the list. Continue clicking several times until the required item appears. This will enable the user to scan through the list in search for a particular text item.
   - Or, position the cursor on the current item on the right of the **'on :'** item, then press and hold the RIGHT MOUSE BUTTON to pop up the menu. While holding the MOUSE BUTTON down drag the MOUSE until the required text item is highlighted. Release the BUTTON to select the highlighted item.

3. Once the item is selected, move the cursor to the 'Help' button and click the LEFT MOUSE BUTTON to display the available information on the item.

E.2.3 THE CONFIGURATION ANALYSIS WINDOW

This window is activated every time the Configuration Module is executed (see 'Execution Mode' on section E.2.1.2.2). It is used to control the execution of the configuration analysis process. The window consist of two sections :

- The Display Sub-window
- The Configuration Control Panel.

as shown in the foreground of Fig. E.7.

E.2.3.1 The Display Sub-window

The Display Sub-window is located at the top of the window. It is actually a Canvas on which text is written. It is mainly used to display the result and the reasoning process involved in each design step.

E.2.3.2 The Configuration Control Panel

This panel is located at the bottom of the window. This layout is selected to enable the user to control the execution without any clutter on the screen caused by overlapped windows. The panel consists of a **'Design Step :'** panel choice item and several panel buttons. The buttons are activated by following *Procedure 1* in section E.1.2.1 :

- Move the cursor to the button,
- Click the LEFT MOUSE BUTTON to enable the operation.

Design Step :

This item shows the design step which is about to be executed. Once the execution of the current step is completed the item changes automatically to point to the next step on the queue. The user however can select the design step manually by either one of the following steps :

- Place the cursor on the current item of the **'Design Step :'** and click the LEFT MOUSE BUTTON to advance to the next step on the queue. Click the BUTTON a few times until the required step is displayed.
- Or, position the cursor on the current step of the **'Design Step :'**, then press and hold the RIGHT MOUSE BUTTON to pop up the list of steps on the queue. While pressing the BUTTON, drag the MOUSE until the required step is highlighted. Release the BUTTON to select the step.

Execute Step

This will initiate the inference procedure to look for solutions for the step selected.

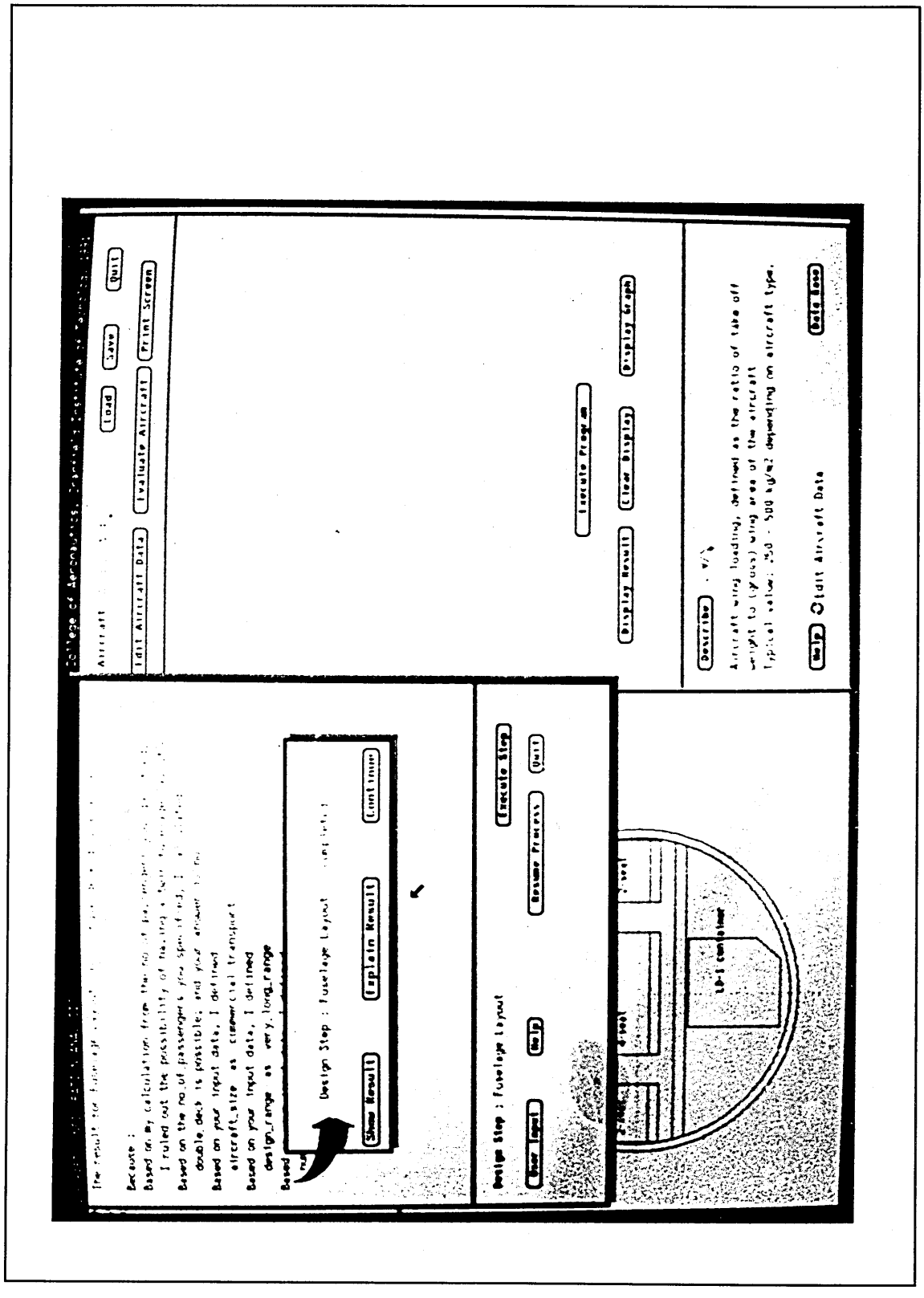


Fig. E.7 The aircraft configuration window

User Input

This button will activate the enquiry for the required data from the user. The data will be used as additional information to help solve a particular problem.

Suspend Step

This button will suspend the execution of the current step. The system will place the step back on the queue, call the next step from the queue and set it as the current design step.

Quit

This button enables the user to quit from the Configuration Analysis and close the window. This action however requires a confirmation from the user.

E.2.4 TEXT WINDOW

The Text Window is a Frame used to bring a file text into display. Its main use are :

- To display the information on the selected program module activated by switching the **'Text'** item on (see 'Selecting A Module' on section E.2.1.2.1), and
- To display the output file of the selected module, by activating the **'Display Result'** panel button (see 'Execution Mode' on section E.2.1.2.2).

The format of a Text Window is shown in the foreground of Fig. E.4.

The user can scan through the contents of the text window by scrolling the page up and down. These operations can be done with the help of the MOUSE.

To Scroll The Page Down

Use the LEFT MOUSE BUTTON to scroll the text down :

1. Place the cursor on one of the arrows situated at the top of the scroll bar to the left of the window.
2. Click the LEFT MOUSE BUTTON once to scroll the text down one line. To quickly scroll the text down, press and hold the LEFT MOUSE BUTTON. The scrolling stops once the BUTTON is released.

To go to the bottom of the text file, click the MIDDLE MOUSE BUTTON while the cursor is on one of the arrows at the top of the scroll bar.

To Scroll The Page Up

Use the RIGHT MOUSE BUTTON to scroll the text up :

1. Move the cursor to one of the arrows at the bottom of the scroll bar situated at the left frame of the window.
2. Click the RIGHT MOUSE BUTTON once to scroll the text up one line. Pressing and holding the RIGHT MOUSE BUTTON will quickly scroll the text up until the BUTTON is released.

Moving The Window

The Text Window can be replaced to a certain position on the screen by following this procedure :

1. Position the cursor at the top frame of the window,
2. Press and hold the RIGHT MOUSE BUTTON to pop up the frame menu,
3. While holding the BUTTON down, drag the MOUSE until the item 'Move' is highlighted and then release the BUTTON,
4. Move the cursor to one of the frame corners of the window,
5. Press and hold the MIDDLE MOUSE BUTTON, drag the window with the MOUSE to the new position and release the BUTTON.

Adjusting The Size of The Window

To adjust the size of the Text window, take the following steps :

1. Position the cursor at the top frame of the window,
2. Press and hold the RIGHT MOUSE BUTTON to pop up the frame menu,
3. While holding the BUTTON down, drag the MOUSE down until the item 'Resize' is highlighted. Drag the MOUSE to the right to highlight the pop-right menu item 'Unconstrained' and then release the BUTTON,
4. Move the cursor to the bottom left corner of the window frame,
5. Press and hold the MIDDLE MOUSE BUTTON, drag the MOUSE to adjust the window to the new size, and release the BUTTON.

Closing the Window

To close the Text Window follow this procedure :

1. Move the cursor to the top frame of the window,
2. Press and hold the RIGHT MOUSE BUTTON to pop up the frame menu,
3. While holding the RIGHT MOUSE BUTTON down, drag the MOUSE until the item 'Done' is highlighted and release the BUTTON.

Alternatively, the window can be closed through the operation on the appropriate panel button on the Program Interface Panel (see sections E.2.1.2.1 and E.2.1.2.2).

E.3 THE AIRCRAFT DATABASE

The general data of the aircraft database system is recorded in a file called **'aircraft.data'**. This file contains all names of aircraft data included in the database. The data is listed with the following pattern :

Data symbolic name	Data type	Description
Wto	real	Take-off weight of the aircraft (kg.) :
actype	menu	Aircraft type : Commercial_transport Commuter Business/executive end
Sw	real	Wing (gross) area (m2) :

The data is identified by its symbolic name. The symbolic name should be unique for each data.

The data type specifies the acceptable value of the data. Data of type menu accepts a round number, either integer or real. The list of choices for the menu data has to follow the description. The list should be terminated with 'end'.

The data description has to be terminated by a colon. It can take up to 80 characters including the colon. There should be blank space between the last word of the description and the colon itself.

There is no proper ordering of the data listed in the file. New data can be added easily any where in the list as long as it does not disturb the pattern of the recording of data as shown earlier.

The database specific to a particular aircraft is recorded in a file with extension name **'.data'** and a name the same as the name of the aircraft. For example, the database that belongs to the Airbus-A340 aircraft is stored in a file called **'Airbus-A340.data'**. This file takes almost the same format as the **'aircraft.data'** file :

Data symbolic name	Data value	Description
Wto	255000.0	Take-off weight of the aircraft (kg.) :
actype	1.0	Aircraft type :
Sw	363.2	Wing (gross) area (m2) :

with the data value replaces the data type. For data of type menu, the list of the choices is not recorded in this data file. The value specifies the index of the choice in the list. Again, the data is randomly recorded. The order is necessarily the same as that in the **'aircraft.data'** file.

When loaded, the system will consult the **'aircraft.data'** file and will load the data file with the name as specified in the **'Aircraft :'** item. The symbolic names from both data will be matched to correspond the data and the value.

APPENDIX F
DETAILS OF APPLICATION PROGRAMS IN THE ENGINEERING MODULES

F.1 INTRODUCTION

This apendix contains detailed information on each application program included in the engineering modules. The presentation is given in the following order :

- OPDOT (Optimum Preliminary Design Of Transport Aircraft)
- FUSELAGE
- WEIGHT ESTIMATION
- DELTA PROGRAM (Drag Estimation)
- ACCOST (Aircraft Cost Estimation)
- FLTANAL (Flight Analysis)
- MITCHELL (Stability And Control Analysis)

F.2 OPDOT (Optimum Preliminary Design Of Transport Aircraft)

F.2.1 OPTIMISATION CODE

OPDOT (Optimisation of Preliminary Design of Transports) [29,30] is a computer program written to perform preliminary design and evaluation of transport aircraft using nonlinear programming techniques. It utilises constrained parameter optimisation to minimise a performance index, in this case direct operating cost per block hour, while satisfying operating constraints. The approach in OPDOT uses geometric descriptors as independent design variables. This set of design variables is iterated upon until a minimum of a performance index which satisfies a series of constraint functions has been calculated. The design variables usually consist of geometry characteristics and mission parameters while the constraint function include regulatory performance requirements and handling quality design criteria

The general optimisation scheme of OPDOT program is shown in Fig. F.1. The optimisation is performed by a sequential simplex method which utilises a continuous penalty function for constraint violations. The penalty scheme applied is independent of the number of active constraints. The method, however, is slow to reach convergence in large regions of small gradients of the augmented performance function with respect to the independent design variables.

The general problem is formulated as follows :

Let the unaugmented performance index, **f**, be a function of the independent design

variables, **X**, and design parameters, **P**.

$$\mathbf{f} = \text{fcn}(\mathbf{X}, \mathbf{P}) \qquad \text{F.1}$$

and

$$\hat{g}_i = \begin{cases} 0 & \text{if } c_i \geq S_{\ell i} \text{ and } c_i \leq S_{ui} \\ S_{\ell i} - c_i & \text{if } c_i < S_{\ell i} \\ c_i - S_{ui} & \text{if } c_i > S_{ui} \end{cases} \qquad \text{F.2}$$

then

$$\mathbf{F} = \mathbf{f} + \sum K \left(\hat{g}_i / N_i \right)^2 \qquad \text{F.3}$$

The goal is to find the minimum of the augmented function, **F**, with the gains, **K**, large.

A variable transformation is used to automatically scale the variables and apply "side" constraints, which are inequality constraints applied directly on the design variables. This results in a reduction in the number of iterations required for convergence.

The form of the transformation is as follows :

$$X_i = \frac{V_{ui} - V_{li}}{2} \sin\left(\frac{\pi}{2} Z_i\right) + \frac{V_{ui} + V_{li}}{2} \qquad \text{F.4}$$

where V_{ui} and V_{li} are the upper and lower boundaries of the ith independent variables. So the simplex optimiser iterates on the transformed variables **Z** which spans the set of the allowable values of the independent design variables with the range in Z of 1 to -1. This allows consistency in step size selection and limits the allowed values of the independent design variables.

F.2.2 EVALUATION OF THE UNAUGMENTED PERFORMANCE INDEX

A flow diagram for evaluating the unaugmented performance index is shown in Fig. F.2.

First of all, DOCOST calls subroutine GEO to calculate the geometry of the aircraft from user input data and design variables, and subroutine CGCAL to calculate centre of gravity locations. Having obtained the required geometry and c.g. location, subroutine WEIGHT is called to calculate component weights.

Subroutine WEIGHT estimates the weight of aircraft components based mainly on the empirical methods described in Nicolai [52]. The empty weight of the aircraft is calculated as the summation of component weights. By assuming an initial estimate of the total weight, the total fuel weight required to cover the specified range and reserve is calculated by iteration until the difference in the total weight is less than 0.5 lbs.

Subroutine FUELCAL calculates the total fuel weight required to fly the required range including reserve with the flight procedure as shown in Fig. F.3. It calls subroutines CLIMB, CRUFUEL and DESCENT subsequently to calculate the flight segments corresponding to climb, cruise and descent flights respectively. A modification was made from the original program to enable the user to set the initial cruise altitude and reserve flight conditions.

The climb portion of the flight is estimated in subroutine CLIMB which is a new addition to the original program. The climbing flight is divided into several segments and each segment is carried out at constant equivalent airspeed of 250 KTS. The climb procedure implemented in subroutine CLIMB is shown in Fig. F.4. The distance covered during climb is substracted from the range specified. The climbing flight ends when the initial cruise altitude is reached. The distance covered during climb is subtracted from the range specified.

The cruise portion of the flight is divided into 10 equi-distant segments for the remaining range. Subroutine CRUFUEL is called to calculate the fuel weight and time required to fly the segment. The flight procedure performed during cruise in subroutine CRUFUEL is shown in Fig. F.3. During the first segment, the aircraft flies to reach the economical cruise lift coefficient C_{LB^*} at cruise Mach number specified. Based on a simple drag polar, it is estimated that

$$C_{LB^*} = 0.79\ C_{L\ at\ (L/D)max}$$

Segments 2-5 are flown at lift coefficient for B^* at constant Mach number.

Segment 6 is flown to reach lift coefficient for 98% of maximum lift to drag ratio. Segments 7-10 are flown at lift coefficient for 98% of maximum lift to drag ratio. The lift coefficient for these segments is estimated as

$$C_L = 0.9\ C_{L\ at\ (L/D)max}$$

At the end of each segment, the total lift and weight of the aircraft in subroutine CRUALT is matched to obtain the cruise altitude at the end of the segment.

The flight range of segment 10 is set equal to the difference between the remaining range to be flown and the range covered during descent flight. The descent flight is carried out in subroutine DESCENT in which a constant descent angle of 0.025 rad is maintained. The flight range covered during descent is calculated as follows,

$$R_{DESCENT} = H_{cc} / \tan \gamma$$

where H_{cc} is the cruise altitude at the end of cruise (segment 10), and γ is the descent gradient angle of 0.025.

At each flight condition, subroutine XLOD are called to calculate the aerodynamic performance of the aircraft. Subroutine XLOD calls subroutine CDZL to calculate the parasite drag of the aircraft by assuming flat plate analogy. CDZL calculates the skin-friction drag of each component for the corresponding Mach and Reynolds number. It then calls subroutine STABCOD to estimate stability and control derivatives at the indicated flight configuration by using methods derived in Reference [53]. These derivatives are applied in subroutine TRIM. Having obtained the parasite drag and trimmed lift coefficient, XLOD calculates the induced drag and interference drag by using methods described in References [2] and [53].

TRIM evaluates the trim equation of the aircraft to calculate the tail and wing lift coefficient by iterative until the difference in tail lift coefficient is less than 0.003. The trim equation for canard and three-surface aircraft is obtained from References [34], [35], and [36].

Having known the aircraft geometry and weights, DOCOST calls subroutine AIRCOST to calculate airframe and engine total costs incurred during development and production phase [52]. It then calculates the price of the aircraft for the specified year of evaluation. A more detailed description is given is section F.6.1.

DOCOST then calculates the direct and indirect operating costs, by applying methods described in Reference [44].

F.2.3 EVALUATION OF DESIGN CONSTRAINTS

The program has 53 contraint functions that can be applied to transport aircraft design. The designer specifies an upper and lower boundary for each function as input. The program does a test on all constraint lower boundaries, except if -999 is input for the lower boundary of a constraint function, the constraint is excluded in the penalty function even if it is a violation.

Subroutine CNSTRN calculates the values of all constraint functions. The first set of constraints (constraints no 1-12 and 53) are design constraints which restrict the design to avoid infeasible geometry or to ensure satisfying performance regulations and mission requirements.

The second set is used to study tail size and the impact of flying quality design criteria upon transport aircraft with relaxed static stability augmentation system. The flying quality analysis is started by trimming the aircraft in approach configuration with an altitude of 150 meters by calling XLOD. The non-dimensional stability derivatives for

cruise and approach, which are stored in the database, are converted to dimensional stability derivatives by subroutine DIMDER. The characteristic equation for the fourthorder longitudinal set of equations is assembled by subroutine LONGRT. The four roots are determined by using subroutine RPOLY which utilises a NAG subroutine for finding roots of polynomials.

A pitch-attitude-hold with pitch-rate-command autopilot was selected as a conservative estimate of an augmentation system. The aircraft is arbitrarily augmented to have : $\omega_{sp}^2/(n_z/\alpha) = 1$ and $\zeta_{sp} = 0.7$. The feed back gains K_θ and $K_{\dot{\theta}}$ can be calculated by utilising the short period approximation to the longitudinal dynamics.

$$K_\theta = \omega_{sp}^2 \frac{1 - M_{\dot{w}} U_o}{M_\delta + M_w} + K_{\dot{\theta}} \frac{M_{\dot{w}} M_\delta}{M_\delta + M_w} \omega_{sp}^2$$

$$K_{\dot{\theta}} = 2\zeta_{sp}\omega_{sp}(M_{\dot{w}} U_o - 1) - M_q - M_w U_o + M_{\dot{w}} \omega_{sp}^2 \frac{(1 - M_{\dot{w}} U_o)}{(M_w + M_\delta)} \qquad \text{F.5}$$

$$+1 - \frac{M_{\dot{w}}^2 M_\delta \omega_{sp}}{(M_\delta + M_w)} + 2\zeta_{sp}\omega_{sp} M_w M_\delta - M_\delta - M_w M_\delta$$

The gain are then used to estimate the variance of the elevator position and elevator position rate in cruise and approach. The turbulence is assumed to have a characteristic length of 760 meters (2500 ft.) with a RMS gust load of 0.9 and 2.13 m/sec (3 and 7 ft/sec) in cruise and approach, respectively. These autopilot calcualtions are used to assign the following quantities in cruise and approach to available constraint functions: K_θ, $K_{\dot{\theta}}$, σ_δ and $\dot{\sigma}_\delta$. The constraint functions are used to ensure that enough aerodynamic control exists to stabilise the aircraft to excellent flying qualities and that enough hyraulic capability is available to prevent control surface rate saturation in heavy turbulence.

F.2.4 MODIFICATIONS IN VERSION 2.0

The original program [29] was thought to be somewhat restricted in many areas, in that analysis is tailored for a specific aircraft, i.e a 200-passenger transport aircraft cruising initially at 36000 ft for a range of 3000 nautical miles. Several modifications were attempted to make the program more general purpose. These are mainly on subroutines estimating weight (WEIGHT, FUELCAL) and aircraft costs (DOCOST, AIRCOST, MAINCST). Component weight estimation methods were slightly modified with reference to Ref. [52]. Equations used in subroutine AIRCOST were updated using information obtained in Refs. [2] and [53]. Subroutine MAINCST were updated by

applying the original equations given in Ref. [44].

Several new routines were required to complete the analysis. Subroutines CLIMB and DESCENT were developed and integerated into the program to calculate fuel consumption, time required and range covered in climb and descent phase of the flight, respectively. This in turn removed the assumptions made in the original program about these flight phases.

Although attempts were made to make subroutines DOCOST and MAINCST more general and up to date as mentioned earlier, the empirical equations used are suitable only for transport aircraft with take-off weight of more than 100000 lbs (40000 kg). This restricted the capability of the program in that it was not applicable for commuter aircraft. For this reason, new routines for calculating direct operating cost of commuter aircraft were developed. These are subroutines ADOC, INST and INSTVFW. Two different subroutines are available to estimate maintenance costs of commuter aircraft, namely VFW methods (INSTVFW) and DLH methods (INST). The level of analysis in these new routines, however, is not so detailed as that for larger aircraft, but for the moment it is acceptable due to unavailability of other means.

The engine related parts of the program were also reviewed. The costs of engine system in subroutines MAINCST and AIRCOST were updated to cater for new engines, thanks to the engine cost data given in Ref. [4]. The engine module used in fuel weight and performance calculation were modified to enable the program to use different types of engines. To reduce modification to minimum, the format of the engine performance data is maintained by using generalised performance data : thrust to installed thrust ratio, thrust specific fuel consumption. This will enable the program to rubberise the engine, either to up-rate or down-rate, if necessary. At the moment five types of engines are available : CF6 (the original), CFM-56, Tay 710, Modified Tay 710 and Williams FJ44. Although it is necessary to include engines of a greater spectrum of thrust range, this could not be achieved due to inavailability of data. The same could be said for turboprops engines for commuter aircraft. Although performance data of a few turboprop engines were later obtained, the integration with the program has not been validated.

The other major changes from the original program are aimed to allow the evaluation of new technologies incorporated in the design in an optimal fashion. These include modifying the subroutines calculating drag (CDZL), trimmed lift coefficient (TRIM), aerodynamic performance (XLOD) and non-dimensional stability derivatives (STABCOD) to allow the evaluation of canard-wing and three-surface configurations. The equations applied were obtained mainly from References [34], [35], [36] and [37].

