A COMPUTER SOLUTION TO PARACHUTE DESIGN PROBLEMS

BY

P.J.BROADBENT

The thesis submitted to the University of Leicester for the degree of Doctor of Philosophy

July 1986

A Computer Solution to Parachute Design Problems

by P.J.Broadbent

Abstract

In this thesis a Pascal computer program is presented which calculates a proposed design of parachute from some simple input parameters, of the type specified by a customer to a parachute company. The program reduces by a significant degree time spent by parachute engineers in the preliminary design stages.

Parachute design is a process which (in common with much engineering design) can be regarded as consisting of a number of separate calculations. The most suitable method (or methods) for each calculation were selected after a thorough investigation of parachute design techniques. The chosen methods must be sufficiently accurate and readily conform to a computer treatment. The data required by the program have been collected from various sources and are stored in a number of files on a floppy disk.

The program is applied to requirements received by a parachute company and results obtained compared with the actual parachutes designed. The program is highly interactive with the user who is able to dispute its selection of values for various parameters. Because the designer can make a rapid and objective choice between a number of methods for various calculations, the existence of this program contributes to his knowledge of the relevance of the parameters involved in, and his understanding of, parachute design. Examples of these techniques are given in the text.

Possibilities for expanding and improving the program exist in a number of areas. In some cases the data required for a particular parachute or particular design methods are not available or do not exist. Provision has been made for such data to be included in the program when they are received.

Acknowledgements

Without the help of a large number of people, the completion of this project, in its present form, would have been impossible.

Firstly I would like to thank my supervisor David Cockrell for his valuable criticism and encouragement throughout the course of this research.

I am grateful to my associate supervisor Steve Lingard, Technical Director, GQ Defence Equipment Limited, Woking, for the financial contribution whilst at the University of Leicester and for his advice during the last three years. Dr Lingard's design and development team have also been very helpful, thanks to: Dave Stephens (now at Marconi), Hu Pickerin, Tim Chadderton, David Gibney and Nigel Parker.

I have received assistance in the choice and operation of the computing methods used in the research from various people. Derek Andrews, head of the Computer Studies Department, Leicester University, has spared time to assist me in these areas of the work. I am also grateful to Martin Lee and Paul Manning of the Computer Studies Department and Tony Baxter of the Engineering Department, Leicester University for helping me in this field.

I would like to thank Dick Dennis of the Parachute Section, Flight Systems Department Q153, Royal Aircraft Establishment, Farnborough, who has provided me with some parachute data and his advice on various topics.

I am grateful to my colleagues Maher Higazy and Sisira Polpitiye for their helpful criticisms and suggestions.

The work presented in this thesis was initiated at the University of Leicester and I am grateful to Professor G.D.S.MacLellan the Head of the Engineering Department for allowing me to use facilities in the department for my research.

Contents

	Page
Acknowledgements	i
Contents	iii
List of Tables	viii
List of Figures	x
Nomenclature	xii
Chapter 1 Introduction	1
1.1 Historical Review	1
1.2 Aims of the Project	4
Chapter 2 A Description of the Computing Methods Used in the Research	6
2.1 Choice of Machine	6
2.2 Choice of Computer Language	7
2.3 Choice of Operating System	8
Chapter 3 A Detailed Examination of Parachute Design Techniques	10
3.1 Type of Parachute	13
3.2 Drag Coefficient	16
3.3 Area	18
3.4 Clusters	20
3.5 Rigging Line Length	24
3.6 Number of Rigging Lines	26
3.7 Staging	27
3.8 Opening loads	28
3.8.1 A Review of Opening Load Calculation Methods	29
3.8.2 Semi-Empirical Opening Distance Methods	31
3.8.2.1 The "Mass Ratio" Method	32
3.8.2.2 The Canopy Loading Method	32
3.8.2.3 The "Pflanz" Method	33
3.8.3 A comparison of Four Different Load Calculation Methods	34

3.8.4 Determination of the Forces on the Store, and the Trajectory, of a Two Stage Parachute System, Using Lingard's Semi-Empirical Opening Load Calculation Method 37

3.8.5 Cluster Opening Forces 42

3.9 Reefing 43

3.10 Structural Analysis 49

3.10.1 Design Factor 49

3.10.2 Solid Cloth Circular Parachutes 50

3.10.3 Cruciform Parachute 54

3.10.4 Slotted Canopies 56

3.10.5 Parawing Parachute 58

3.10.6 Reefing Line Stressing 59

3.11 Parachute Materials 61

3.11.1 Fabric 61

3.11.1.1 Fabric Strength 62

3.11.2 Tapes and Webs 63

3.11.3 Cord 63

3.11.4 Reserve Factors 64

3.12 Landing Control 64

3.13 Weight and Volume 64

3.13.1 Parachute Weight 65

3.13.2 Parachute Packing Density and Parachute Volume 66

3.13.3 Comparison With Approximate Methods 67

3.13.4 Geometric Porosity 67

3.14 Cost 69

3.15 Stability 70

3.16 Parachute Reliability 72

3.16.1 General Definitions 72

3.16.2 Preliminary Considerations 73

3.16.3 The Overall System Reliability Method 73

3.16.4 The Component Reliability Method 74

3.16.4.1 Parallel Components and Clusters 77

Chapter 4 The Parachute Design Program "paradesign" 79

4.1 Structure Diagram 79

4.2 Writing, Organising and Testing of the Program 80

4.2.1 Main Memory Management 80
4.2.2 Data Table Structure 83
4.2.3 Testing and Checking 84
4.3 Input and Output 84
4.3.1 Input Data 84
4.3.2 Output 85
4.4 Alteration and Development of the Program 85
4.4.1 Equations and Tables 85
4.4.2 General Development 86

Chapter 5 Results and Discussion 87
5.1 A Parachute Canopy for the X-RAE2 Remotely Piloted Vehicle 88
5.2 A Parachute System for the Sparrowhawk and Snipe Mk.3 Remotely Piloted Vehicles 95
5.3 A Mail Dropping Canopy for the Royal Netherlands Navy 102
5.4 A Parachute Canopy for the Plessey Marine SSQ 954 Sonobuoy 109
5.5 A Parachute System Incorporating A Cluster of Three Canopies 115
5.6 Reefed Airborne Forces Parachute 121
5.7 Staging of an Airborne Forces Parachute 123
5.8 Ribbon Parachute 126
5.9 6.2m Flying Diameter Aeroconical Gliding Parachute 129
5.10 Discussion of Results 133

Chapter 6 Recommendations for Further Work 137

Chapter 7 Conclusions 138

References 139

Appendix A Types of Parachute 1A
A.1 Type 1 - Flat Circular Parachute 1A
A.1.1 Canopy Weight Calculations 2A
A.1.2 Inputs Required for the Computer Program 4A

A.2 Type 2 - Conical Parachute 5A
A.2.1 Parachute Weight 5A
A.2.2 Inputs Required for the Computer Program 5A
A.3 Type 3 - Bi-Conical Parachute 6A
A.3.1 Canopy Weight Calculation 6A
A.3.2 Inputs Required for the Computer Program 9A
A.4 Type 4 - Tri-Conical Parachute 10A
A.4.1 Canopy Weight Calculation 10A
A.4.2 Inputs Required for the Computer Program 13A
A.5 Type 5 - Extended Skirt 10% Flat Parachute 14A
A.5.1 Canopy Weight Calculation 14A
A.5.2 Inputs Required for the "paradesign" Computer Program 16A
A.6 Type 6 - Extended Skirt 14.3% Full Parachute 16A
A.6.1 Canopy Weight Calculation 16A
A.6.2 Inputs Required for the Computer Program 17A
A.7 Type 7 - Hemispherical Parachute 18A
A.7.1 Canopy Weight Calculation 18A
A.7.2 Inputs Required for the "paradesign" Computer Program 20A
A.8 Type 11 - Cruciform Parachute 21A
A.8.1 Canopy Weight Calculation 21A
A.8.2 Inputs Required for the Parachute Design Program 22A
A.9 Type 12 - Flat Ribbon Parachute 23A
A.9.1 Canopy Weight Calculation 23A
A.9.2 Inputs Required for the Parachute Design Program 25A
A.10 Type 13 - Conical Ribbon Parachute 25A
A.10.1 Canopy Weight Calculation 26A
A.10.2 Inputs Required for the "paradesign" Parachute Design Program 26A
A.11 Type 15 - Hemisflo (Hemispherical Ribbon) Parachute 27A
A.11.1 Canopy Weight Calculation 27A
A.11.2 Inputs Required for the Parachute Design Program 30A
A.12 Type 16a - Flat Ringslot Parachute 30A
A.12.1 Canopy Weight Calculation 31A

A.12.2 Inputs Required for the "paradesign" Computer Program 32A
A.13 Type 16b - Conical Ringslot Canopy 32A
A.13.1 Canopy Weight Calculation 33A
A.13.2 Inputs Required for the "paradesign" Computer Program 33A
A.14 Type 17 - Ringsail Canopy 33A
A.14.1 Parachute Weight Calculation 34A
A.14.2 Inputs Required for the Parachute Design Program 35A
A.15 Type 18 - Disk-Gap-Band Parachute 36A
A.15.1 Canopy Weight Calculation 36A
A.15.2 Inputs Required for the Computer Program 38A
A.16 Type 19 - Rotafoil Parachute 39A
A.16.1 Canopy Weight Calculation 39A
A.16.2 Inputs Required for the Computer Program 41A
A.17 Type 21 - Gliding Aeroconical Parachute 41A
A.17.1 Canopy Weight Calculation 42A
A.17.2 Inputs Required for the "paradesign" Computer Program 42A
A.18 Type 22 - Le Moigne Parachute 43A
A.18.1 Canopy Weight Calculation 43A
A.18.2 Inputs Required for the Parachute Design Program 43A
A.19 Type 23 - Parawing (Single Keel) Parachute 44A
A.19.1 Canopy Weight Calculation 44A
A.19.2 Inputs Required for the Computer Program 46A

Appendix B Brief Description of the Warnier-Orr Structured Program Design Method 1B

Appendix C Structure Diagram 1C

Appendix D Data Tables Used in the "paradesign" Program 1D

Appendix E "paradesign" Program Listing 1E

List of Tables

		Page
Table 2.1	Comparison of Fortran and Pascal	7
Table 3.1	Parachute Types	14
Table 3.2	Drag Coefficient: Non-Gliding Parachutes	17
Table 3.3	Force Coefficient: Gliding Parachutes	17
Table 3.4	Effect of Cluster on Drag Coefficient. Flat Circular Parachutes. Rate of Descent < 7.62 m/s	23
Table 3.5	Effect of Cluster on Drag Coefficient. Flat Circular Parachutes. Rate of Descent > 7.62 m/s	23
Table 3.6	Effect of Cluster on Drag Coefficient. Ringsail Parachutes. Diameter <9.30m	23
Table 3.7	Effect of Cluster on Drag Coefficient. Ringsail Parachutes. Diameter >9.30m	23
Table 3.8	Line Length Data	25
Table 3.9	Values of Fill Constant and Opening Force Coefficient for Various Canopies	33
Table 3.10	A Comparison of Four Opening Load Calculation Methods	36
Table 3.11	Values of the Ratio of Projected to Nominal Diameter	53
Table 3.12	Values of the Ratio of Flying Diameter to Nominal Diameter	58
Table 5.1	Specification for the X-RAE2 RPV	90
Table 5.2	Design Solution for the X-RAE2 RPV Canopy	91
Table 5.3	Specification for the Sparrowhawk and Snipe Mk.3 RPVs	97
Table 5.4	Design Solution for the Sparrowhawk and Snipe Canopy	98
Table 5.5	Design Requirement Data for a Mail Dropping Parachute	104
Table 5.6	Design Solution for a Mail Dropping Canopy	105
Table 5.7	Specification for the Plessey SSQ 954 Canopy	110

Table 5.8	Design Solution for the Plessey SSQ 954 Canopy	111
Table 5.9	System C Specification	116
Table 5.10	Computer Solution for a Cluster of Parachutes	117
Table 5.11	Airborne Forces Parachute Specification	120
Table 5.12	Design Solution for a Reefed Airborne Forces Parachute	121
Table 5.13	Design Solution for Staging of an Airborne Forces Parachute	124
Table 5.14	Ribbon Parachute Design Solution	127
Table 5.15	Design Requirements for an Emergency Escape Parachute	130
Table 5.16	Design Solution for a Gliding Aeroconical Parachute	131
Table D.1	Fabric Material Data Used in the Parachute Design Program	4D
Table D.2	Tape and Web Material Properties	5D
Table D.3	Cord Material Data	9D
Table D.4	Material Reliability Data	11D
Table D.5	Gradients of the Non-Central t-Distribution Line	12D
Table D.6	Constants of the Non-Central t-Distribution Line	13D

List of Figures

	Page
Figure 3.1 Flow Chart to Illustrate the Stages of Parachute Design	11
Figure 3.2 A Flow Chart for Cluster Design	21
Figure 3.3 Diagram of a Store and Canopy System	38
Figure 3.4 Aerodynamic Forces on the Store	40
Figure 3.5 Vent Reefing	44
Figure 3.6 Skirt Reefing	44
Figure 3.7 Reefing Flowchart	46
Figure 3.8 Cruciform Parachute Dimensions	55
Figure 3.9 Reefed Canopy Configuration	60
Figure 3.10 Bias and Block Construction Methods	65
Figure 3.11 Ribbon Parachute Gore Configuration	68
Figure 4.1 Parachute Design Represented in Warnier-Orr Notation	80
Figure 4.2 paradesign Text File Organisation	82
Figure 4.3 A Data Table in Warnier-Orr Notation	83
Figure A.1 Flat Circular Canopy Configuration	1A
Figure A.2 Conical Canopy Configuration	5A
Figure A.3 Bi-Conical Parachute Configuration	6A
Figure A.4 Tri-Conical Parachute Configuration	10A
Figure A.5 Extended Skirt Canopy Configuration	14A
Figure A.6 Full Extended Skirt Canopy Configuration	16A
Figure A.7 Hemispherical Parachute Configuration	18A
Figure A.8 Cruciform Canopy Configuration	21A
Figure A.9 Flat Ribbon Parachute Configuration	23A
Figure A.10 Conical Ribbon Parachute Configuration	25A
Figure A.11 Hemisflo Parachute Configuration	27A
Figure A.12 Flat Ringslot Parachute Configuration	30A
Figure A.13 Conical Ringslot Parachute Configuration	32A
Figure A.14 Ringsail Parachute Configuration	33A
Figure A.15 Disk-Gap-Band Parachute Configuration	36A
Figure A.16 Rotafoil Parachute Configuration	39A
Figure A.17 Aeroconical Parachute Configuration	41A

Figure A.18 Le Moigne Parachute Configuration	43A
Figure A.19 Parawing Parachute Configuration	44A
Figure B.1 Sequence in Warnier-Orr	1B
Figure B.2 Repetition in Warnier-Orr	2B
Figure B.3 Selection in Warnier-Orr Notation	2B
Figure D.1 Data tables Used in the Program	2D

Nomenclature

A	Altitude
a	Cluster line length ($=l_e - l_c$)
aa	Gore width (parachute type 19)
ACU	Total cut-out area of canopy
A_F	Aerodynamic force in x direction
AR	Cruciform parachute aspect ratio
area(x)	Fabric panel areas
$area_T(x)$	Upper fabric panel areas
$area_M(x)$	Middle fabric panel areas
$area_B(x)$	lower fabric panel areas
A_T	Table value (reliability calculation)
A_X	Opening force coefficient
b	Gore width (parachute type 19)
C	Constant
C_D	Drag coefficient (nominal)
C_{DC}/C_D	Drag coefficient correction for the effects of a cluster
C_{DD}	Drogue drag coefficient
C_{Dfinal}	Final corrected drag coefficient of a parachute in a cluster
$C_D H$	High C_D
$C_D L$	Low C_D
C_{D0}	Drag coefficient (nominal)
C_{D0}/C'_{D0}	Drag coefficient correction for effects of the rigging lines
C_{Dr}	Reefed drag coefficient
C_{DP}	Projected drag coefficient
C_{DP}/C_{DP0}	Drag coefficient ratio
C_{DS}	Store drag coefficient
$(C_D\ S)_D$	Drogue drag area
$(C_D\ S)_r$	Reefed drag area
C_F	Force coefficient
CF	Fabric cost
C_{FR}	Force coefficient (gliding parachutes)
C_L	Lift coefficient

C_N	Normal force coefficient
C_M	Moment coefficient
COF	Fabric material cost
C_R	Force coefficient
C_s	Constant
C_k	Opening force factor
C_x	Opening force coefficient
C_{xdr}	Opening force coefficient at disreef
C_{xL}	Cluster opening force coefficient
C_{xr}	Reefed opening force coefficient
C_{x1}	Opening load factor
C_Y	Ratio C_{xL}/C_x
D	Canopy nominal diameter
d	Canopy diameter
d_b	Store base diameter
D_c	Canopy constructed diameter
D_{cm}	Canopy maximum diameter
D_F	Design factor
D_{max}	Maximum drag
D_0	Canopy nominal diameter
D_p	Canopy projected diameter (in flight)
dpack	Packing density
D_{pr}	Estimated canopy inflated diameter
D_r	Reefed diameter
D_r	Drag
D_v	Vent diameter
e_s	Maximum gore width
e(x)	Gore widths
$e_B(x)$	Gore widths
$e_M(x)$	Gore widths
e_{max}	Maximum gore width
$e_T(x)$	Gore widths
e_v	Gore width at the vent
e_1	Gore width
e_2	Gore width
F	Opening load
f	Degrees of freedom

f'	Reefing line load
F_A	Maximum allowable load
F_D	Design load
F_{dr}	Maximum force at disreef
F_l	Load in the lines
Fr	Froude number
F_r	Reefed opening force
F_s	Steady state force
F_{sc}	Cluster opening force
fx	Force coefficient
Fx	Force
f_{xx}	Degrees of freedom of material reliability data
f_y	Degrees of freedom of the trial loads
f_1	Degrees of freedom
f_2	Degrees of freedom
g	Acceleration due to gravity
G_A	Maximum load factor
gr	Ratio f_1/f_2 (reliability variable)
GW	Gap width
gxr	Confidence coefficient
h_c	Inflated part of the reefed canopy
h_g	Gore height, from skirt to vent
h'_g	Gore height
HRS	Horizontal ribbon strength
HRW	Horizontal ribbon width
h_s	Gore height
h'_s	Gore height
h_x	Non-inflated part of the reefed canopy
h_1	Gore height
h_2	Gore height
h'_2	Gore height
h_3	Gore height
h_4	Gore height
h_{1g}	Gore height
I_{xx}	Store Moment of inertia in x axis
I_{yy}	Store Moment of inertia in y axis
I_{zz}	Store Moment of inertia in z axis

k	Gore height ratio (h_1/h_2)
k_f	Store mass ratio
k_p	Canopy mass ratio
Kr	Reliability variable
K_s	Constant
k_x	Fill constant
K_1	Constant
k_1	Gore height ratio (h_1/h_2)
K_2	Constant
k_2	Gore height ratio (h_3/h_2)
L	Lift force
l_c	Cluster line length
l_e	Rigging line length
l_c/D	Ratio of cluster line length to nominal diameter
l_e/D	Ratio of rigging line length to diameter
l_r	Reefing line length
m	Store mass
M	Aerodynamic moment
MA	Canopy mouth area
M_{dampF}	Aerodynamic damping moment
m_f	Store mass
m_F	Store mass
m_p	Canopy mass
mr	Maximum number of canopies that can fail without causing the mission to fail
N	Number of gores
n	Number of lines
N_F	Aerodynamic force in z direction
NHR	Number of horizontal ribbons
NLP	Number of lower fabric panels per gore
NMP	Number of middle fabric panels per gore
NP	Number of fabric panels per gore
Nr	Reliability variable
NR	Number of rings
N_T	Number of trials
NUP	Number of upper fabric panels per gore
NVT	Number of vertical tapes

N_x	Number of tests (materials)
N_y	Number of tests (parachute trials)
N_1	Number of tests
N_2	Number of tests
ONP(x)	Number of each fabric panel missing in canopy types 21 and 22
p	Differential pressure
P_d	Probability of failure of a cluster
p_F	Angular velocity around x axis
p_{max}	Maximum differential pressure
P_r	Probability of failure of q canopies in a cluster
pr	Probability of failure of a single canopy
q	Number of canopies in cluster (reliability)
q_{dr}	Dynamic pressure at disreef
q_e	Equilibrium dynamic pressure
q_F	Angular velocity around y axis
$\bar{q}F$	Dynamic pressure
q_M	Maximum dynamic pressure
q_o	Deployment dynamic pressure
r	Radius of curvature
R	System reliability
Rb	Total reliability
R_c	Component reliability
R_{cx}	Component reliability
r_F	Angular velocity around z axis
R_M	Mass ratio
r_{max}	Maximum radius of curvature
R_{Mdr}	Mass ratio at disreef
R_{ML}	Cluster mass ratio ($=R_M/\eta_C$)
R_{Mr}	Reefed mass ratio
R_p	Fully inflated canopy radius
R_{pr}	Operational reliability
rr	Ratio S_1^2/S_2^2
R_{sys}	System reliability
RW	Ring width
S	Standard deviation
S_b	Store base area

SB	Strengthening band strength
SBS	Skirt band strength
S_D	Drogue canopy area
$S_\bullet$	The area of each parachute in a cluster
S_{ee}	Estimated area of a canopy in a cluster
S_F	Safety factor
S_f	Filling distance
S_f	Fabric strength
S_{final}	Final (corrected for the effects of the store) area
SG	Total gore area
S_l	Line strength
S_0	Canopy constructed, or nominal, area
S_{0D}	Drogue area
S_p	Canopy projected area
S_{pr}	Inflated reefed area
S_r	Reefed area
S_s	Store cross sectional area
STR	Total exposed material area
S_x	Standard deviation of material properties
S_y	Standard deviation of test results
S_1	Standard deviation
S_2	Standard deviation
t	Time
T	Tension
TAPEHR	Overlap of radial tape on horizontal ribbon
t_{cru}	Cruciform canopy arm width
t_f	Inflation time
T_F	Critical fabric load
THR	Total horizontal ribbon length
t_{inf}	Inflation time
T_L	Line strength
TLP	Total lower panel area
TOF	Total fabric area
TOT	Total radial tape length
TOL	Total rigging line length
TOVL	Total vent band length
T_r	Maximum tension in the canopy