F.2.5 PROGRAM IN USE

The OPDOT is currently available in the College of Aeronautics VAX 11/750 computers. An attempt has been made to install a version of it for use in the Sun

workstations, but this is so far unsuccessful due to unavailibility of suitable library routines. The College's Sun workstations are not supplied with NAG FORTRAN library routines. An attempt to write similar subroutines, using the IEEE mathematical library routines that are available in the Sun, has been made but the accuracy is not up to the standard, which sadly degrades the performance of the programs. For this reason, the program has not been implemented as part of the Aircraft Configuration Design System.

The following gives a description of how to run the program in the College VAX computers.

The design variables are

1 Wing area
2 Wing aspect ratio
3 Fuselage length
4 Horizontal tail area
5 Horizontal tail aspect ratio
6 Total engine thrust
7 Aft c.g. limit

The program can also optimise aircraft of unconventional design such as that of canard-wing configuration by replacing the tail related variables with those of canard (variables 4 and 5), or three-surface configuration by adding two extra variables representing canard geometry.

For three-surface configuration the following variables are included :

8 Canard area
9 Canard aspect ratio

Additional design variables can be included by inserting the appropriate statements into the program :

10 Mach number
11 Wing sweep
12 Wing thickness ratio
13 Wing taper ratio
14 Fuselage diameter

Some 53 constraints are included in the program, of which the first 52 constraints are applicable for conventional tail-aft and canard-wing configuration. The 53th constraint is used only for optimising aircraft with three-surface configuration. Some are of design constraints such as

1	Cruise thrust requirement	
2	Second segment climb gradient	- thrust requirement
3	Missed approach climb gradient	- thrust requirement
4	Landing field length	- wing loading requirement
5	Take-off field length	- wing loading requirement
6	Landing gear	- aft c.g. limit

APPENDIX F

7	Passenger volume limit	
8	Cruise altitude	
9	Cruise altitude for $(L/D)_{max}$	
10,11	Wing lift coefficient	- cruise, approach
12	Tail aspect ratio limit	- AR_{tail}/AR_{wing}
53	Canard aspect ratio limit	- AR_{canard}/AR_{wing}

Handling quality constraints are

13,14	Static stability	- cruise, approach
15,16	Manoeuvre margin	- cruise, approach
17	Tail lift	- approach
18	Nose gear unstick	
19,20	Dynamic stability	- cruise, approach
21,22	Phugoid frequency	- cruise, approach
23,24	Phugoid damping	- cruise, approach
25,26	Short period frequency	- cruise, approach
27,28	Short period damping	- cruise, approach
29,30	Time to double	- cruise, approach
31,32	Time to half	- cruise, approach
33	Flight path stability	- approach
34,35	Vertical gain	- cruise, approach
36,37	Time sub theta 2	- cruise, approach
38,39	$Frequency^2$ / vertical gain	- cruise, approach
40	T(1) parameter	- approach
41,42	Ratio of mode frequencies	- cruise, approach
43-45	Elevator variance	- cruise
46-48	Elevator variance	- approach
49,50	Variance of elevator rate	- cruise, approach
51,52	Trimmed elevator deflections	- cruise, approach

Executing the Program

The program runs on CoA VAX in uninteractive mode. Before submitting the job into the system batch, the user needs to prepare the input file called OPDOT.DAT (see section below). This can be done by typing

$ *edit opdot.dat* [return]

The system will then display the editor's window, ready to receive input from the user. To save the contents of the editor's window the user just typed in, press *Ctrl-Z* simultaneously to get into the line mode and type

* *exit* [return]

The user is now ready to run the program.

APPENDIX F

To submit the program execution, type

$ *submit opdot* [return]

and wait until the system message appears on the terminal screen.

The program normally takes about 20 minutes to run. The output of the execution is written into a file called OPDOT.RES. To view the result on the screen, the user can load the file into the editor by typing

$ *edit opdot.res* [return]

To go out of editor without saving the contents of the file, the user need to press *Ctrl-Z* simultaneously and type

* *quit* [return]

in the editor's line mode.

For more information of how to run the program, consult Ref. [30]. Ref. [30] also includes the data required as input for the program.

F.3 FUSELAGE

FUSELAGE program is a program used for aircraft fuselage sizing. It was originally developed by the author [38] in Turbo Prolog for IBM PCs and compatibles under DOS operating system with the objective of applying the Knowledge Based System (KBS) techniques. It was later rewritten in FORTRAN for the College VAX and Sun workstations for use in the system. Unfortunately, some of the built-in functions of Prolog, mainly those related to the KBS, cannot be encoded in Fortran.

The program works from the inside to the outside. It works on the principle that the outside has to be bigger than the inside, in that fuselage should be able to house all items that needs to be placed in it. It first determines the internal size of the passenger cabin for a particular cabin layout specified by the user, as shown in Fig. F.5.

Having set the inside diameter of the cabin, the available cargo volume underneath the floor is examined to see if any standard cargo container can be fitted into. The user can take full control of the selection as to what to do with the underfloor cargo bay. Several options are available :

- To chop off a larger container than the ones that can be fitted into the available volume
- To add a second arc at the bottom surface to accomodate a larger container size
- To add a second arc at the bottom surface to flatten the fuselage lower surface if the

cargo volume underneath the floor is not required
- To increase diameter altogether to increase cargo volume
- To decrease diameter to reduce the unused space.

In any case the user may use a specially designed container if necessary.

Once the internal layout of the fuselage has been determined, the outside dimension can be calculated. The external dimensions of the fuselage can be estimated by adding the allowance for the width of the fuselage frame to the internal dimensions. The leght of the fuselage can be calculated as the total length of the passenger and cargo section, flight deck, unpressurised nose and tail section of the fuselage.

F.4 WEIGHT ESTIMATION

Weight estimation during the conceptual and preliminary design phases is mainly statistical. The actual weight of the aircraft cannot be calculated until the detailed design and production of each component are completed. However it is important to estimate the weight as accurate as possible.

The simplest method is the crude statistical technique used to initially estimate the take-off and empty weights from performance requirements. This method is normally employed in a parametric study for initial estimates. The take-off weight is estimated as follows :

$$W_{to} = C_1 \, W_{pl}^{C_2} \, R^{C_3} \, S_{to}^{C_4} \, V_{cr}^{C_5} \qquad \text{F.6}$$

where

W_{to} = Take off weight of the aircraft
W_{pl} = Total payload weight of the aircraft
R = Design range of the aircraft
S_{to} = Take-off distance of the aircraft
V_{cr} = Maximum cruising speed of the aircraft

The coefficients C_1 to C_5 are statistically derived from existing aircraft. The values are different depending upon the classification of aircraft, which is based upon the type of engine and the operational role of the aircraft, as shon in Table F.1. This equation gives an accuracy of 10% off the actual take-off weight, which is quite reasonable for initial stage.

The empty weight of the aircraft is predicted as follows :

$$W_{oe} = C_6 \, W_{to}^{C_7} \qquad \text{F.7}$$

where W_{oe} is the operational empty weight of the aircraft and the coefficients are :

C_6 = 0.96 for turboprops
1.02 for jet transports
C_7 = -0.05 for turboprops
-0.06 for jet transports

Table F.1

Coeffi-cients	Jet Transports	Executive Aircraft		Turboprop transports	
		Turboprop	Jet	Small	Large
C_1	1.0000	0.998424	1.39167	0.998426	1.0002
C_2	0.753313	0.30758	0.4785	0.30758	0.78284
C_3	0.356088	0.077421	1.09154	0.077421	0.077421
C_4	0.260264	0.37484	0.017133	0.37484	0.085976
C_5	-0.147147	0.57226	-0.49264	0.57226	0.48487

These equations are adequate for first estimate since the information available is normally limited to only performance requirements. More sophisticated weight estimation methods are required to estimate the weight of individual components. This component weight estimation is required not only to reflect the influence of each component's geometry but also to obtain the distribution of weight for balancing and centre of gravity calculation. Furthermore, the moments of inertia of the aircraft along its principal axes need to be calculated.

The method used at present is a simple method in which most of the component weights are calculated as proportional to the take-off weight. The empty weight is broken down into three major groups : structures, propulsion and equipment. The structure group consists of the load carrying components of the aircraft : wing, fuselage, tail surfaces, control surfaces, nacelles and landing gear. The propulsion group includes only the engine related components, e.g installed engines and their accessories, fuel systems and propellers, if available. The equipment group contains all the equipment components, e.g air conditioning system, furnishing and others. The component weight is calculated as follows :

$$W_{c_i} = C_8 \, W_{to}^{C_9} \qquad \text{F.8}$$

where W_{ci} is the component weight and the coefficients are specified in Table F.2.

The only deviations from this are the equations used for wing, fuselage, engines, propellers and nacelles.

The wing weight is calculated as follows :

$$W_w = 4.3\ AR^{1.06} \left[\frac{W_{to}}{S}\right]^{0.3} \qquad \text{F.9}$$

where AR is aspect ratio of the wing and W_{to}/S is the wing loading.

Table F.2

		C_8	C_9
Tail unit	Conventional	0.14	0.83
	High tail	0.16	0.83
Fuel System		0.068	0.8
Furnishing and Equipment	Long range	0.005	1.2
	Others	0.0088	1.2
Air conditioning system		0.044	0.88
Miscelaneous system and equipment weight	Turboprop, long range	0.11	1.0
	Other turboprops	0.12	1.0
	Jet aircraft, long range	0.09	1.0
	Other jet aircraft	0.12	1.0

The weight of the fuselage is calculated as follows :

$$W_f = C_{10}\left[(H_f + D_f)^{0.5} L_f \right]^{0.9} \qquad \text{F.10}$$

where C_{10} is 0.024 for long range aircraft and 0.02 for other aircraft.

The weights of nacelles, engines and propellers are assumed as proportional to the total engine gross thrust or power.

$$W_i = C_{11}\ T_{to}^{C_{12}} \qquad \text{F.11}$$

where T_{to} represents the total thrust for jet engines or total power for turboprop engines. The coefficients are specified in Table F.3.

Table F.3

		C_{11}	C_{12}
Engine Nacelles	Jet engines	0.03	1.0
	Propeller engines	0.034	1.0
Engine weight	Jet engines	0.15	1.0
	Propeller engines	0.25	1.0
Propeller system		0.11	1.0

F.5 DELTA (Drag Estimation Program)

The DELTA program is based on the Delta method of drag estimation which was developed by R.C. Feagin, et al of Lockheed for NASA in 1978 [39]. The program was rewritten in Fortran by G.D. Frosst [40] as part of his MSc thesis in the College of Aeronautics. It was then updated by A.J. Hartland to relax the limiting wing sweep angle [41].

The aircraft drag is broken down into the following component :

$$C_D = C_{Dmin} + C_{DL} \qquad \text{F.12}$$

where

C_{Dmin} = zero lift drag coefficient
= $C_{Df} + C_{Dc} + C_{Dadd}$
C_{Df} = friction drag, calculated by conventional method
C_{Dc} = compressibility drag
C_{Dadd} = additional zero lift drag cofficient due to miscellaneous factors, e.g stores, flap deflections, etc.
C_{DL} = lift dependent drag coefficient
= $C_L^2 / \pi_x AR + C_{Dp}$
C_{Dp} = pressure drag

DELTA first calls the INPUT subroutine which reads in all the data. The subroutine also loads into memory the data tables of the Delta method which are extensively used to interpolate values for different variables.

The program then enters the main loop of the calculation. It calculates the design lift coefficient and Mach number for the specified airfoil geometry.

The subroutines BUFFET, ZERO and LIFT are then called. Subroutine BUFFET calculates the buffet onset lift coefficient which may limit the lift coefficient for cruise. Subroutine ZERO calculates the zero-lift drag coefficient. Subroutine LIFT estimates the required lfit coefficient for the specified flight conditions.

F.6 ACCOST (Aircraft Cost Estimation)

F.6.1 PROGRAM DESCRIPTION

ACCOST [43] is a computer program written for estimating subsonic commercial transport aircraft and engine related costs. It is written in FORTRAN and installed in the College of Aeronautics VAX computers. The program can be used to calculate the following cost data :

1 aircraft cost and price for a projected break-even point
2 aircraft operating costs including direct and indirect operating costs
3 engine related costs such as engine and propeller price
4 return on investment for a specified percentage of profit and taxation

The program is based on cost estimation methods described in Raymer [1], Nicolai [52], Roskam [53], Maddalon [44]. The core of the program was obtained from OPDOT program [30] as explain in section F.2.2. Some cost factors were updated to give a more reasonable value for current aircraft as suggested in Reference [53]. These include cost escalation factor with respect to datum year quoted in the original routines. Using the updated value, the datum year is set to be 1989 rather than 1976 in the original program.

The flow diagram of the program is shown in Fig. F.6.

The MAIN program reads input from user through an input data file and calls subroutine DOCOST to calculate various cost parameters. The user input data is probably necessary to be converted into the unit system used in the program which is mainly Imperial unit system.

DOCOST is the main subroutine used to calculate operating costs and return on investment for a large-sized transport aircraft. It requires aircraft cost data from AIRCOST to calculate aircraft and engine price in the year of evaluation specified by the user.

It then calculates various elements of direct operating cost for the required flight time and length. The direct operating cost is calculated based on the American Airlines methods described in Reference [44], which is suitable for jet-engined aircraft with take off weight of more than 100000 lbs. (40000 kg.). The maintenance costs are obtained from subroutine MAINCST.

For a propeller driven commuter aircraft, DOCOST calls subroutine ADOC to calculate direct operating cost. Two different methods are used in ADOC to calculate maintenance costs of commuter aircraft. The first is the DLH method which is in subroutine INST. The other, in subroutine INSTVFW, is based on VFW method. The VFW method is more elaborate in that it requires more data than the DLH method. The user can specify either method to use in the analysis.

Subroutine AIRCOST calculates aircraft and engine unit costs. It calculates the costs incurred during development and production phase of the aircraft using the methods described in References [1] and [53]. The total cost is divided by the specified number of aircraft expected to be sold in order to reach break even point to obtain the aircraft cost per aircraft for the datum year.

Total aircraft cost can be divided into cost components [1] which are calculated as follows:

Engineering costs :

$$C_D(1) = 4.86\ T_1\ V_S 0.894\ Q_D^{0.163}\ W_E^{0.777}\ D_{REH}\ F_{DIF} \qquad \text{F.13}$$
$$CP(1) = 4.86\ T_1\ V_S 0.894\ Q_{P1}^{0.163}\ W_E^{0.777}\ D_{REH}\ F_{DIF} - C_D(1)$$

where

T_1 = cost factor accounted for active control system installation
= 1.15 with active control
= 1.0 otherwise

V_S = maximum diving speed of the aircraft [knots]

Q_D = number of aircraft produced/used during development phase for ground/flight testing

Q_P = number of aircraft produced during production/service phase

Q_{P1} = $Q_D + Q_P$

W_E = Empty weight of the aircraft [lbs.]

D_{REH} = dollar rate per engineering hour
= $59.00/hour in 1989

F_{DIF} = cost factor accounted for the level of difficulty of the aircraft project
= 1.0 Conventional technology aircraft
= 1.5 Moderately advanced
= 2.0 Highly advanced technology

and D denotes development phase and P denotes production phase.

Development Support :

$$C_D(2) = 45.42\ V_S^{1.3}\ W_E^{0.630}\ F_{DIF} \qquad \text{F.14}$$
$$C_P(2) = 0.0$$

where

C_{EF} = cost escalation factor in 1989 relative to that in 1976
= 2.04

Flight test operation :

$$C_D(3) = 1243.03\ T_3\ T_{32}\ V_S^{0.822}\ Q_D^{1.21}\ W_E^{0.325}\ F_{DIF}\ F_{OBS} \qquad \text{F.15}$$
$$C_P(3) = (Q_{P1} - Q_D)\ C_{OPSHR}\ T_{PFT}\ F_{FTOH}$$

where

T_3 = cost factor accounted for active control system installation
= 2.0 with active control
= 1.0 otherwise

T_{32} = cost factor accounted for c.g. control system installation
= 1.1 with c.g. control
= 1.0 otherwise

F_{OBS} = cost factor accounted for 'stealth' technology
= 1.0 without 'stealth' technology (default)
= 3.0 otherwise

C_{OPSHR} = operating cost per hour [$/hr]
= 2500 assumed

F_{PFT} = flight test hour flown by the manufacturer before delivering each aircraft to customer [hr]
= 10.0 Commercial transports

F_{FTOH} = cost factor accounted for flight test overhead cost
= 4.0

Tooling cost :

$$C_D(4) = 5.99\ V_S^{0.696}\ Q_D^{0.263}\ W_E^{0.777}\ D_{RTH}\ F_{DIF} \qquad \text{F.16}$$
$$C_P(4) = 5.99\ V_S^{0.696}\ Q_{PI}^{0.263}\ W_E^{0.777}\ D_{RTH}\ F_{DIF} - C_D(4)$$

where

D_{RTH} = dollar rate per tooling hour
= $60.70/hour in 1989

Manufacturing Labour :

$$C_D(5) = 7.37\ V_S^{0.484}\ Q_D^{0.641}\ W_E^{0.777}\ D_{RMH}\ F_{DIF} \qquad \text{F.17}$$
$$C_P(5) = 7.37\ V_S^{0.484}\ Q_{PI}^{0.641}\ W_E^{0.777}\ D_{RMH}\ F_{DIF} - C_D(5)$$

where

D_{RMH} = dollar rate per manufacturing hour
= $55.40/hour in 1989

Quality Control :

Total man hour for quality control is assumed to be proportional to the total manufacturing labour.

$$C_D(6) = 7.37\ C_{QC}\ V_S^{0.484}\ Q_D^{0.641}\ W_E^{0.777}\ D_{RQH}\ F_{DIF} \qquad \text{F.18}$$
$$C_P(6) = 7.37\ C_{QC}\ V_S^{0.484}\ Q_{PI}^{0.641}\ W_E^{0.777}\ D_{RQH}\ F_{DIF} - C_D(6)$$

where

C_{QC} = 0.076 for cargo aircraft

= 0.133 for passenger transport

D_{RQH} = dollar rate per quality control hour
= \$50.10/hour in 1989

Manufacturing Material :

$$C_D(7) = 11.0\, V_S^{0.621}\, Q_D^{0.799}\, W_E^{0.921} \qquad \text{F.19}$$
$$C_P(7) = 7.37\, V_S^{0.621}\, Q_{P1}^{0.799}\, W_E^{0.921} - C_D(7)$$

Subroutine ENGCOST is called by subroutine AIRCOST to calculate engine and propeller costs. The empirical method used in this subroutine is obtained from Reference [53]. This replaces the engine related cost equation used in the original OPDOT program which was based on Reference [52].

The total development cost and production cost of the aircraft is then calculated as follows :

$$C_{total} = \sum_{i=1}^{7} [\, C_D(i) + C_P(i) \,] + (Q_P+Q_D)\, N_{oe} C_{eng} + (Q_P+Q_D)\,(C_{av} + C_{act}) + C_{profit} + C_{interest} \qquad \text{F.20}$$

where

N_{oe} = number of engines per aircraft
C_{eng} = cost per engine unit
C_{av} = avionics system cost per aircraft
C_{act} = active control system installation cost per aircraft
C_{profit} = cost allowance for profit
$C_{interest}$ = cost allowance for interest charges of the bank loan to finance the aircraft project.

The purchase price of the aircraft can then be estimated as follow :

$$C_{aircraft} = \frac{C_{total}}{Q_P} \qquad \text{F.21}$$

Subroutine MAINCST calculates maintenance costs of commercial transport aircraft. Various cost elements are estimated using the equations described in reference [3] which were derived from the American Airlines cost data. A cost escalation factor of 2.04 is used to convert all costs to the value of dollar in the datum year 1989 from 1976 dollar. This escalation factor is obtained from Ref. [53] based on consumer's price index in 1989 relative to that in 1976.

F.6.2 PROGRAM IN USE

The program is written in FORTRAN and currently available in the CoA VAX computers. A version of the program also runs at Sun workstations available in the College of Aeronautics. The following section explains how to execute the program in the Aircraft Configuration Design System environment. The program however can be executed in its own (see Ref. [43]).

F.6.2.1 Executing the Program

As mentioned earlier, the program reads user input from a data file. The output is also written into a result file. For this purpose, when the program is executed the user needs to specify the name of input and output data file when requested.

The program can be executed while the system is on **Execution Mode** (see section E.2.1.2.2 in Appendix E), by activating the **'Execute Program'** panel button. The system subsequently calls the program interface to translate the data into a data file in a format acceptable to the program. To display the output, select and activate the **'Display Result'** panel button.

F.6.2.2 Input Data

The input data can be prepared partly in the **Editing Mode** of the system and partly in the **Execution Mode** (see sections E.2.1.2.1 and E.2.1.2.2 in Appendix E). In the editing mode, the general data of aircraft is specified. The input specified in the execution mode are only those that control the execution of the program.

The data required for ACCOST are listed in the following in the format as written in the input data file. These data are specified while the system is on the editing mode.

USYS

USYS = unit system used in the data file
- 0 Metric unit
- 1 Imperial unit

INTR, NPAX, WTO, WTE, WTL, WPL, WAMPR, CARGO

INTR Type of aircraft
- = 0 Commercial transports
- = 1 Executive/business aircraft
- = 2 General aviation

NPAX Number of passengers
WTO Take off weight [lbs or kg]
WTE Empty weight of aircraft [lbs or kg]
WTL Maximum landing weight [lbs or kg]
WPL Payload weight [lbs or kg]

WAMPR Manufacturer's airframe weight [lbs or kg]
If 0 is input this data will be estimated by the program from the take off weight data.
CARGO Cargo weight [lbs or kg]

IENG, SW, VS, ICGX, IACT
IENG type of propulsion
= 0 jet engines
= 1 turboprop
= 2 piston engines
SW Wing area [sq.ft or m^2]
VS Maximum diving speed [kts or km/hr]
ICGX = 1 if centre-of-gravity control is installed
= 0 otherwise
IACT = 1 if active control is installed
= 0 otherwise

NE, IPROP, TTO
NE Number of engines
IPROP Propeller type
= 0 Not applicable
= 1 Metal propeller
= 2 Composite propeller
TTO Thrust per engine [lbs or kg], or
Power per engine [HP or kW]

RANGE, VCR, FLTIME, BLKHR, BLHYR, FLFUEL, FLOAD
RANGE Aircraft mission range [naut. miles or km]
VCR Aircraft cruise speed [kts or km/hr]
FLTIME Mission flight time [hr]
BLKHR Mission block hour [hr]
BLHYR Aircraft utilisation per year [hrs/yr]
FLFUEL Mission fuel weight excluding reserve [lbs or kg]
FLOAD Passenger load factor

DSM, FPR, CX, CXC, PRICE, CPWP, NCST, IVFW, NYRE
DSM Fare [$/seat-naut.mile]
FPR Fuel price [$/gallon]
CX Purchase price cost boost (%Boost = CX * 5%)
CXC Maintenance cost boost (% Boost = CXC * 5%)
PRICE Aircraft purchase price [$], enter ZERO if unknown
CPWP Powerplant unit cost [$], enter ZERO if unknown
NCST Element of cost data produced (specified in the execution mode):
= 1 DOC per Block hour
= 2 DOC per flight
= 3 Return on Investment

APPENDIX F

= 4 Passenger fare
= 5 Indirect Operating Cost
= 6 Aircraft price
= 7 Investment
= 8 Airframe cost
= 9 Powerplant system cost
= 10 Total maintenance

IVFW Methods used in maintenance cost estimation
= 0 American Airlines' model (for large aircraft)
= 1 Fokker VFW model
= 2 DLH model

NYRE Year of evaluation (4 digits)

The following data are compulsory for estimating production cost :

QD, RD, QP, RP

QP Number of aircraft used for flight testing during development phase of the aircraft

RD Rate of aircraft production per month during development phase

QP Expected number of aircraft produced to reach break even point during production/service phase of the aircraft

RP Rate of aircraft production per month during production phase

IDIFF, ICAD, IMAT, CAV, FFIN, FPRO

IDIFF Level of technology complexity/difficulty introduced in the aircraft
= 1 Conventional technology
= 2 Moderately advanced
= 3 Highly advanced technology

ICAD Use of CAD system in design
= 1 For manufacturers which are in CAD learning process
= 2 No CAD system used
= 3 CAD system used extensively

IMAT Type of material used
= 1 Conventional Aluminium alloys
= 2 Stainless steel
= 3 Conventional composite, Li/Al or ARAL material
= 4 Carbon composite

CAV Avionics system cost [$]

FFIN Interest rate for financing cost [%]

FPRO Percentage of profit expected [%]

If IACT = 1 (active control technology installed) this input is required :

CACT Active control system cost [$]

In the execution mode, the user can specify the control input that selects the type of

calculation requested from the analysis :

NCST Element of cost data produced :
= 1 DOC per Block hour
= 2 DOC per flight
= 3 Return on Investment
= 4 Passenger fare
= 5 Indirect Operating Cost
= 6 Aircraft price
= 7 Investment
= 8 Airframe cost
= 9 Powerplant system cost
= 10 Total maintenance

A sample of input data is given below, which is for X-91 aircraft.

```
0
1 295  269894.220  132064.300  220000.000  27639.000  102439.000      0.000
0     383.200     954.170 1 1
4 1   20111.000
  13805.000    848.48     16.270     16.700   3200.000  99881.750      0.650
   0.090   0.570   0.000    0.000     0.570 2000000.000 8  0  1991
     6.000      0.500    700.000     4.000
2 3 1  5000000.000      10.000      10.000
 1000000.000
```

F.7 FLTANAL (Flight Analysis Program)

F.7.1 PROGRAM DESCRIPTION

FLTANAL is a computer program written to perform flight analysis of subsonic transport aircraft [45]. It performs calculations of fuel weight, time spent and range covered during climb, cruise, descent and reserve mission flight. The core of the program was obtained from OPDOT program [29,30].

Several new routines were required to complete the analysis. Subroutine DESCENT was developed and integrated into the program to calculate fuel consumption, time required and range covered in descent phase of the flight. This in turn removed the assumptions made in the OPDOT program about this flight phase.

A flow diagram for FLTANAL program is shown in Fig. F.7.

The MAIN routine reads input from a data file specified by the user. It then calls subroutine WEIGHT to calculate take-off weight of the aircraft. Subroutine WEIGHT assumes that the empty weight and payload weight of the aircraft are fixed. The total take-off weight of the aircraft is calculated as the sum of the following :

$$W_{to} = W_{oe} + W_{pl} + W_f \qquad \text{F.22}$$

where

W_{to} = Take-off weight
W_{oe} = Empty weight
W_{pl} = Payload or fixed weight
W_f = Total fuel weight including reserve.

This calculation is performed by iteration until the difference between the old and new estimation of the take-off weight is less than 0.5 lbs.

Subroutine FUELCAL used in this program is the same as the one used in OPDOT. It calculates the total fuel weight required to fly the required range including reserve. It follows the flight plan shown in Fig. F.3. It calls subroutines CLIMB, CRUFUEL and DESCENT subsequently to calculate fuel burned in the flight segments corresponding to climb, cruise and descent flights respectively. Subroutines CLIMB and CRUFUEL are the same as in OPDOT.

The descent flight carried out in subroutine DESCENT follows the procedure shown in Fig. F.8 in which a constant descent angle of 0.025 rad. is maintained. Two different procedures are offered. In the first procedure, the descent speed (true airspeed) is set constant at 80% of cruise speed. In the second procedure, the descent speed (indicated airspeed) is the maximum of the following :

$$V_D \text{ (IAS)} = \begin{cases} 0.8\, V_{cr} \\ M\, a\sqrt{\sigma} \end{cases} \qquad \text{F.23}$$

where V_D is the descent speed, V_{cr} is the cruise speed, M is the cruise Mach number, and a and σ are the speed of sound and the relative density at the corresponding altitude, respectively.

At each flight condition, subroutines CDZL and XLOD are called to calculate the aerodynamic performance of the aircraft. CDZL calculates the zero lift drag coefficient at the corresponding Mach number, while subroutine XLOD calculates the drag coefficient of the aircraft by using the drag equation specified by the user.

F.7.2 PROGRAM IN USE

The program is written in FORTRAN and currently available in the CoA VAX computers. A version of the program also runs at Sun workstations available in the College of Aeronautics. The following section describes how to run the program in Sun computers in the Aircraft Configuration Design System environment. The program

however can be executed in its own (see Ref. [45]).

F.7.2.1 Executing the Program

As mentioned earlier, the program reads user input from a data file. The output is also written into a result file. For this purpose, the user needs to specify name of the input and output data file when requested. To execute the program,

- Change the system mode into execution mode (see **'Execution Mode'** in section E.2.1.2.2 in appendix E),
- Select and activate the **'Execute Program'** panel button.

The system subsequently calls the program interface to translate the data into a data file. Once the data file has been saved, it then changes the default directory in the TTY window into the directory in which the program exists and executes the program.

F.7.2.2 Input Data

The data required for FLTANAL are listed in the following in the format as written in the input data file. These data can be edited while the system is on editing mode (see section E.2.1.2.1 in Appendix E).

USYS

USYS	= unit system used
	= 0 Metric unit
	= 1 Imperial unit

NPAX, WTO, WTEMPTY, WTFIXED

NPAX	= Number of passengers
WTO	= Initial estimate of take-off weight [lbs or kg]
WTEMPTY	= Empty weight of the aircraft [lbs or kg]
WTFIXED	= Fixed or payload weight of the aircraft [lbs or kg]

IENG, AR, SW, ICD

IENG	= type of propulsion
	= 0 jet engines
	= 1 turboprop
	= 2 piston engines
AR	= wing aspect ratio
SW	= wing gross area [sq.ft or m^2]
ICD	= option for drag polar equation
	= 0 for : $C_D = C_{D0} + K\,C_L + K'\,C_L^2$
	no Mach number influence
	= 1 for : $C_D = C_{D0}(M) + K(M)\,C_L + K'(M)\,C_L^2$
	with Mach number influence

The following data are required ONLY if ICD = 0 :

CDS(1), XK(1), XKB(1)

CDS(1)	= zero-lift drag coefficient C_{D0}
XK(1)	= coefficient K for drag polar equation
XKB(1)	= coefficient K' for drag polar equation

The following data are required ONLY if ICD = 1 :

NM

NM	= number of Mach number in the table of drag polar

The following data are required ONLY if ICD = 1 (as many as NM, one set at each line) :

XMC(i), CDS(i), XK(i), XKB(i)

XMC(i)	= the i^{th} Mach number in the table
CDS(i)	= zero-lift drag coefficient C_{D0} for the i^{th} set
XK(i)	= coefficient K for drag polar equation for the i^{th} set
XKB(i)	= coefficient K' for drag polar equation for the i^{th} set

The following data are compulsory for any case :

NOE, ETYPE, TTO, PX

NOE	= number of engines
ETYPE	= engine type selector
	For jet engines :
	= 0 General Electric CF6 turbofan
	= 1 Rolls Royce Tay 710
	= 2 Uprated Rolls Royce Tay 710
	= 3 General Electric/Snecma CFM56-40
	= 4 Williams International FJ44
TTO	= installed gross thrust [lbs or kg]
PX	= specific fuel consumption reduction factor

MACH, RANGE, ALTCR, ICR, RRES, HRES

MACH	= cruise Mach number
RANGE	= mission range (naut. miles or km)
ALTCR	= initial cruise altitude (ft.)
ICR	= cruise method selector
	= 0 constant Mach number
	= 1 optimised lift coefficient
RRES	= range for reserve flight (naut. miles or km)
HRES	= altitude for reserve flight (ft.)

A sample input data is given in the following for the X-91 aircraft in the format as written in a data file :

```
0
295   264731.400   132064.3    27639.000
0       9.650      388.200 0
      0.0166         0.000        0.0429
4 0    14041.000         0.000
       0.800    13805.000        9.150 1      370.000        7.000
```

F.8 MITCHELL (Aircraft Stability And Control Analysis)

F.8.1 INTRODUCTION

Mitchell program is a modified version of a computer program developed by C.G.B Mitchell at the Royal Aircraft Establishment [46] for a rapid prediction of the stability characteristics of aircraft based only on information that would be available in the early stage of design. The program has been modified for use on the Cranfield DEC VAX systems [47]. The major modifications concern the ICL library routines used in the original program being replaced by the NAG Fortran library routines and the input/output part of the program.

The program is intended to allow a rapid assessment of a particular design with given geometry. It uses the information about the aircraft geometry, weight and flight conditions to estimate the stability derivatives, to calculate the characteristics of lateral and longitudinal stick-fixed modes, to estimate the response to control inputs and some of the limiting speeds for low speed flight. The program however does not indicate what changes are required to modify a particular aspect of the predicted stability and control.

A version of the program has been modified by the author for use in the Sun microsystems. Again these modifications related to the library routines, due to the unavailability of the NAG library in the system. However, one subroutine which calculate the first order differential equations (subroutine D02DBF) could not be replaced by a suitable routine. A number of attempts has been made without meeting the required level of accuracy as the NAG library routine provides. For this reason, the part of the program that uses the routines has been scraped altogether. This obviously limits the capability of the program slightly, in that it cannot predict the response to control input. However, the presence of the program in the Aircraft Configuration Design System is necessary to complete the analysis.

The part of the program that relates to the engines is also slightly modified. This enables the program to deal with any number of engines as compared to only two on the original program. The performance and geometric data of the engines are specified in pairs assuming that the engines are symmetrical along the longitudinal (fuselage) axis

of the aircraft.

The following section describes the program as it is in the Sun microsystems.

F.8.2 PROGRAM DESCRIPTION

The program can perform a number of operation as follows :

- Calculation of geometric quantitites
- Calculation of aerodynamic derivatives
- Estimation of the moments of inertia
- Calculation of the eigen values (roots) of the equations for longitudinal and lateral motions with the control stick fixed
- Calculation of the limiting speeds for low speed flight at sea level

The MAIN routine calculates the geometric quantities after reading the input data. It then estimates the aerodynamic derivatives of the aircraft based on the published data. The ESDU Aerodynamic data sheets and the USAF Stability and Control Handbook (DATCOM) are used extensively in this calculation.

The program calculates the moments of inertia by either one of the following :

- DATCOM method, if the weight breakdown of the major components and their loads are known
- a simple method using the radii of gyration in roll, pitch and yaw, if the weight breakdown is not known.

The program however overrides this calculation if the moments of inertia of the aircraft along its principal axes are given as input.

The stability of the aircraft with fixed controls is then determined by finding the eigen values of the determinants of the equations of motion. The equations are first separated into the longitudinal and lateral motions. The eigen values and the eigen vectors are calculated by subroutine EIGEN, a subroutine developed by the author to replace the NAG library routine, F02AGF. Subroutine EIGEN follows the same procedure as the FO2AGF subroutine which enables the accuracy of the two to be almost equal.

The low speed handling of the aircraft is calcualted in subroutine LSPD. It calculates the stalling speed and the incidence angle, the tail load and elevator angle at the stall. It estimates the nose-wheel lift speed with tailplane set either to trim at 1.2 V_s or as near the trim position as it can be moved. The minimum control speed for engine failure and the steady side slip angle that can be held at 1.3 V_s is also calculated with either full rudder or 80% of full aileron and the corresponding cross-wind components.

F.8.3 PROGRAM IN USE

A slight modification of input/output facility is required to enable the program to be integerated in the Aircraft Configuration Design System. This modification however is not in the program but in the System itself. The program itself has been left as it is to allow the user to execute the program in its own.

The input data is prepared in both the editing mode and execution mode of the System. In the editing mode the geometric data of the aircraft is displayed. The execution mode allows the user to specify the data for each pair of the engines and the data for each flight case considered. The execution mode also allows the user to select the operation to be executed out of the options listed earlier.

F.8.3.1 Input Data in Editing Mode

Fuselage geometry :

B(1)	Length of the fuselage parallel section, aft of datum
B(2)	Length of the fuselage parallel section, forward of datum
B(3)	Length of the fuselage, aft of datum
B(4)	Length of the fuselage, forward of datum
B(5)	Maximum width of fuselage
B(6)	Maximum height of fuselage
B(7)	Location of wing apex below datum
B(8)	Location of main landing gear, aft of datum
B(9)	Fuselage incidence angle on ground (degree)

Wing geometry :

W(1)	Wing root chord
W(2)	Wing tip chord
W(3)	Leading edge sweep angle (degree)
W(4)	Semi span
W(5)	Average thickness ratio (t/c) of wing section
W(6)	Wing incidence angle (degree)
W(7)	Wing dihedral angle (degree)
W(8)	Root chord of aileron, at hinge line
W(9)	Tip chord of aileron, at hinge line
W(10)	Semi span to inboard end of aileron
W(11)	Semi span to outboard end of aileron
W(12)	Maximum upward deflection of aileron (degree)
W(13)	Maximum downward deflection of aileron (degree)
W(14)	Root chord of spoiler, at spoiler line
W(15)	Tip chord of spoiler, at spoiler line
W(16)	Semi span to inboard end of spoiler

W(17) Semi span to outboard end of spoiler
W(18) Spoiler projection area

Fin geometry :

F(1) Fin leading edge location at fuselage centre line
F(2) Chord at fuselage centre line
F(3) Tip chord
F(4) Leading edge sweep angle
F(5) Fin height
F(6) Tailplane height from fuselage centreline
F(7) Fin thickness ratio (t/c)
F(8) Fuselage top height from centreline at quarter chord line
F(9) Root chord of rudder, at hinge line
F(10) Tip chord of rudder, at hinge line
F(11) Height of lower end of rudder
F(12) Height of top end of rudder
F(13) Maximum deflection angle of rudder (degree)

Tailplane geometry :

T(1) Location of tailplane apex, aft of datum
T(2) Root chord of tailplane
T(3) Tip chord of tailplane
T(4) Leading edge sweep angle (degree)
T(5) Semi span
T(6) Root chord of elevator, at hinge line
T(7) Tip chord of elevator, at hinge line
T(8) Semi span to inboard end of elevator
T(9) Semi span to outborad end of elevator
T(10) Thickness ratio of tailplane
T(11) Tailplane incidence angle (degree)
T(12) Maximum upward deflection of elevator (degree)
T(13) Maximum downward deflection of elevator (degree)
T(14) Maximum upward deflection of tailplane trailing edge (degree)
T(15) Maximum downward deflection of tailplane trailing edge (degree)

Aerodynamic Data :

ALF0 Wing zero-lift incidence angle (degree)
CD0 Zero lift drag C_{Do} for the complete aircraft
CM0 Zero lift pitching moment C_{mo} for the complete aircraft
CLMW Maximum lift coefficient C_{Lmax} for the complete wing
DCLF C_L for the flap on a 2-D wing section
(only required if the aircraft has spoilers)

APPENDIX F

Moments of Inertia :

If the moments of inertia are not known, set input to ZEROs.

I_x	Moment of inertia along the X axis (kg m^2)
I_y	Moment of inertia along the Y axis (kg m^2)
Iz	Moment of inertia along the Z axis (kg m^2)
I_{xz}	Moment of inertia along the X-Z axis (kg m^2)
	I_{xz} is positive is the principal axis is inclined nose down

Weight Data :

The following variables are required for the estimation of the moments of inertia. If the moments of inertia are known, the variables are not required. If the weiight breakdown is not known, enter ZEROs as input.

WB	Weight of the fuselage group
WW	Weight of the wing group
WF	Weight of the fin group
WTP	Weight of the tailplane group
WE	Installed weight of one powerplant
WFUEL	Weight of fuel
WP	Weight of the payload in fuselage
LFUEL	Ratio of the span of fuel tanks to wing span

Engine and Case Data :

NE	Number of engines
NCASES	Number of flight conditions

F.8.3.2 Execution Mode

The execution mode enables the user to enter the engine and flight case data as many as specified in the editing mode.

Engine Data :

These data are required for each engine pair. Thrust of each engine at sea level (N) is estimate by the equation :

$$F = E(4) + E(5)\ V + E(6)\ V^2$$

with V in m/s.

E(1)	Distance from fuselage centreline
E(2)	Centre of gravity location aft of datum

E(3)	Thrust line below datum
E(4)	Coefficient E(4) in the thrust equation
E(5)	Coefficient E(5) in the thrust equation
E(6)	Coefficient E(6) in the thrust equation
E(7)	Distance of nacelle nose, forward of wing leading edge
E(8)	Nacelle diameter

Flight Conditions :

The following data are required for each flight case as specified in the editing mode :

HT	Flight altitude (m)
X	Flight Mach number of speed (m/s)
WT	Aircraft weight (kg)
LCG	Distance of centre of gravity aft of datum
HCG	Distance of center of gravity above datum
AINC	Inclination of the principal inertia axis (degree), positive for nose down

Type of Operation :

The type of operation can be selected from the following :

- Calculate Aerodynamic derivatives only
- Calculate Aerodynamic derivatives and Low speed handling
- Calculate Aerodynamic derivatives and Eigen values
- Calculate Aerodynamic derivatives, Eigen values and Low speed handling.

F.8.3.3 Executing the Program

The program can be executed by activating the **'Execute Program'** panel button (see section E.2.1.2.2 in Appendix E). This will call the interface program to translate the specified input into the input file in a format which is acceptable to the program. The output of the execution can be displayed by activating the **'Display Result'** panel button.