TR	Total reinforcing length
T_{RL}	Reefing line strength
TSB	Total skirt band length
T_T	Radial tape strength
TUP	Total upper panel area
TVB	Total vent band length
TVT	Total vertical tape length
TW	Radial tape width
U	Air velocity through the canopy fabric
u_F	Velocity in x direction
U_F	Ultimate safety factor
U/φ	Reliability Variable
V	Rate of descent
VBS	Vent band strength
V_D	Deployment velocity
v_F	Velocity in y direction
VFR	Volumetric flow rate
V_H	Horizontal velocity
VLS	Vent line strength
V_m	Velocity at inflation to maximum diameter
V_o	Deployment velocity
VOL	Volume
V_s	Snatch velocity (velocity at the start of inflation)
V_T	Total velocity
VTAPEHR	Overlap of vertical tape on horizontal ribbon
VTS	Vertical tape strength
VTW	Vertical tape width
V_x	Speed
v_x	Vent area as a percentage of S_0
V_1	Velocity
W	Store weight
w_F	Velocity in z direction
WF	Total fabric weight
WHR	Total horizontal ribbon weight
WL	Total rigging line weight
WR	Total reinforcing weight

WRL	Reefing line weight
W_S	Store weight
WSB	Total skirt band weight
WT	Total weight of the radial tapes
WTF	Fabric material weight
WTHR	Horizontal ribbon material weight
WTL	Rigging line material weight
WTR	Reinforcing band material weight
WTOT	Total canopy weight
$WTOT_C$	Cluster weight
WTRL	Reefing line material weight
WTSB	Skirt band material weight
WTT	Vertical tape material weight
WTVB	Vent band material weight
WTVL	Vent line material weight
WTVT	Vertical tape material weight
WVB	Total weight of the vent band
WVL	Total vent line weight
WVT	Total vertical tape weight
x	Counter
$\bar{x}$	Mean of material breaking strengths
x_i	Material breaking strength in ith test
Xr	Reliability variable
X_1	Opening force reduction factor
Z	Number of lines
zr	Reliability variable
zz	Gore height (type 15)
α	Line angle to vertical
α_F	Angle of attack
β	Gore angle
β_1	Gore angle
β_2	Gore angle
β_3	Gore angle
γ	Angle of attack (at tail of forebody)
Δx	Distance from store centre of gravity to centre of pressure

ΔP Pressure drop across the canopy fabric

ζ Reefing ratio

η_c Number of parachutes in a cluster

η_2 Fill constant

θ Pitch angle

θ_T Trajectory angle

θ_p Construction angle (parachute type 23)

θr Ratio N_2/N_1 (reliability)

θ_x Construction angle (parachute types 5 and 6)

θ_{x1} Construction angle (parachute type 15)

θ_{x2} Construction angle (parachute type 15)

λ_g Geometric porosity

μ Canopy constructed cone angle

μ_1 Canopy constructed cone angle

μ_2 Canopy constructed cone angle

μ_3 Canopy constructed cone angle

ρ Air density

ΣF_{Fx} Force in x direction

ΣF_{Fy} Force in y direction

ΣF_{Fz} Force in z direction

ΣM_{Fx} Moment about x

ΣM_{Fy} Moment about y

ΣM_{Fz} Moment about z

τ Dimensionless time

τ_o Dimensionless time of the peak of the lines taut snatch force

τ_r Reefing line ratio

φ Roll angle

φ_r Canopy lines conversion angle (reefed)

ψ Yaw angle

ψ_r Radial member conversion angle (reefed)

Superscript

. Derivative with respect to time

Chapter 1

Introduction

1.1 Historical Review

The first design of a parachute appears in the sketchbook of Leonardo da Vinci in 1514. As far as is known this device was never manufactured and put into practice. Towards the end of the eighteenth century the first parachute jumps were made from balloons. During the nineteenth century exhibition jumps from high buildings and (especially) balloons became very popular. Jumping from aircraft by stunt men was more dangerous because the man was moving on exit from the plane, and opening of the parachute had to be delayed until it was clear of the aircraft. During the 1914-18 War pilot´s lives were saved by the use of parachutes, and soon after this war it became compulsory for airmen to carry parachutes. The first technical analysis of parachutes was done in Germany and between the wars much research, mainly on opening behaviour, was performed.

In 1942 the British Parachute Section was established at the Royal Aircraft Establishment, Farnborough, under the leadership of W.D.Brown. In 1946 Johns (a member of this team) published "Parachute Design"[1]. In this paper the important design considerations are given as: opening, drag, strength and stability. These four would generally be regarded as the most important characteristics in the present day. Johns also describes the four types of parachute in use at that time: the gathered parasheet, ungathered parasheet, flat parachute and shaped parachute. The first step in the design (after choice of type of parachute presumably) is given by Johns as the choice of fabric porosity. After that the design procedure is essentially that presented in the modern parachute design guides.

In 1951 "Parachutes" by Brown[2] was published. This is the first book written on the subject and describes most aspects of parachutes including: design, manufacture, aerodynamics and testing. In the introduction to the design chapter Brown states "...we are forced, for the time being, to extrapolate from empirical relationships which are not very reliable and are certainly limited to speed ranges and dimensions completely outside present day requirements". For many of the calculations in parachute design this problem is still present. The design section splits parachutes up into those which open near their release speed, and those which open after a substantial reduction in speed. In both these cases the procedure is similar to that given by Johns, although Brown´s methods require a greater amount of empirical data.

In 1951 the first parachute design handbook was published, its second revision appeared in 1963[3]. This book contains an excellent design chapter and some useful worked examples. It contains data for most commonly used parachutes and is extensively used at present.

By 1960 many different types of parachute were in use as well as other drag-generating systems such as rotor blades and inflatables. Parachutes became regarded as one type of "aerodynamic decelerator". Ibrahim[4] defines parachutes as "...flexible, elastic bodies; their inflated shape depends on the flow conditions and vice versa". An engineering review of aerodynamic decelerators was published by Pepper and Maydew[5] in 1971. This paper contains design information for slotted (ribbon,ringslot,ringsail) parachutes, and 215 references. A similar, more recent and restricted, review was published by Dennis[6] in 1983.

By this time numerous tests were being performed on various types of parachutes (mainly in the United States). In order to keep track of the results the Parachute Design and Performance Data Bank[7] was set up (1970-1973). Test results from 105 documents are held in this data bank and the data are available to parachute design engineers on request. Because databases only became available in 1975-76, the software documented in reference 7 is crude by modern day standards, however any attempt to cut down on expensive parachute testing is useful. A similar system using a database is now available[8].

In 1978 a further revision of the parachute design handbook was published[9]. This was essentially the same as the previous version but includes information on new types of parachutes and new design techniques (28 different types in general use are discussed). The design chapter contains a number of useful examples. Many of the techniques and data from this publication have been used in the present analysis.

The Kevlar design guide[10], published in 1982, contains much useful information about Kevlar 29 and ribbon parachute design. Some of the formulae in this report can be applied to any type of parachute.

Lecture notes by Knacke[11] are another important source of parachute design data and formulae. These notes will form the basis of a forthcoming "Recovery System Design Manual" for the Naval Weapons Center, China Lake, California. These notes can be regarded as an "up-dated" version of the Recovery Systems Design Guide (reference 9), the section on ribbon parachutes is especially good.

Due to obvious difficulties, the assessment of parachute design work in Russia and Japan is impossible. In these countries, from a limited number of translated papers, it can be seen that useful work has been done in this field.

A large number of different design methods are contained in the pages of these publications. Most are based on experimental data, and may require some inputs from tests of a similar or scaled down parachute to the one being designed. In some cases the data used in the design has come from obscure experiments performed many years ago and hence may not be very reliable. Also the basis for some of the curves of data presented in these design guides is not known. Whether values gleaned from interpolation and certainly from extrapolation from these curves and other parachute design data are reliable is a debatable point. A review of these techniques to select those most suitable for use by parachute design engineers, and to present a standardised design technique, is required. By clearly demonstrating where data are unreliable the present design analysis will aid this selection process.

There has been little use of computer methods in parachute design. The notable exceptions are in the fields of:

(i) Stressing - an interpolation method, based on a Fortran computer program is becoming increasingly used.

(ii) Inflation - Many computer methods have been used for the calculation of parachute inflation and trajectory characteristics. A computer program to calculate the inflation history of a parachute is used at the premises of the co-operating body (G.Q. Defence Equipment Ltd., Woking, Surrey). A version of this program is incorporated into the parachute design program presented in this thesis.

(iii) Design - The author is aware of a parachute design program used by Irvin Great Britain Ltd. at Letchworth.

1.2 Aims of the Project

(i) To devise a system, consisting of a number of separate programs, for the design of parachutes. This system requires some simple input data, and performs a number of separate calculations to give a theoretical design of parachute.

(ii) To make a critical assessment of the computing techniques available in order to select the most suitable computer language, operating system and machine for the program.

(iii) To collect the data required for the programs described in aim (i) above from literature and known design procedures. These data can then be stored in a computer in such a form that they can readily be added to and updated.

(iv) To perform a critical examination of the methods available to parachute designers. Thus the methods most suitable for inclusion in the design program are selected. Additionally a standard design technique for parachutes making the best use of the methods available, as well as consistency between designs and design engineers, is ensured.

Chapter 2

A Description of the Computing Methods Used in the Research

The software to be produced must be operational at the premises of the co-operating body (G.Q. Defence Equipment Ltd., Woking, Surrey), and must conform to this company's requirements.

2.1 Choice of Machine

A number of different computer systems are available both at Leicester University and the co-operating body. To a limited extent computer programs written on one machine can be transferred to and used on a different system.

At Leicester University the mainframe computer consists of two DEC VAX 8600's, each with 20 megabytes of memory. The Computer Studies Department has a number of RML Nimbus and ACT Sirius micro-computers. Elsewhere in the university various mini-computers and micro-computers are available. At G.Q. Defence Equipment Ltd. Hewlett-Packard 9836 and ACT Sirius micro-computers are available.

One possibility was to write the software on the VAX computer at Leicester and transfer program files to Woking via British Telecom. However the equipment for doing this is not available so the software had to be written on a machine that is available at both Leicester and Woking. Therefore the Sirius micro-computer was chosen. An added problem that arises if the software is written on the VAX is that its storage is much larger than that of a micro-computer, and programs on the VAX would have to be checked on a microcomputer at Leicester before transferring them to Woking.

The Sirius I, manufactured by the American company Applied Computer Technology was introduced in 1981. At one time it was the 16-bit market leader in Europe. In America it is known as the Victor 9000. The Sirius's used at Leicester have 384k bytes of memory, the one at Woking has 512k bytes of memory.

2.2 Choice of Computer Language

The choice of the computer language to be used in the project was between Fortran,Basic and Pascal. Basic is not compiled and is therefore slower than the other two languages in this survey. It is also not really suitable for large programs and hence Basic was discounted. Fortran and Pascal are compared in table 2.1.

	advantages	disadvantages
Fortran	scientific	not structured old fashioned(1954)
Pascal	easy to use structured modern(1971)	not scientific

Table 2.1 Comparison of Fortran and Pascal.

Structured programming is a systematic approach to good program design. It is used to write large and complex programs in a manner that avoids the errors that plague programming in older languages such as Basic and Fortran. To write programs using structured programming methods a language like Pascal is required. Pascal contains a large number of flow-of-control statements (if...then...else, and while...do for example). It also supports various data structures not used in Fortran and Basic (records, pointers etc.). To write a large computer program (typically one over 2000 lines of code) some forward planning using structure diagrams is essential and transferring from these diagrams to Pascal code is relatively simple. Pascal is becoming increasingly popular for use with micro-computers such as the Sirius. After consultation with staff at the Computer Studies Department, Leicester University, and the co-operating body (G.Q. Defence Equipment Ltd.) it was chosen as the language to be used. The main disadvantage of using Pascal for this project is that it is not scientific, and programming mathematical formulae is rather long winded (for example the facility to raise values to a power is not available).

2.3 Choice of Operating System

The three commonly used micro-computer operating systems are: CP/M, MS-DOS, and UCSD. There are a number of different implementations of Pascal available with CP/M and MS-DOS: UCSD can only be used with UCSD Pascal. UCSD allows separately compiled portions of code to be incorporated into the main program (UCSD units[12]), so this operating system was chosen to be used for the parachute design program. This facility is not available in standard Pascal.

Sets of pre-programmed routines can be grouped together in a separate 'unit' in such a way that any of the routines (procedures and functions) can be used as if they had been declared within the 'using' Pascal program (i.e. the program that uses the unit). Several units may be grouped together into a disk file called a 'library'. A unit consists of two parts: interface and implementation. The interface part is 'public', i.e. it is available to the 'using' program. The implementation part is 'private' to the unit, not available directly to the 'using' program. Because units are pre-compiled their use saves time whilst writing and checking a program; when an alteration is made to the program only the unit in which the change has been made needs to be re-compiled. As well as providing the facility to store parts of the program that may not be used in every run in separate units, units can be used to split up the program as it becomes too large for the compiler. Also, using a unit, programs already written in UCSD Pascal can be incorporated into the main parachute design program.

Other advantages of UCSD Pascal over standard Pascal are listed in appendix D4 of reference 13. The disadvantage of UCSD is that the Pascal code is not translated into machine code but into a sort of ´intermediate´ language called p-code. This is much slower to run than machine code.

Chapter 3

A Detailed Examination of Parachute Design Techniques

Figure 3.1 has been constructed in order to set out, in a chronological form, the processes involved in parachute design. As well as the requirements listed in this figure some construction details are required, to enable the weight and volume of the parachute to be calculated. These details are different for each type of parachute.

In this chapter each stage in figure 3.1 is examined and the best method (or methods) for performing the parachute design calculations chosen from those available. The criteria used for choosing these methods are:

(i) accuracy.
(ii) reliability.
(iii) suitability for computer treatment.

Sections 3.1, 3.2, and 3.3 of this chapter outline the initial selections a parachute designer is required to make: choice of type of parachute and drag coefficient, as well as the calculation of the parachute area. Clusters of canopies are discussed in section 3.4. The length and numbers of the rigging lines are usually determined using methods given in sections 3.5 and 3.6. Staging is discussed in section 3.7 and opening loads in 3.8. Parachute reefing is the subject of section 3.9 and stressing calculations are given in section 3.10. The choice of parachute materials is outlined in section 3.11. Sections 3.12 and 3.13 contain discussions of landing control and canopy weight and volume. Finally cost, stability and reliability are outlined in sections 3.14, 3.15 and 3.16 respectively.

Some of the equations and tables in this section and in appendices A and C have been given an extra identifier. This identifier refers directly to the parachute design program

REQUIREMENTS:
The following parameters are assumed to be be given:-
Rate of descent.
Horizontal velocity (gliding parachutes).
Size and weight of store.
Velocity, altitude and attitude at deployment.
Allowable volume and weight of parachute.
Allowable loading.
Permitted range of oscillations.

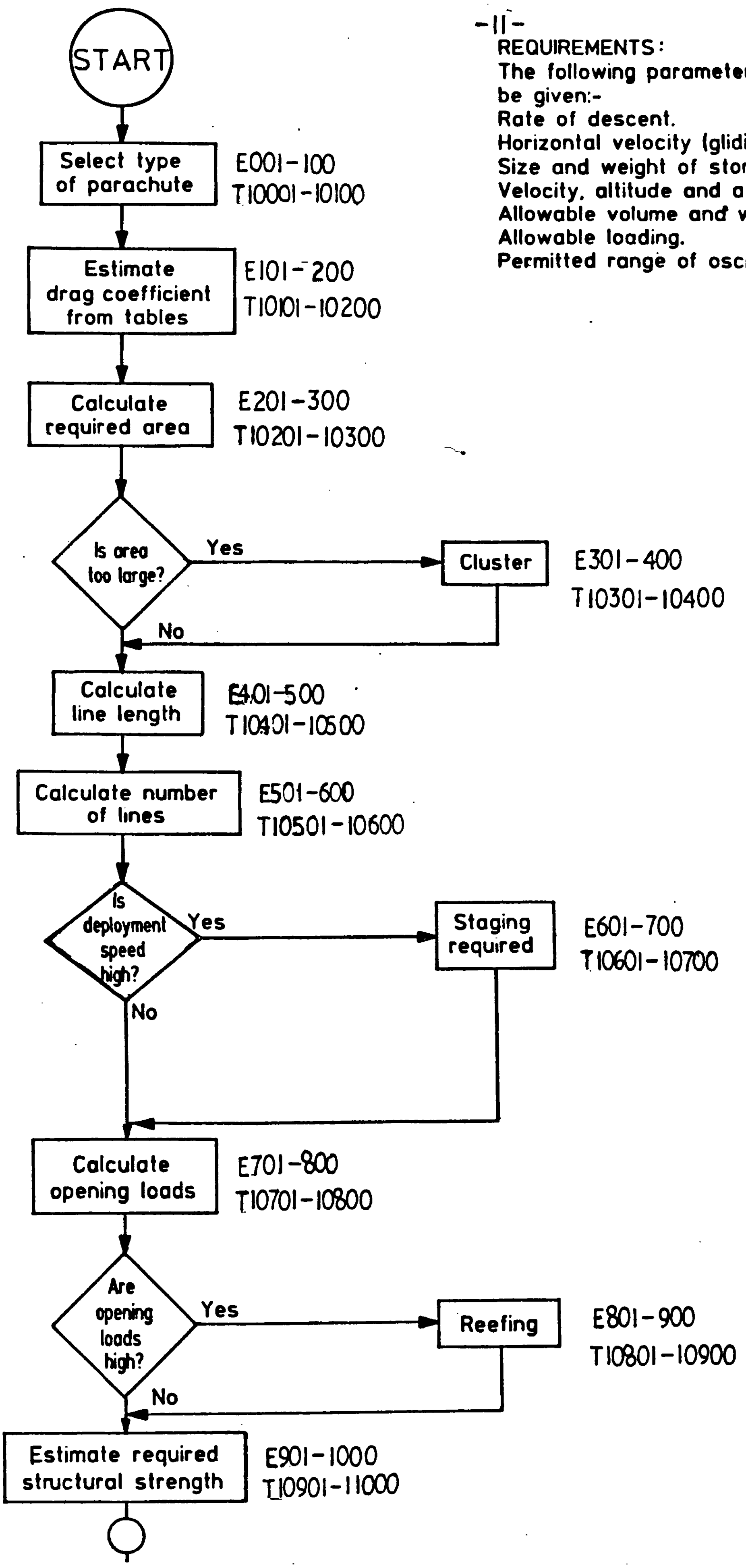


Figure 3.1 Flow chart to illustrate the stages of parachute design

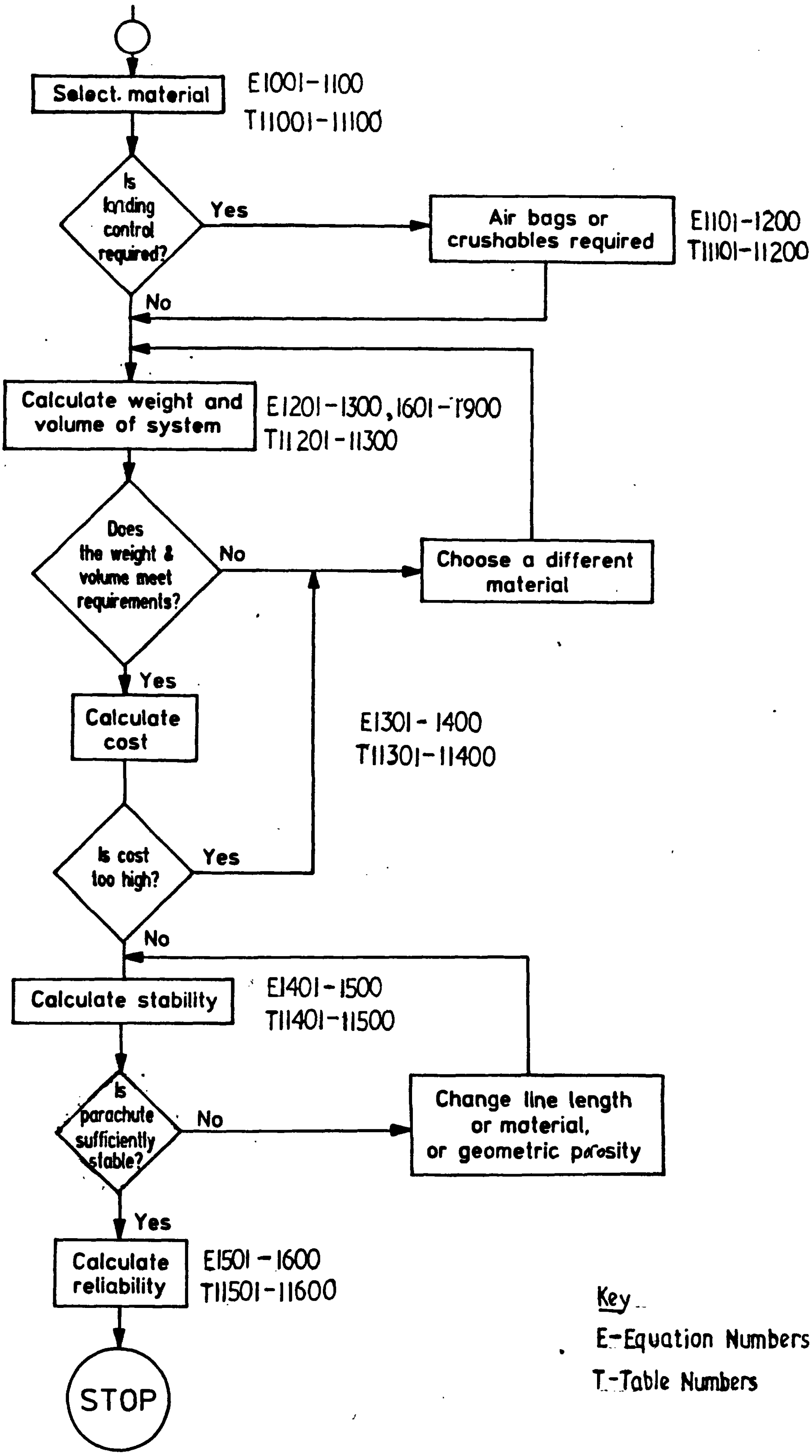


Figure 3.1 (continued)

code (listed in appendix E), and to the program structure diagram (listed in appendix C). The identifier is a three or four figure number for an equation and a five figure number for a table. The number is dependent on the calculation in which the table or equation is first used, the numbering convention being given in figure 3.1.