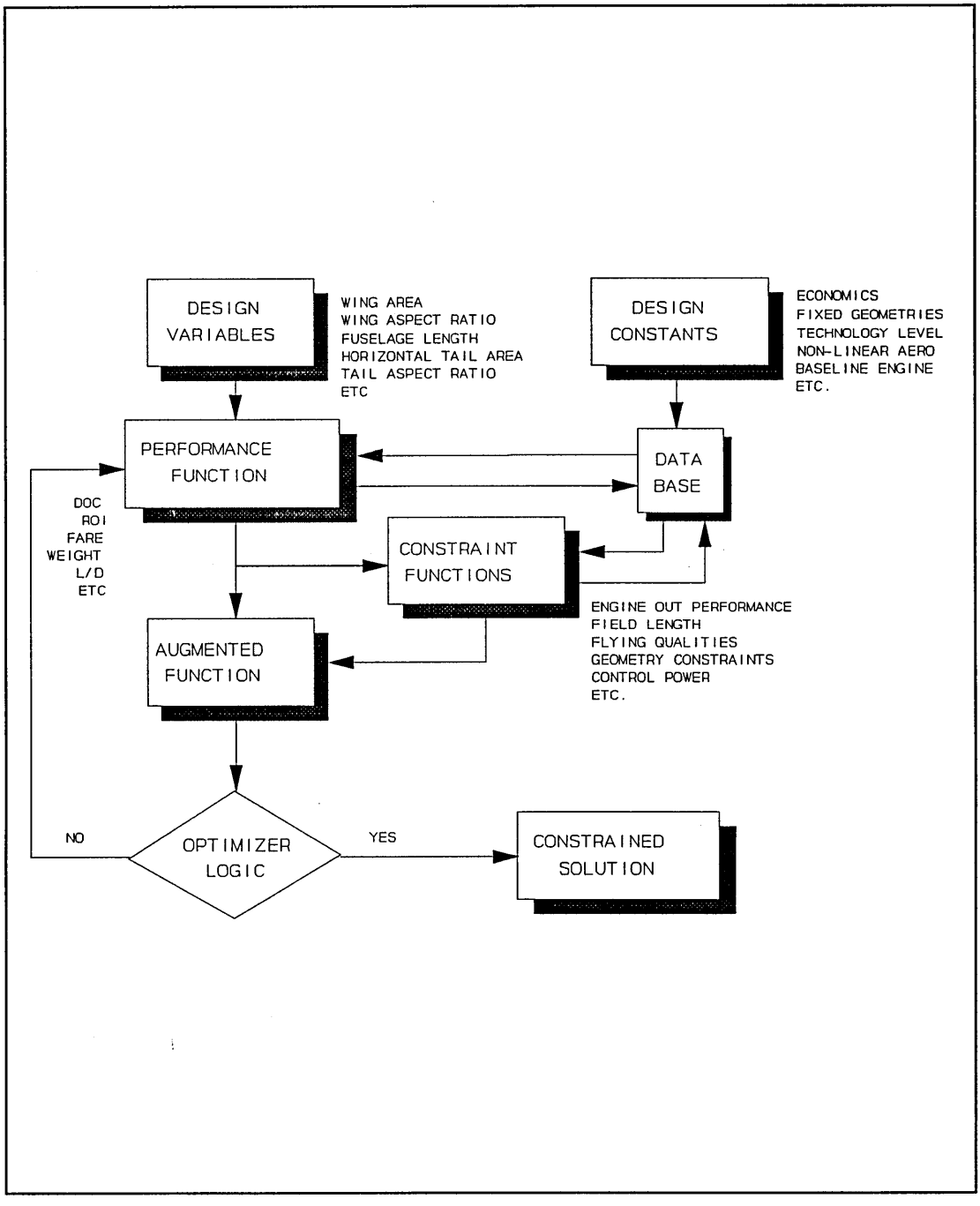


Fig. F.1 General optimisation scheme in OPDOT program

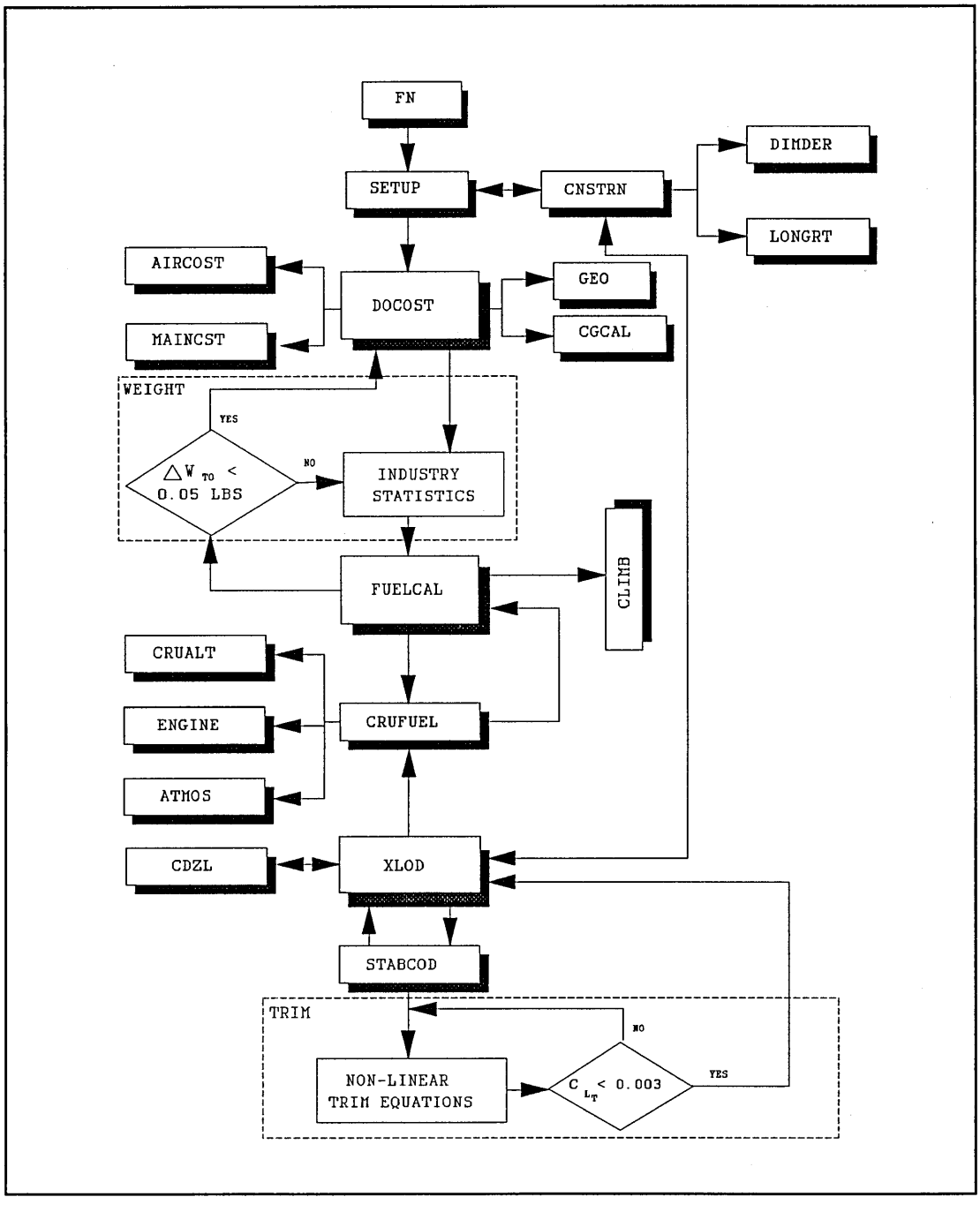


Fig. F.2 Flow diagram for evaluating the unaugmented performance index

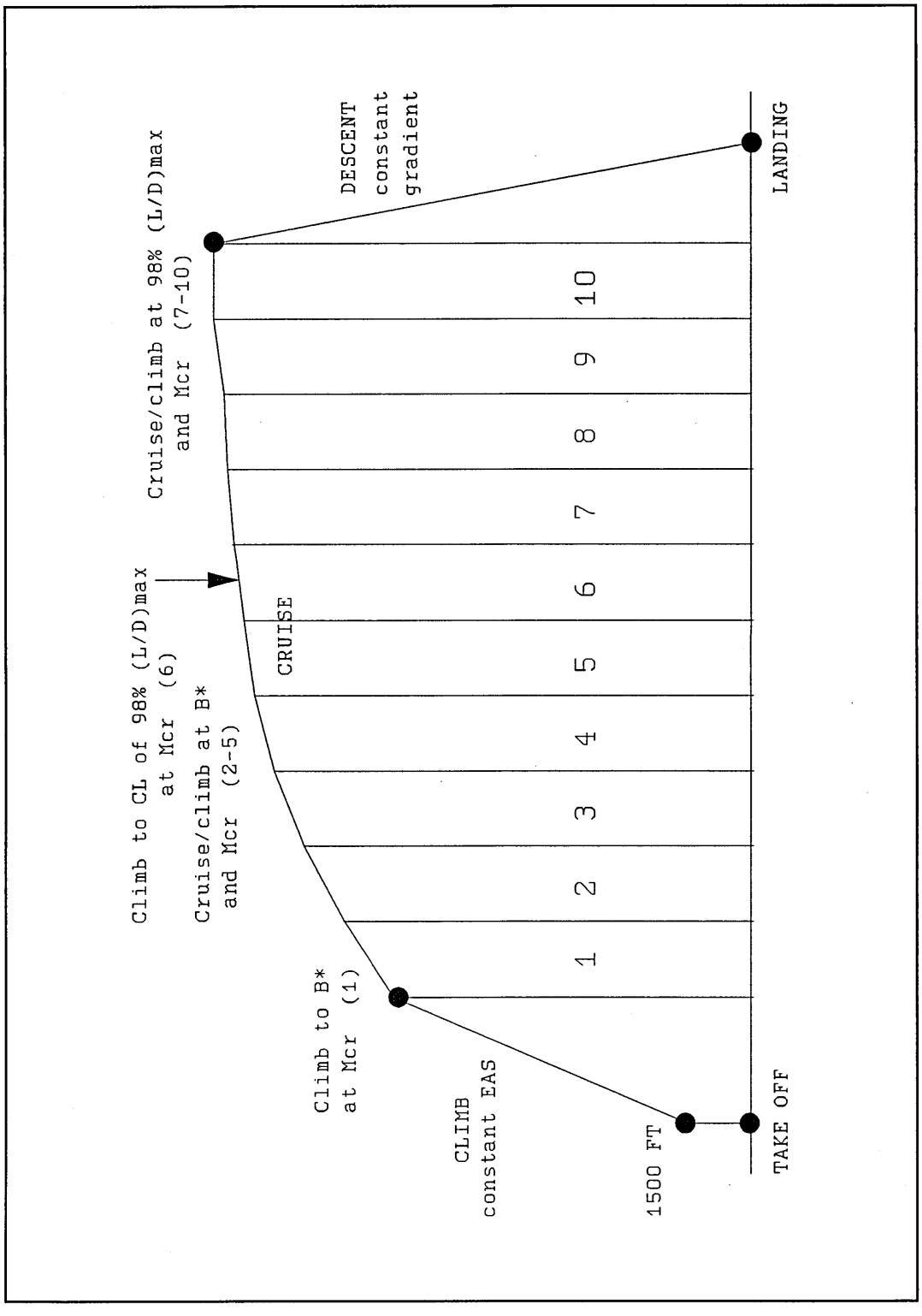


Fig. F.3 Flight plan used in OPDOT and FLTANAL programs

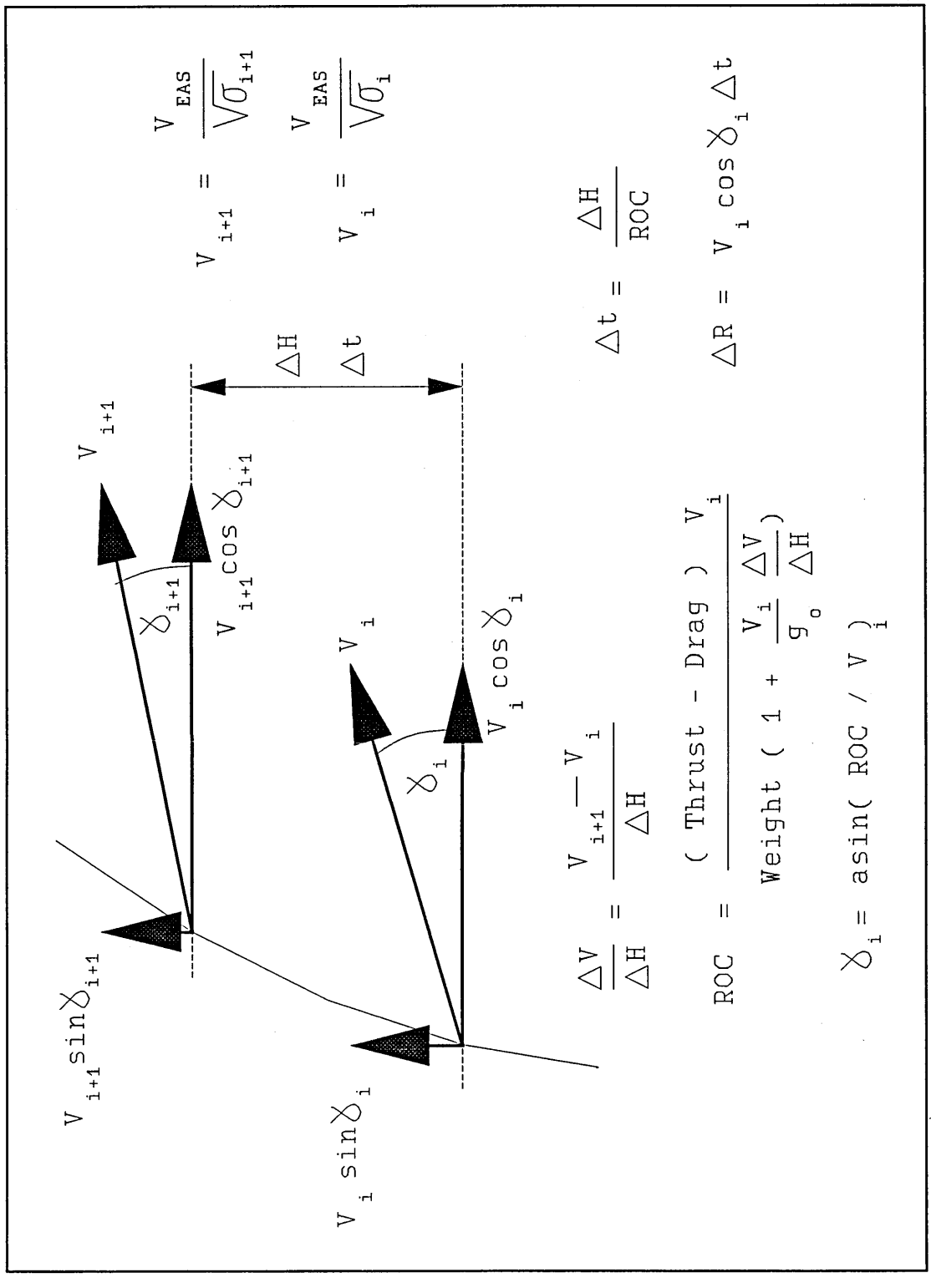


Fig. F.4 Climb procedure in subroutine CLIMB used in OPDOT and FLTANAL programs

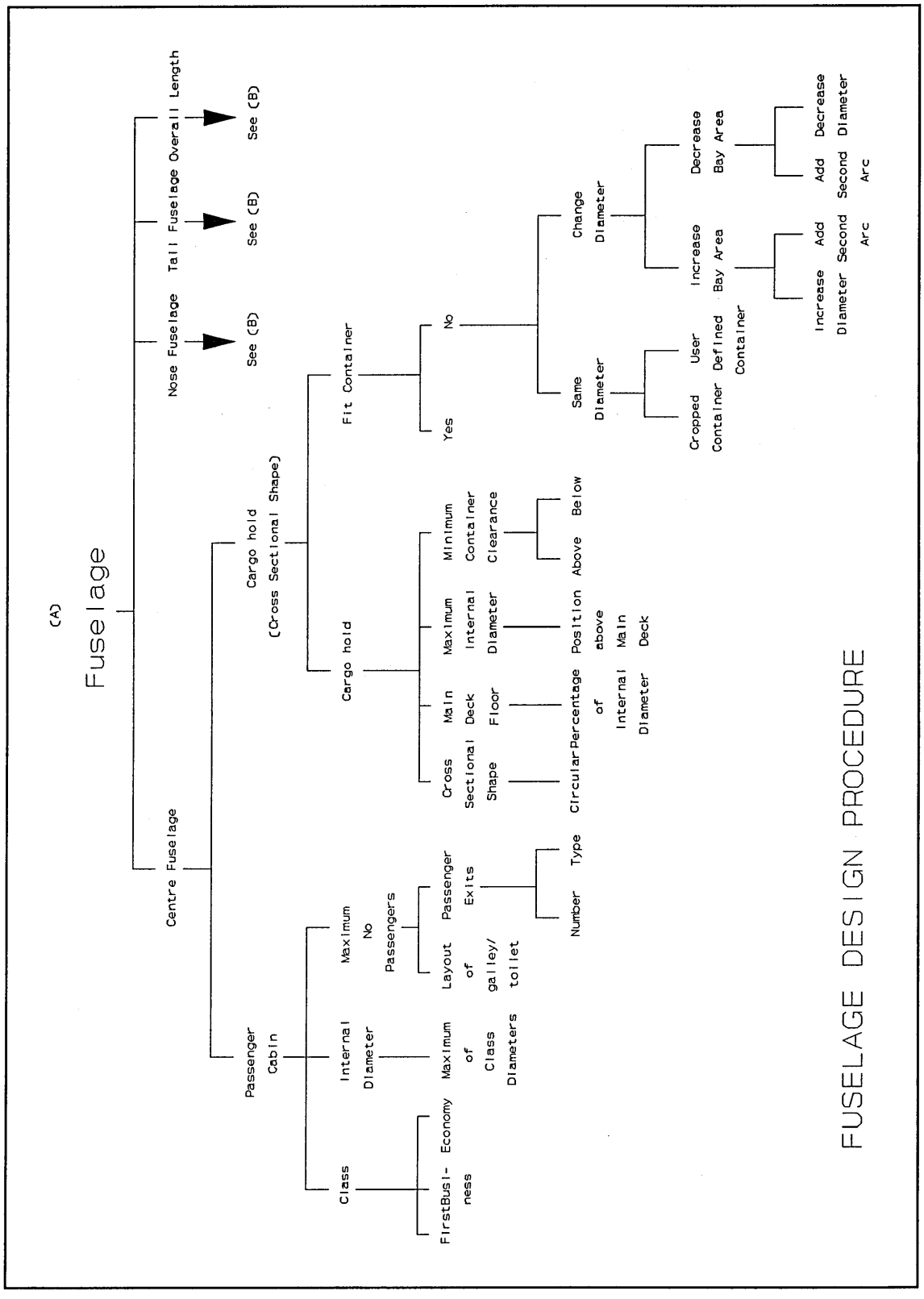


Fig. F.5 Fuselage design procedure

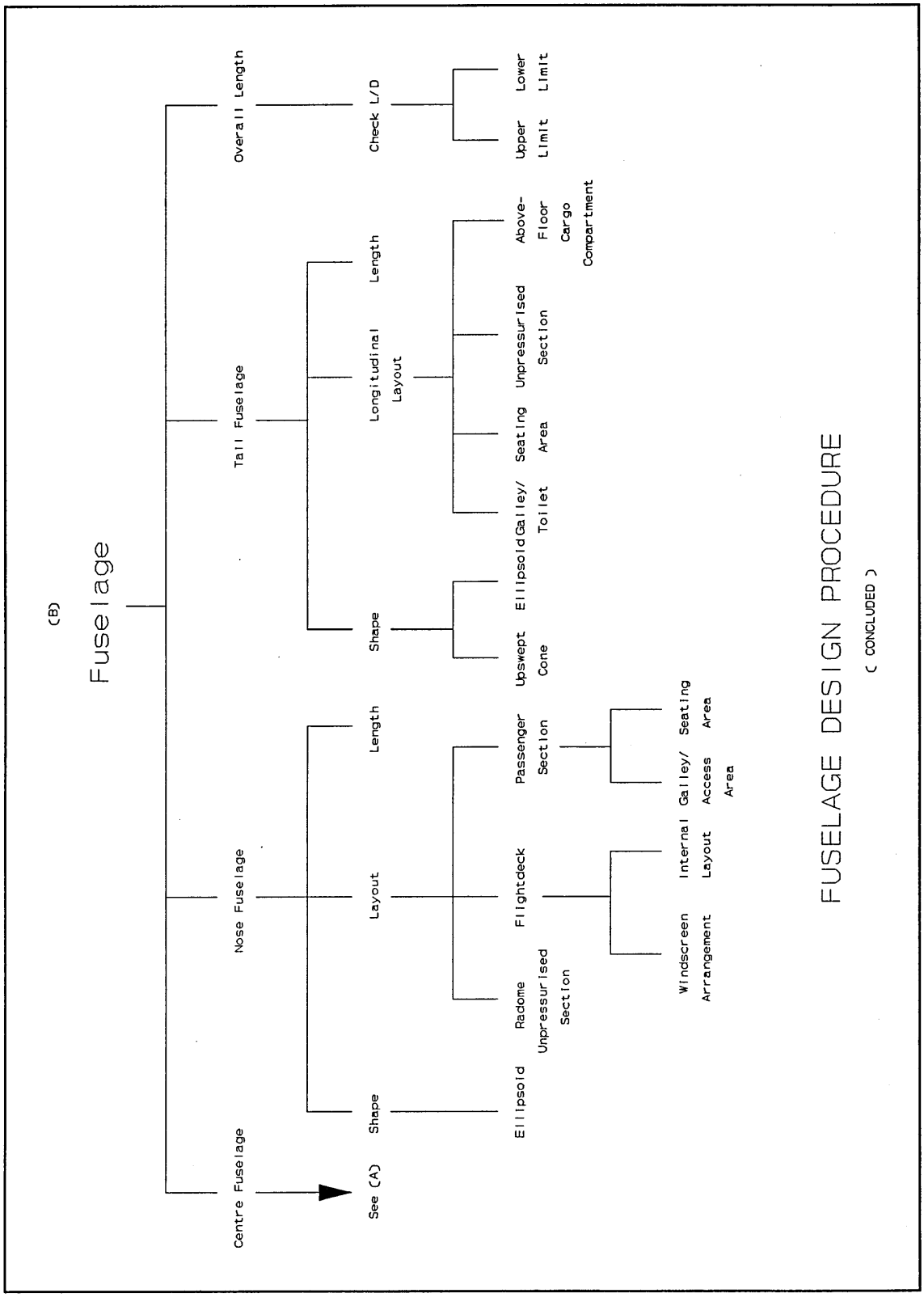


Fig. F.5 Fuselage design procedure (Continued)

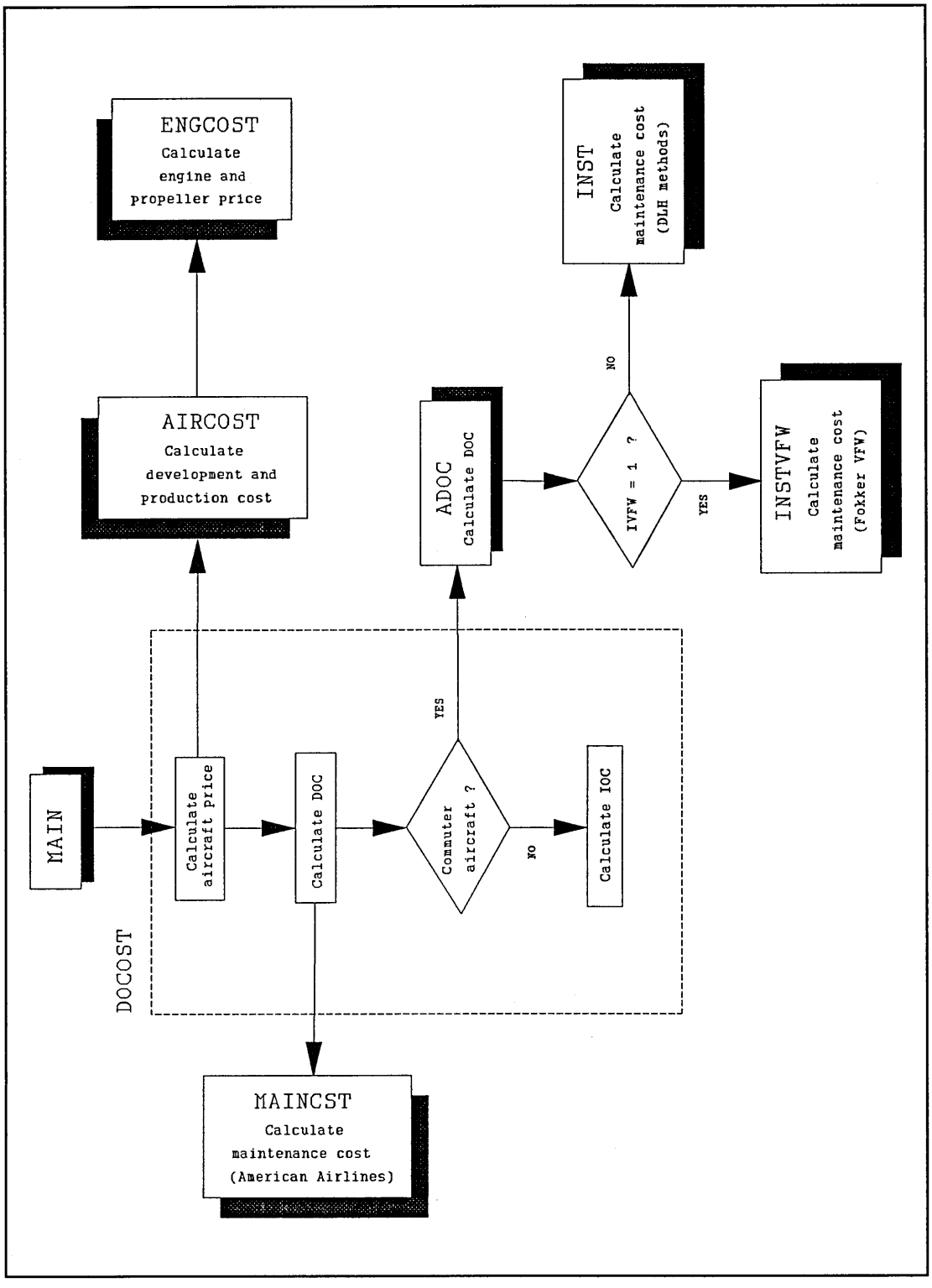


Fig. F.6 ACCOST program flow diagram

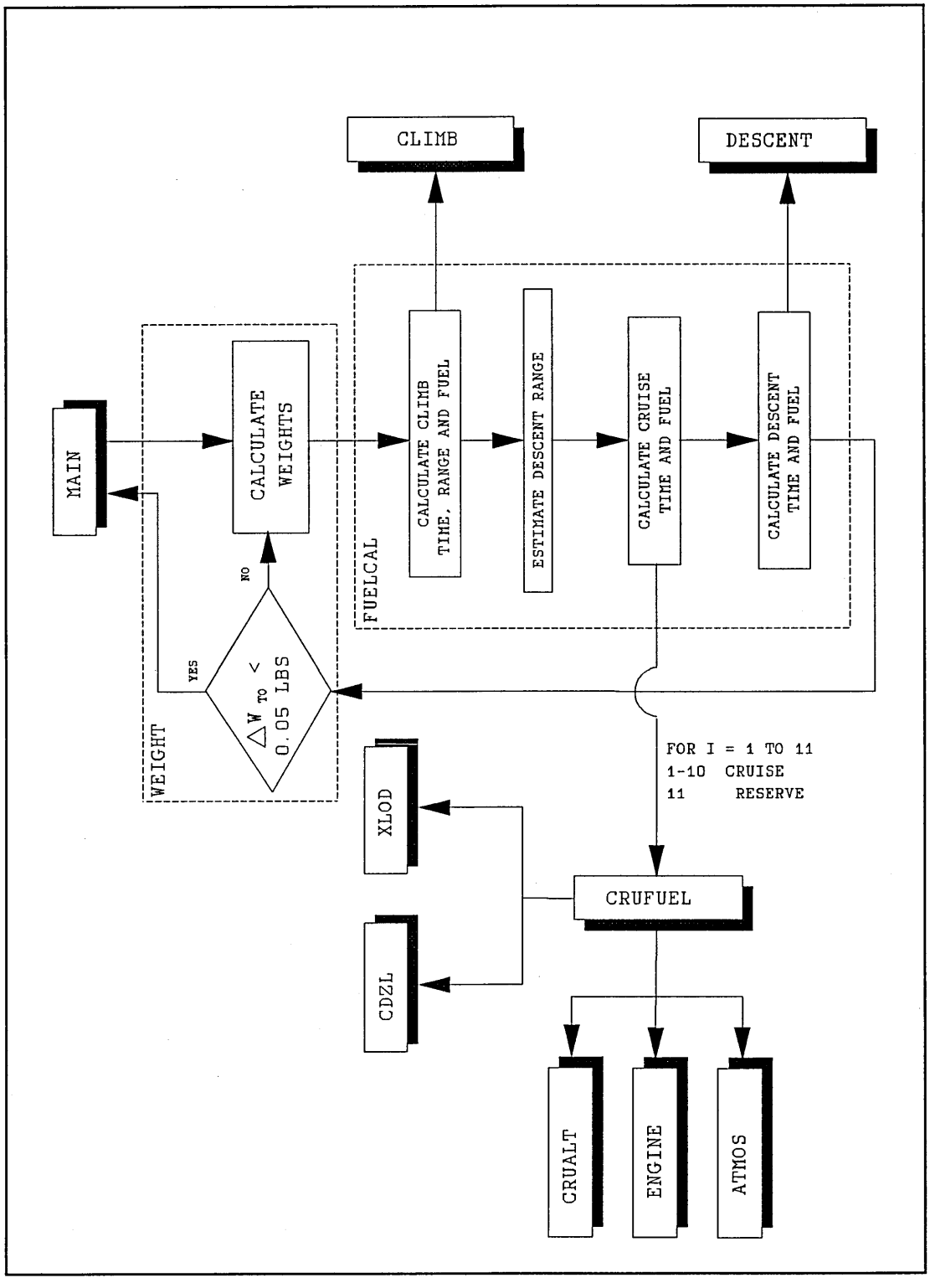


Fig. F.7 FLTANAL program flow diagram

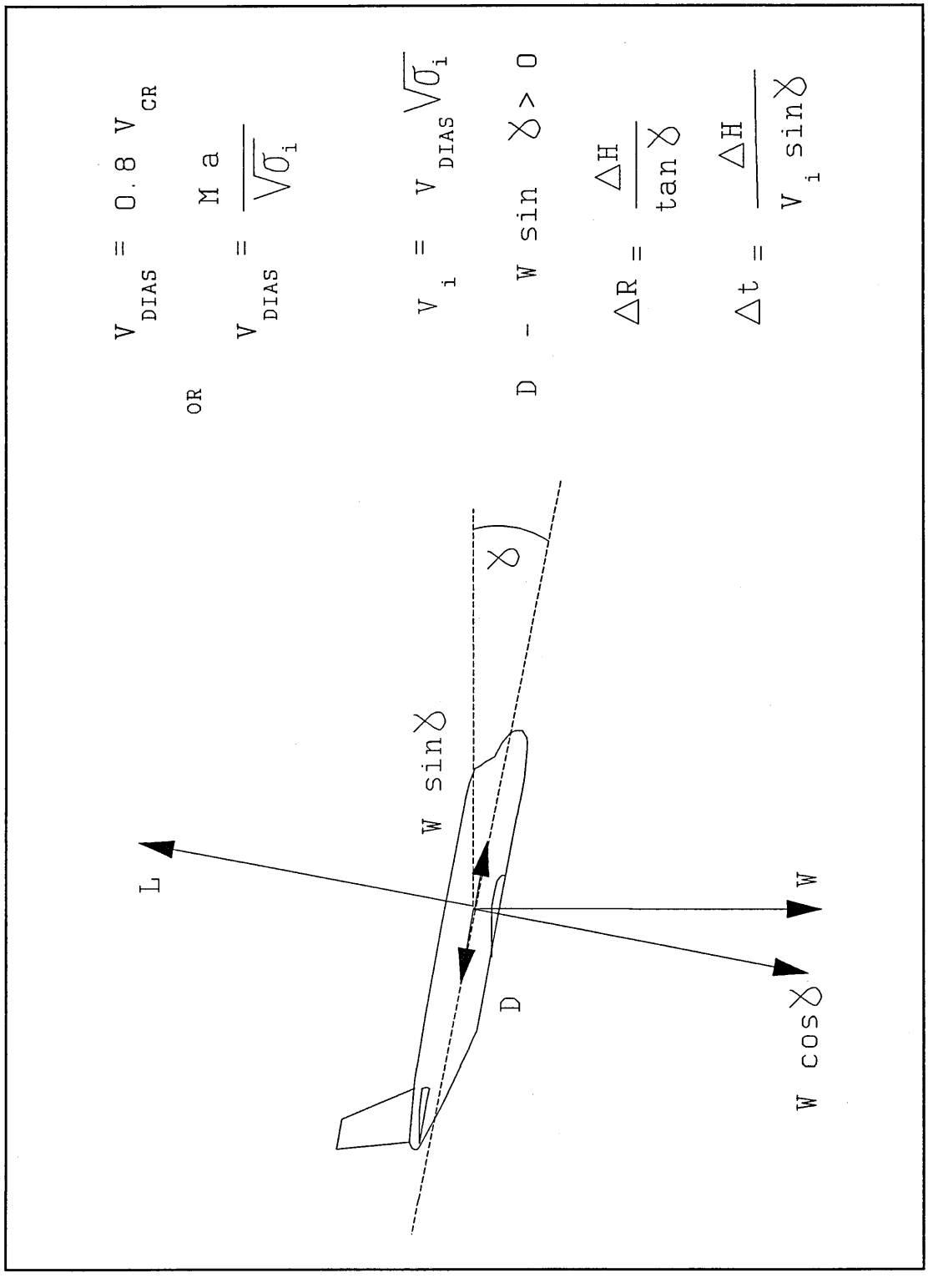


Fig. F.8 Descent procedure in subroutine DESCENT of FLTANAL program

APPENDIX G
DETAILS OF PROGRAM OUTPUT

G.1 INTRODUCTION

This apendix gives the detailed output of the execution of the program as reported in Chapter 6. The presentation is given in the following order :

- Fuselage Program
- Weight Estimation Program
- Drag Estimation Program
- Flight Analysis Program
- Cost Estimation Program

Each presentation is preceded by the input data used in each calculation.

G.2 FUSELAGE DESIGN

G.2.1 INPUT DATA

The following data are used in the calculation :

Aircraft Type	= Commercial Transport
Range (km)	= 13805.0
Height of cabin (m)	= 2.08
Width of fuselage frame (m)	= 0.15
Floor depth (m)	= 0.20
Number of passenger class	= 3

	First class	Business	Economy class
Number of passengers	18	81	196
No. of seats abreast	6	7	8
Seat pitch (m)	1.5	0.92	0.86
Aisle width (m)	0.54	0.52	0.52
Tabel width (m)	0.0	0.0	0.0
Seat type	2	1	1

G.2.2 OUTPUT DATA

The output is presented in the following pages.

```
************************************************************
*                                                          *
*                        FUSELAGE                          *
*            program for aircraft fuselage design          *
*                                                          *
*                    developed based on                    *
*   [1]    Pasaribu, A Computer Augmented Procedure for    *
*        Subsonic Aircraft Fuselage Design Synthesis       *
*     [2]    Nicolai, Fundamentals of Airplane Design      *
*          [3]    Roskam, Airplane Design Part 8           *
* [4]    Torenbeek, Synthesis of Subsonic Airplane Design  *
*                          by                              *
*                      H. M. Pasaribu                      *
*                         March 1990                       *
*                                                          *
************************************************************
```

******* PASSENGER CABIN **********

INTERNAL DIMENSION

INTERNAL DIAMETER DEFINED BY CABIN DIMENSION FOR ECONOMY CLASS

INTERNAL DIAMETER	=	5.37 M
EXTERNAL DIAMETER	=	5.67 M
FLOOR LEVEL	=	0.64 M BELOW CENTRELINE

CLASS	INTERNAL DIAMETER	FLOOR HEIGHT
FIRST CLASS	5.19	0.64
BUSINESS CLASS	5.37	0.64
ECONOMY CLASS	5.37	0.64

NUMBER OF PASSENGER (MIXED CLASS)= 295

******* UNDERFLOOR CARGO VOLUME ********

****** STANDARD CONTAINER LIST *******

TYPE	BOX	SYMMETRY	TOP WIDTH	BASE WIDTH	BIG HEIGHT	SIDE HEIGHT
TYPE-727	NO	YES	94.00	45.50	43.00	26.40
TYPE-LD1	NO	NO	92.00	61.00	64.00	31.50
TYPE-LD2	NO	NO	61.50	47.00	64.00	47.40
TYPE-LD3	NO	NO	79.00	61.50	64.00	44.40
TYPE-LD4	YES	YES	96.00	96.00	64.00	41.00
TYPE-LD6	NO	YES	160.00	125.00	64.00	44.40
TYPE-LD8	NO	YES	125.00	96.00	64.00	47.40
TYPE-747	NO	YES	186.00	125.00	64.00	31.50

**** CALCULATED DIMENSION OF CONTAINER LIST ****

TYPE	X TOP	X BASE	Y CENTRE	Y SIDE	AREA
TYPE-727	1.19	0.58	1.09	0.67	0.76
TYPE-LD1	1.56	0.77	1.63	0.80	1.58
TYPE-LD2	0.97	0.60	1.63	1.20	1.05
TYPE-LD3	1.23	0.78	1.63	1.13	1.38
TYPE-LD4	1.22	1.22	1.63	1.04	1.98
TYPE-LD6	2.03	1.59	1.63	1.13	2.69
TYPE-LD8	1.59	1.22	1.63	1.20	2.06
TYPE-747	2.36	1.59	1.63	0.80	2.90

****** LIST OF CONTAINER *******
IN ORDER OF INCREASING AREA

TYPE	X TOP	X BASE	Y CENTRE	Y SIDE	AREA
TYPE-727	1.19	0.58	1.09	0.67	0.76
TYPE-LD2	0.97	0.60	1.63	1.20	1.05
TYPE-LD3	1.23	0.78	1.63	1.13	1.38
TYPE-LD1	1.56	0.77	1.63	0.80	1.58
TYPE-LD4	1.22	1.22	1.63	1.04	1.98
TYPE-LD8	1.59	1.22	1.63	1.20	2.06
TYPE-LD6	2.03	1.59	1.63	1.13	2.69
TYPE-747	2.36	1.59	1.63	0.80	2.90

THE CONTAINER(S) THAT CAN BE FITTED INTO UNDERFLOOR
BELLY COMPARTMENT (IN ORDER OF INCREASING AREA) :
TYPE-727
TYPE-LD2

THE BIGGEST CONTAINER TYPE THAT CAN BE FITTED IS TYPE-LD1

MAXIMUM NUMBER OF PASSENGER = 360

****** PASSENGER AND EMERGENCY EXITS ******

	INDEX	TYPE	QUANTITY
1	1	TYPE-A	3
2	2	TYPE-I	1

DOOR TYPE AT NOSE SECTION = TYPE-A
AT TAIL SECTION = TYPE-A

PASSSENGER CLASS AT NOSE SECTION = FIRST CLASS
TAIL SECTION = ECONOMY CLASS
(IF APPLICABLE)

***** GALLEY AND TOILET SERVICE ******

	FRONT	MID	REAR	TOTAL
NO OF TOILETS	1	2	5	8
TOILET AREA	1.500	3.000	7.500	12.000
TROLLEY AT CLASS	50	14	22	
NO OF TROLLEY	14	14	22	50
NO OF TROLLEYS	14	20	16	50
GALLEY AREA	6.739	9.411	7.630	23.780

TOTAL GALLEY LENGTH = 3.647
TOTAL TOILET LENGTH = 1.782

TOTAL GALLEY AND TOILET LENGTH = 5.429

****** FUSELAGE NOSE SECTION ******
(ALL IN METER)

TOTAL LENGTH	=	10.378
RATIO OVER DIAMETER	=	1.830
NOSE SECTION COMPRISES OF :		
RADOME LENGTH	=	0.624
FLIGHT DECK LENGTH	=	3.500
PASSENGER DOOR WIDTH	=	0.762
GALLEY/TOILET WIDTH	=	1.211
NUMBER OF PASSENGERS	=	24
NUMBER OF SEAT ROWS	=	4
DELTA LENGTH	=	11.254

****** FUSELAGE TAIL SECTION ******

(ALL IN METER)

TOTAL LENGTH	=	17.013
RATIO OVER DIAMETER	=	3.000
TAIL SECTION COMPRISES OF :		
UNPRESSURISED SECTION LENGTH	=	7.486
GALLEY/TOILET WIDTH	=	2.223
PASSENGER DOOR WIDTH	=	1.067
NUMBER OF PASSENGERS	=	56
NUMBER OF SEAT ROWS	=	7
DELTA LENGTH	=	9.527

FUSELAGE DIMENSION (ALL IN METER) :

MAXIMUM DIAMETER	=	5.67
MAXIMUM HEIGHT	=	5.67
PASSENGER AREA LENGTH	=	38.79
CABIN LENGTH (EXCL. FLIGHTDECK)	=	48.18
FLIGHT DECK LENGTH	=	3.50
GALLEY AND TOILET LENGTH	=	5.43
EXIT LENGTH	=	3.96
TOTAL LENGTH	=	58.29
FINENESS RATIO	=	10.279
NUMBER OF PASSENGER (MIXED) IN 3 CLASS LAYOUT	=	295
MAXIMUM NUMBER OF PASSENGER	=	360

G.3 WEIGHT ESTIMATION

G.3.1 INPUT DATA

The following data are used in the calculation :

Type of aircraft	= 1 (Commercial Transport)
Type of powerplant	= 0 (Jet engines)
Number of Passenger	= 335
Weight of Pasengers (kg.)	= 31825.000
Weight of cargo (kg.)	= 13675.000
Cruise Mach number	= 0.800
Cruise altitude (km.)	= 9.150
Range (km.)	= 13805.000
Take-off distance (m.)	= 2956.000
Aspect ratio	= 9.650
Wing area (m^2)	= 388.200

G.3.2 OUTPUT DATA

The output is given in the following pages.

AIRCRAFT WEIGHT BREAKDOWN
==========

STRUCTURE GROUP :

Wing	=	29098.7	kg
Fuselage	=	25516.3	kg
Tail unit	=	5265.0	kg
Surface controls	=	1638.6	kg
Nacelles	=	2046.2	kg
Landing gear	=	9418.5	kg
Others	=	2189.5	kg
Total structure weight	=	75172.8	kg

PROPULSION GROUP :

Engines	=	10231.2	kg
Fuel system	=	1536.4	kg
Others	=	353.0	kg
Total propulsion weight	=	12120.6	kg

EQUIPMENT AND SYSTEM :

Furnishing	=	16981.3	kg
Air Conditioning system	=	2709.3	kg
Others	=	25080.3	kg
Total equipment weight	=	44770.9	kg

TOTAL EMPTY WEIGHT	=	132064.3	kg

USEFUL PAYLOAD :

Passengers + luggage	=	31825.0	kg
Cargo	=	13675.0	kg
Total payload weight	=	45500.0	kg

TOTAL FUEL WEIGHT	=	99449.5	kg

TOTAL TAKE-OFF WEIGHT	=	277013.8	kg

G.4 AERODYNAMIC ANALYSIS

G.4.1 INPUT DATA

The following data are used in the calculation :

Unit system used	= 1 (Metric)
Type of output	= 1 (Report)
Temperature difference from ISA condition (°C)	= 0.0
Additional drag coefficient	= 0.0
Type of airfoil	= 2 (Advanced)
Wign span (m.)	= 61.21
Wing area (m^2)	= 388.200
Wing wetted area (m^2)	= 648.350
Wing mean aerodynamic chord (m.)	= 6.960
Wing aspect ratio	= 9.650
Wing sweep angle at quarter chord (°)	= 30.0
Wing taper ratio	= 0.300
Wing thickness ratio	= 0.120
Wing camber	= 0.000
Fuselage length (m.)	= 58.290
Fuselage maximum height (m.)	= 5.670
Fuselage maximum diameter (m.)	= 5.670
Fuselage base area (m^2)	= 0.0
Fuselage cross sectional area (m^2)	= 25.260
Fuselage wetted area (m^2)	= 1426.360
Tailplane mean areodynamic chord (m.)	= 5.0
Tailplane wetted area (m^2)	= 180.0
Tailplane thickness ratio	= 0.1
Fin mean aerodynamic chord (m.)	= 4.500
Fin wetted area (m2.)	= 135.0
Fin thickness ratio	= 0.1

G.4.2 OUTPUT DATA

The output is given in the following pages.

DELTA METHOD DRAG ESTIMATE VERSION 2.2 - INPUT DATA

TITLE : Airbus-A340 Aerodynamic Characteristics

INPUT DATA UNITS 0=imperial 1=metric	OUTPUT TYPE 1=report 2=detailed	TEMP DIFF FROM ISA CONDITIONS C	ADDITIONAL ZERO LIFT DRAG COFFICIENT
1	1	0.0	0.0000

WING :	AEROFOIL TYPE 1=conventional 2=advanced	SPAN m or ft	AREA m2 or ft2	WETTED AREA m2 or ft2	MEAN AERO. CHORD m or ft
	2	61.210	388.20	648.35	6.960

WING :	ASPECT RATIO	C/4 SWEEP	LE SWEEP	TAPER RATIO	t/c	CAMBER 100{z(u)-z(l)}/2c @ 70% semi-span
	9.650	30.0	0.0	.300	.120	0.000

FUSELAGE :	LENGTH	HEIGHT	WIDTH	BASE AREA	MAX XSECT AREA EXCL INLETS	WETTED AREA
	m or ft	m or ft	m or ft	m2 or ft2	m2 or ft2	m2 or ft2
	58.290	5.670	5.670	0.00	25.26	1426.36

TAILPLANE :	MAC m or ft	WETTED AREA m2 or ft2	t/c %	FIN :	MAC m or ft	WETTED AREA m2 or ft2	t/c %
	5.000	180.00	.100		4.500	135.00	.100

CASE DATA :	CASE No.	MACH No.	ALTITUDE m or ft	AIRCRAFT MASS Kg or lb
	1	0.80	9150.0	264000.0
	2	0.82	9150.0	264200.0

DELTA METHOD DRAG ESTIMATE : Airbus-A340 Aerodynamic Characteristics

CASE 1 OF 2

FREE STREAM MACH No. = 0.80
ALTITUDE = 9150.0 m
AIRCRAFT MASS = 264000.0 Kg
TRUE AIRSPEED = 242.6 m/s
AIR TEMPERATURE = 228.7 K
AIR DENSITY = 0.458 Kg/m3
UNIT REYNOLDS NO. = 7.472E+06 per m

ZERO LIFT DRAG COEFFICIENT :

CDF = 0.0154
CDC = 0.0012
CDADD = 0.0000

CDMIN = 0.0166

OVERALL DRAG POLAR : CD = .0166 + .0429 CL2

CL	CD
0.1039	0.0176
0.2039	0.0182
0.3039	0.0203
0.4039	0.0233
0.4539	0.0251
0.5039	0.0275
0.5539	0.0304
0.6039	0.0334
0.7039	0.0413
0.8039	0.0557

SPECIFIC CASE OUTPUT :

CASE CD = 0.0270
CASE CL = 0.4945
BUFFET ONSET CL = 0.6633
DESIGN CL = 0.5039
DRAG RISE MACH No. = 0.8402
CASE DRAG FORCE = 142.4 KN

DELTA METHOD DRAG ESTIMATE : Airbus-A340 Aerodynamic Characteristics

CASE 2 OF 2

FREE STREAM MACH No. = 0.82
ALTITUDE = 9150.0 m
AIRCRAFT MASS = 264200.0 Kg
TRUE AIRSPEED = 248.7 m/s
AIR TEMPERATURE = 228.7 K
AIR DENSITY = 0.458 Kg/m3
UNIT REYNOLDS NO. = 7.659E+06 per m

ZERO LIFT DRAG COEFFICIENT :

CDF = 0.0153
CDC = 0.0015
CDADD = 0.0000

CDMIN = 0.0169

OVERALL DRAG POLAR : CD = .0169 + .0432 CL2

CL	CD
0.1039	0.0179
0.2039	0.0184
0.3039	0.0206
0.4039	0.0236
0.4539	0.0254
0.5039	0.0278
0.5539	0.0309
0.6039	0.0341
0.7039	0.0432
0.8039	0.0565

SPECIFIC CASE OUTPUT :

CASE CD = 0.0262
CASE CL = 0.4710
BUFFET ONSET CL = 0.6398
DESIGN CL = 0.5039
DRAG RISE MACH No. = 0.8402
CASE DRAG FORCE = 144.8 KN

G.5 FLIGHT ANALYSIS

G.5.1 INPUT DATA

The following data are used in the calculation :

Number of passengers	= 295
Take-off weight estimate (kg.)	= 264731.4
Empty weight (kg.)	= 132064.3
Weight of payload (kg.)	= 27639.0
Propulsion Type	= 0 (Jet Engines)
Wing Aspect ratio	= 9.65
Wing gross area (m^2)	= 388.2
Drag equation	= 0 (independent of Mach number)
C_{Do}	= 0.0166
$\partial C_D/\partial C_L$	= 0.0
$\partial C_D/\partial C_L^2$	= 0.0429
Number of engines	= 4
Engine Specification Name	= 0 (GE CF-6)
Gross thrust per engine (kg.)	= 14041.0
Reduction in SFC (%)	= 0.0
Cruise Mach number	= 0.8
Range (km.)	= 13805.0
Initial cruise altitude (km.)	= 9.15
Cruise method	= 1 (Constant Mach number)
Diversion range (km.)	= 370.0
Altitude for diversion flight (km.)	= 7.0

G.5.2 OUTPUT DATA

The output is given in the following pages.

```
*********************************************************
*                                                       *
*                   FLIGHT ANALYSIS                     *
*       a computer program for flight analysis of       *
*            subsonic commercial transports             *
*                                                       *
*                   developed based on                  *
*          [1]    NASA TM 81857, September 1980        *
* [2]    Torenbeek, Synthesis of Subsonic Airplane Design *
*          [3]    Roskam, Airplane Design Part 8       *
*                      by                               *
*                  H. M. Pasaribu                       *
*                     June 1990                         *
*                                                       *
*********************************************************
```

*** FUNCTION INPUT ***

ALL INPUT IN METRIC UNIT

NUMBER OF PASSENGERS		=	295
INITIAL TAKE-OFF WEIGHT	(KG.)	=	264731.4
EMPTY WEIGHT	(KG.)	=	132064.3
PAYLOAD WEIGHT	(KG.)	=	27639.0

GENERAL DATA :

ENGINE TYPE		=	JET ENGINES
ASPECT RATIO		=	9.65
WING AREA	(SQ.M)	=	388.20
AIRCRAFT DRAG MODEL		=	0

DRAG POLAR :

$$CD = 0.01660 + 0.00000\ CL + 0.04290\ CL^2$$

FLIGHT DATA :

MACH NUMBER		=	0.80
RANGE	(KM.)	=	13805.00
INITIAL CRUISE ALTITUDE	(KM.)	=	9.1
CRUISE METHOD SELECTION		=	1
RESERVE RANGE	(KM.)	=	370.0
ALTITUDE FOR RESERVE FLIGHT	(KM.)	=	7.0

ENGINE DATA :

ENGINE TYPE NO.		=	0
NUMBER OF ENGINE		=	4
GROSS THRUST PER ENGINE	(KG.)	=	14041.0
SFC REDUCTION FACTOR		=	0.00

*** WEIGHT ESTIMATION ***

EMPTY WEIGHT	132064.3
FIXED WEIGHT	27639.0
FUEL	110190.9
TAKE OFF WEIGHT	269894.2
NO. OF ITERATIONS REQUIRED	6

******** CLIMB HISTORY *********

ALT(BEG) KM.	ALT(END) KM.	V KM/H	MACH	ROC M/SEC	GAMA	WBEG KG.	WTFUEL KG.	TIME HR	RANGE KM.
0.46	0.89	475.37	0.388	11.560	0.0880	261797.4	210.92	0.010	4.95
0.89	1.33	485.55	0.399	11.102	0.0828	261586.5	221.08	0.011	5.26
1.33	1.76	496.05	0.409	10.640	0.0776	261365.4	232.27	0.011	5.61
1.76	2.20	506.89	0.420	10.174	0.0726	261133.1	244.66	0.012	6.00
2.20	2.63	518.08	0.432	9.706	0.0678	260888.5	258.41	0.012	6.43
2.63	3.07	529.64	0.444	9.235	0.0631	260630.0	273.77	0.013	6.91
3.07	3.50	541.58	0.456	8.761	0.0585	260356.3	290.99	0.014	7.45
3.50	3.93	553.93	0.469	8.285	0.0541	260065.3	310.42	0.015	8.06
3.93	4.37	566.68	0.482	7.806	0.0498	259754.8	332.47	0.015	8.75
4.37	4.80	579.88	0.496	7.325	0.0457	259422.4	357.69	0.016	9.55
4.80	5.24	593.53	0.511	6.842	0.0417	259064.7	386.75	0.018	10.46
5.24	5.67	607.66	0.526	6.358	0.0378	258677.9	420.57	0.019	11.53
5.67	6.11	622.29	0.541	5.872	0.0341	258257.4	460.35	0.021	12.79
6.11	6.54	637.45	0.558	5.383	0.0305	257797.0	507.78	0.022	14.29
6.54	6.98	653.15	0.575	4.894	0.0271	257289.2	565.19	0.025	16.11
6.98	7.41	669.43	0.592	4.402	0.0238	256724.0	636.00	0.027	18.35
7.41	7.85	686.31	0.611	3.909	0.0206	256088.0	725.41	0.031	21.19
7.85	8.28	703.83	0.630	3.414	0.0175	255362.6	841.65	0.035	24.89
8.28	8.72	722.01	0.650	2.916	0.0146	254521.0	998.64	0.041	29.89
8.72	9.15	740.88	0.672	2.418	0.0118	253522.4	1221.80	0.050	36.99
TOTAL							9496.82	0.420	265.47

WEIGHT AT END OF CLIMB = 252300.56 KG.