3.1 Type of Parachute

In table 3.1 twenty-eight types of parachute are listed together with their construction details and uses. This information has been taken from references 9 and 11.

The two main uses of parachutes are descent and deceleration. Descent can be regarded as the delivery of a store; deceleration applications are usually at high velocities, e.g. the deceleration of aircraft. Generally solid cloth parachutes (flat circular, conical etc.) are used for descent applications and slotted parachutes (ribbon, ringslot etc.) are used for deceleration applications. However there are special cases such as emergency escape: this is a high velocity application for which a solid cloth parachute is employed.

Parachutes are divided into two sets: gliding and non-gliding. Gliding parachutes are those which impart a horizontal velocity or 'drive' to the parachute and load system. Non-gliding or conventional parachutes possess solely a drag generating role. The design of these two types differs in the initial stages but generally, after this, the same techniques are used.

Stability is an important criterion for the choice of the type of canopy. Some types of parachute are more stable than others. So if the required oscillation amplitude, expressed as a permitted range of oscillations, is less than $\pm$ 5 degrees, a stable parachute is required. Cruciform (or cross)

Type	Construction	Use
1. Flat Circular	Solid textile	Descent (obsolete)
2. Conical	Solid textile	Descent
3. Bi-conical	Solid textile	Descent
4. Tri-conical	Solid textile	Descent
5. Extended skirt flat	Solid textile	Descent
6. Extended skirt full	Solid textile	Descent
7. Hemispherical	Solid textile	Descent (obsolete)
8. Guide surface (ribbed)	Solid textile	Drogue, stabilization
9. Guide surface (ribless)	Solid textile	Drogue
10. Annular	Solid textile	Descent
11. Cross (cruciform)	Solid textile	Descent
12. Flat ribbon	Slotted textile	Descent,deceleration, drogue (obsolete)
13. Conical ribbon	Slotted textile	Descent,deceleration
14. Conical ribbon varied porosity	Slotted textile	Descent,deceleration, drogue
15. Hemisflo (ribbon)	Slotted textile	Drogue,supersonic
16. Ringslot	Slotted textile	Extraction, deceleration
17. Ringsail	Slotted textile	Descent
18. Disk-gap-band	Slotted textile	Descent
19. Rotafoil	Slotted textile	Drogue
20. Vortex ring	Slotted textile	Descent
21. TU slotted	Slotted textile	Descent
22. Le moigne	Slotted textile	Descent
23. Parawing single keel	Solid textile	Descent
24. Parawing twin keel	Solid textile	Descent
25. Parafoil (ram-air)	Solid textile	Descent
26. Sailwing	Solid textile	Descent
27. Volplane	Solid textile	Descent
28. Balloon (ballute)	Solid textile	Drogue,stabilization

Table 3.1 (10001) Parachute Types

parachutes with appropriate arm ratios and ribbon parachutes are known to be highly statically stable, and coupled with the fact that cruciform parachutes are very simple to make, means that this type is being used for an increasing number of applications. Most of the typical parachute systems studied in the results (chapter 5) are cruciform. The expression of stability in terms of a range of oscillations is meaningless in practical terms. It is sensible to design all parachutes to be stable and to check their stability by testing. The stability of a parachute canopy is highly dependent on the atmospheric conditions at the time of use. A high wind will greatly affect the parachute's performance and stability.

After consultation with the co-operating body some changes were made to the list of parachutes, table 3.1, for use in the parachute design program. Types 8, 9 and 10, the guide surface and annular parachutes, are not used by the co-operating body. Type 14 (conical ribbon, varied porosity) and type 20 (vortex ring) are of very complicated construction, so these five types have been removed from the table. An emergency escape parachute manufactured by the co-operating body is the GQ Aeroconical canopy. This canopy is of similar construction to Type 21, the Tojo, or TU slotted canopy, and replaces it in the table. Types 24 and 25, parawing(twin keel) and parafoil(ram-air) are again very complicated and require completely different design techniques, beyond the scope of this project. Types 26 and 27, sailwing and volplane, have no force coefficient data available, so these four types, 24 to 27, are also removed. If the force coefficient information is obtained parachute types 26 and 27 can be included in the program. Type 28, balloon(ballute), is not in general use and does not satisfy the requirements of a parachute as defined by Ibrahim (section 1.1), it is also removed. The ringslot parachute can be either flat (type 16a) or conical (type 16b).

3.2 Drag Coefficient

A parachute's drag coefficient indicates how effectively it produces drag with a minimum of cloth area.

The value of the drag coefficient for a particular parachute is obtained from curves or tables presented in the parachute design guides. Reference 9 contains curves of rate of descent versus drag coefficient for nine types of parachute. Each curve has been compiled using data from a limited number of tests, because these data are sparse a drag coefficient value chosen from a particular curve may be inappropriate for the parachute that is being designed.

A complete set of drag coefficient values are contained in tables 2.1 to 2.5 of reference 9. Table 3.2 has been compiled from these values. This table contains drag coefficient data for non-gliding parachutes and represents the results from rate of descent trials on a number of parachutes. Following the practice of reference 9 two values are given. These may represent an uncertainty band as the accurate measurement of parachute drag coefficient is difficult. The higher value ($C_D H$) is the drag coefficient at a low rate of descent, about 17ft/sec (5.18m/s), the lower one ($C_D L$) is at a high rate of descent, about 30ft/sec (9.14m/s). So if the rate of descent given, V, lies between these values, the drag coefficient (C_D) can be estimated from equation 3.1.

$$C_D = C_D H-(C_D H-C_D L)(V-5.18)/3.96 \qquad (3.1)(101)$$

If V is higher than 9.14 m/s then C_D=$C_D L$ (103), if V is lower than 5.18 m/s then C_D=$C_D H$ (102).

For gliding parachutes a force coefficient which relates to both horizontal and vertical motion is taken from table 3.3. These values have also been extracted from reference 9, they represent average force coefficients from a number of tests.

Type	Drag coefficient range
1. Flat circular	0.800 - 0.750
2. Conical	0.900 - 0.750
3. Bi-conical	0.920 - 0.750
4. Tri-conical	0.960 - 0.800
5. Extended skirt (10% flat)	0.870 - 0.780
6. Extended skirt (14.3% full)	0.900 - 0.750
7. Hemispherical	0.770 - 0.620
8. Guide surface (ribbed)	0.420 - 0.280
9. Guide surface (ribless)	0.340 - 0.300
10. Annular	1.000 - 0.950
11. Cross (cruciform)	0.820 - 0.600
12. Flat ribbon	0.500 - 0.450
13. Conical ribbon	0.550 - 0.500
14. Conical ribbon (varied porosity)	0.650 - 0.550
15. Hemisflo (ribbon)	0.460 - 0.300
16. Ringslot	0.650 - 0.560
17. Ringsail	0.900 - 0.750
18. Disk-gap-band	0.580 - 0.520
19. Rotafoil	0.990 - 0.850
20. Vortex ring	1.800 - 1.500
28. Balloon (ballute)	1.200 - 0.510

Table 3.2 (10101) Drag Coefficient: Non-Gliding Parachutes

Type	C_{FR}
21. Aeroconical	0.875
22. Le moigne	0.950
23. Parawing (single keel)	1.000
24. Parawing (twin keel)	1.050
25. Parafoil (ram-air)	0.800
26. Sailwing	-
27. Volplane	-

Table 3.3 (10102) Force Coefficient: Gliding Parachutes

The drag coefficient of a parachute is dependent on its construction. Alterations, especially cut-outs such as drive slots in the side of the canopy and vents in the top of the canopy, will change the drag coefficient, because the rate of descent is increased. When a parachute has been manufactured its drag coefficient is determined experimentally. In some cases this value is found to be very different from the original drag coefficient assumed for the parachute. This is demonstrated in reference 14 where the expected drag coefficient is 0.72-0.75 and the value calculated by experiment is 0.49-0.59, a decrease of as much as 35%. So care must be taken to choose the correct drag coefficient in order to realise the required rate of descent.

The values of drag and force coefficients presented in this section are only intended as a guide. In many cases the parachute designer will already know the coefficient he wishes to use from past experience.

3.3 Area

Equation 3.2 is the basic drag coefficient definition for non-gliding parachutes.

$$D_r = C_0 \, 1/2 \, \rho \, V^2 \, S_0 \tag{3.2}$$

Where:

D_r - Drag, which is equal to the weight of the store being delivered plus the weight of the parachute (W_s), in steady descent.

ρ - Air Density.

S_0 - The constructed, or nominal, area, i.e. the total canopy area inclusive of any cut-outs.

The air density can be taken as the sea level value or calculated from equation 3.3 for the troposphere, where A is the altitude (metres)[15].

$$\rho = 1.225(1 - 2.2605X10^{-5}A)^{4.256} \qquad (3.3)(201)$$

Re-arranging equation 3.2

$$S_0 = 2\,W_S/(\rho\,V^2\,C_D) \qquad (3.4)(202)$$

C_D is often expressed as C_{D0} to indicate that it refers to the constructed, or nominal, area.

For gliding parachutes the force coefficient refers to the total velocity (V_T). This can be calculated from equation 3.5 as both the vertical velocity (rate of descent (V)) and horizontal velocity (V_H) are specified.

$$V_T = (V^2 + V_H^2)^{1/2} \qquad (3.5)$$

Then if C_{FR} is the canopy force coefficient, using a similar method to that used for non-gliding parachutes:

$$S_0 = 2\,W_S/(\rho\,V_T^2\,C_{FR}) \qquad (3.6)(206)$$

Many parachutes are designed for the recovery of remotely piloted vehicles (RPV´s). These (and other) stores have a high cross sectional area which can be taken into account in the parachute area calculation, as they will inpart a certain amount of drag to the system. If S_S is the store cross-section area, and C_{DS} the store drag coefficient, nominal area S_0 is calculated from equation 3.4 or 3.6 and the final area with the store area taken into account (S_{final}) from equation 3.7.

$$S_{final} = (C_D S_0 - C_{DS} S_S)/C_D \qquad (3.7)(206a)$$

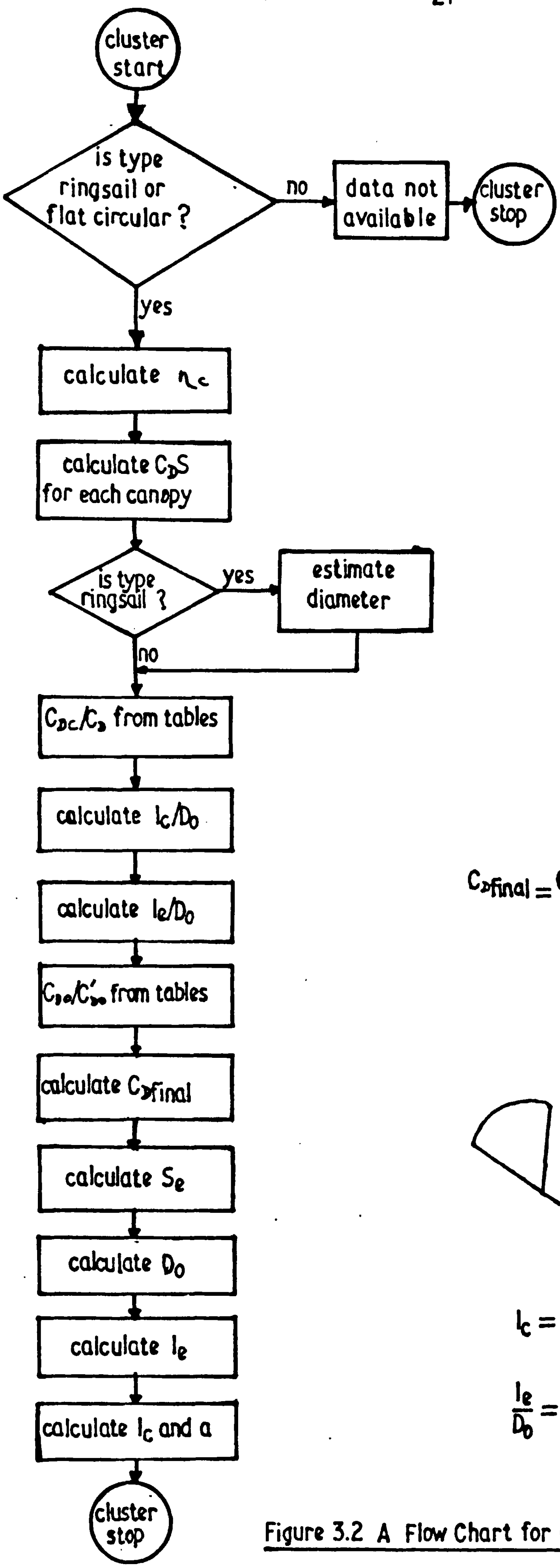
3.4 Clusters

A single parachute of nominal area greater than 30,000 square feet (2787 m^2) (301) creates stowage problems due to its weight and bulk. It opens very slowly and the type generally used for large parachutes (flat circular) is invariably unstable in descent. Parachutes of this size are replaced by a cluster of smaller canopies. The reliability of a parachute system can be improved by the use of a cluster because redundancy can be built into the system by the inclusion of extra parachutes. Flat circular parachutes can be used in clusters because they are flying at an angle of attack and their tendency to become unstable is suppressed. The use of a cluster speeds up inflation and hence the height loss during parachute opening is reduced. Two disadvantages of clusters are reduced drag efficiency and high inflation loads, these high loads are encountered because the canopies open separately.

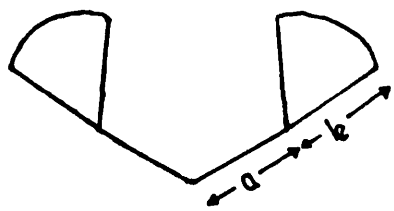
The method used for cluster design is given in the Recovery Systems Design Guide[9]. Corrections are made to the drag coefficient selected for a single parachute to take into account the effects of the cluster and the effects of the rigging lines. From the final drag coefficient calculated the area of each canopy in the cluster can be determined. Figure 3.2 has been constructed to illustrate the calculations required in cluster design.

Data are available for flat circular and ringsail canopies. For flat circular parachutes there are data for a maximum cluster of four canopies. For large (>9.30 metre diameter) ringsail canopies data are available for a maximum cluster of two canopies, and for ringsail canopies of less than 9.30 metre diameter data are available for a maximum cluster of three canopies.

The number of parachutes in the cluster, η_c, is calculated from equation 3.8 which compares the largest sensible canopy



$$C_{D}\text{final} = C_D \cdot \frac{C_{Dc}}{C_D} \cdot \frac{C_{Do}}{C'_{Do}}$$



$$l_c = l_e + a$$

$$\frac{l_e}{D_0} = \frac{98}{100} \cdot \frac{l_c}{D_0}$$

Figure 3.2 A Flow Chart for Cluster Design

($2787\ m^2$) with the area calculated for a single canopy.

$$\eta_C = |(S_0/2787)| + 1 \qquad (3.8)(302)$$

$C_{D final}$ is the corrected drag coefficient:

$$C_{D final} = C_D\ (C_{DC}/C_D)\ (C_{DO}/C'_{DO}) \qquad (3.9)(310)$$

Where (C_{DC}/C_D) is the correction made to the drag coefficient for the effects of the cluster (reduced drag efficiency). It is taken from tables 3.4-3.7 (figure 6.30 of reference 9). If the parachute is a ringsail canopy then its nominal diameter D must be estimated. Let S_{ee} be the estimated constructed area,

$$S_{ee} = S_0/\eta_C \qquad (3.10)(304)$$

and hence D (305).

(C_{DO}/C'_{DO}) is the correction for the effect of the length of the rigging lines on the canopy drag area. It is taken from a curve of (C_{DO}/C'_{DO}) versus l_e/D (figure 6.61 of reference 9, equations 308-309 in the parachute design program). l_e is the parachute line length which is generally 98% of the cluster line length, l_c (see figure 3.2). Therefore:

$$(l_e/D) = (98/100)\ (l_c/D) \qquad (3.11)(307)$$

For good practice the ratio of cluster line length (l_c) to nominal diameter D is taken from equation 3.12.

$$l_c/D = (\eta_C)^{1/2} \qquad (3.12)(306)$$

Hence the corrected drag coefficient of each parachute in the cluster ($C_{D final}$). The area of each parachute is:

$$S_e = (C_D\ S_0)/(\eta_C\ C_{D final}) \qquad (3.13)(303,311)$$

n_c	C_{Dc}/C_D
1	1.000
2	0.980
3	0.965
4	0.920

Table 3.4 (10301)

(Rate of descent <7.62 m/sec)

n_c	C_{Dc}/C_D
1	1.000
2	0.890
3	0.840
4	0.780

Table 3.5 (10302)

(Rate of descent >=7.62 m/sec)

Effect of Cluster on Drag Coefficient.
Flat Circular Parachutes

n_c	C_{Dc}/C_D
1	1.000
2	0.990
3	0.960

Table 3.6 (10303)

(Diameter <9.30 m)

n_c	C_{Dc}/C_D
1	1.000
2	0.930

Table 3.7 (10304)

(Diameter >=9.30 m)

Effect of Cluster on Drag Coefficient
Ringsail Parachutes

hence D, l_e, l_c and a (312-315), where:

$$l_c = l_e + a \qquad (3.14)(315)$$

3.5 Rigging Line Length

The length of a parachute's rigging lines can affect its drag coefficient and its stability. Altering the length of the lines will change the drag area of the canopy and hence its drag efficiency. Shortening the rigging lines will decrease the amplitude of the parachute's oscillations whilst it descends.

Reference 9 suggests designing the line length to minimise canopy weight. However the rigging lines are of little importance to the total parachute weight, increasing the line length by 10% will only increase the parachute weight by about 2%.

The method chosen for use in the parachute design computer program is to calculate the canopy line length as the product of the canopy diameter in feet and a factor. Values of this factor, l_e/D or l_e/D_c, for most types of parachute are given in reference 9. In general l_e/D is 1.0 for solid cloth circular parachutes. These values along with others obtained from the co-operating body are listed in table 3.8. The constructed diameter for the Guide Surface, Vortex Ring and Parawing (single keel) canopies are calculated from equations 3.15-3.18.

Guide Surface (ribbed) (type 8)

$$D_c = 0.63\ D \qquad (3.15)(402)$$

Guide Surface (ribless) (type 9)

$$D_c = 0.66\ D \qquad (3.16)(404)$$

Vortex Ring (type 20)

$$D_c = 1.9\ D \qquad (3.17)(406)$$

Type	l_e/D_0	l_e/D_c
1.Flat Circular	1.00	-
2.Conical	1.00	-
3.Bi-conical	0.95	-
4.Tri-conical	0.90	-
5.Flat Extended Skirt	0.85	-
6.Full Extended Skirt	0.95	-
7.Hemispherical	1.00	-
8.Guide Surface (ribbed)	-	1.33
9.Guide Surface (ribless)	-	1.33
10.Annular	1.25	-
11.Cruciform (cross)	1.50	-
12.Flat Ribbon	1.00	-
13.Conical Ribbon	1.50	-
14.Conical Ribbon (varied porosity)	1.50	-
15.Hemisflo	2.00	-
16.Ringslot	1.00	-
17.Ringsail	1.20	-
18.Disk-gap-band	1.69	-
19.Rotafoil	1.00	-
20.Vortex Ring	-	1.00
21.Aeroconical	1.00	-
22.Le Moigne	1.00	-
23.Parawing (single keel)	-	1.00

Table 3.8 (10401) Line Length Data

Parawing (single keel) (type 23)

$$D_c = (S_0/0.692)^{1/2} \qquad (3.18)(408)$$

As for the drag coefficients given using equation 3.1, these values of the ratio of line length to diameter are only intended as a guide, the parachute designer may already know the figure he wishes to use.

A correction can be made to the drag coefficient of the parachute to take into account the effect of the rigging lines. This procedure, incorporating the correction factor, (C_{D0}/C'_{D0}), is described in the cluster calculations, section 3.4. No correction is made for single parachutes because l_e/D_0 is usually 1.0 and at this value the correction factor is also 1.0.

3.6 Number of Rigging Lines

For circular parachutes the number of lines is equal to the number of gores. One method used to determine the number of rigging lines is to calculate the number of lines required to give a gore width of 1 metre at the skirt, which is a convenient length for parachute packing. Parachutes usually have two risers and the number of lines must invariably be even. In some cases (see table 5.5) four or more attachment points may be required.

The parachute design program uses an old rule of thumb given in reference 9 which states that the number of lines should be equal to the canopy nominal diameter (in feet). This gives a gore width of π feet which is approximately 1 metre. In addition the program determines a number of lines which is divisible by four or six (501-506).

The number of lines on each arm of a cruciform canopy is dependent on the allowable fabric width, which is generally about one metre. If the arm width (t_{cru}) is less than 1.5m 12 lines are used. If t_{cru} is greater than 1.5m equation 3.19, based on fabric widths, is used to calculate the number of lines (Z).

$$Z = 4\ (|(1.14t_{cru} + 1.29)| + 1) \qquad (3.19)(916a)$$

For the parawing (single keel) parachute (type 23, figure A.19) the number of gores, N, is calculated from equation

3.20 (D_c is known from equation 3.17), and Z calculated from equation 3.21.

$$N = |(D_c \; 0.823/0.864)| + 1 \qquad (3.20)(932a)$$

$$Z = 3(2N + 1) \qquad (3.21)(932b)$$

3.7 Staging

If the deployment speed is high, instability of the store can occur so a two, or more, stage system may be required. Also the opening load of a large parachute at high speed may be excessive and damage can occur. Staging involves the deployment of a small parachute called a drogue or auxiliary which stabilises the store and provide some initial deceleration. The drogue can be in the form of a spring loaded parachute or a solid body such as a cone. After a short time, usually about half a second, the main parachute is deployed.

The method used to calculate the drogue size is that given in the Recovery Systems Design Guide[9]. This method requires the maximum allowable loading and the store base area to be supplied. It assumes that the maximum dynamic pressure, q_M, is equal to 1.15 times the steady state dynamic pressure, q_e, when the drogue is fully deployed.

$$q_e = W_S/((C_D \; S)_D + (C_{DS} \; S_S)) \qquad (3.22)$$

Where:

$(C_D \; S)_D$ - drogue drag area
$(C_{DS} \; S_S)$ - store drag area

Putting q_M=1.15q_e and re-arranging 3.22 gives an equation for the drogue drag area:

$$(C_D S)_D = (1.15W_S/q_M) - (C_{DS} \; S_S) \qquad (3.23)(607)$$

The maximum dynamic pressure is calculated from equation 3.24.

$$q_M = F_A/((C_D\ S_0)\ C_{X1}) \qquad (3.24)(606)$$

F_A is the maximum allowable load (in Newtons). C_{X1}, the opening load factor is taken from a mass ratio, R_M, versus C_{X1} curve, figure 6.25 of reference 9. Using a curve fitting routine this figure has been represented by equations (707-707c) in the parachute design program. Mass ratio R_M is the ratio of a measure of the mass of air included within the main canopy to the store mass:

$$R_M = ((C_D\ S_0)^{3/2}\ \varrho)/(W_S/g) \qquad (3.25)(604)$$

Using this method $(C_D\ S)_D$ is calculated. If it is less than zero no drogue is required. Otherwise the drogue area, S_{0D}, and diameter can be calculated by assuming that the drogue drag coefficient equals 0.5 (608). The trailing distance of the drogue is seven times the store base diameter (610), to take wake effects into account.

In the parachute design program the user is given the option to initialise this staging routine if the deployment velocity is greater than 100 m/s.

3.8 Opening Loads

The calculation of the opening force of a parachute is one of the most important stages in its design. Once the opening forces are known, the strength of the materials required can be determined. If an error is made in the calculation of the opening force the consequences could be catastrophic.

Established methods of predicting canopy opening loads require specification of its shape and its variation with time as an empirical input. Because the canopy surface is highly elastic and its interaction with the surrounding airflow complex, accurate values for canopy shapes during inflation are difficult to obtain. In advanced opening load calculation techniques (post 1970) the canopy shape is determined as an output. However in these techniques other data, such as the pressure distribution around the inflating canopy, are required, and these are not readily available.

In this section a number of old and new inflation theories are examined and compared so that the most suitable ones for inclusion in a parachute design computer program can be selected.

3.8.1 A Review of Opening Load Calculation Methods

The earliest inflation load calculation methods applied conservation of mass principles to a control volume defined by the parachute canopy. A good example of this technique is given by O´Hara[16] in 1948. These methods required knowledge of the shape of the canopy as it inflates, which O´Hara represented as a series of truncated cones. O´Hara´s and other similar methods are known as "filling distance theories".

A second set of load calculation methods are known as "filling time theories". These methods use similar mechanical models to O´Hara, but appeared much later (1960´s). They include an equation which determines the volume flow rate into the canopy during opening. One such method is that of Heinrich[17], this method is contained in the United States Air Force Parachute Handbook (reference 3). The results presented in this paper show that Heinrich´s method compares well with measured forces for low deployment altitudes (below 10000ft). A useful summary of these and other opening load calculation methods was made by Roberts and Reddy[18] in 1975.

More recently Purvis[19] has devised an opening load calculation method including fluid kinetics. In his paper Purvis predicts the introduction of an expanded finite-element inflation model. No experimental input is required but this and Heinrich's methods are too complicated for contemporary design use.

Two similar methods are those of Lingard[20] and Wolf[21]. Both these analyses make the assumptions of inviscid flow and inelastic materials. Dimensional analysis techniques are used to identify the important parameters in parachute inflation as:

Froude number $F_r = V_o/(g\ R_p)^{1/2}$ (3.26)

Force coefficient $f_x = F_x/(q_o\ S_p)$ (3.27)

Forebody (or store) mass ratio
$k_f = 3\ m_f/(4\ \varrho\ \pi\ R_p^3)$ (3.28)

Canopy mass ratio $k_p = 3\ m_p/(4\ \varrho\ \pi\ R_p^3)$ (3.29)

where:

- V_o - System initial velocity.
- R_p - Fully inflated canopy radius.
- S_p - Canopy area.
- F_x - Axial force.
- q_o - Initial dynamic pressure.
- ϱ - Atmospheric density.
- m_f - Forebody mass.
- m_p - Canopy mass.

In Wolf's method a system of four differential equations are derived. These equations must be solved in order for inflation forces to be determined.

Lingard's semi-empirical method has been specially formulated for design use. It assumes a unique form of the force-coefficient dimensionless time curve.

dimensionless time $\tau = V_s (t - t_{inf})/D_0$ (3.30)

V_s - Snatch velocity (velocity at start of inflation)
D_0 - Nominal diameter.
t - Time.
t_{inf} - Inflation time.

Hence if the force coefficient - dimensionless time curve is known for one type of parachute, the opening force for a geometrically similar parachute can be calculated at any time t. The data required are q_0, S_p, V_s, D_0 and t_{inf}.

$$t_{inf} = k_x \ D_0/V_s \qquad (3.31)$$

where k_x is a constant, different for each type of parachute.

3.8.2 Semi-Empirical Opening Distance Methods

In these methods the peak opening force in infinite mass conditions, F_x, is defined as the steady state force F_s multiplied by an opening force coefficient C_x[11].

$$F_x = F_s \ C_x \qquad (3.32)$$

Values of C_x are given in tables 2.1 and 2.2 of reference 9, which are summarized in table 3.9.

The opening force in finite mass conditions is obtained by multiplying F_x by a dimensionless opening force reduction factor X_1:

$$F_x = 1/2 \ \rho \ V^2 \ C_D \ S_0 \ C_x \ X_1 \qquad (3.33)$$

This section considers three semi-empirical opening distance load calculation methods. Each requires no specialised input data. They are often used in design applications. If a quick calculation is required, or data for more complicated methods are not available then one of the following methods can be employed.

3.8.2.1 The "Mass Ratio" Method

This method is described in the Recovery Systems Design Guide[9]. The mass ratio R_M is the ratio of a measure of the mass of air included within the canopy to the store mass. It is calculated using equation 3.34. Once this mass ratio is known, the opening force factor C_{X1} or C_k, a combination of C_X and X_1, is obtained from a R_M-C_{X1} curve, for example figure 6.25 of reference 9, which has been fitted to equations (707-707c) in the parachute design program. Finally the maximum opening load, F, is calculated from equation 3.35.

$$R_M = (C_D\ S_0)^{3/2}\ \varrho/(m) \qquad (3.34)(706)$$

$$F = (C_D\ S_0)(1/2\ \varrho\ V^2)(C_{X1}) \qquad (3.35)(708)$$

m - mass of store.

3.8.2.2 The Canopy Loading Method

This method, developed by Knacke, is outlined in reference 11. Knacke argues that for a given canopy shape, the opening force reduction factor, X_1, is a function of canopy loading:

$$\text{Loading} = W/(C_D\ S_0) \qquad (3.36)$$

W - weight of store

A curve of X_1 versus canopy loading is given in reference 11. Typical values are: X_1=1.0 for an aircraft decelerator (loading 14k Pa), X_1=0.33 for a supply dropping parachute (loading 200 Pa) and X_1=0.03 for a man carrying parachute (loading 25 Pa).

3.8.2.3 The "Pflanz" Method

This method was developed in Germany during the 1939-1945 war by Pflanz and Knacke. It assumes that the drag area versus time relationship for an inflating parachute canopy can be expressed in one of a variety of simple, definable forms. It is described in reference 11 but originally appeared in reference 22. X_1, the opening force reduction factor, is calculated from a curve of X_1 vs A_x (fitted to equations (703-704b) in the parachute design program) where:

$$A_x = 2\, W_s / ((C_D\, S_0)\, \rho\, g\, V_1\, t_f) \qquad (3.37)(702)$$

V_1 - velocity.

t_f - inflation time: the time between line stretch and the canopy reaching its first steady state diameter.

From Scheubel's[23] concept that the filling distance, S_f, can

Parachute Type	n_2	C_x
1.Flat Circular	9.0	1.80
5.Extended Skirt 10% Flat	9.0	1.40
6.Extended Skirt 14.3% Full	12.0	1.40
9.Guide Surface (Ribless)	5.0	1.40
11.Cruciform	11.8	1.20
12.Flat Ribbon	14.0	1.05
15.Hemisflo	20.0	1.15
16.Ringslot	14.0	1.05
17.Ringsail	7.0	1.10

Table 3.9 Values of Fill Constant and Opening Force Coefficient for Various Canopies

be expressed in terms of the canopy's nominal diameter, D by:

$$S_f = \eta_2 D \quad (3.38)$$

the filling time, t_f, is expressed in terms of the deployment velocity V_1 as:

$$t_f = (\eta_2 D)/V_1 \quad (3.39)(701)$$

The fill constant, η_2, is also taken from table 3.9 (extracted from reference 11).

3.8.3 A Comparison of Four Different Load Calculation Methods

In order to choose the most suitable load calculation method for inclusion in the parachute design program, results obtained using the Lingard (section 3.8.1 and 3.8.4), mass ratio (section 3.8.2.1), canopy loading or Knacke (section 3.8.2.2) and Pflanz (section 3.8.2.3) methods, were compared with known, experimentally determined, values. Three different sets of input data have been used for this check:

(i)C9 Flat Circular Parachute

The C9 is a 28ft diameter flat circular man carrying parachute. Reference 20 contains results of tests on this parachute, and the determination of its opening force using the 'Lingard' method. The case chosen was:

V_s = 77.46 m/s
m = 199.5 kg
Altitude = 1830 m
(air density = 1.023 kg/m^3 (equation 3.3))
S_0 = 57.21 m^2
D_0 = 8.53 m
t_f = 0.99 sec
Canopy filling time (equation 3.39 (taking η_2=9 for a flat circular canopy, table 3.9))

C_D = 0.775 (drag coefficient estimation)

<u>(ii)Aeroconical Gliding Parachute (Test I)</u>

A check on the program written for the 'Lingard' method (section 3.8.4) was made using the following data for an Aeroconical gliding parachute.

V_s = 50 m/s
m = 125 kg
Altitude = 3000 m (ϱ = 0.9085 kg/m^3 (equation 3.3))
S_0 = 38.5 m^2
D_0 = 7.38 m
t_f = 1.33 sec
(equation 3.39, assuming η_2=9 as for a flat circular parachute)
C_R (=C_D) = 0.875 (drag coefficient estimation)

<u>(iii)Aeroconical Gliding Parachute (Test II)</u>

Reference 20 contains experimental results and an opening force analysis using the 'Lingard' method of the Aeroconical gliding parachute. A case from this report was chosen to be analysed by the other three methods.

V_s = 72.11 m/s
m = 135 kg
Altitude = 460 m (ϱ = 1.172 kg/m^3 (equation 3.3))
S_0 = 21.24 m
D_0 = 5.2 m
t_f = 0.65 sec (equation 3.39, assuming η_2=9)
C_D = 0.875 as assumed above

From the results given in table 3.10 it can be seen that of the four opening load calculation methods tested, three are suitable for inclusion in the parachute design program. The canopy loading or Knacke method is discarded. The results obtained using this method differed vastly from those obtained using the other three. It underestimated on all three checks.

Method	Lingard	Mass Ratio	Knacke	Pflanz
Requires	τ-C_F curve	-	C_x	C_x and t_f
Constraints	incompress-ible flow	-	Altitude < 20000 ft	-
Results (N)				
Test 1 C9	25500 (2)	24493 (2)	20819 (20)	30072 (20)
Test 2 Aer-oconical I	7361 (*)	7651 (*)	5008 (*)	8780 (*)
Test 3 Aer-oconical II	20000 (10)	19821 (11)	16414 (34)	22623 (3)
Advantages	Low Error	Low Error Simple	Simple	Low Error
Disadvanta-ges	Complicated c.p. With Approximate Methods	Approximate Underestim-ates	Approximate Underestim-ates	t_f required

Key

Figures in brackets indicate the percentage error compared with experimental results.

* - there was no experiment in test 2.

Table 3.10 A Comparison of Four Opening Load Calculation Methods

Results obtained using the 'Lingard' method have been shown to compare well with experimental results. Therefore in cases where the $C_F-\tau$ curve is available this method should be used. In cases where the 'Lingard' method cannot be used, either the simple 'mass ratio' method or the more complicated 'Pflanz' method will be adequate. Results obtained using these two methods were sufficiently accurate to indicate that the possible errors will be small, compared with the design factors (typically 2-3) applied. Therefore these three methods are incorporated in the parachute design program and the choice between them made by the user whilst running the program.

3.8.4 Determination of the Forces on the Store, and the Trajectory, of a Two Stage Parachute System, Using Lingard's Semi-Empirical Opening Load Calculation Method

Lingard's method of calculating the opening load of an inflating parachute canopy, as described in section 3.8.1, has been incorporated in a Basic program on a Hewlett-Packard computer at the co-operating body. This program has been expanded in the following way to include the effects of staging (a drogue canopy can be incorporated) and the aerodynamic characteristics of the store. A trajectory calculation is also included. This procedure is incorporated into the parachute design program as described in section 4.2.1.

For reasons described in section 4.3.2 the output from this trajectory calculation is only directed to the Sirius computer screen, it does not appear in a datafile and hence in the optional printed output. Also due to its complexity this part of the parachute design program is slow. If a quick design solution is required one of the other, more approximate, load calculation methods is recommended. However a trajectory calculation is often very important to the design of a parachute system. Altering the program to enable

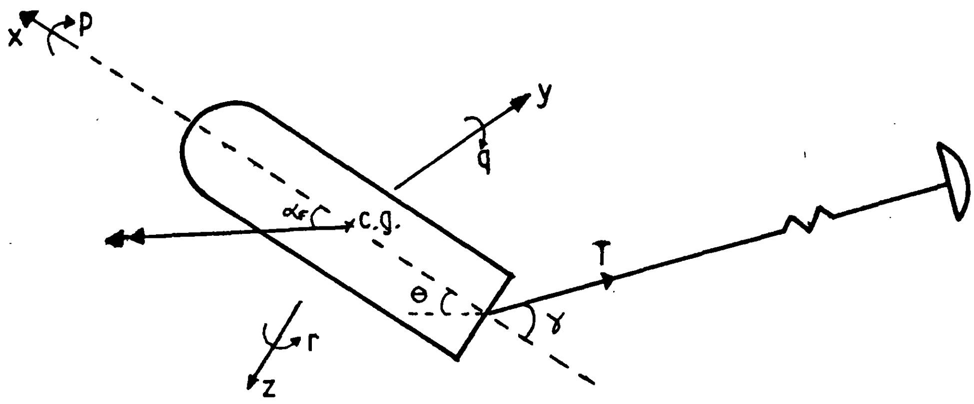


Figure 3.3 Diagram of a Store and Canopy System

this output to be sent to a datafile should not prove too difficult.

For the canopy trajectory, Schatzle and Curry[24] give the following six equations of motion, and three Euler equations, in three dimensions relative to axes fixed in the body, for a store (or forebody) in a forebody-parachute system (see figure 3.3).

$$\dot{u}_F = \Sigma F_{Fx}/m_F + r_F\ v_F - q_F\ w_F$$

$$\dot{v}_F = \Sigma F_{Fy}/m_F + p_F\ w_F - r_F\ u_F$$

$$\dot{w}_F = \Sigma F_{Fz}/m_F + q_F\ u_F - p_F\ v_F$$

$$\dot{p}_F = \Sigma M_{Fx}/I_{xx} \qquad (3.40)$$

$$\dot{q}_F = (\Sigma M_{Fy} + (I_{yy} + I_{xx})\ p_F\ r_F)/I_{yy}$$

$$\dot{r}_F = (\Sigma M_{Fz} - (I_{yy} - I_{xx})\ p_F\ q_F)/I_{yy}$$

$$\theta = (q_F \cos\varphi - r_F \sin\varphi) \sec\psi$$

$$\psi = q_F \sin\varphi - r_F \cos\varphi \qquad (3.40)$$

$$\varphi = p_F - (q_F \cos\varphi - r_F \sin\varphi) \tan\psi$$

Where:

x,y,z	- axes
p,q,r	- angular velocities around x,y,z
u,v,w	- velocities along x,y,z
θ,ψ,φ	- pitch, yaw and roll angles
$\Sigma F_{Fx},\Sigma F_{Fy},\Sigma F_{Fz}$	- forces in x,y,z
m_F	- mass of forebody
I_{xx},I_{yy},I_{zz}	- moment of inertia in x,y,z
$\Sigma M_{Fx},\Sigma M_{Fy},\Sigma M_{Fz}$	- moment about x,y,z
α_F	- angle of attack
γ	- angle of attack (at tail of forebody)
T	- tension (exerted by parachute)

In a two-dimensional analysis $v=0,\psi=0$ and $r=0$. Rolling moment is neglected so $\varphi=0,p=0$ and equations 3.40 can be reduced to:

$$\dot{u}_F = \Sigma F_{Fx}/m_F - q_F w_F$$

$$\dot{w}_F = \Sigma F_{Fz}/m_F + q_F u_F \qquad (3.41)$$

$$\dot{q}_F = \Sigma M_{Fy}/I_{yy}$$

$$\theta = q_F$$

and:

$$\Sigma F_{Fx} = - A_F - m_F g \sin\theta - T \cos\gamma$$

$$\Sigma F_{Fz} = - N_F + m_F g \cos\theta - T \sin\gamma \qquad (3.42)$$

$$\Sigma M_{Fy} = M_{dampF} - N_F \Delta x - \bar{x} T \sin\gamma + M$$

Where:

A_F, N_F - Aerodynamic force on the store in x and z directions

Δx - Distance from store centre of gravity to centre of pressure

M - Aerodynamic moment

M_{dampF} - Aerodynamic damping moment (if required)

From figure 3.4 the aerodynamic forces on the store can be calculated as follows.

$$A_F = D_F \cos\alpha_F - L \sin\alpha_F$$
$$N_F = D_F \sin\alpha_F + L \cos\alpha_F \qquad (3.43)$$

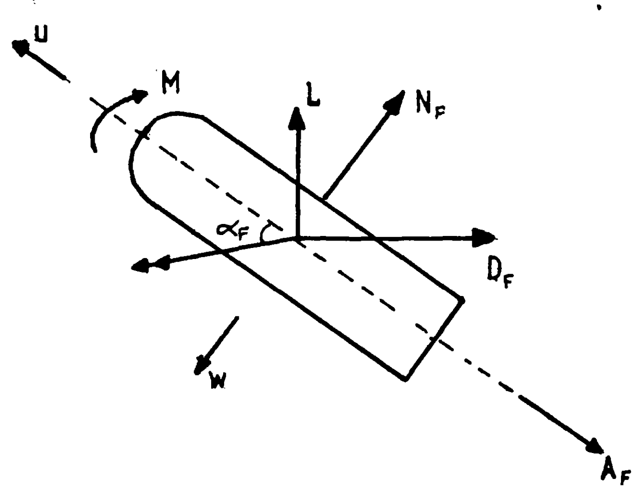


Figure 3.4 Aerodynamic Forces on the Store

Expressing equation 3.41 in coefficient form:

$$A_F = (C_D \cos\alpha_F - C_L \sin\alpha_F)\, \bar{q}_F\, S_b$$
$$N_F = (C_D \sin\alpha_F + C_L \cos\alpha_F)\, \bar{q}_F\, S_b \qquad (3.44)$$

and

$$M = C_M \ \bar{q}_F \ S_b \ d_b \qquad (3.44)$$

Where:

D_F - Drag force on store
L - Lift force on store
C_D - Store drag coefficient
C_L - Store lift coefficient
$\bar{q}_F$ - Store dynamic pressure
S_b - Store base area
C_M - Store moment coefficient
d_b - Store base diameter

The displacement of the forebody in fixed (earth-based) axes can be calculated from:

$$dx = (u \cos\theta + w \sin\theta)\ dt$$
$$dz = (w \cos\theta - u \sin\theta)\ dt \qquad (3.45)$$

re-arranging:

$$\dot{x} = u \cos\theta + w \sin\theta$$
$$\dot{z} = w \cos\theta - u \sin\theta \qquad (3.46)$$

Where:

x,z - displacement in x (horizontal) and z (vertical) directions.

The parachute exerts a tension T on the system (figure 3.3). The tension exerted by the drogue canopy is calculated from equation 3.47, the drogue being assumed to be instantaneously deployed to its flying diameter using a spring or similar mechanism.

$$T = C_{DD} \ 1/2 \ \rho \ V_T^2 \ S_D \qquad (3.47)$$

Where:

C_{DD} - drogue drag coefficient
V_T - total velocity
S_D - drogue canopy area

After a certain time delay the main canopy is deployed. The tension this parachute exerts on the system is calculated using the method of Lingard[20]. τ_o, dimensionless time of the peak of the lines-taut snatch force, is calculated from equation 3.48, based on empirical results.

$$\tau_o = f(V_T) \tag{3.48}$$

The inflation time

$$t_{inf} = - \tau_o \; D_0/V_T \tag{3.49}$$

and

$$t/t_{inf} = \tau/\tau_o \tag{3.50}$$

so at any time t, τ/τ_o can be calculated and the tension exerted by the parachute obtained from a τ/τ_o-C_F curve of the type of canopy in question.

Equations 3.41 and 3.46 are solved using a Runge-Kutta technique.