TIME FOR GROUND MANOEUVRE = 0.43 HOURS

*** CRUISE ANALYSYS ***

LEG	MACH NO	CL	V KM/H	TIME HR	L/D	TSFC KG/KG/HR	T/T(IN)	ALT(BEG) KM.	ALT(END) KM.	WT(BEG) KG.	WT(END) KG.	DIST KM.	TCON	GAMA RAD.	CLM
1	0.80	0.475	877.2	1.490	18.08	0.221	0.282	9.1	9.4	253522.4	243542.0	1306.7	1.129	0.0001	0.692
2	0.80	0.475	873.9	1.495	18.07	0.220	0.275	9.4	9.7	243542.0	233953.7	1306.7	1.143	0.0001	0.692
3	0.80	0.474	870.6	1.501	18.07	0.219	0.267	9.7	9.9	233953.7	224741.4	1306.7	1.157	0.0001	0.692
4	0.80	0.474	867.4	1.507	18.07	0.219	0.259	9.9	10.2	224741.4	215890.1	1306.7	1.169	0.0001	0.692
5	0.80	0.474	864.2	1.512	18.06	0.218	0.251	10.2	10.4	215890.1	207385.3	1306.7	1.180	0.0001	0.692
6	0.80	0.473	860.9	1.518	18.06	0.217	0.243	10.4	10.7	207385.3	199213.4	1306.7	1.188	0.0001	0.692
7	0.80	0.473	857.7	1.524	18.05	0.216	0.235	10.7	10.9	199213.4	191361.4	1306.7	1.195	0.0001	0.692
8	0.80	0.472	854.5	1.529	18.05	0.216	0.232	10.9	11.2	191361.4	183796.7	1306.7	1.228	0.0001	0.692
9	0.80	0.474	853.8	1.530	18.06	0.217	0.223	11.2	11.5	183796.7	176510.9	1306.7	1.229	0.0001	0.692
10	0.80	0.475	853.8	1.530	18.08	0.217	0.214	11.5	11.8	176510.9	169494.0	1306.7	1.229	0.0001	0.692

****** DESCENT FLIGHT ******

ALTITUDE KM.	WEIGHT KG.	SPEED KM/H	CL	TIME HR.	FUEL KG.	RANGE KM.
11.77	169423.20	853.819	0.476	0.029	70.771	24.484
11.16	169349.20	853.819	0.432	0.029	74.002	24.484
10.55	169270.64	859.560	0.393	0.028	78.562	24.484
9.94	169185.98	867.281	0.357	0.028	84.658	24.484
9.33	169094.09	874.934	0.326	0.028	91.894	24.484
8.72	168993.80	882.520	0.297	0.028	100.295	24.484
8.11	168883.89	890.041	0.272	0.028	109.906	24.484
7.50	168763.11	897.500	0.249	0.027	120.789	24.484
6.89	168630.09	904.897	0.228	0.027	133.024	24.484
6.28	168483.39	912.234	0.209	0.027	146.712	24.484
5.67	168323.45	914.135	0.195	0.027	159.934	24.484
5.06	168161.92	884.200	0.194	0.028	161.539	24.484
4.45	167998.69	855.684	0.194	0.029	163.240	24.484
3.84	167833.64	828.500	0.194	0.030	165.038	24.484
3.23	167666.70	802.567	0.194	0.031	166.935	24.484
2.63	167497.78	777.809	0.194	0.031	168.935	24.484
2.02	167326.73	754.159	0.194	0.032	171.041	24.484
1.41	167153.47	731.550	0.193	0.033	173.256	24.484
0.80	166977.89	709.923	0.193	0.034	175.584	24.484
0.19	166923.31	689.224	0.193	0.011	54.577	7.506
TOTAL =				0.565	2570.692	472.709

TOTAL MISSION RANGE (KM.) = 13805.00
CLIMB DISTANCE (KM.) = 265.47
DESCENT DISTANCE (KM.) = 472.71

*** MISSION RESERVE ***

LEG	MACH NO	CL	V KM/H	TIME HR	L/D	TSFC KG/KG/HR	T/T(IN)	ALT(BEG) KM.	ALT(END) KM.	WT(BEG) KG.	WT(END) KG.	DIST KM.	TCON	GAMA RAD.	CLM
11	0.52	0.547	589.3	0.628	18.58	0.176	0.393	7.0	7.0	169494.0	167281.7	370.0	2.417	0.0000	0.692

** FUEL WEIGHT ANALYSIS **

TAKE OFF	269894.22
START-CRUISE	261797.38
MID-CRUISE	207385.28
END-CRUISE	169493.98
AFTER-RESERVE	167281.66
AFTER DESCENT/TAXI	166923.31
NET FUEL WEIGHT (KG.)	110190.90

* CRUISE ALTITUDE *

LEG1	9.15
LEG2	11.77
RESERVE LEG	7.00

FLIGHT LENGTH (HR)	16.27
AVERAGE SPEED (KM/H)	848.48
BLOCK TIME (HR)	16.70
BLOCK SPEED (KM/H)	826.71
BLOCK FUEL (KG.)	99881.75
BLOCK FUEL (GALS)	34409.71
KM./GAL	0.40
SEAT KM./GAL	0.00
INSTALLED THRUST	56164.00
NO. OF ENGINES	4
ENGINE THRUST (KG.)	14041.00
REFERENCE ENGINE (KG.)	18640.91
SCALE FACTOR	0.753

```
FUEL FLOW RATE      ( KG. /BLKHR) =   5981.391
MID CRUISE L/D                    =     18.061
MID CRUISE DRAG COEFFICIENT       =      0.026
MID CRUISE LIFT COEFFICIENT       =      0.474
MID CRUISE ALTITUDE FOR (L/D) MAX =  42251.973
MID CRUISE WEIGHT          ( KG. ) = 207385.281
MID CRUISE ALTITUDE        ( KM. ) =     10.443
MID CRUISE SPEED           (KM/HR) =    864.153
LANDING WEIGHT             ( KG. ) =   161915.6
```

********** END OF RESULTS **********

G.6 COST ESTIMATION

G.6.1 INPUT DATA

The following data are used in the calculation :

Type of aircraft	= 1 (Commercial Transport)
Number of passengers	= 295
Take off weight (kg.)	= 259894.22
Empty weight of aircraft (kg.)	= 132064.3
Maximum landing weight (kg.)	= 220000.0
Payload weight (kg.)	= 27639.0
Manufacturer's airframe weight (kg.)	= 102439.0
Cargo weight (kg.)	= 0.0
Type of propulsion	= 0 (Jet Engines)
Wing area (m^2)	= 388.2
Maximum diving speed (km/hr)	= 954.17
Centre-of-gravity control	= 1 (installed)
Active control system	= 1 (installed)
Number of engines	= 4
Propeller type	= 1 (Not applicable)
Thrust per engine (kg.)	= 14041.0
Aircraft mission range (km.)	= 13805
Aircraft cruise speed (km/hr)	= 848.48
Mission flight time (hr)	= 16.27
Mission block hour (hr)	= 16.71
Aircraft utilisation per year (hrs/yr)	= 3200
Mission fuel weight excl. reserve (kg)	= 99881.75
Passenger load factor	= 0.65
Fare ($/seat-naut.mile)	= 0.09
Fuel price ($/gallon)	= 0.57
Purchase price cost boost	= 0.0
Maintenance cost boost	= 0.0
Aircraft purchase price ($)	= 0.0 (unknown)
Powerplant unit cost ($)	= 2000000.0
Element of cost data produced	= 8 (All cost)
Methods used in maint. cost estimation	= 0 (American Airlines' model)
Year of evaluation (4 digits)	= 1991
Number of aircraft used for flight testing during development phase of the aircraft	= 6
Rate of aircraft production per month during development phase	= 0.5
Expected number of aircraft produced during production phase of the aircraft	= 700
Rate of aircraft production per month during production phase	= 4

Level of technology complexity/difficulty introduced in the aircraft	= 2 (Moderately advanced)
Use of CAD system in design	= 3 (used extensively)
Type of material used	= 1 (Conventional Al alloys)
Avionics system cost ($)	= 5000000.0
Interest rate for financing cost (%)	= 10.0
Percentage of profit expected (%)	= 10.0
Active control system cost ($)	= 1000000.0

G.6.2 OUTPUT DATA

The output is given in the following pages.

```
************************************************************
*                                                          *
*                        ACCOST                            *
*       program for calculation of aircraft costs          *
*                                                          *
*                     developed based on                   *
*            [1]    NASA TM 81857, September 1980         *
*      [2]    Nicolai, Fundamentals of Airplane Design     *
*           [3]    Roskam, Airplane Design Part 8          *
*                [4]   NASA TM 78694, May 1878             *
*          [5]    Raymer, Airplane Conceptual Design       *
*                          by                              *
*                     H. M. Pasaribu                       *
*                        April 1990                        *
*                                                          *
************************************************************
```

** AIRCRAFT COST ESTIMATES **

	DEVELOPMENT	PRODUCTION
ENGINEERING	3104441344.0	549257216.0
DEVELOPMENT SUPPORT	775259584.0	0.0
FLIGHT TEST	362783520.0	70000000.0
TOOLING	1187187968.0	2972889088.0
MANUFACTURING LABOUR	1084841984.0	21964328960.0
QUALITY CONTROL	144283984.0	2921255680.0
MATERIALS	239733216.0	10579100672.0
ENGINE	39219576.0	3660493824.0
AVIONICS	34500000.0	4024999936.0
ACTIVE CONTROLS SYSTEM	6000000.0	700000000.0
INTERIOR	0.0	613305024.0
TEST, SIMULATION FACILITY	697825216.0	0.0
PROGRAM FINANCING	697825216.0	4805562880.0
PROFIT	6978252.0	48055628.0
TOTAL	8380880384.0	49458253824.0

TOTAL COST PER AIRCRAFT = $ 82627336.00

** MAINTENANCE OPERATING COST **

NO.	SYSTEM	LABOUR	MATERIAL
1	INSPECTION	119.54	19.54
2	AIR CONDTN	10.41	9.22
3	AUTO PILOT	27.39	6.42
4	COMMUNICTN	16.61	7.10
5	ELECTRICAL	8.78	11.73
6	FURNISHING	69.71	39.62
7	FIRE PROT.	28.46	4.47
8	FLT CONTRL	18.67	13.09
9	FUEL	9.93	4.81
10	HYDR POWER	6.79	8.06
11	ICE & RAIN	1.75	2.21
12	INSTRUMENT	3.62	1.39
13	LAND. GEAR	48.43	109.78
14	LIGHTING	10.01	8.47
15	NAVIGATION	23.64	16.81
16	OXYGEN	2.65	4.53
17	PNEUMATIC	0.87	3.15
18	WAT/WASTE	2.91	4.62
19	AIRBRN APU	0.64	0.94
20	STRUCTURE	8.79	0.00
21	DOORS	5.95	5.51
22	FUSELAGE	15.45	1.19
23	NACELLES	2.75	1.14
24	WINGS	6.01	0.26
25	STABILIZER	1.70	0.76
26	WINDOWS	1.82	21.79

LABOUR COST	88.88
MATERIAL COST	60.11
ENGINE LABOUR COST	107.52
ENGINE MATERIAL COST	132.43

MAINTENANCE DOC IN 1989 DOLLARS PER HOUR = 388.95

ALL COSTS AND PRICE ARE CONVERTED TO DOLLARS IN 1991

** DIRECT OPERATING COSTS -- DOLLARS/FLT. HOUR **

	$ / FLT. HR	%
DEPREC	2276.39	37.66
SUPPORT	310.42	5.14
SPARES	155.21	2.57
DELAY	24.03	0.40
INSURANCE	362.15	5.99
FUEL COST	1174.34	19.43
MAINTENANCE	545.52	9.03
LANDING FEE	58.05	0.96
CREW	352.03	5.82
ATTENDANTS	591.64	9.79
FUEL SERVICE	180.26	2.98
CONTROL	14.15	0.23
TOTAL DIRECT OPERATING COSTS	$ 6044.19	100.00

* INDIRECT OPERATING COSTS -- DOLLARS/FLT. HOUR *

	$ / FLT. HR	%
MAIN BURDN	572.79	32.74
FOOD COST	337.65	19.30
MOVIE	16.46	0.94
PASS INSUR	62.42	3.57
MISC PASS.	4.46	0.25
ADVERTISE	328.11	18.76
COMMISSION	282.10	16.13
RESERVAT.	70.86	4.05
PASS HDLG.	46.22	2.64
BAG HDLG.	21.10	1.21
CARGO HDLG	0.00	0.00
SERVICING	7.12	0.41
TOTAL INDIRECT OPERATING COSTS	1749.30	100.00

** PERFORMANCE FUNCTION SUMMARY **

REVENUE PER BLOCK HOUR	14265.86
TOTAL COST PER BLOCK HOUR	7793.49
RETURN ON INVESTMENT	0.1033

1	DOC/HR	6044.188
2	DOC/FLT	100937.945
3	ROI	0.103
4	FARE	0.201
5	IOC	1749.300
6	PRICE	115889144.000
7	INVESTMENT	104300224.000
8	AIRFRAME	108476232.000
9	POWERPLANT	7412915.000
10	TOT. MAINT	1118.313

********** END OF RESULTS **********

G.7 CONFIGURATION ANALYSIS

The following is the detail of the example run of the configuration analysis module as briefly reported in section 6.3.2 of Chapter 6.

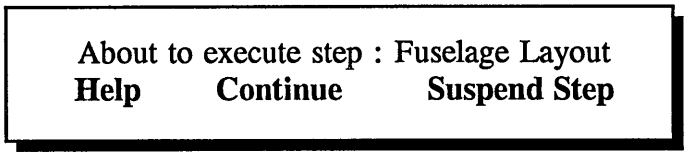
G.7.1 EDITING MODE

While in the editing mode, a few input data are required for the analysis. These could be updated by overwriting the current values on display. The current values, which were obtained from the database of the X-91 aircraft, were displayed on the screen as shown in Fig. G.1 in the following page. Once the editing of data was completed, the database need to be saved by selecting the **'Save'** button with the LEFT MOUSE BUTTON. It was decided to start the execution by switching into execution mode with the **'Evaluate Aircraft'** button.

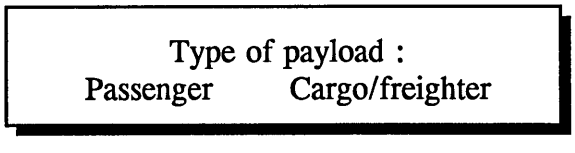
G.7.2 EXECUTION MODE

The Configuration Analysis window popped out once the execution was triggered with the **'Execute Program'** button which is available in the execution mode.

The first step suggested by the system was *Fuselage Layout*, which was written on the right of the **'Design Step :'** item. An explanation about this step can be obtained by activating the **'Help'** button on the bottom panel of the window. To start the analysis, the **'Execute Step'** button was activated. The system displayed the *execution menu* window with the following options :



Again an explanation can be requested through the **'Help'** button. This is in fact the same as the previous help facility which gives information about the active design step. To start the inference, select the **'Continue'** button. The system started the query for the fuselage layout by displaying a menu :



The 'Passenger' option was selected, to inform the system that the X-91 aircraft is a passenger transport aircraft. And then the following *question menu* was displayed :

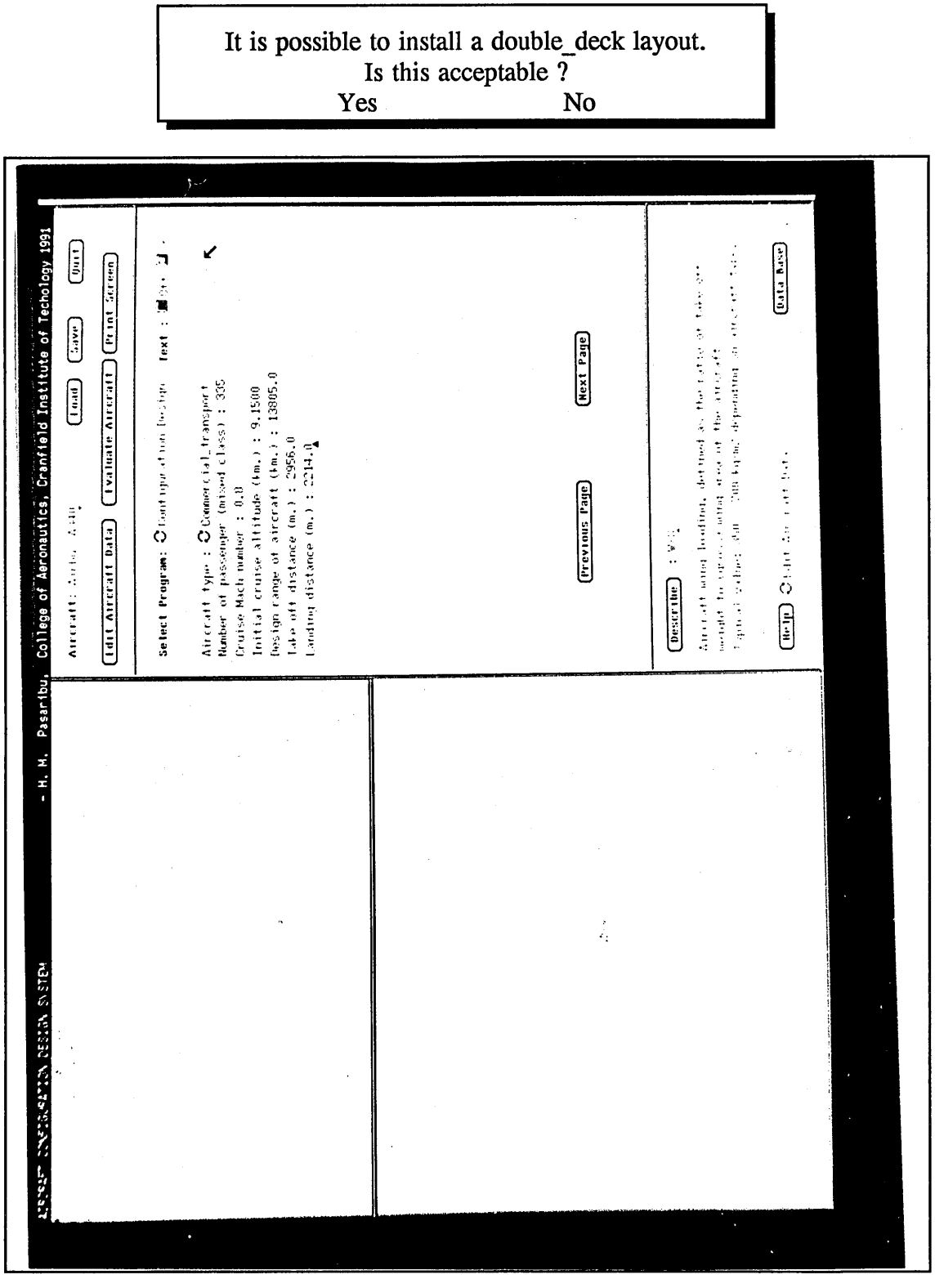


Fig. G.1 Editing mode for the configuration analysis module

APPENDIX G

The answer was *no*. However, the A340 is offered with a option of having a passenger cabin at the lower deck, underneath the main cabin floor, which can accommodate up to 50 passengers in five seat abreast layout. This will obviously take up some space previously allocated for the cargo containers. This possibility is also be offered for the X-91 aircraft as shown in the reasoning.

When the system completed the analysis, the following *control menu* was displayed as shown in Fig. G.2 as follow :

Design step : Fuselage Layout completed
Show Result Explain Result Continue

When the 'Explain Result' option was selected, the following information was displayed on the panel as shown in Fig. G.2 :

The result for Fuselage Layout is single_deck_double_aisle
Because :
Based on my calculation for the no_of_passengers you specified,
I ruled out the possibility of having a twin fuselage
Based on the no_of_passengers you specified, I calculated
double_deck is possible, and your answer is no
Based on your input data of aircraft_size, I defined
the aircraft as commercial_transport
Based on your input data of design_range, I defined
the aircraft as long_range
Based on your input data of number_of_passengers, I defined
the aircraft as intercontinental_aircraft

The system again displayed the same *control menu* to give a chance to the user to select other options. When the 'Show Result' option was pressed, the following information was presented in a pop-up window as shown in Fig. G.3 :

FUSELAGE LAYOUT
SINGLE BODY DOUBLE AISLE
A passenger seating layout with double aisles is normally required for large passenger aircraft with a single fuselage. With two aisles, the seats can be arranged so that the passenger is no further than one seat away from the aisle. This is necessary to increase the level of comfort around the cabin, through ease of movement. Only if triple seats are installed at each side of the cabin, the passengers seating next to the cabin wall will be two seats away from the aisle. This is however compensated by the fact that they have the luxury of sitting next to the windows.