3.8.5 Cluster Opening Forces

Parachutes in clusters always open independently of each other and are therefore subject to high loads for a short time. The cluster opening forces are calculated using a method given in the Recovery Systems Design Guide. This method is based on the mass ratio opening load calculation method, section 3.8.2.1. Mass ratio, R_M, and opening load factor, C_X, are calculated as described in this section. If all the parachutes in the cluster were to open together (synchronous opening) the opening force could be calculated from equation 3.51.

$$F_{SC} = (C_D\ S_0)\ 1/2\ \varrho\ V_D^2\ C_X/n_C \qquad (3.51)(711)$$

Here $(C_D\ S_0)$ is the total nominal drag area of the cluster. For the more realistic non-synchronous opening case (the parachutes open independently of each other) R_{ML} is defined:

$$R_{ML} = R_M/n_C \qquad (3.52)(712)$$

C_{XL} is calculated from figure 6.25 of reference 9 using R_{ML} (curve fit equations (707-707c)). Then:

$$C_Y = C_{XL}/C_X \qquad (3.53)(714)$$

And the non-synchronous opening load:

$$F = F_{SC}\ C_Y \qquad (3.54)(715)$$

3.9 Reefing

Parachute reefing is a process in which the canopy opens in a number of separate controlled stages. It is similar to staging in as much as it serves to reduce the deployment loads on the canopy. The amount of reefing can be controlled to realise a maximum deployment load. This facility is especially useful for aircraft landing deceleration parachutes. The parachute is disreefed at touchdown to provide a high braking load.

There are two types of reefing: vent reefing and skirt reefing (see figures 3.5 and 3.6). In vent reefing a line attached to the centre of the vent is pulled down to reef the parachute.

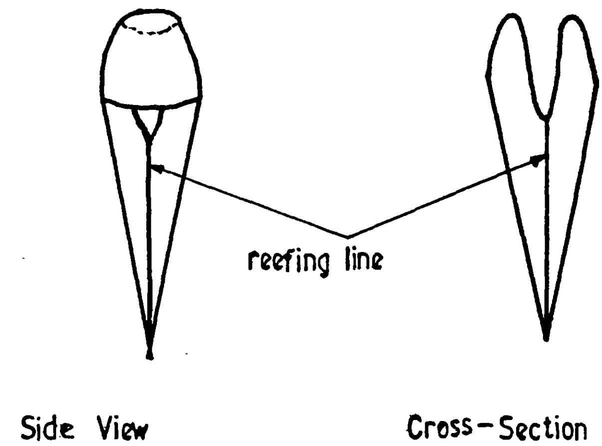


Figure 3.5 Vent Reefing

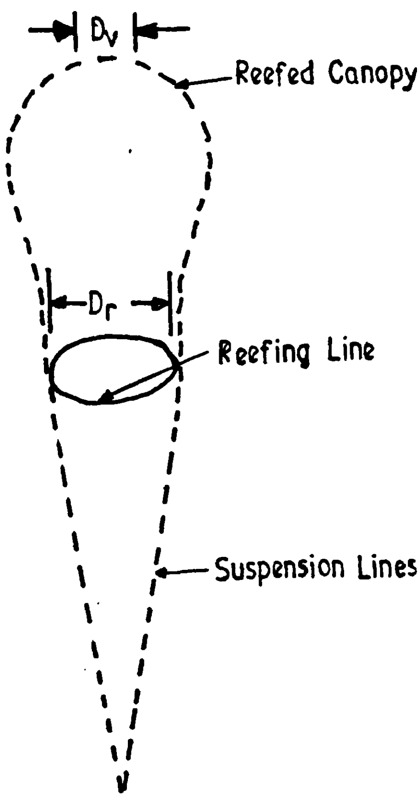


Figure 3.6 Skirt Reefing

The most common method of reefing, and the method assumed by the parachute design program, is skirt reefing. A reefing line which restricts the opening of the canopy is attached to the skirt. Mechanical reefing line cutters are used to cut this line at a specified time, the canopy then fully inflates.

The method from reference 9 that is used to calculate drogue size (section 3.7) is also used to calculate the required reefed canopy area. This method involves equating the equilibrium (reefed or drogue deployed) dynamic pressure to a constant multiplied by the maximum dynamic pressure experienced by the system. The calculations used in this routine are listed in figure 3.7. q_{dr}, the maximum dynamic pressure which is assumed to occur at disreef, i.e. when the reefing line is cut, is estimated to be 1.1 times the steady state reefed dynamic pressure:

$$q_{dr} = 1.1\ W_s/(C_D\ S)_r \tag{3.55}$$

$(C_D\ S)_r$, the reefed drag area can be calculated from this equation.

The maximum dynamic pressure (at disreef) q_{dr} is obtained from equation 3.56:

$$q_{dr} = F_{dr}/(C_D\ S_0\ C_{Xdr}) \qquad (3.56)(809)$$

F_{dr} is the allowable maximum force at disreef, it is estimated to be equal to the weight of the store multiplied by the maximum allowable load factor:

$$F_{dr} = W_S\ (G_A - \sin\theta_T) \qquad (3.57)(806)$$

G_A is the maximum load factor calculated from the allowable maximum load, F_A, which is assumed to be supplied.

$$G_A = F_A/W_S \qquad (3.58)(802)$$

θ_T is the trajectory angle, if it is unknown vertical (downwards) deployment is assumed, i.e.

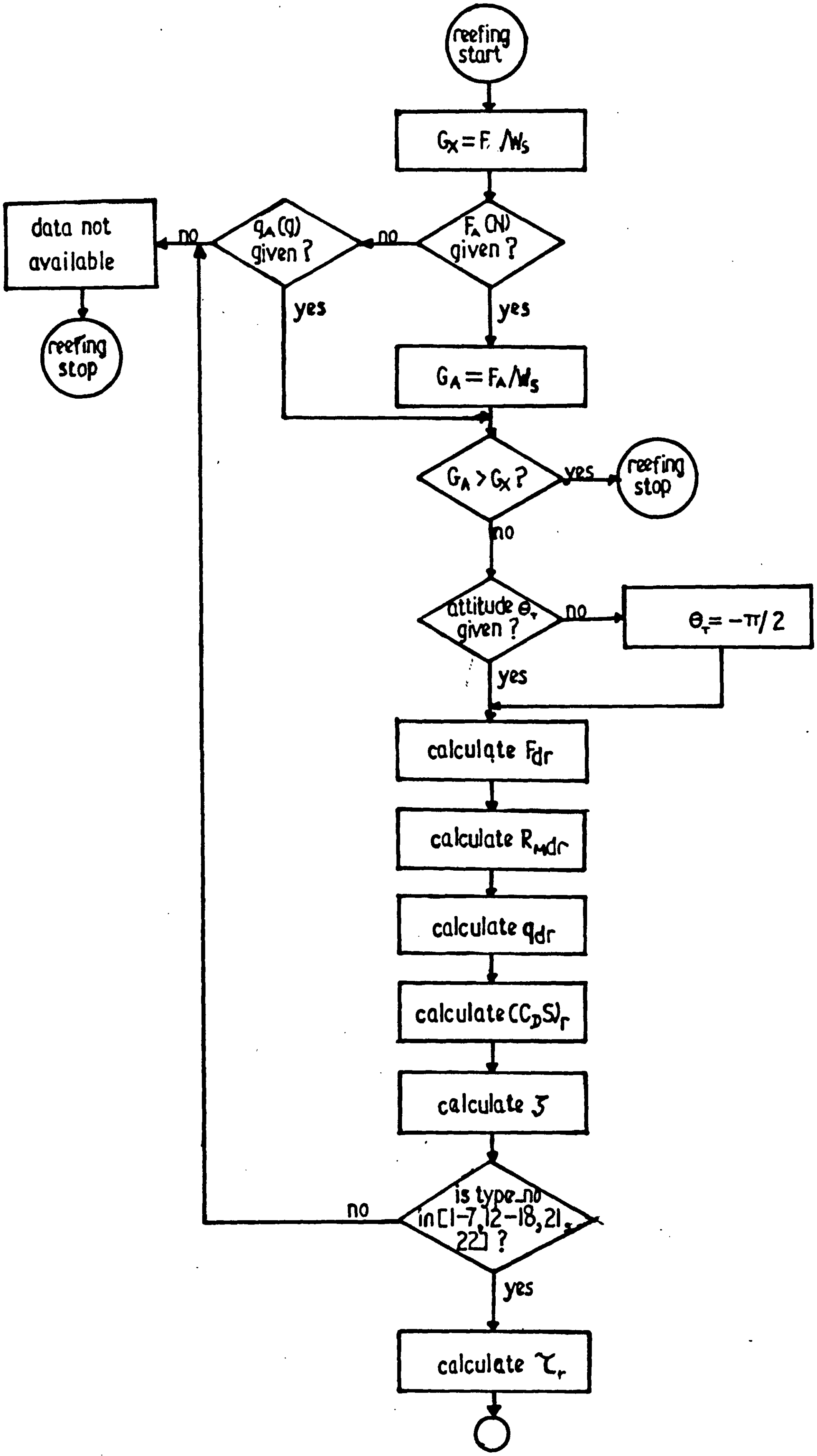


Figure 3.7 Reefing Flowchart

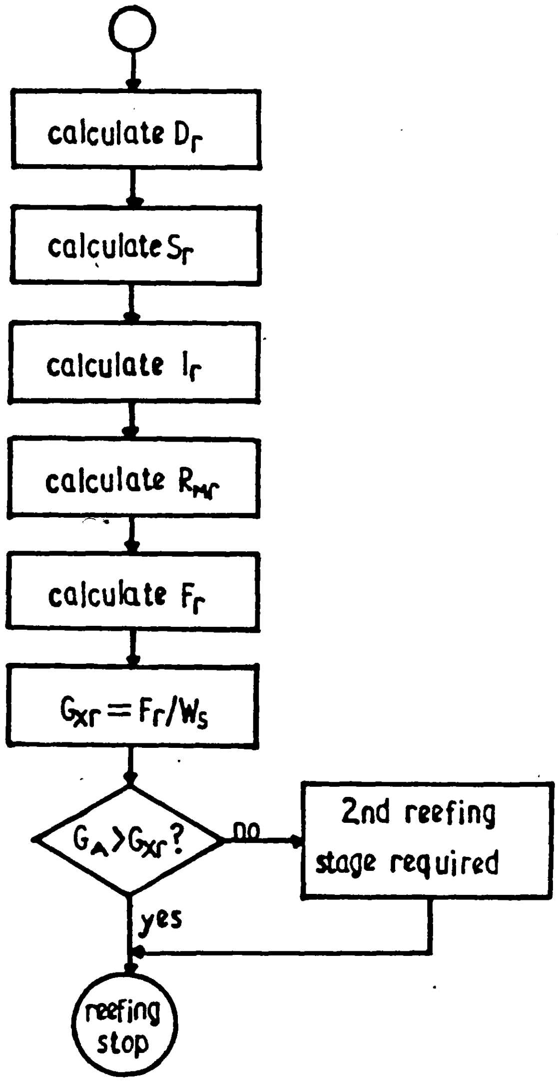


Figure 3.7 (continued)

$$\theta_T = -\pi/2 \qquad (3.59)(805)$$

C_{xdr} is the opening load factor at disreef, it is taken from a curve of C_{xdr} versus R_{Mdr}, figure 6.25 of reference 9 (this curve is fitted to equation (808) in the parachute design program). R_{Mdr} is the mass ratio on disreef,

$$R_{Mdr} = ((C_D\ S_0)^{3/2}\ \varrho)/(W_S/g) \qquad (3.60)(807)$$

The reefing ratio ζ is defined as the ratio of the reefed to unreefed drag area.

$$\zeta = (C_D\ S)_r/(C_D\ S_0) \qquad (3.61)(811)$$

The reefing line ratio, τ_r, is defined as the ratio of the reefed to the nominal canopy diameter.

$$\tau_r = D_r/D \qquad (3.62)$$

τ_r is obtained from a curve of ζ versus τ_r, figure 6.64 of reference 9 (the curves in this figure have been defined by equations (812-816) in the parachute design program). Data in this figure are available for a limited number of parachutes (see figure 3.7). Hence reefed diameter, D_r (817), the canopy reefed area, S_r (818) and the reefed drag coefficient, C_{Dr}.

The length of the reefing line, l_r, is taken from equation 3.63.

$$l_r = \pi\ D_r \qquad (3.63)(819)$$

The force encountered by the canopy opening to its reefed area is calculated using the mass ratio method, section 3.8.2.1. R_{Mr} is the reefed mass ratio.

$$R_{Mr} = ((C_D\ S)_r^{3/2}\ \varrho)W_S/g \qquad (3.64)(820)$$

Hence C_{xr} from figure 6.25 of reference 9 (fitted to equations (707-707c) in the parachute design program).And the reefed opening force:

$$F_r = (C_{xr})(1/2\ \rho\ V^2)(C_D\ S)_r \qquad (3.65)(822)$$

If F_r is greater than the allowable maximum force, F_A, a second stage of reefing is required. In the parachute design program only a single stage of reefing is assumed as the reefing routine is seldom used. If more stages of reefing become increasingly required a modification will have to be made to the program to accommodate a multi-stage reefing facility.

The parachute design program enters this reefing routine if the calculated opening load is greater than the maximum load specified by the user.

3.10 Structural Analysis

The stressing of parachute canopies is made difficult by the same complications as those encountered in the calculation of opening loads, i.e. the canopy shape is constantly changing during inflation. So the parachute designer can only guess at the shape of the canopy at the time of maximum load. In addition the canopy load distribution is determined by the canopy shape and the shape itself depends on the canopy load distribution. A possible solution to this problem is to use an iterative technique. Such a method, for slotted canopies, is described in section 3.10.4.

3.10.1 Design Factor

The design load, F_D, is obtained by multiplying the maximum opening force obtained, F, by a design factor, D_F.

$$F_D = F.D_F \qquad (3.66)(901)$$

The design factor includes an ultimate factor and a safety factor:

$$D_F = U_F \cdot S_F \tag{3.67}$$

values of these factors are given in table 8.6 of reference 9.

The size of the design factor depends on whether the parachute is manned or not. Taking the maximum design factors from table 8.6 of reference 9.

D_F = 3.1 man carrying parachute

D_F = 2.3 unmanned application

These factors can be changed by the designer if he so wishes. Design factors are often as low as 1.33 to 1.5, for military applications, but a factor of at least 2.0 is recommended, bearing in mind the accuracy of the opening load calculation.

3.10.2 Solid Cloth Circular Parachutes

Johns[1] calculated the maximum tension in the canopy, T_r as:

$$T_r = 1.5\ \pi\ C_D\ \rho\ d\ V_x^2/(\ 8\ C) \tag{3.68}$$

and the line load, F_l as:

$$F_l = (3\ D_{max})/(2\ n\ \cos\alpha) \tag{3.69}$$

Where:

- d - canopy diameter
- V_x - speed
- D_{max} - maximum drag
- n - number of lines
- α - angle the lines make with the vertical
- C - a constant which depends on the type and construction of the parachute

Brown[2] gives the following empirical formulae:

$$\text{Fabric Strength } S_f = (3/8)\ (D_{cm}/K_s)\ (V_m/100)^2 \qquad (3.70)$$

$$\text{Line Strength } S_l = 21\ (D_{cm}^2/n)\ (V_m/100)^2 \qquad (3.71)$$

Where:

D_{cm} - canopy maximum diameter
V_m - velocity at inflation to maximum diameter
K_s - constant, dependent on the type of parachute

Both these methods are from the 1940's and values of the constants C and K_s are only supplied for the small number of parachutes in use at that time. Parachutes have altered substantially in the last 40 years. Different materials, construction techniques and types are commonplace. Therefore the validity of Johns' theoretical and Brown's empirical formulae, in the present day, is doubtful.

A more recent and advanced stressing method is the 'Inflation Energy Transfer Method'. This was used by Houmard[25] for the analysis of the Viking, Mars soft landing, parachute. This method calculates the work applied by the inflation gas (Mars atmosphere) during opening. This work is equated to the strain energy absorbing capacity of the primary structural components. Knowing this and the inflation time the cloth stress can be determined. To correctly formulate this method the canopy shape during inflation is required. Houmard obtained this from film of trials, but for general design use this information is unavailable. This method is therefore too complicated for inclusion in the parachute design program.

The stressing method generally used for solid cloth circular parachutes is similar to Johns'[1] method and given in the Recovery Systems Design Guide[9]. Using a membrane analysis technique, for a canopy of general curvature the critical fabric load T_F is taken from equation 3.72.

$$T_F = C_s \, (p \, r)_{max} \tag{3.72}$$

Where:

r - radius of curvature

$$r_{max} = D_p/2 \tag{3.73}$$

D_p - projected canopy diameter (in flight)
C_s - constant
p - differential pressure

The canopy is assumed to be hemispherical and therefore C_s = 0.5. If S_p is the projected canopy area then:

$$p_{max} = F_D/S_p \tag{3.74}$$

Substituting for r_{max}, p_{max} and S_p in 3.72:

$$T_F = F_D/(D_p \, \pi) \qquad (3.75)(903)$$

Fabric tensile strengths are usually quoted in (N/mm)*50 so the value of T_F, obtained from equation 3.75, must be multiplied by 50 and divided by 1000. Values of projected (inflated) diameter are given in table 3.11, they have been taken from tables 2.1-2.5 of reference 9.

The line strength, T_L, is calculated using a similar method to that of Johns'.

$$T_L = F_D/(Z \cos\alpha) \qquad (3.76)(905)$$

Where α is the inclination of the rigging lines to the vertical. It is unknown at the time of maximum load so the worst case is assumed - full inflation.

$$\alpha = \sin^{-1}(D_p/(2 \, l_e)) \qquad (3.77)(904)$$

The strengths of the other components of the parachute canopy

Type	D_p/D_0
1.Flat Circular	0.66
2.Conical	0.70
3.Bi-conical	0.70
4.Tri-conical	0.70
5.Extended skirt 10% flat	0.68
6.Extended skirt 14.3% full	0.68
7.Hemispherical	0.66
18.Disk-gap-band	0.65
19.Rotafoil	0.90
21.Aeroconical	0.66
22.Le moigne	0.66

Table 3.11 (10901) Values of the Ratio of Projected to Nominal Diameter

are factored from the line strength:
Radial tape strength, T_T. A factor of 0.9 is used, as in the Recovery System Design Guide[9].

$$T_T = 0.9\ T_L \qquad (3.78)(912)$$

The radial tapes are very important for the rotafoil and disk-gap-band parachutes so for these two types:

$$T_T = T_L \qquad (3.79)(911)$$

Factors for the skirt band, vent band and vent line strengths are taken from the Kevlar Design Manual[10].

$$\text{Vent Band } VBS = 2.27\ T_L \qquad (3.80)(906)$$

$$\text{Vent Line } VLS = 1.00\ T_L \qquad (3.81)(907)$$

Equation 3.81 assumes that the line strength is less than 6000N which is true for the majority of cases.

Skirt Band $SBS = 1.25\ T_L$ (3.82)(908)

The above factors are designed for ribbon parachutes and may be high for solid cloth circular canopies. The user of the parachute design program is able to change these stressing factors if he so wishes. However, the skirt and vent bands are very important, if either of them fails extensive canopy damage usually follows. Therefore an the use of an overstrength skirt and/or vent band is common practice.

A horizontal reinforcing band can be included if required. Since the main load path is from the canopy fabric to the radial tapes the strength of these bands does not need to be very high. The factor chosen for this component is the one that is used for horizontal ribbons in ribbon parachute design, so from reference[10], strengthening band strength:

one use of parachute $SB = 0.55\ T_L$ (3.83)(909)

many uses of parachute $SB = 0.46\ T_L$ (3.84)(910)

3.10.3 Cruciform Parachute

The stressing of this canopy uses similar techniques to those used for solid cloth circular parachutes (section 3.10.2). The geometry of the cruciform canopy must be determined, AR is the arm ratio:

$$AR = D_{cru}/t_{cru} \qquad (3.85)(914)$$

Knowing the arm ratio the arm width can be calculated as follows:

$$t_{cru} = (S_0/(2\ AR - 1))^{1/2} \qquad (3.86)(913)$$

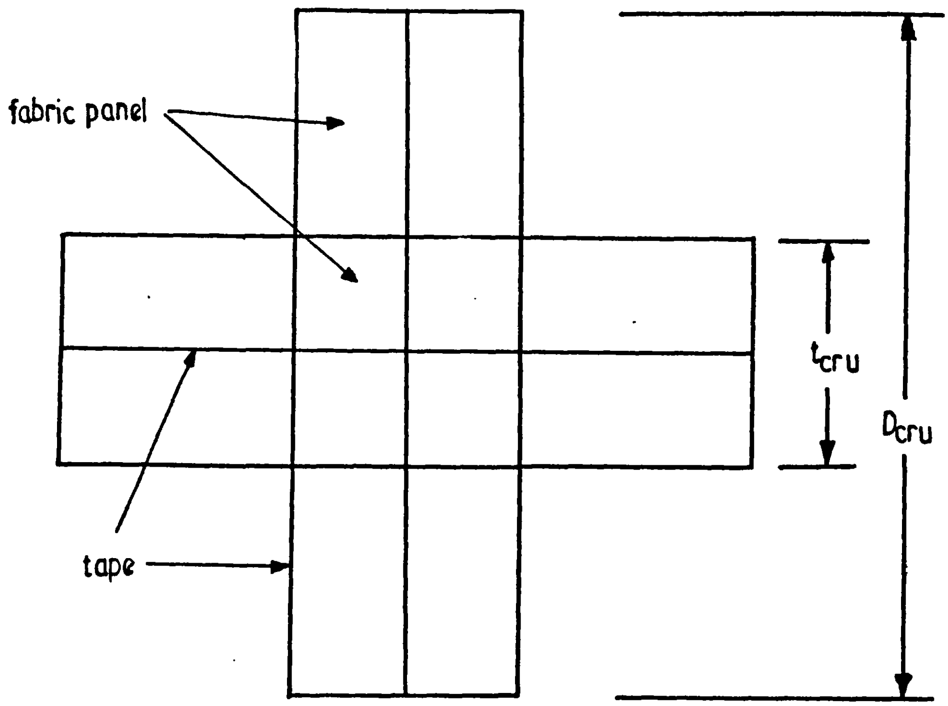


Figure 3.8 Cruciform Parachute Dimensions

and hence the arm span, D_{cru}, from equation 3.85.

The fabric strength (N/mm) is obtained from equation 3.87, based on Johns[1].

$$T_F = 1.7\ F_D/(D_{cru}\ 1000) \qquad (3.87)(915)$$

Line and tape strengths as for solid cloth circular parachutes:

$$T_L = F_D/(Z\ \cos\alpha) \qquad (3.88)(917)$$

where $\alpha = \sin^{-1}(0.69\ D/(2\ l_e))$ (3.89)(916)

$$T_T = 0.9\ T_L \qquad (3.90)(918)$$

A vent is unnecessary on the cruciform parachute, and the tape acts as skirt band and strengthening band, so the stressing of this type of canopy is very straightforward.

3.10.4 Slotted Canopies

One method that can be used to determine the stresses in ribbon parachutes is CANO. This is a Fortran code which uses a combination of finite element and iterative techniques to stress ribbon parachutes. It is based on the Pressure Strain Equilibrium Stressing method, and was devised by Mullins and Reynolds[26] for the structural analysis of the Apollo earth landing system.

In this method the canopy is split into separate members (radial, vertical and horizontal) and the canopy pressure distribution is estimated. A trial pressure and skirt diameter are assumed.

Equilibrium of the canopy is established, first at the skirt then working vertically through the members to the vent. This solution is then compared with boundary conditions at the vent. This process is repeated, choosing a new diameter and pressure, until the boundary conditions are satisfied.

The results of tests on the Apollo landing parachute were inconclusive, in many cases CANO did not converge to give an answer. Garrard and Muramoto[27] devised an improved version of CANO, CANO 2, which included a Newton-Raphson procedure to speed up the iteration process. Results from this program were compared with those obtained from experiments by Konicke and Garrard[28]. The comparison between experimental and calculated stresses was poor.

A listing of a further improved version of CANO, CANOWG[29], has been obtained. This version appears to produce results that agree well with experiment. However this program is not working on the Sirius computer at the time of writing and therefore cannot be used in the parachute design program. Also the input required for CANO includes the pressure distribution curve for the canopy which may prove difficult to obtain.

The methods used in the parachute design program for the stressing of slotted canopies are taken from the Kevlar Design Guide[10]. Flying diameter, D_p, is taken from table 3.12, and the rigging line angle, α, and hence the required rigging line strength, T_L, is calculated as for solid cloth circular parachutes, equations 3.76 and 3.77 (919-921). Horizontal ribbon strength, HRS, is factored from the line strength using equations 3.91 and 3.92.

$$\text{repeated use HRS} = 0.55\ T_L \qquad (3.91)(922)$$

$$\text{one use HRS} = 0.46\ T_L \qquad (3.92)(923)$$

This horizontal ribbon strength must be converted to a required fabric strength for the ringslot and ringsail canopies, as fabric is used for the rings as opposed to tape. To calculate this strength the parachute gore construction details are required. These and the required fabric strength are calculated in appendix A and the materials section (3.11).

The radial tape strength, assuming a double thickness of tape is used, for all parachutes except the ringsail (which has no radial tapes), T_T:

$$T_T = 0.506\ T_L \qquad (3.93)(924)$$

The skirt band, vent band and vent line strengths are calculated using equations 3.80 to 3.82 (906,908,927).

The vertical tapes incorporated in ribbon parachutes, except the ringsail, carry little load. Their main contribution is to the geometric porosity of the canopy. Their numbers and the number of horizontal ribbons can be altered to realise a required geometric porosity (see section 3.13.4). So for all parachutes except ringsail, vertical tape strength, VTS:

$$\text{VTS} = 0.25\ T_L \qquad (3.94)(928)$$

Type	D_p/D_0
12.Flat ribbon	0.67
13.Conical ribbon	0.70
14.Conical ribbon (varied porosity)	0.70
15.Hemisflo	0.62
16.Ringslot	0.68
17.Ringsail	0.69

Table 3.12 (10902) Values of the Ratio of Flying Diameter to Nominal Diameter

The vertical tapes are important for ringsail parachutes because there are no radial tapes so:

$$VTS = 0.506\ T_L \qquad (3.95)(929)$$

3.10.5 Parawing Parachute

Due to its construction (figure A.19) the parawing requires a unique stressing method. It is assumed that flying diameter D_p is equal to 0.66 D_c, as for most parachutes, tables 3.11 and 3.12. The constructed diameter, D_c, has been calculated in equation 3.18 (930). Fabric strength T_F in (N/mm):

$$T_F = F_D/(0.66\ \pi\ D_c\ 1000) \qquad (3.96)(931)$$

and line strength:

$$T_L = F_D/(z\ \cos\alpha) \qquad (3.97)(933)$$

where $\alpha = \sin^{-1}(0.66\ D_c/(2\ l_e))$ (3.98)(932)

Radial tape strength, T_T, is calculated as for the other types of parachute:

$$T_T = 0.9\ T_L \tag{3.99)(934}$$

and skirt band strength:

$$SBS = 1.25\ T_L \tag{3.100)(935}$$

There is no vent on this parachute.

3.10.6 Reefing Line Stressing

Using a method taken from the Recovery Systems Design Guide[9], the ratio of the load in the reefing line, f´, to the maximum opening load, F, is:

$$f'/F = ((\tan\psi_r - \tan\varphi_r)/(2\ \pi)) \tag{3.101)(949}$$

ψ_r is the angle of conversion of the canopy radial members, and φ_r is the convergence angle of the canopy lines (figure 3.9).

$$\psi_r = \sin^{-1}((D_{pr} - D_r)/(2\ h_x)) \tag{3.102)(948}$$

$$\varphi_r = \sin^{-1}(D_r/(2\ l_e)) \tag{3.103)(947}$$

h_x is the non-inflated part of the canopy (see figure 3.9).

$$h_x = D/2 - h_c \tag{3.104)(946}$$

Where the inflated part of the canopy:

$$h_c = \pi\ D_{pr}/4 \tag{3.105)(945}$$

D_{pr} is an estimation of the inflated diameter of the reefed canopy. This is calculated from the inflated reefed area, S_{pr}, which is estimated to be the ratio of the reefed drag

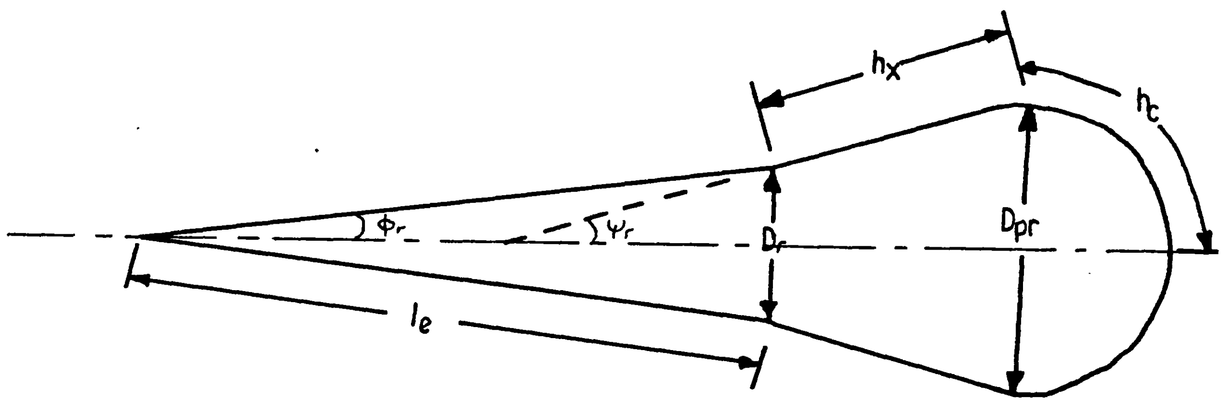


Figure 3.9 Reefed Canopy Configuration

area to the drag coefficient based on the projected canopy area:

$$S_{pr} = (C_D S)_r / C_{DP} \qquad (3.106)(943)$$

$(C_D S)_r$ is known from equation 3.55. C_{DP} is calculated using the drag coefficient ratio, C_{DP}/C_{DPO}. This is determined from figure 6.65 of reference 9 (this figure is defined by equations (936-940) in the parachute design program).

$$C_{DPO} = C_D / (D_p/D_0)^2 \qquad (3.107)(941)$$

D_p is known from tables 3.11 and 3.12 C_{DP} is calculated using equation 3.107 and the drag coefficient ratio.

The required reefing line strength, T_{RL}, is calculated by multiplying f′ by the design factor previously selected.

$$T_{RL} = f' \, D_F \qquad (3.108)$$

3.11 Parachute Materials

Nylon and polyester are ideal materials for parachute applications because they have high strength to weight ratios. Problems with nylon can occur at the high temperatures caused by very high velocity applications, its strength is reduced at temperatures of over 250^{o}F and it melts at 415^{o}F. In such cases kevlar can be used. This material is more expensive than nylon but it is stronger and operates well in temperatures in excess of 500^{o}F. Ribbon parachutes are generally used for high velocity, high temperature applications (e.g. aircraft deceleration), so kevlar is often used in their construction. The Kevlar Design Guide[10] is intended for ribbon parachute design.

3.11.1 Fabric

The choice of canopy fabric depends on a number of criteria: strength, weight, width, cost, porosity, colour and availability. Most fabrics are supplied in a variety of colours so this criteria is not very important. The cost of a particular material changes every few months or so, keeping a list of costs up to date would prove difficult. Therefore, although it is important to the designer, material cost has not been included in the design program at this, choice of materials, stage. The availability of a particular material changes from day to day, so in order to use the materials most likely to be available at the co-operating body the materials chosen for the program have been taken from a list of GQ Defence Equipment preferred materials.

The two most important criteria in the choice of a parachute fabric are width and strength. Fabrics are supplied to a minimum width. The maximum panel width of the canopy must be calculated from the ratio of the gore height to the number of fabric panels used (Appendix A), only fabrics of a greater minimum width can be used. The fabric ultimate tensile strength must be greater than the calculated required fabric strength (section 3.11.1.1).

The user of the parachute design program has the choice of either selecting a fabric from those available to the program or inputting a fabric specification. The parachute design program accesses a table of fabrics (table D.1 (11001)). from this table all the understrength and too narrow materials are removed. The remaining materials are displayed for the user to select one for use. He may require a low weight or highly porous (improved stability) material.

3.11.1.1 Fabric Strength

The required canopy fabric strength, T_F, has been calculated for all the types of parachute except the ringslot and ringsail canopies. A horizontal ribbon strength has been given for these parachutes. Now that their gore construction details are known from appendix A, the equivalent required fabric strength can be calculated.

The canopy ring width, RW, is calculated in section A.12.1 for the flat and conical ringslot parachutes and in section A.14.1 for the ringsail parachute.

fabric strength (N/mm) T_F = HRS/(RW 1000) (3.109)(1044,1048, 1053)

T_F can be compared with fabric ultimate tensile strengths, and hence a suitable canopy fabric chosen.

For the ringslot and ringsail parachutes the minimum required canopy fabric width, including a sewing allowance of 50mm, is equal to RW + 0.05 meters (1054a).

3.11.2 Tapes and Webs

Nylon tapes and webs are used for the following parachute components: radial tape, skirt band, vent band, horizontal ribbon, vertical ribbon and reinforcing band. The criteria for choosing the tapes are, in order of importance: strength, weight, width, availability and colour. The tapes and webs available to the computer program are taken from a list of GQ Defence Equipment preferred materials in order to satisfy the availability requirements - although the availability of a particular material at a particular time cannot be guaranteed.

The user of the program can either choose a material from those stored on disk or input a known material specification. The program compares the required strength of a component with a list of materials (table D.2 (11002)) and neglects all the understrength tapes and webs, although a slightly understrength material is sometimes allowed if the safety factor used is high (2-3). The user is then able to choose from the remainder of the materials. A wide tape (2 inches or more) may be required for the skirt band, vent band, or horizontal ribbon, or low weight materials may be the important criteria.

3.11.3 Cord

Nylon cord is used for parachute rigging and vent lines. Criteria for choosing the cord for a particular application, in order of importance, are: strength, weight, availability and colour. The colour of the cord is relatively unimportant as most cords are white. The availability requirement is satisfied by taking cords from the GQ preferred materials list. The user has the choice of obtaining a material from those available to the program (table D.3 (11003)) or inputting an alternative specification. The program takes the required rigging line or vent line strength and removes all the understrength cords in its table. The remainder are displayed to the user who generally chooses the lightest cord possible.

3.11.4 Reserve Factors

A reserve factor is calculated for each component by dividing the strength of the material used by the calculated required strength (1056-1065). Reserve factors of 1.0 represent the best possible use of materials. If a component has a reserve factor of less than 1.0 it is understrength, a factor of much greater than 1.0 indicates redundancy.

3.12 Landing Control

If the store is fragile its impact with the ground can be softened using a variety of methods: crushable materials, airbags or retro-rockets. The use of any of these accessories will add weight to the parachute system. A calculation of the effects of including landing control could be added to the parachute design program if required. At present no such calculation is included.

3.13 Weight and Volume

A parachute's weight can be estimated from its nominal diameter. This procedure is shown to be inaccurate in section 3.13.3. The weight calculation method employed in the program uses the gore dimensions and rigging line length, together with the canopy material properties, to calculate the weight of each component of the parachute system (fabric, tape, lines, etc.). The parachute volume is then determined from the total weight and the parachute packing density.

3.13.1 Parachute Weight

The calculation of total canopy weight, WTOT, for each type of parachute used in the parachute design program (table 3.1) is included in appendix A.

The canopy weight is dependent on the construction methods used. The canopy can be of either "block" or "bias" construction, see figure 3.10. In bias construction the gore

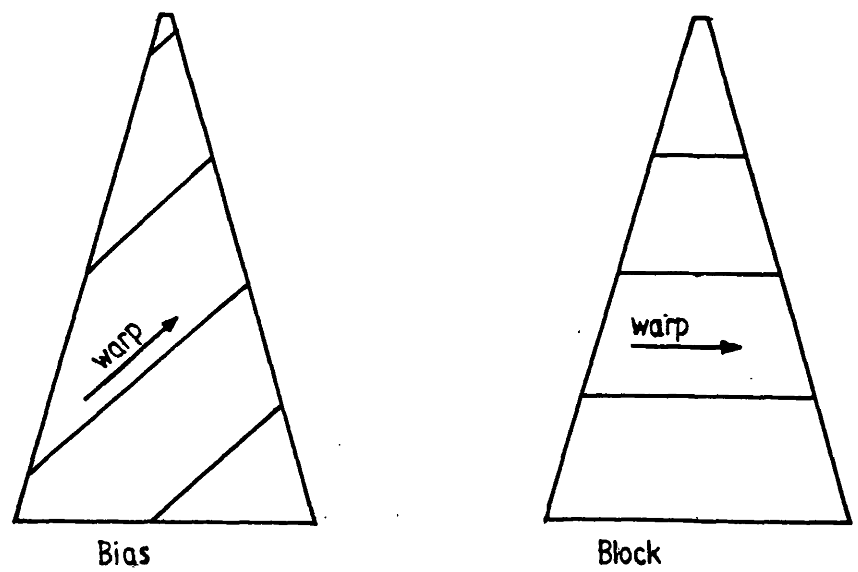


Figure 3.10 Bias and Block Construction Methods

is divided into panels which are cut on the bias, i.e. the fabric threads are at 45 degrees to the meridian of the gore. In block construction the fabric threads are parallel to, and at right angles to, the meridian of the gore. The computer program assumes the canopy is block constructed as shown in figures A.1, A.2, etc.

The number of gores, N, is also required for the canopy weight calculation. This has already been calculated for the parawing (single keel) parachute (equation 3.20). For the cruciform parachute (see figure 3.8) the number of gores per arm is usually:

$$N = Z/4 - 1 \qquad (3.110)(1030)$$

more gores can be added to cruciform parachutes if required. For all the other types of parachute listed in table 3.1:

$$N = Z \qquad (3.111)(1001,1004,1008,1014,1022,1025,1027,1032,1035)$$

If the parachute is reefed, reefing line weight (kg):

$$WRL = l_r/WTRL \qquad (3.112)(1845)$$

where WTR is the reefing line material weight (m/kg). WRL is added to the total weight, WTOT (1846).

The weight of a cluster of η_c parachutes:

$$WTOT_c = \eta_c \, WTOT \qquad (3.113)(1848)$$

3.13.2 Parachute Packing Density and Parachute Volume

Parachutes are either hand packed or pressure packed. A typical hand packing density is 320 kg/m^3, this is the value normally used in parachute volume calculations by the co-operating body. Using a hydraulic press driven by a fluid pump a packing density of 720 kg/m^3 can be attained[9]. Care must be taken not to damage the canopy during pressure packing.

If dpack is the parachute packing density, the parachute volume:

$$VOL = WTOT/dpack \qquad (3.114)(1641)$$

3.13.3 Comparison With Approximate Methods

The weight of a parachute can be estimated from its nominal diameter using figures 8.13 and 8.14 of reference 9. Data are available for flat circular, extended skirt, ringsail, ribbon and ringslot canopies. Two examples of flat circular parachutes have been taken from the results presented in this thesis (chapter 5).

The flat circular parachute designed for the X-RAE 2 remotely piloted vehicle is of 5.7m nominal diameter and the parachute design program gives a weight of 1.4kg. The weight given by figure 8.13 of reference 9 is 1.9kg, 36% higher. The flat circular canopy designed for the Sparrowhawk and Snipe remotely piloted vehicles by the parachute design program has a nominal diameter of 9.2m and a weight of 4.3kg. Figure 8.13 of reference 9 gives a weight of 5.9kg, 37% higher. So this approximate method gives a parachute weight of a third greater than expected and should be used with caution.

3.13.4 Geometric Porosity

Geometric porosity is defined as the ratio of the open canopy area to the total canopy area. This property is important for ribbon canopies because altering the geometric porosity will affect parachute weight and performance. Increasing the geometric porosity increases the canopy stability. The geometric porosity can be changed by altering the number of horizontal ribbons and vertical tapes used in the canopy's construction. This facility is available in the parachute design program. The geometric porosity is calculated for one gore of the canopy by determining the total exposed material area. This is subtracted from the gore area, hence the open area which is expressed as a percentage of the gore area.

From figure 3.11, total gore area, SG:

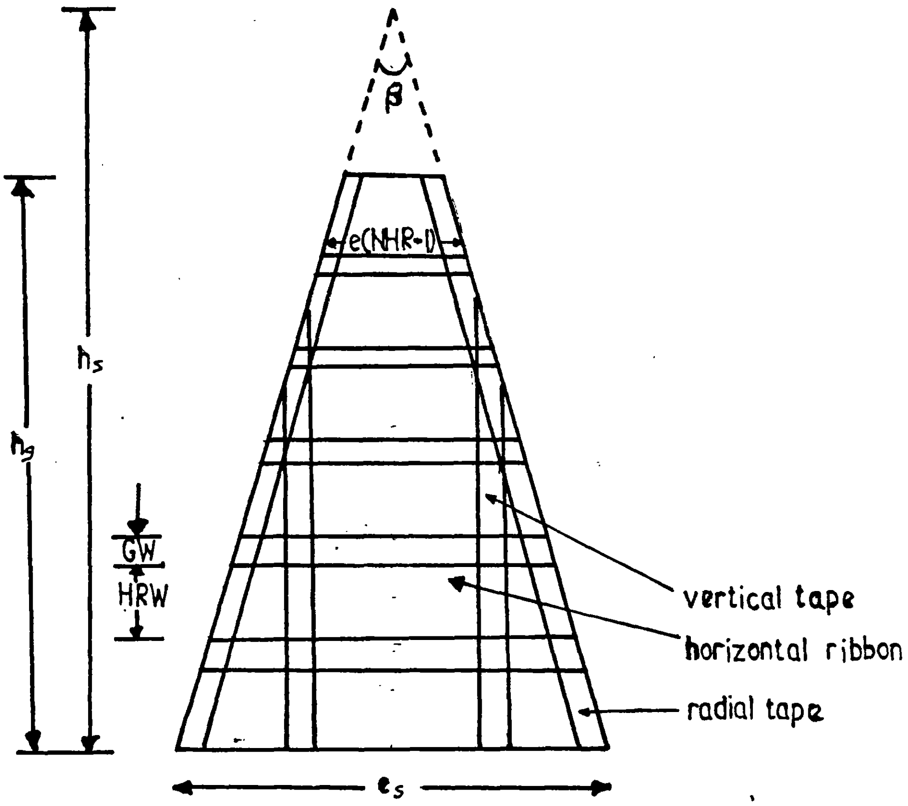


Figure 3.11 Ribbon Parachute Gore Configuration

$$SG = h_g \, ((e_s + e(NHR-1))/2) \qquad (3.115)(1853)$$

The total exposed material area:

$$STR = ((THR - 0.2\ NHR)/N)\ HRW + ((TOT - 0.2\ N)/N)\ TW + ((TVT - N\ NVT\ 0.2)/N)\ VTW - TAPEHR - VTAPEHR \qquad (3.116)(1861)$$

where:

TW - radial tape width
VTW - vertical tape width
TAPEHR - overlap of radial tape on horizontal ribbon
VTAPEHR - overlap of vertical tape on horizontal ribbon and radial tape
TOT - total radial tape length (appendix A)
THR - total horizontal ribbon length (appendix A)
TVT - total vertical tape length (appendix A)

$$TAPEHR = HRW.TW.NHR \qquad (3.117)(1855)$$

if NVT is odd then:

$$VTAPEHR = |((NVT + 1)/2)|\ NHR.HRW.VTW + (NVT - 1)\ (TW/2 - HRW)\ VTW \qquad (3.118)(1857)$$

if NVT is even then:

$$VTAPEHR = (\sum_{x=1}^{|NVT/2|} 2.x)\ (2\ NHR.HRW.VTW/(NVT + 1)) + NVT.VTW\ (TW/2 - HRW) \qquad (3.119)(1859,1860)$$

Then geometric porosity (%):

$$\lambda_g = ((SG - STR)/SG)\ 100 \qquad (3.120)(1862)$$

The effects of the skirt and vent band have been neglected as they are usually overlapped by horizontal ribbons.

3.14 Cost

For reasons of commercial security a detailed cost analysis is not included in this project. However, determination of the total cost of the materials used in the construction of a particular canopy is a simple calculation. If a large number of canopies are required, this unit cost must be kept as low as possible.

If COF is the fabric cost (£/m^2), then the total cost of the fabric in a canopy is:

$$CF = COF.TOF \qquad (3.121)(1301)$$

This calculation is repeated for all components of the system and hence the total unit cost determined (1302-1312).