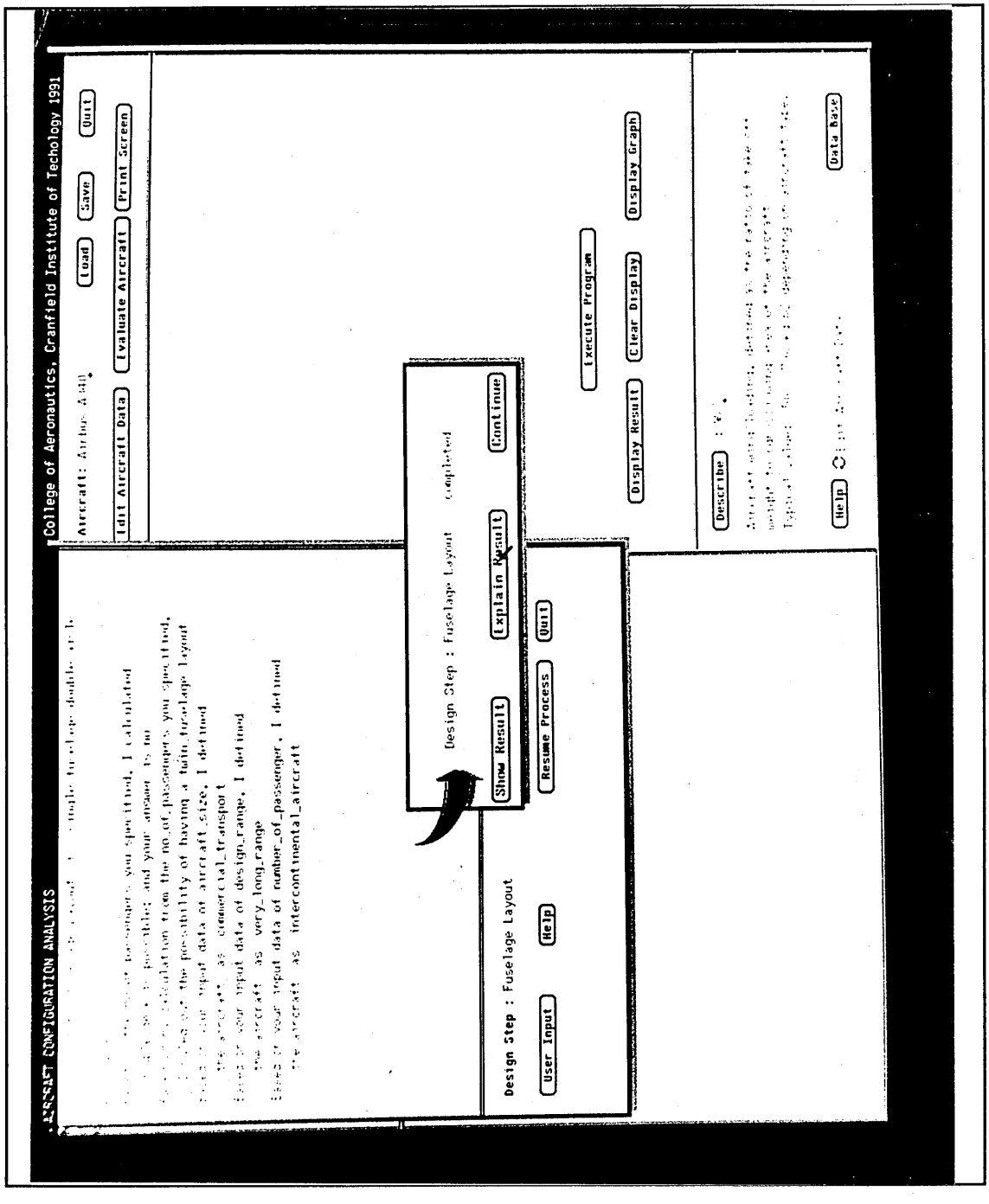


Fig. G.2 The explanation of the decision for the 'Fuselage Layout' step

Again the *control menu* was displayed. This time it was decided to continue the execution to the next design step. The next step suggested by system was *Propulsion Type*. At this stage, it was decided to suspend the analysis in the 'Propulsion Type' step, and continue to the next step.

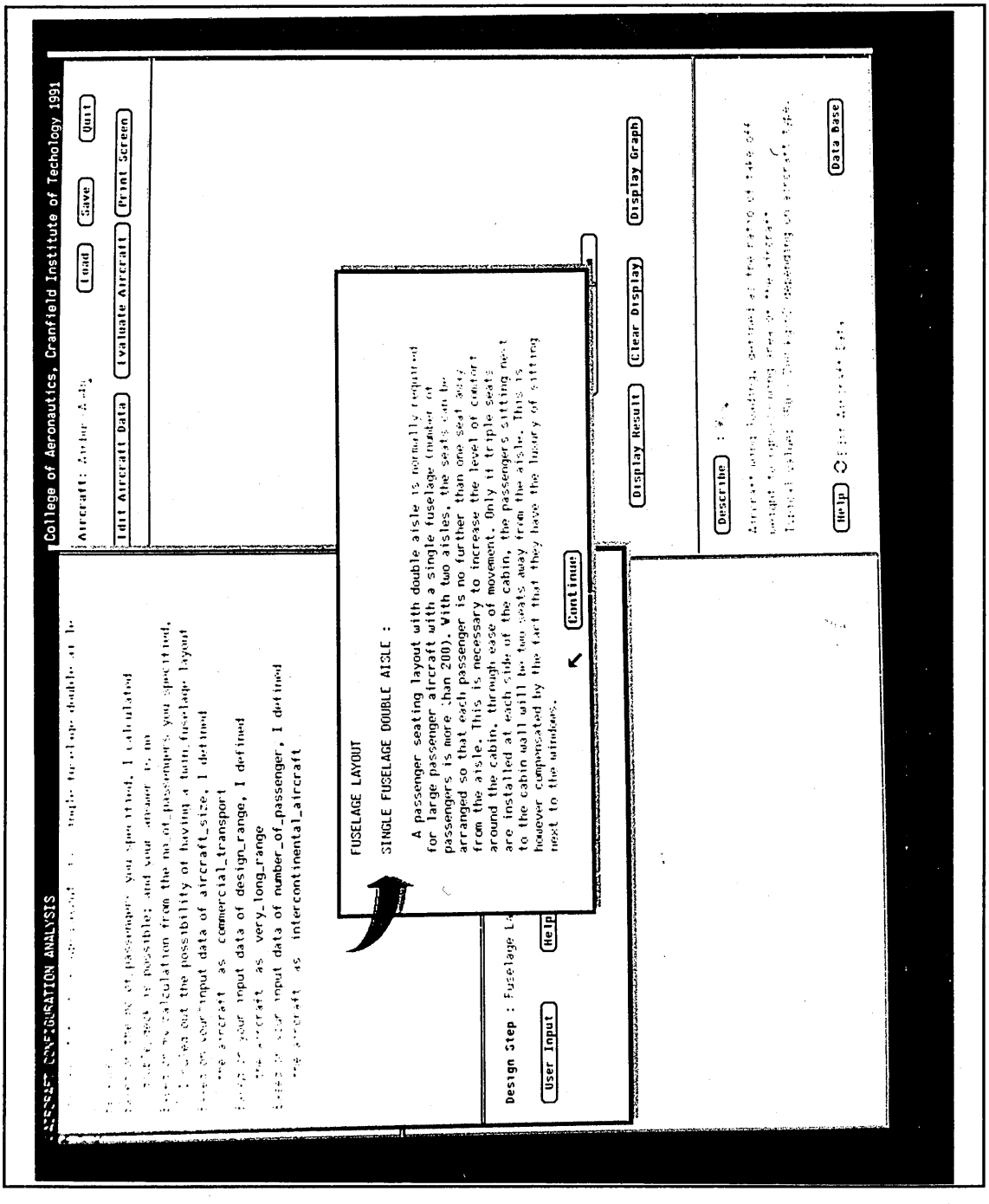


Fig. G.3 The explanation of the result for the 'Fuselage Layout' step

The next step suggested by system was *Wing Position*. An explanation about this design step was given when requested. This was done by selecting the 'Help' option in the *execution menu*. The following information was presented in a pop-up window :

WING POSITION
GENERAL :

Wing vertical location with respect to the fuselage is generally set by the real-world environment in which the aircraft will operate. Although the aerodynamic and structural differences are not without importance, they can only be deciding factors when the choice between high, low and mid wing is not dictated by considerations of maximum operational flexibility.

When executed, the system popped out the question menu :

Is it operating on semi-prepared runway ?
Yes No

The answer to this question was *no*. The result for the wing position, as shown in Fig. G.4, was explained as follows :

The result for Wing Position is low
Because :
Your answer to the question
'Is it operating on semi-prepared runway ?' is no
Based on your input data of aircraft_type, I defined
the aircraft as passenger aircraft

When requested, the following information was given for the 'Show Result' option as shown in Fig. G.5 :

WING POSITION
LOW WING :

Low wing configuration offers several advantages. The landing gear stowage can be easily provided. With a low wing, the trunnion about which the gear is retracted can be attached directly to the wing box which, being strong already, will not require much extra strengthening to absorb the landing load. Since the leg is comparatively short the retracted gear can be stowed easily inside the wing itself, thus eliminating the external blisters usually needed with high wing.

To provide the adequate engine and propeller ground clearance, the fuselage must be placed farther off the ground than a high wing aircraft. This will require a slightly longer landing gear leg. In most cases, a certain degree of dihedral is required to increase ground clearance, especially for the wing tips.

Since the fuselage compartment underneath the floor is normally large enough, the wing box can pass through the fuselage to reduce drag penalty. This layout removes the structural complexity associated with high and mid wing configurations.

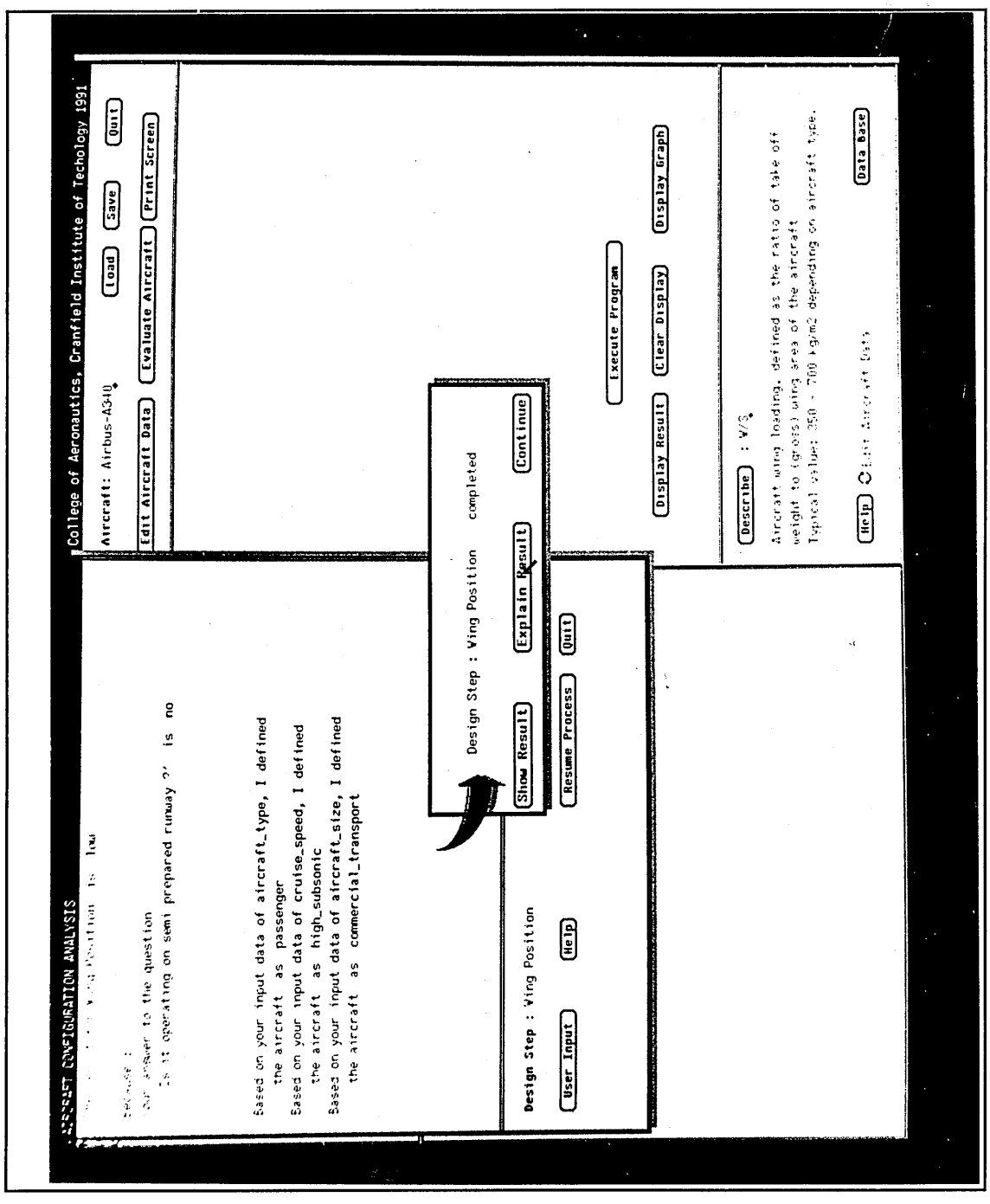


Fig. G.4 Explanation of decision in the 'Wing Position' step

Once the wing position had been set, the next step to tackle was *Propulsion Type*. Again the execution menu was displayed. An explanation for this step was given when requested as follows :

PROPULSION TYPE
GENERAL :

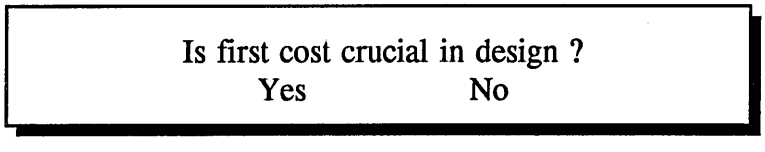
The selection of propulsion type depends strongly on the operational and performance requirements of the aircraft. There are currently two types of propulsion system available for application in civil aircraft : thrust producing propulsion system (jet engines) and power producing propulsion, in which the power is used to rotate the propellers to provide the required forward thrust (propeller engines).

The aircraft maximum speed is normally the dominant factor for selecting either one of these two types, in which the latter is effective for low speed aircraft since the propeller efficiency decreases as the speed increases.

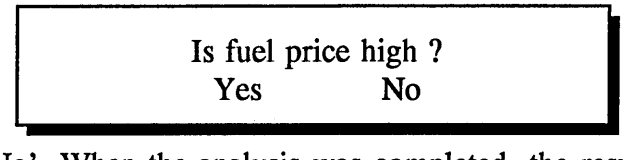
Having understood the activity involved in this step, the step was executed. After a little while, the system informed that the analysis was completed by displaying the control menu. When requested the conclusion for the propulsion type, as shown in Fig. G.6, was explained as follows :

The result for Propulsion Type is jet_engines
Because :
Based on your input data of aircraft_size, I defined
the aircraft as commercial aircraft
Based on your input data of cruise_altitude, I defined
the aircraft as high
Based on your input data of cruise_speed, I defined
the aircraft as high_subsonic
The decision for wing_position is low

Again the *control menu* was displayed. This time it was decided to continue the execution to the next design step. The next step to deal with was *Engine Type*. Again the inference began when the **'Continue'** option in the execution menu was pressed. The system displayed a question window :



The answer was 'Yes' since one of the design criteria was minimum cost. The system asked another :



The answer was 'No'. When the analysis was completed, the result was displayed in Fig. G.7 as follows :

The result for Engine Type is turbofan
Because :
The decision for propulsion_type is jet_engines
The decision for wing_position is low
Based on your input data of aircraft_size, I defined
the aircraft as commercial_aircraft
Based on your input data of cruise_altitude, I defined
the aircraft as high
Based on your input data of cruise_speed, I defined
the aircraft as high_subsonic
Your answer to the question
'Is first cost crucial for design ?' is yes
Your answer to the question
'Is fuel price high ?' is no

The explanation given for the Show Result option for this step was :

ENGINE TYPE
TURBOFANS :

Like turboprop engines, turbofan engines use a turbine to extract mechanical power from the exhaust gasses. This mechanical power is used to accelerate a larger mass of outside air with a ducted fan of one or several stages. This accelerated air is then split, with part remaining in the engine to further compression and burning, and the remainder being 'by passed' around the engine to exit unburned.

The 'by pass ratio', i.e the mass-flow ratio of the by passed air to the air that goes into the core of the engine, ranges from as high as 6 for most of the current turbofan engines to as low as 0.25 (the so-called 'leaky turbojet').

The specific fuel consumption of turbofan engines is relatively lower than that of turbojet engines, although its thrust diminishes quite rapidly at very high speed (supersonic).

The next step was Engine Position. When executed, the execution menu was displayed, from which the **'Continue'** button was selected to proceed with the analysis. The result of the decision was displayed in Fig. G.8 and explained as follows :

The result for Engine Position is wing
Because :
Based on my calculation from the engine_size,
I concluded that the engine_fuselage_clearance is not_ok
Based on my calculation from the engine_size,
I concluded that the engine_ground_clearance is ok
The decision for wing_position is low
The decision for propulsion_type is jet_engines