3.15 Stability

The static stability characteristics in pitch for a descending parachute canopy can be determined from a wind tunnel test in which the component of normal force is measured over a range of angles of attack. The condition for static stability is that at the angle of attack at which the canopy is in equilibrium, i.e. $C_N=0$, that $dC_N/d\alpha$ shall be positive. Any shape of parachute canopy can be made to exhibit static stability by a suitable increase of its porosity.

A rule of thumb for static stability is that formulated at the co-operating body by Lingard, who proposes that for stability the canopy effective porosity (ratio of velocity through the fabric to rate of descent) be >5-6%. This is equivalent to:

$$VFR/(MA.V) > 0.1 \qquad (3.122)(1412)$$

where:

VFR - Volumetric air flow rate through the canopy.
MA - Mouth area.
V - Rate of descent.

The canopy mouth area, MA, can be calculated from the projected, in flight, diameter, D_p, which is obtained from tables 3.11 and 3.12. The volumetric flow rate:

$$VFR = (S_0 - ACU)U + ACU.V \qquad (3.123)(1411)$$

where:

ACU - The total cut out area (drive slots and vent)
U - The air velocity through the canopy fabric.

U can be determined using a relationship between the pressure drop across, and the velocity through, the fabric, quoted by Payne[30]:

$$\Delta P = K_1 U^2 + K_2 U \quad (3.124)$$

where:

ΔP - pressure drop across the fabric

U - fluid velocity through the fabric

K_1, K_2 - constants

To determine the constants K_1 and K_2 for a particular canopy the porosity of the canopy fabric must be known in both U.S. and U.K. units. In the United States fabric porosity (U in $ft^3/(ft^2\ sec)$) is measured at a pressure of 1/2 inch water ($\Delta P = 2.6012\ lb/ft^2$). In the United Kingdom porosity is measured at 10 inches of water ($\Delta P = 52.0236\ lb/ft^2$).

In steady state descent Lingard proposes that:

$$\Delta P = 1/2\ \rho\ V^2 \quad (3.125)(1403)$$

This method can also be used to calculate the stability of ribbon parachutes. U, the velocity through the fabric, is assumed to be zero. The total open area of the ribbon parachute is calculated and VFR obtained from equation 3.123. The stability can then be calculated, as for solid cloth circular parachutes, using equation 3.122.

No stability calculation is available to the program at present for the cruciform and parawing (single keel) parachutes. However cruciform canopies, with appropriate arm ratios, are known from tests to be highly statically stable.

Dynamic stability characteristics in pitch, roll and yaw can be determined by formulating the full equations of motion about these axes. From the solutions to these equations the periods and damping times for these motions are obtained. Various mathematical models exist with which dynamic stability characteristics can be obtained: that developed at Leicester University[31,32] for example, requires as inputs the full aerodynamic and inertial data for the canopy. Since it has been written in Fortran it cannot be included for the present in this parachute design program. However provided the necessary inputs are available it has been used by the author as a supplement to this program.

3.16 Parachute Reliability

A reliability analysis can provide a parachute designer with some useful results. Firstly, the overall reliability of the system, together with the confidence in this figure, is calculated. Secondly, using the method proposed for parachutes in this section, the components of the system which possess the poorest reliability are pinpointed. Hence time spent on improving these components will be of benefit to the system reliability as a whole.

3.16.1 General Definitions

Reliability: an inverse measure of the expected failure rate, i.e. the figure obtained by subtracting the expected failure rate from unity.

Success (and failure): defining success (and failure) of a parachute mission is not simple. One possible statement is "the safe delivery of the store". But what is "safe delivery" if the store is a bomb? The best definition of failure is: "the failure of any portion of the parachute construction which will cause an unsuccessful drop, or a use in which the parachute was improperly packed so that the parachute deployed in such a manner as to cause failure of the system"[33].

Confidence coefficient: represents the fact that not all the manufactured systems are used in the reliability tests. It is good practice to work to 90% confidence, i.e. 90% chance of the true reliability lying between the figure quoted and unity.

Series Components: failure of these components results in failure of the whole system.

Parallel components: more than one component with the same function is present. The successful operation of one of these components will result in the successful operation of the entire system (provided there are no other failures).

3.16.2 Preliminary Considerations

The first stage in a reliability analysis is to obtain a full definition of the system by considering the following points:

(i) Limits of Applicability (system boundaries). e.g. "from deployment to touchdown", would neglect the separation from an aircraft and any ground disconnect mechanism.

(ii) Conditions of Use. e.g. maximum load, maximum deployment speed, etc.

(iii) Atmospheric Conditions (rain, snow, etc.). These are usually neglected but their effects can be included if enough data are available (very unlikely).

(iv) Success (and failure). To be defined as in section 3.16.1.

3.16.3 The Overall System Reliability Method

In this method a number of systems are tested. The failure rate from these tests, with allowance for confidence, is assumed to be the failure rate of all identical systems. If there are F_T failures in N_T trials, using F_T and the desired confidence coefficient, a value A_T is obtained from table I of reference 33. The system reliability, R, is then:

$$R = 1 - A_T / N_T \tag{3.126}$$

at the chosen confidence coefficient.

This method is not suitable for parachute systems because of the large number of tests required.

3.16.4 The Component Reliability Method

This method involves calculating the reliability of each component of the system, and then calculating the system reliability from the reliability of the components and the operational reliability of the combination of components.

$$R = R_{pr} \; R_{c1} \; R_{c2} \; \ldots \; R_{cN} \tag{3.127}$$

where:

R - system reliability

R_{pr} - operational reliability

$R_{c1} \ldots R_{cN}$ - component reliabilities

The first step in the analysis is to separate the parachute system into smaller systems (components). A preliminary analysis is performed to determine the components and sub-components most likely to fail. The other components can be neglected. Generally the parachute components most likely to fail are: risers, bridles, suspension lines, reefing line cutters (appendix B of reference 33 contains much reefing line cutter data) and mechanical disconnect systems. Components that can be ignored (unlikely to fail) are: deployment bags, reefing lines, break cords and radial canopy reinforcing.

The operational term is based on the rate of human error in the parachute packing process. It can be calculated using the method described in section 3.16.3. Appendix A of reference 33 contains some parachute packing reliability data, this is used in the parachute design program for the operational term, R_{pr}.

The material properties of the components must be known. Taking the suspension (or rigging) lines as an example. The line cord breaking strength is assumed to have a normal distribution. Its standard deviation is then:

$$S_x = (\sum_{i=1}^{N_x}(x_i - \bar{x})^2)/(N_x - 1))^{1/2} \tag{3.128}$$

where:

x_i - breaking strength of the ith test.
N_x - number of tests
$\bar{x}$ - mean of the breaking strength

and f_{xx} (= N_x - 1) is defined as the number of degrees of freedom of the tests. Appendix B of reference 33 contains much material data.

It is also assumed that the rigging line loads obtained from parachute drop tests have a normal distribution. Therefore:

$\bar{y}$ - mean of the loads
S_y - standard deviation of the loads
N_y - number of drop tests
and $f_y = N_y - 1$

The set of data, loading or material, with the smallest standard deviation becomes S_1, N_1 and f_1. The other set is S_2, N_2, f_2. Then:

$$zr = (\bar{x} - \bar{y})/S \tag{3.129}$$

where $S = (S_x^2 + S_y^2)^{1/2}$ (3.130)

$$rr = S_1^2/S_2^2 \tag{3.131}$$

$$gr = f_2/f_1 \tag{3.132}$$

$$\theta r = N_2/N_1 \tag{3.133}$$

$$Nr = N_2 \ (1 + rr)/(1 + \theta r.rr) \tag{3.134}$$

$$f = f_2 \; (1 + rr)^2/(1 + gr.rr^2) \tag{3.135}$$

$$Xr = (Nr/f)^{1/2} \; zr \tag{3.136}$$

Using Xr, Kr is calculated from graphs of the non-central t distribution[33,34], at the required confidence coefficient, gr. Then:

$$U/\sigma = Kr \; ((f + 1)/Nr)^{1/2} \tag{3.137}$$

The reliability R_c is then obtained from tables in reference 33, and the system reliability, R, from equation 3.127. The complex mathematical basis for this method is given in appendix C of reference 33.

If no detailed line load data are available the calculated required line load can be used (section 3.10). This load is $\bar{y}$. The standard deviation of the breaking strength of the line material (S_x) = S. The number of materials tested (N_x) = Nr, and f = N_x - 1. The reliability is then calculated using equations 3.129, 3.136 and 3.137.

Tables of the non-central t-distribution[34] are only available at confidence coefficients of 90%, 95% and 99%. A linear Xr versus Kr curve for each value of f can be constructed at each of these confidence coefficients, as shown in reference 33.

In order to obtain the value of Kr at any confidence coefficient two tables (D.5 and D.6) were constructed containing the gradients and constants of these lines. The gradient and constant of the Xr versus Kr line at any confidence coefficient is then obtained by interpolating between the table values.

The exact calculation of the overall confidence coefficient is complex (details in reference 33), so an assumption that the overall confidence coefficient is the product of the component confidence coefficients is made. If an overall confidence coefficient of 90% is required for Nr components, the confidence coefficient of each component should be $(0.9)^{1/Nr}$. So for two components, packing and rigging lines, the confidence of each is 0.9487.

3.16.4.1 Parallel Components and Clusters

A parachute may have two or more reefing line cutters each of reliability $R_{c,gxr}$ where gxr is the confidence coefficient. The reliability of two such components is $(1 - (1 - R_{c,gxr})^2)$ at confidence coefficient gxr. For Nr parallel components the total reliability is:

$$R_b = 1 - (1 - R_{b1,gxr1})(1 - R_{b2,gxr2})\ldots(1 - R_{bN,gxrN}) \qquad (3.138)$$

If a system consisting of a cluster of parachutes can only operate successfully with all the parachutes in the cluster deployed then each canopy is represented by a series term in the model. So in a cluster of η_c parachutes, if R is the reliability of a single parachute and R_{pr} is the packing reliability (operational term), the system reliability is:

$$R_{sys} = (R\, R_{pr})^{\eta c} \qquad (3.139)(1520)$$

If one or more canopies can fail then the probability of failure of q identical canopies, P_r, out of a total of Nr parachutes in the cluster when the probability of failure of a single canopy is pr is:

$$P_r = (Nr!/(q!\ (Nr - q)!))\ pr^q\ (1-pr)^{(Nr-q)} \qquad (3.140)$$

If mr is the maximum number of canopies that can fail without causing the mission to fail then the probability of failure of the entire cluster is:

$$P_d = \sum_{r=mr+1}^{Nr} P_r \qquad (3.141)$$

and system reliability:

$$R_{sys} = (1 - (1 - R\,R_{pr})^{mr+1})(R\,R_{pr})^{\eta C - mr - 1} \qquad (3.142)(1521)$$

Chapter 4

The Parachute Design Program "paradesign"

Using the flow chart for parachute design, figure 3.1, the design methods outlined in chapter 3 were defined by a set of equations and tables of data. Each equation and table was given a unique number dependent on the calculation it refers to, see figure 3.1 for the numbering convention.

Before writing a program of the size of "paradesign" some forward planning is essential, especially when a highly structured language like Pascal is being used. So a structure diagram was first written as described in section 4.1. Using this diagram and the list of equations and tables of data the Pascal code was constructed, organised and tested, as described in section 4.2. Input to and output from the program are described in section 4.3. Section 4.4 contains details of possible alterations and improvements to the program.

4.1 Structure Diagram

A structure diagram for the parachute design program was written using the Warnier-Orr method[35], see appendix B for details of this notation. The problem of parachute design can be represented in Warnier-Orr notation as shown in figure 4.1. It is easily broken down into fifteen smaller problems. In Pascal each of these parts of the parachute design problem are ideal candidates for subroutines. They can be further split up using the Warnier-Orr technique.

This technique also gives the variables required by the program. A listing of the structure diagram is given in appendix C.

$$\text{parachute design} \left\{ \begin{array}{l} \text{select type} \\ \text{calculate area} \\ \text{cluster calculations} \\ \text{calculate line length} \\ \text{calculate number of lines} \\ \text{staging calculations} \\ \text{opening load determination} \\ \text{reefing calculations} \\ \text{stressing calculations} \\ \text{selection of materials} \\ \text{landing control} \\ \text{weight and volume calculations} \\ \text{cost} \\ \text{stability calculation} \\ \text{reliability calculation} \end{array} \right.$$

Figure 4.1 Parachute Design Represented in Warnier-Orr Notation

4.2 Writing, Organising and Testing of the Program

4.2.1 Main Memory Management

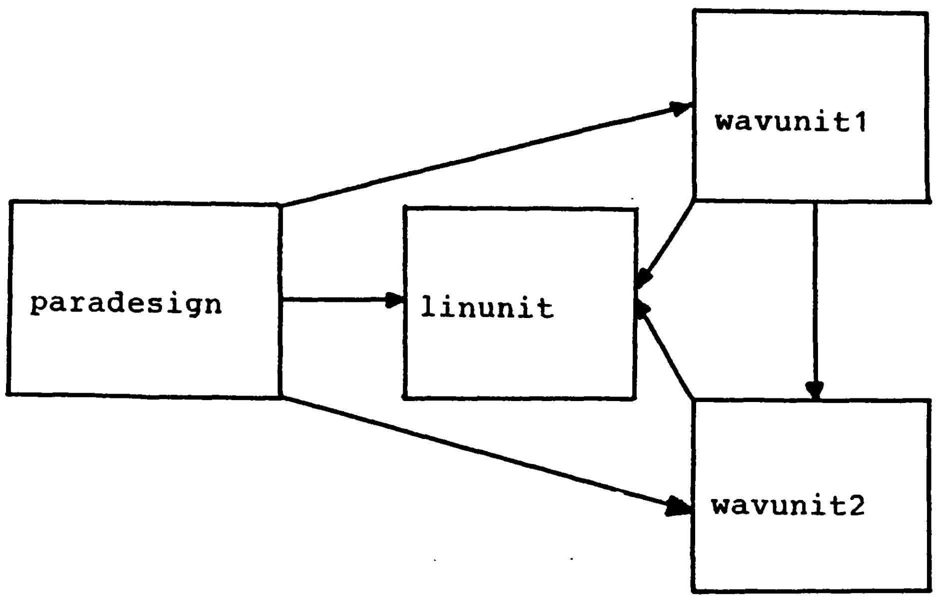
Because of the length of the program and the large number of variables involved, three techniques have been employed in order to use the Sirius's memory economically and to make writing, editing and testing the program as easy as possible. See reference 12 for full details.

(i) Include files. The Pascal code is written in text files which are then compiled to form code files. The program is executed by reading a code file from disk into the computer's memory and then running it. Portions of Pascal code can be kept in separate text files which are known as include files. When compiling a Pascal program an include directive, (*$I filename *), tells the compiler to treat all the program statements in filename as if they were within the text file that contains the include directive. This procedure splits the code up into smaller files which are easier to edit than large ones.

(ii) Segments. UCSD Pascal programs consist of a number of segments of code. Only one segment needs to be contained in the Sirius's memory at once, the others can reside on disk. A Pascal program is a single code segment unless some of its routines have been declared as separate segments. Suitable candidates to be defined as segments are procedures that are only used once in a run of the program - as are many procedures in "paradesign". Some of the procedures in "paradesign" have been declared as segments and hence reside on disk until they are called.

(iii) Units. Separately compiled UCSD units are described in section 2.3. A Pascal version of the "Lingard" opening load method was written before "paradesign" was started and is incorporated in a unit "used" by "paradesign". Two other units containing most of the weight and volume calculations are "used" by the program. These units are called "linunit", "wavunit1" and "wavunit2" respectively.

As explained in section 2.3 the "interface" part of the unit, which contains variable and subroutine declarations is available to the "using" program. So, in order for all the variables and subroutines in "paradesign" to be available to all its program units the units must "use" other units: wavunit2 uses linunit, wavunit1 uses wavunit2 and linunit and paradesign uses all three units. The organisation of the text files, include files and units used in the program is shown in figure 4.2. Arrows indicate the using programs.



paradesign has the following include files: pdesignvr1, pdesignvr2, pdesign1, pdesign2, pdesign3.
linunit contains the include file pjbpasinf1.
So there are ten files of code in total.

Calculations contained in each file:

paradesign	: stability, reliability and printout.
pdesignvr1	: a-s variables.
pdesignvr2	: s-z and greek variables.
pdesign1	: select type, drag coefficient, area, cluster, line length, number of lines, loading and reefing.
pdesign2	: structural strength.
pdesign3	: materials and landing control.
wavunit1	: circular and cruciform parachute weight and volume.
wavunit2	: ringslot, ringsail, disk-gap-band, rotafoil, parawing and ribbon weight and volume.
linunit and pjbpasinf1	: Lingard inflation method.

Figure 4.2 paradesign Text File Organisation

The variables contained in files pdesignvr1 and pdesignvr2 can only be used in the paradesign unit of the program. For a variable to be used in all parts of the program it must be declared in linunit.

4.2.2 Data Table Structure

Each table of data, as listed in appendix D, is contained in a text file on disk. The name of the file refers to the table´s identifying number. For example file #5:t10701.text contains table 10701, opening load data. The first line of the file contains a single integer giving the number of sets of data in the file, this can be represented in Warnier-Orr notation by figure 4.3.

$$\begin{array}{l}\text{data} \\ \text{table}\end{array}\left\{\begin{array}{l} n \\ \\ \text{data set} \\ (n)\end{array}\right.$$

where n is an integer.

Figure 4.3 A Data Table in Warnier-Orr Notation

When it is required the file is accessed by the program and this integer is compared with a constant, m10701 in this example, contained in the program. If the two are not equal there is an error and the program is terminated.

4.2.3 Testing and Checking

The program has been extensively tested during every stage of its development, from the basic structure shown in figure 4.1 to its final state. From reference 35 testing a program can be regarded as consisting of three operations: developing test cases, developing expected results and comparing expected results with actual results. Therefore much of the testing of the parachute design program has involved a comparison of computer results with hand calculations, for each type of parachute included in the program. When completed, a version of the program was given to the co-operating body where it was subjected to further testing, a number of improvements were suggested and made.

The operating system does not check the inputs given from the keyboard to the program. If, by mistake, a character is given where an integer is required the program, and operating system, will abort. To overcome this problem a routine was written to check all inputs to the program. This routine is contained at the beginning of linunit so that it is available to the whole program, see figure 4.2.

4.3 Input and Output

4.3.1 Input Data

The program requires the following input data for each run:

(i) Type of parachute.

(ii) Rate of descent (m/s).

(iii) Store frontal area (m^2). This is especially important for remotely piloted vehicles (RPV's). If the store size is not known a guess ($0.3m^2$ - small store, $1m^2$ - large store) will suffice .

(iv) Mass of store (kg).

(v) Deployment velocity (m/s).

(vi) Deployment altitude (m).

(vii)Allowable mass of parachute (kg).
(viii)Allowable volume of parachute (m^3).
(ix)Maximum shock load (N or g's).
(x)Horizontal velocity, gliding parachutes only (m/s).

Each parachute type requires some unique data as listed in appendix A.

4.3.2 Output

Whilst running the program output is presented to the user via the screen. After all the calculations have been completed the user is given an option to create a datafile of results which can be printed out. The output given by the program could be sent to a printer, however this procedure makes interaction with the program difficult as no output then appears on the Sirius screen.

4.4 Alteration and Development of the Program

4.4.1 Equations and Tables

All the equations used have been numbered in the program text files, see the convention in figure 3.1, so that if a change is required the equation in question can readily be accessed using the UCSD editor. New calculation methods for parachute design problems can be introduced by removing the old equations and inserting new ones, if new variables are required they must be declared in the program.

Additions and alterations to the tables of data can easily be made. The integer contained in the first line of each file, and its corresponding constant in the program, as described in section 4.2.2, must be changed if the number of sets of data in the file is changed. An expansion of the tables of materials data is expected. The introduction of new data tables can be accomplished if the conventions listed in this section are followed.

4.4.2 General Development

The UCSD operating system only requires 256k of the Sirius's 384k of memory, the rest is not used. About 128k of memory is available to the program and in its present state the program is stretching the Sirius's memory to the limit. The length of each unit of the program code, in 512 byte blocks is:

linunit	40	blocks
wavunit1	41	"
wavunit2	18	"
paradesign	117	"

this is a total of 216 blocks or 110k bytes. Not all the code is contained in the memory at once, but this 110k does not include the storage of the variables.

Further expansion of the "paradesign" part of the program will involve use of another unit as this part of the program is too large to be contained in one unit of code. Disk space to store another unit is scarce due to the increasing presence of the datafiles. This means that a major expansion of the program will not be possible unless the program text files are stored on a separate disk, which in turn would make alteration and development much more complicated.

For these reasons major expansions to the program are not recommended unless some compensation can be gained by the possible removal of unwanted parts of code (calculations or types of parachute).

Chapter 5

Results and Discussion

In order to demonstrate the possible use of the 'paradesign' program in a parachute company it has been employed to specify a selection of parachute systems from typical input data. The output from these computer solutions is presented in this chapter.

Input data for runs of the program have been collected from various sources. The most important input data is that received by a parachute company in the form of a specification. The parachute company takes these data to design a proposed system, and a formal proposal is made in the form of a published document. Four of these documents have been obtained and used for the basis of 'paradesign' runs. The input data taken from each proposal has been used for computer solutions with different parachute canopies (cruciform, flat circular, extended skirt and hemispherical) and/or design methods. The results from these runs can provide the design engineer with a variety of possible systems to satisfy the initial specifications. He can easily make comparisons using the printed output given by the program, and from these possibilities he can choose the one most suitable for use depending on his main required criteria: cost, stability, ease of manufacture, weight, etc. (or a combination of these). In this way the program is a time-saving device; repeating the design calculations by hand for a number of parachutes is a tedious and time consuming task.

The requirements for an airborne forces parachute have been obtained from the Royal Aircraft Establishment. These data have been used to demonstrate reefing and staging. The staging and reefing routines contained in the program have been taken from the Recovery Systems Design Guide[9], as described in chapter 3. Examples of these techniques are given in this reference, but the author has discovered some errors in these calculations. For example the sine of -90 degrees has been taken to be 1.0 instead of -1.0, and 351 divided by 7017 is given as 0.0421 instead of 0.05. An example of a cluster is taken from reference 9 in order to demonstrate the cluster routine contained in the program, which is described fully in chapter 3. Some data were assembled to demonstrate ribbon parachute design.