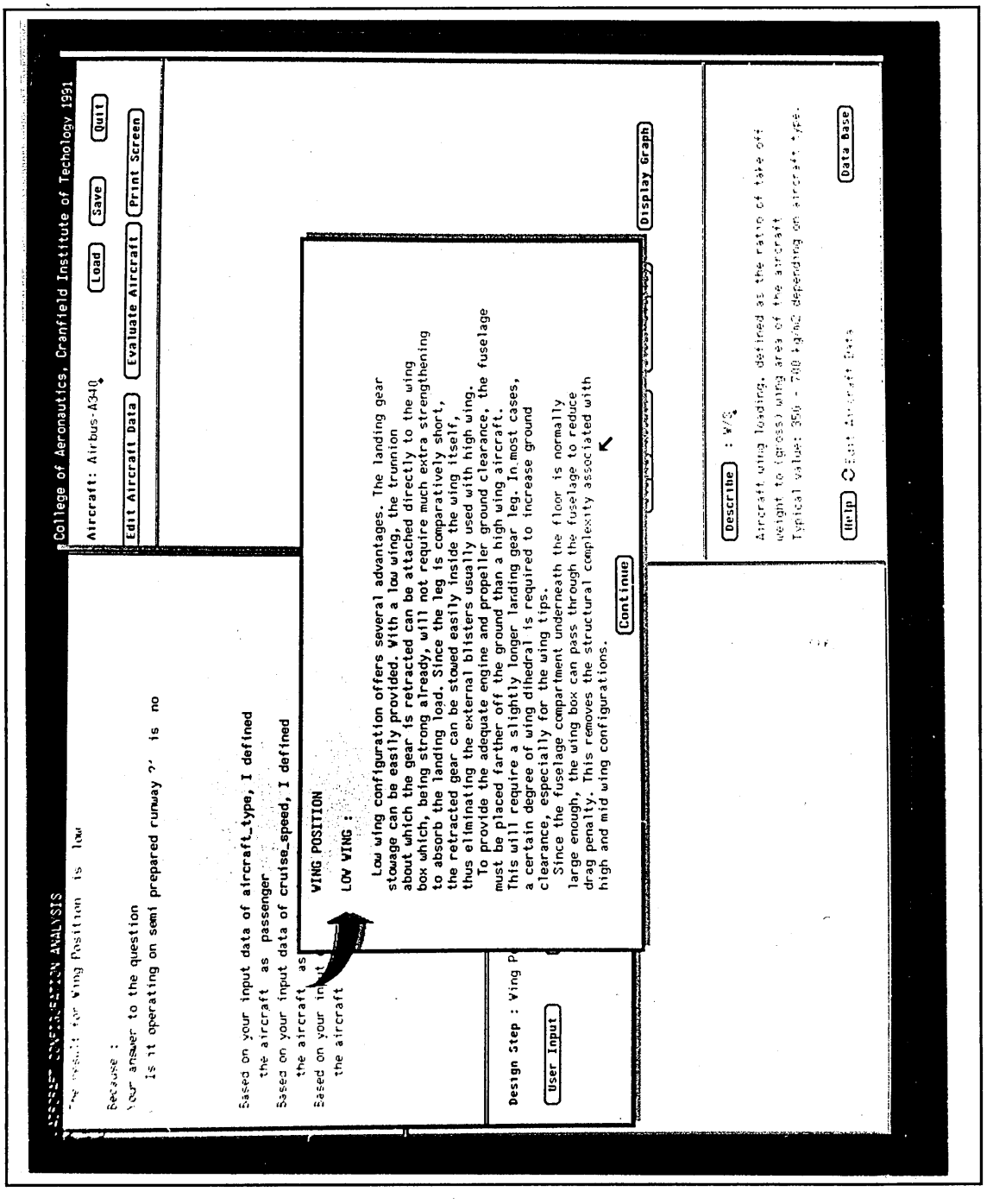


Fig. G.5 Result of the 'Wing Position' step

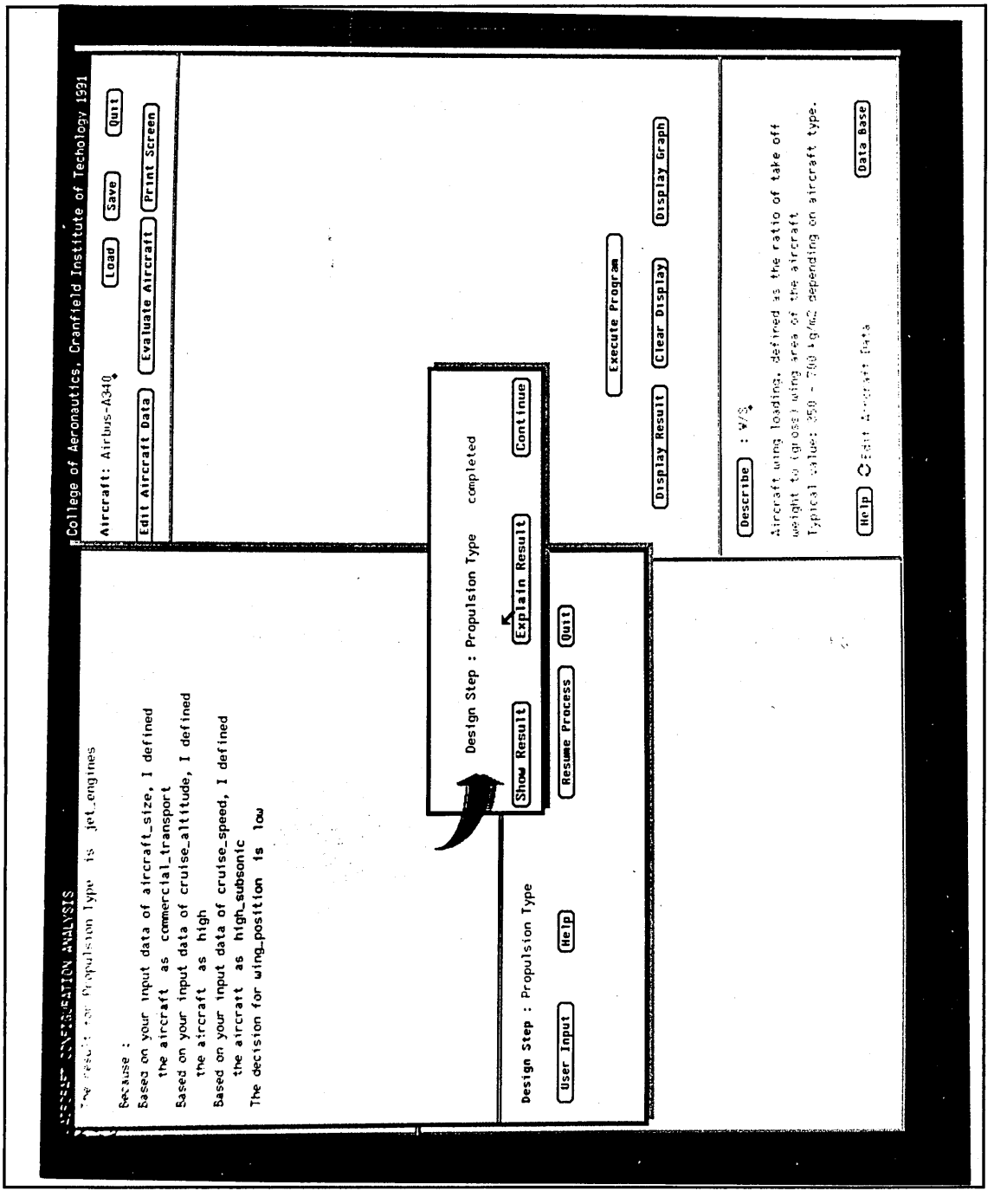


Fig G.6 The explanation for the 'Propulsion Type' decision

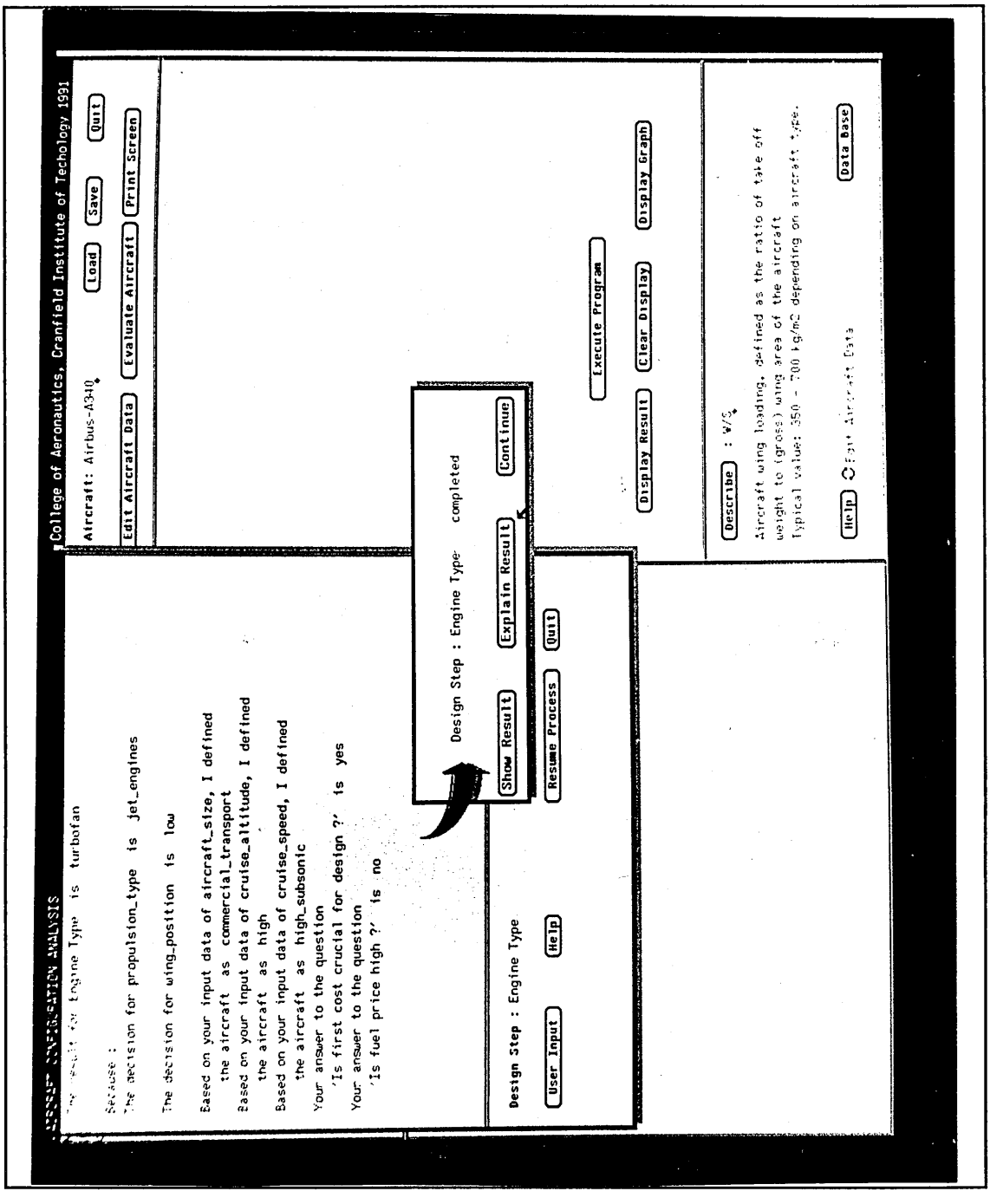


Fig G.7 The explanation for the 'Engine Type' decision

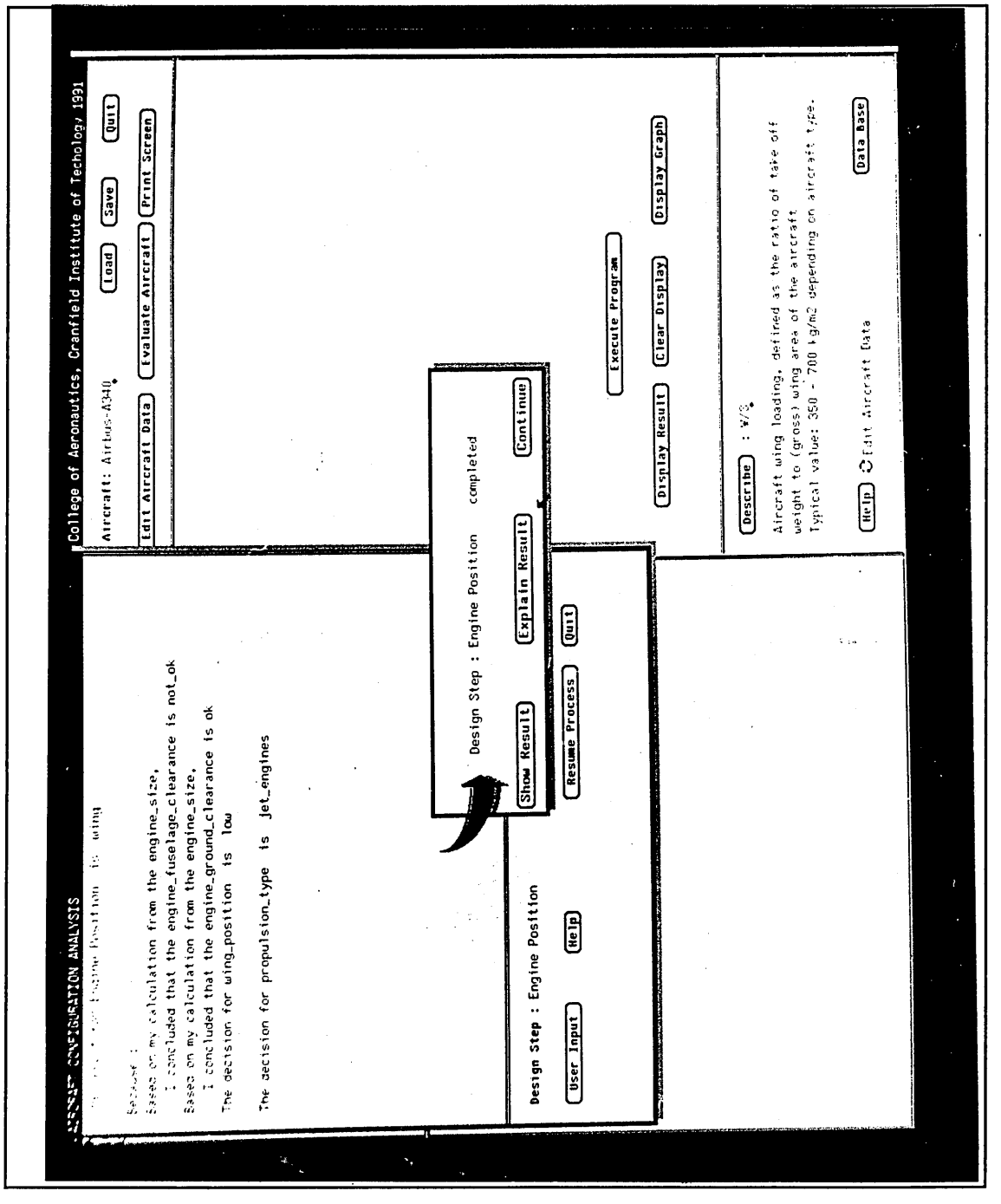


Fig G.8 The explanation for the 'Engine Position' decision

The explanation of the result was given when the '**Show Result**' was requested as shown in Fig. G.9.

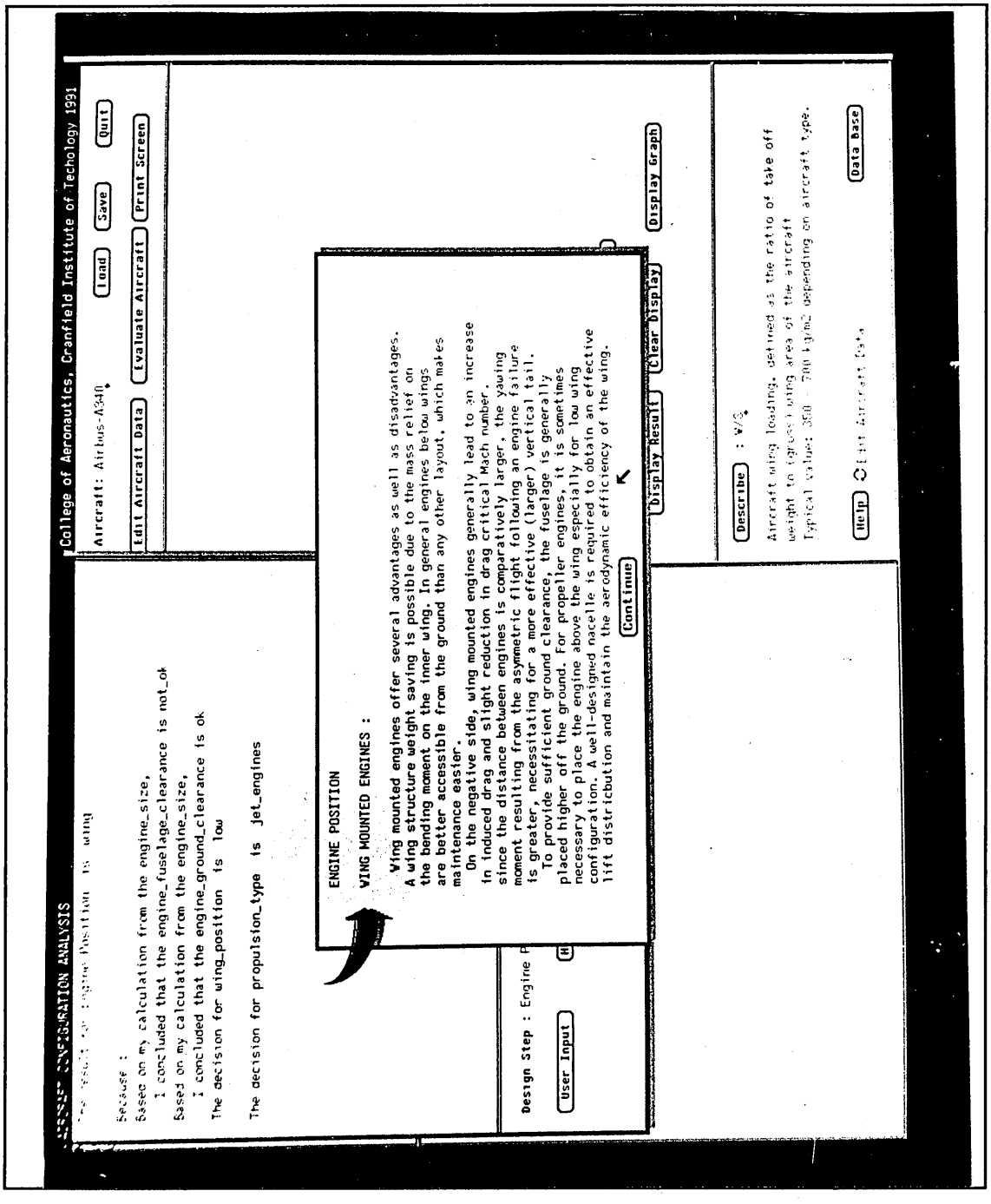


Fig. G.9 The result of the 'Engine Position' step

The next step to tackle was *Tail Layout*. An explanation of the step was given on request in the following :

TAIL LAYOUT
GENERAL :

The design of the tail surfaces probably depends more on the general arrangement and the detail layout of the aircraft than any other major component. Because of their location, their effectiveness is influenced by the wing and the operation of the engines, particularly in the case of propeller driven aircraft. The way in which the empennage is mounted to fuselage affects the structural layout of the tail surfaces and that of the fuselage.

For most aircraft designs, the conventional single-fin tail usually provides adequate stability and control at the lightest weight, but there are other reasons to opt for a different layout.

The result was explained in the following :

The result for Tail Layout is low_tail
Because :
Based on your input data of aircraft_size. I defined
the aircraft as commercial_transport
The decision for engine_type is turbofan
The decision for wing_position is low

The next step was Landing Gear Type, for which the decision was a retractable landing gear. This decision was explained as follows :

The decision for Landing Gear Type is retractable_gear
Because
Based on my calculation from the cruising_speed,
I concluded that the retractable is ok
Based on your input data of aircraft_size, I defined
the aircraft as commercial_transport

The next step to deal with was then the Landing Gear layout. Again the execution menu appears when the **'Execute Step'** was pressed. The result was given in Fig. F.10, and explained as follows :

The result for Landing Gear Layout is tricycle
Because :
The decision for landing_gear_type is retractable_gear
Based on your input data of aircraft_type, I defined
the aircraft as passenger
Based on your input data of cruise_speed, I defined
the aircraft as high_subsonic
Based on your input data of number_of_passenger, I defined
the aircraft as intercontinental_aircraft

Based on your input data of aircraft_size, I defined
the aircraft as commercial_transport

The explanation for the tricycle layout was given when the **'Show Result'** was selected. This is shown in Fig. G.11.

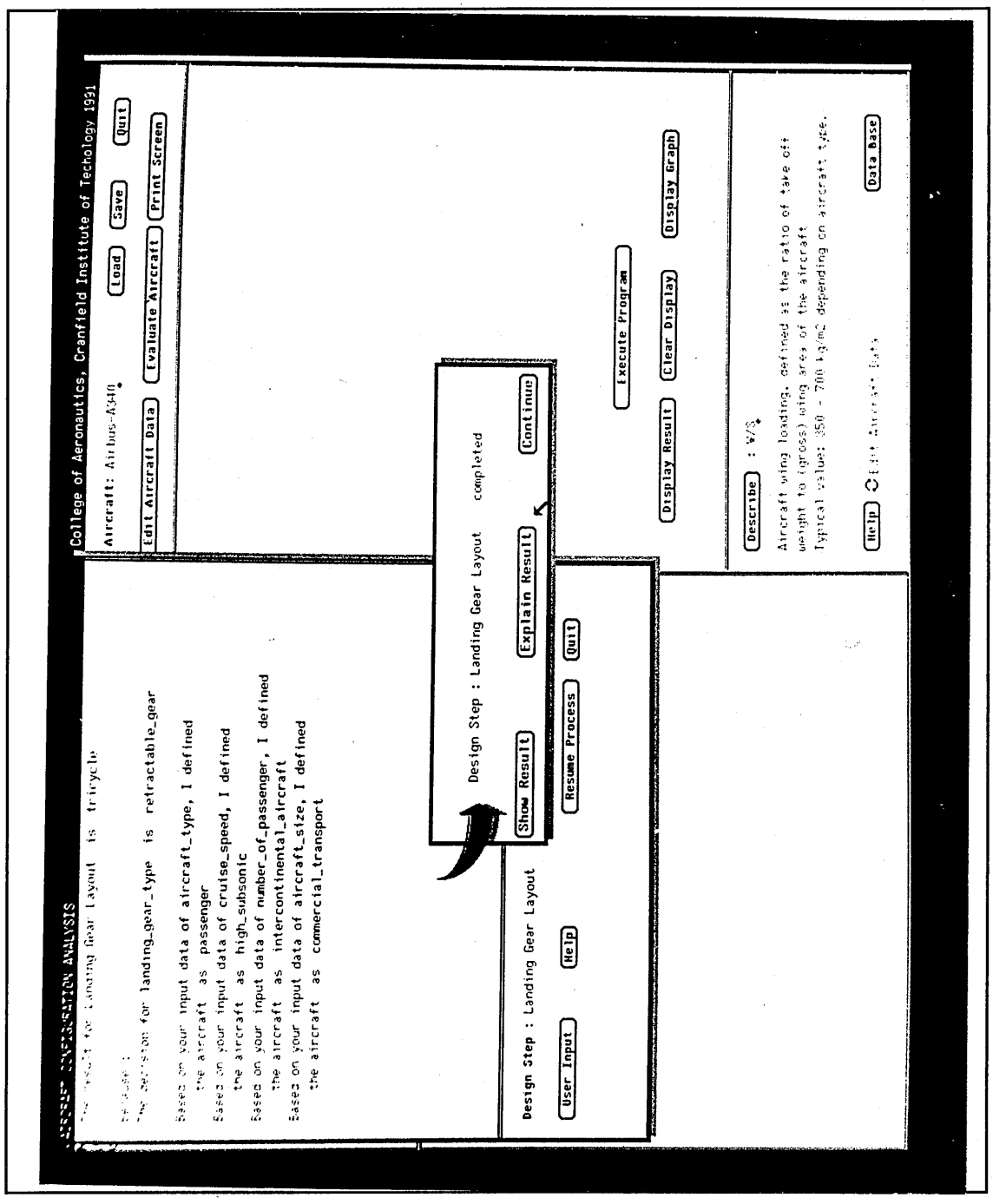


Fig G.10 The explanation for the 'Landing Gear Layout' decision

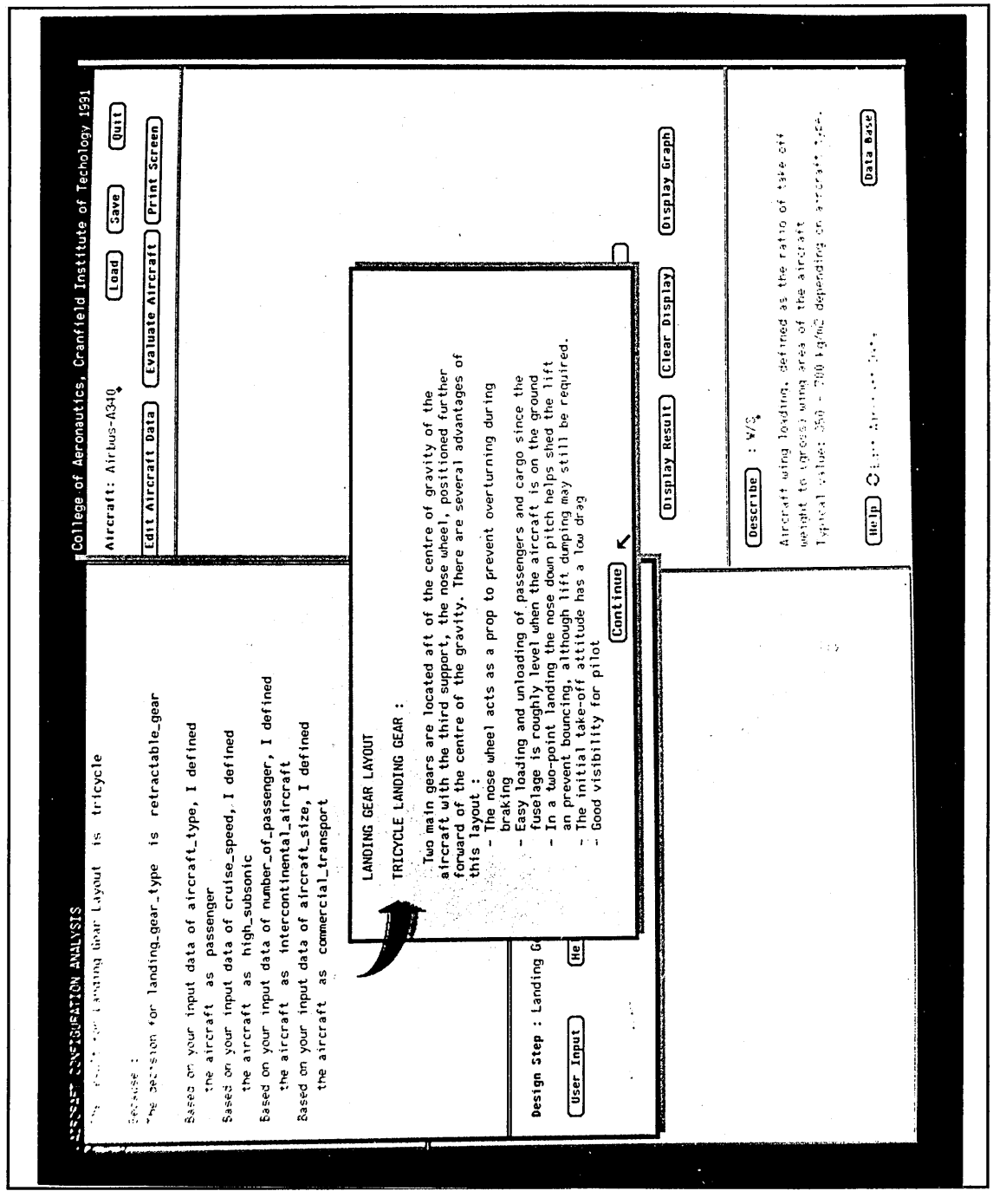


Fig G.11 The explanation for result of the 'Landing Gear Layout' decision

When the execution commenced, the system suggested the next step, which was the Landing Gear Position. Again the analysis on this step can be carried out by selecting the '**Continue**' option in the execution menu. When the analysis was completed, the result was explained as follows :

The result for Landing Gear Position is wing_fuselage
Because :
Based on your input data of aircraft_size, I defined
the aircraft as commercial_transport
The decision for engine_type is turbofan
The decision for engine_position is wing
The decision for wing_position is low
Based on my calculation from the take_off_weight,
I concluded that the landing_gear_size is triple_bogey
The decision for landing_gear_type is retractable_gear
The decision for landing_gear_layout is tricycle

When the activity in the 'Landing Gear Position' step was completed, the system displayed the next step to tackle, which was the Flap Type. The execution began when the '**Continue**' button was selected from the execution menu. When the analysis finished, the control menu appeared on the screen, through which the explanation of the result could be obtained by selecting the '**Explain Result**' option, as follows :

The result for Flap Type is fowler_flap
Because :
Based on your input data of number_of_passenger, I defined
the aircraft as intercontinental_aircraft
Based on your input data of cruise_speed, I defined
the aircraft as high_subsonic
Based on your input data of aircraft_size, I defined
the aircraft as commercial_transport
The decision for wing_position is low

When the activity in this step had been completed, the control returned to the system. When the '**Execute Step**' was selected, the following message :

The Aircraft Configuration Analysis has been completed
All the steps on the Queue has been evaluated

Continue

At this stage, the arrangement of the major component had been defined, and it was thought to be adequate to continue the analysis in other modules. When leaving from the Aircraft Configuration window, the system prompts the message :

Do you want to quit ?
Confirm Cancel

APPENDIX G