A gliding parachute has been designed by the program using requirements for an emergency escape parachute. The canopy obtained is compared with that used in practice (the G.Q. 6.2m flying diameter Aeroconical canopy).

5.1 A Parachute Canopy for the X-RAE2 Remotely Piloted Vehicle

A design proposal based on a 4:1 arm ratio cruciform canopy has been specified for the recovery of this remotely piloted vehicle[36]. The specification is given in table 5.1.

Using this input data three different computer solutions have been formulated. The output is listed in table 5.2.

(i) Cruciform parachute. This computer solution is almost identical to the proposal made by the parachute company. The required tape and line strengths calculated using the computer model are greater than those in the proposal because the angle the rigging lines make with the vertical has been taken into account, this the parachute company failed to do. However the same materials are used throughout by the computer solution as were used in the proposal, so this discrepancy has no effect on the canopy's weight and volume. The fabric weight obtained is 5.6% less using the computer method, due to a difference in the allowance made for sewing. Since the imporous 4:1 arm ratio cruciform canopy is known to be highly statically stable the stability required will be obtained using this parachute. The cruciform is also the easiest parachute to manufacture.

(ii)Flat circular parachute. This parachute is more difficult to manufacture than the cruciform canopy, 32 fabric panels are required for the former as opposed to 12 for the latter. The same materials are used as for the cruciform canopy. The canopy weight was found to be 0.7% less than than that of the cruciform, and this parachute may be preferable as a low weight is required. The disadvantage of using a flat circular canopy is that a large vent (4.5% of the nominal area (S_0)) is needed to ensure the parachute has the required stability. This vent may increase the rate of descent to an unacceptable value since it will affect the canopy drag coefficient. To combat this a drag coefficient 5% less than that recommended by the program was selected, hence ensuring a high area which in turn would give a lower rate of descent if the vent were not present. Also use of a vent of this size (22% of the parachute nominal diameter D_0) will severly restrict the parachute's opening. A better way to stabilise the canopy is to use a highly porous material, at present no highly porous wide material is available to the program. Parachutes of this kind are often stabilised by having a crown of low porosity material with high porosity elsewhere.

(iii)Conical parachute. The materials used for this canopy are the same as those used for the flat circular canopy, but due to its construction the conical parachute weighs less, i.e. it is more drag efficient than the flat circular canopy which is generally regarded as obsolete. The opening load has been calculated using the mass ratio method because the C_F-τ data required for the 'Lingard' method is not available for the conical parachute. However the result obtained is very similar to the opening load obtained using the 'Lingard' method for the flat circular parachute. Stability of this parachute is ensured by the use of a large vent (5% of the nominal area (S_0)). As with the flat circular canopy the use of this vent may increase the rate of descent to an unacceptable value, the initial drag coefficient has been reduced as for the flat circular. The size of the vent could be reduced if a more porous fabric were used, but for consistency with the original proposal the same fabric has been employed for all three systems. A better way to stabilise this parachute is the inclusion of symmetrical slots. The canopy would then be an aeroconical with zero drive. A canopy with a 5% of S_0 vent would probably never open.

Rate of Descent	6 m/s
Mass of Vehicle	40 kg
Stability	$\pm 5^{\circ}$
Deployment Speed - Normal	50 KEAS
Deployment Speed - Power Drive	150 KEAS
Allowable Volume	$0.0062 m^3$

In addition the parachute recovery system must be as light as possible.

Table 5.1 Specification for the X-RAE2 RPV

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	4:1 Cruciform	Flat Circular	Conical	4:1 Cruciform
Main Dimensions				
Nominal Area (m^2)	26.57	25.51	23.91	26.57
Nominal Diameter (m)	5.82	5.70	5.52	5.82
Drag Coefficient	0.72	0.75	0.80	0.72
Line Length (m)	7.79	5.70	5.52	7.80
Number of Lines	12	16	16	12
Opening Load (N)	4476	5390	5592	4500
Opening Load Method	Lingard	Lingard	Mass Ratio	Lingard
Strength Calculations				
Safety Factor	1.5	1.5	1.5	1.5
Fabric Strength ((N/mm)*50)	73.23	34.21	34.56	-
Skirt Band Strength (N)	-	481.76	503.69	-
Vent Band Strength (N)	-	481.76	503.69	-
Tape Strength (N)	521.11	481.76	503.69	507
Rigging Line Strength (N)	579.01	535.29	559.66	563
Vent Line Strength (N)	-	535.29	559.66	-
Fabric				
Specification	GQ MS 309	GQ MS 309	GQ MS 309	GQ MS 309
Strength ((N/mm)*50)	400	400	400	400
Weight (gm/m^2)	39	39	39	39
Width (m)	1.22	1.22	1.22	1.22

Table 5.2 Design Solution for the X-RAE2 RPV Canopy

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	4:1 Cruciform	Flat Circular	Conical	4:1 Cruciform
Fabric (continued)				
Porosity at 10 in H_2O ($ft^3/ft^2 sec$)	0	0	0	0
Porosity at 1/2 in H_2O ($ft^3/ft^2 sec$)	0	0	0	0
Reserve Factor	5.5	11.7	11.6	-
Skirt Band				
Specification	-	IAC S/15	IAC S/15	-
Strength (N)	-	670	670	-
Weight (gm/m)	-	2.6	2.6	-
Width (mm)	-	15	15	-
Reserve Factor	-	1.4	1.3	-
Vent Band				
Specification	-	IAC S/15	IAC S/15	-
Strength (N)	-	670	670	-
Weight (gm/m)	-	2.6	2.6	-
Width (mm)	-	15	15	-
Reserve Factor	-	1.4	1.3	-
Tape				
Specification	IAC S/15	IAC S/15	IAC S/15	IAC S/15
Strength (N)	670	670	670	670
Weight (gm/m)	2.6	2.6	2.6	2.6
Width (mm)	15	15	15	15
Reserve Factor	1.3	1.4	1.3	1.3
Rigging Lines				
Part No.	P00107 213 1	P00107 213 1	P00107 213 1	P00107 213 1
Specification	DTD 5620 SB603	DTD 5620 SB603	DTD 5620 SB603	DTD 5620 SB603
Strength (N)	670	670	670	670
Weight (m/kg)	588.2	588.2	588.2	588.2

Table 5.2 (continued)

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	4:1 Cruciform	Flat Circular	Conical	4:1 Cruciform
Rigging Lines (continued)				
Reserve Factor	1.2	1.3	1.2	1.2
Vent Lines				
Part No.	-	P00107 213 1	P00107 213 1	-
Specification	-	DTD 5620 SB603	DTD 5620 SB603	-
Strength (N)	-	670	670	-
Weight (m/kg)	-	588.2	588.2	-
Reserve Factor	-	1.3	1.2	-
Construction Details				
Vent Area (% of So)	-	4.5	5	-
Gore Width at Vent (m)	-	0.24	0.24	-
Maximum Gore Width (m)	-	1.13	1.06	-
Gore Height (m)	-	2.23	2.20	-
Cone Angle (degrees)	-	-	20	-
Number of Panels per Gore	-	2	2	-
Arm Span (m)	7.79	-	-	7.80
Arm Width (m)	1.95	-	-	1.95
Material Weights (kg)				
Fabric	1.133	1.065	0.998	1.188
Skirt Band	-	0.047	0.044	-
Vent Band	-	0.010	0.010	-
Tapes	0.147	0.103	0.101	0.148
Lines	0.163	0.160	0.156	0.164
Vent Lines	-	0.022	0.023	-

Table 5.2 (continued)

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	4:1 Cruciform	Flat Circular	Conical	4:1 Cruciform
Material Weights (kg) (continued)				
Total	1.42	1.41	1.33	1.50
Volume (m^3)	0.004	0.004	0.004	0.005
Packing Density (kg/m^3)	320	320	320	320
Stability	Assumed Stable	Calculated Stable	Calculated Stable	Assumed Stable

Table 5.2 (continued)

5.2 A Parachute System for the Sparrowhawk and Snipe Mk.3 Remotely Piloted Vehicles

A proposal based on a 3.5:1 arm ratio cruciform canopy has been specified for the recovery of both the Snipe Mk.3 and Sparrowhawk remotely piloted vehicles[37]. A summary of the specification is given in table 5.3.

Using this input data the parachute design program was used to give three computer solutions to this problem.

(i) Cruciform parachute. There are differences between the canopy proposed by the parachute manufacturer and the computer solution in the calculations of area and opening load. In the manufacturer's proposal the nominal area of the canopy has been calculated using a different method to that recommended in section 3.3. The area of the vehicle being delivered has been taken into account by subtracting its drag area from the canopy nominal area rather than the canopy drag area; which is the procedure followed in the program. Hence the nominal area calculated by the program is 1% less than that given in the proposal. The opening load calculated by the program is 22% greater than that given in the proposal. This may be partly due to the smaller canopy. It could also be due to the value of τ_o - the dimensionless time at the start of inflation. It is possible that this value was different in the proposal load calculation method and the computer program load calculation method, as τ_o is known to change with the size of parachute canopy. These differences have little effect on the weight and volume as the same materials are used for the fabric and the lines in the computer model as were used in the proposal; slightly heavier material being used for the tapes. This makes the total calculated volume 3.5% larger than that in the proposal but comfortably within the requirements.

(ii)Flat circular parachute. The same materials are used for this canopy as for the proposed canopy. Due to construction differences the weight and volume of the flat circular canopy are less than that of the cruciform canopy, this may make it preferable as the weight is specified to be as low as possible. However the stability of this parachute cannot be calculated as porosity data are only available in U.K. units for the fabric used (impression N8726), porosity data in both U.K. and U.S. units are required for a stability calculation to be made (section 3.15). Comparing results from the X-RAE 2 RPV flat circular canopy (section 5.1) it is unlikely that this parachute will be stable without the use of cut-outs or a highly porous fabric used in all or part of the canopy.

(iii)Hemispherical parachute. The stability problem mentioned for the flat circular canopy is solved by using this type of parachute. A porous fabric is used, and with a vent of 3.5% of the constructed area the parachute is calculated to be stable. The opening load calculated for this parachute is less than that for the flat circular canopy because a different method has been used, data not being available for the 'Lingard' method. However the load obtained using the mass ratio method is very similar to the load given using the 'Lingard' method for the cruciform parachute in the original proposal. The weight and volume obtained using this canopy are the lowest of the parachutes investigated and hence this type is preferable. The disadvantage of the hemispherical is that, being shaped, it is more complicated to manufacture than either the cruciform or flat circular parachutes.

Rate of Descent	5 m/s
Mass of Vehicle	81.6 kg
Stability	$\pm 5^{o}$
Glide Ratio	Nil
Recovery Velocity - Maximum	174 Knots
Recovery Velocity - Minimum	35 Knots
Maximum Volume	$0.022m^3$

Table 5.3 Specification for the Sparrowhawk and Snipe Mk.3 RPVs

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	3.5:1 Cruciform	Flat Circular	Hemispherical	3.5:1 Cruciform
Main Dimensions				
Nominal Area (m^2)	75.92	66.43	69.02	76.62
Nominal Diameter (m)	9.83	9.20	9.37	9.88
Drag Coefficient	0.70	0.80	0.77	0.70
Line Length (m)	12.49	9.20	9.37	12.51
Number of Lines	16	24	24	16
Opening Load (N)	9574	10956	7834	7800
Opening Load Method	Lingard	Lingard	Mass Ratio	Lingard
Strength Calculations				
Safety Factor	2.0	2.0	2.0	2.0
Fabric Strength ((N/mm)*50)	130.74	57.46	40.31	106.00
Skirt Band Strength (N)	-	870.49	622.44	-
Vent Band Strength (N)	-	870.49	622.44	-
Tape Strength (N)	1119.20	870.49	622.44	877.50
Rigging Line Strength (N)	1243.56	967.21	691.60	975.00
Vent Line Strength (N)	-	967.21	691.60	-
Fabric				
Part Number	-	-	P00115 303 4	-
Specification	N 8726	N 8726	MIL C 7020 Type 1	N 8726
Strength ((N/mm)*50)	400	400	370	400
Weight (gm/m^2)	40	40	37	40

Table 5.4 Design Solution for the Sparrowhawk and Snipe Canopy

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	3.5:1 Cruciform	Flat Circular	Hemispherical	3.5:1 Cruciform
Fabric (continued)				
Width (m)	1.42	1.42	0.950	1.42
Porosity at 10 in H_2O (ft^3/ft^2 sec)	-	-	11.00	-
Porosity at 1/2 in H_2O (ft^3/ft^2 sec)	-	-	1.33	-
Reserve Factor	3.1	7.0	9.2	3.8
Skirt Band				
Part Number	-	P00167 050 7	-	-
Specification	-	MIL T 5038 Type 3	IAC S/15	-
Strength (N)	-	890	670	-
Weight (gm/m)	-	3.7	2.6	-
Width (mm)	-	9.5	15	-
Reserve Factor	-	1.0	1.1	-
Vent Band				
Part Number	-	P00167 050 7	-	-
Specification	-	MIL T 5038 Type 3	IAC S/15	-
Strength (N)	-	890	670	-
Weight (gm/m)	-	3.7	2.6	-
Width (mm)	-	9.5	15	-
Reserve Factor	-	1.0	1.1	-
Tape				
Part Number	P00167 591 3	p00167 050 7	-	p00167 050 7
Specification	MIL T 5038 Type 3	MIL T 5038 Type 3	IAC S/15	MIL T 5038 Type 3
Strength (N)	1112	890	670	890
Weight (gm/m)	4.7	3.7	2.6	3.7
Width (mm)	13	9.5	15	9.5
Reserve Factor	1.0	1.0	1.1	1.0

Table 5.4 (continued)

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	3.5:1 Cruciform	Flat Circular	Hemispherical	3.5:1 Cruciform
Rigging Lines				
Part No.	P00107 185 0	P00107 185 0	P00107 185 0	P00107 185 0
Specification	DTD 5620 CA103	DTD 5620 CA103	DTD 5620 CA103	DTD 5620 CA103
Strength (N)	1350	1350	1350	1350
Weight (m/kg)	270.0	270.0	270.0	270.0
Reserve Factor	1.1	1.4	2.0	1.4
Vent Lines				
Part No.	-	P00107 185 0	P00107 185 0	-
Specification	-	DTD 5620 CA103	DTD 5620 CA103	-
Strength (N)	-	1350	1350	-
Weight (m/kg)	-	270.0	270.0	-
Reserve Factor	-	1.4	2.0	-
Construction Details				
Vent Area (% of So)	-	2	3.5	-
Gore Width at Vent (m)	-	0.17	0.16	-
Maximum Gore Width (m)	-	1.21	0.86	-
Gore Height (m)	-	3.94	4.59	-
Number of Panels per Gore	-	3	6	-
Arm Span (m)	12.45	-	-	12.51
Arm Width (m)	3.56	-	-	3.57
Material Weights (kg)				
Fabric	3.217	2.899	2.432	3.171
Skirt Band	-	0.108	0.054	-
Vent Band	-	0.016	0.011	-
Tapes	0.546	0.370	0.299	0.433
Lines	0.752	0.835	0.851	0.755

Table 5.4 (continued)

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	3.5:1 Cruciform	Flat Circular	Hemispherical	3.5:1 Cruciform
Material Weights (kg) (continued)				
Vent Lines	-	0.076	0.073	-
Total	4.51	4.30	3.72	4.36
Volume (m^3)	0.014	0.014	0.012	0.014
Packing Density (kg/m^3)	320	320	320	320
Stability	Assumed Stable	-	Calculated Stable	Assumed Stable
Reliability at 90% Confidence	0.9949	0.9949	0.9949	-

Table 5.4 (Continued)

5.3 A Mail Dropping Canopy for the Royal Netherlands Navy

A design proposal based on a conical canopy has been specified for a mail dropping system[38]. The design requirements are given in table 5.5.

Using this design requirement data three different computer solutions have been made. The output is listed in table 5.6.

(i) Conical Parachute. The proposal and the computer solution use different stressing methods in the calculation of both fabric and line strengths. In the fabric strength calculation a factor of 1/2.15 has been used in the proposal as a method derived from Johns[1] has been employed. A factor of $1/\pi$ (the normal factor for circular parachutes, see section 3.10.2) has been used in the computer program. This results in the computer program giving a required fabric strength 27% lower than that in the proposal. In the calculation of line strength the computer model takes into account the angle the lines make with the vertical, which is not done in the proposal, and hence a 12% higher required line (and tape) strength is given in the computer solution. The fabric and line materials are the same as used for the proposed canopy. The lines in the computer solution have a reserve factor of 0.9, but when this is compared with the high safety factor of 2.0, for an unmanned application, this reserve factor is acceptable. Tape GQ MS 158 replaces GQ MS 193 for the skirt band, vent band and radial tapes, this is a heavier material. The fabric weight is calculated to be 14% higher by the computer program because a larger width is allowed for sewing (0.025m as opposed to 0.02m in the proposal). This, the heavier tape, and inclusion of the vent line weight, makes the total parachute weight 2.62kg (the weight is 2.42kg in the proposal). Adding in the mass of the parachute sleeve and connecting line, the weight becomes 3.02kg, and at a packing density of $320kg/m^3$ the volume is $0.0094m^3$, well within the $0.0118m^3$ maximum. The parachute is calculated to be unstable but no stability requirement has been made in the specification. If the vent were enlarged sufficiently a stable parachute would ensue.

(ii) Cruciform Parachute. When the mass of the parachute sleeve and connecting line is included the volume of this parachute is 7% over that required. This is due to the high opening load (24717N) calculated using the mass ratio method. The 'Lingard' method gives an opening load of 14184N, a decrease of 43%, this will be different if the τ_o, dimensionless inflation time, value was to be changed. The Pflanz method gives a load of 19427N, a decrease of 21%. If the lower load were used in the calculation the parachute volume would be within that required. However the mass ratio method was used to be consistent with the original proposal, and it is good practice to use the highest load obtained. The advantages of a 3.5:1 cruciform parachute are that it is easy to manufacture and is highly statically stable.

(iii) Extended Skirt 10% Flat. This canopy is similar to the conical canopy. The same materials are used for this canopy as in the original conical canopy proposed, apart from the fabric. Fabric MIL C 7020 Type 1 is used in preference to George Harris B1 as it is much lighter and more porous. As for the flat circular canopy the skirt band, vent band, tapes and lines have reserve factors of 0.9 but this is acceptable with the safety factor used (2.0). The weight and volume of this parachute are less than the other two considered (13% less than the conical canopy) and this makes the extended skirt canopy preferable for use for this application. Its manufacture is similar in difficulty to the conical canopy and, like the conical canopy, it is unstable.

Rate of Descent	9 m/s
Mass of Store	72.2 kg maximum
	37.2 kg minimum
Deployment Speed	200 knots
Deployment Altitude	92 m (300 ft)
Maximum Deployment Load	45 kN
Volume Available	0.0118 m^3
Attachment	4 Point

Also the mass of the parachute sleeve and connecting line, 0.4kg, must be added to the total parachute mass.

Table 5.5 Design Requirement Data for a Mail Dropping Parachute

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	Conical	3.5:1 Cruciform	Extended Skirt 10% Flat	Conical
Main Dimensions				
Nominal Area (m^2)	18.66	21.33	19.06	18.66
Nominal Diameter (m)	4.87	5.21	4.93	4.87
Drag Coefficient	0.80	0.70	0.78	0.80
Line Length (m)	4.87	6.59	4.93	4.87
Number of Lines	16	16	16	16
Opening Load (N)	24717	24717	24717	23100
Opening Load Method	Mass Ratio	Mass Ratio	Mass Ratio	Mass Ratio
Strength Calculations				
Safety Factor	2.0	2.0	2.0	2.0
Fabric Strength ((N/mm)*50)	230.58	636.78	234.86	315.00
Skirt Band Strength (N)	2968.43	-	2956.83	2599.00
Vent Band Strength (N)	2968.43	-	2956.83	2599.00
Tape Strength (N)	2968.43	2890.24	2956.83	2599.00
Rigging Line Strength (N)	3298.26	3211.38	3285.37	2887.00
Vent Line Strength (N)	3298.26	-	3285.37	-
Fabric				
Part Number	-	P00115 325 5	P00115 303 4	-
Specification	GQ MS 502 (B1)	GQ MS 330	MIL C 7020 Type 1	GQ MS 502 (B1)
Strength ((N/mm)*50)	480	950	370	480

Table 5.6 Design Solutions for a Mail Dropping Canopy

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	Conical	3.5:1 Cruciform	Extended Skirt 10% Flat	Conical
Fabric (continued)				
Weight (gm/m^2)	54	85	37	54
Width (m)	1.17	1.22	0.95	1.17
Porosity at 10 in H_2O (ft^3/ft^{2}sec)	0	3.0	11.0	0
Porosity at 1/2 in H_2O (ft^3/ft^{2}sec)	0	0.2	1.3	0
Reserve Factor	2.1	1.5	1.6	1.5
Skirt Band				
Part Number	P00167 575 1	-	P00167 620 2	P00167 620 2
Specification	GQ MS 158	-	GQ MS 193	GQ MS 193
Strength (N)	3115	-	2670	2670
Weight (gm/m)	10.3	-	8.7	8.7
Width (mm)	12.5	-	8.0	8.0
Reserve Factor	1.0	-	0.9	1.0
Vent Band				
Part Number	P00167 575 1	-	P00167 620 2	P00167 620 2
Specification	GQ MS 158	-	GQ MS 193	GQ MS 193
Strength (N)	3115	-	2670	2670
Weight (gm/m)	10.3	-	8.7	8.7
Width (mm)	12.5	-	8.0	8.0
Reserve Factor	1.0	-	0.9	1.0
Reinforcing				
Part Number	P00167 620 2	-	P00167 620 2	P00167 620 2
Specification	GQ MS 193	-	GQ MS 193	GQ MS 193
Strength (N)	2670	-	2670	2670
Weight (gm/m)	8.7	-	8.7	8.7
Width (mm)	8.0	-	8.0	8.0
Reserve Factor	1.5	-	1.5	1.0

Table 5.6 (continued)

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	Conical	3.5:1 Cruciform	Extended Skirt 10% Flat	Conical
Tape				
Part Number	P00167 575 1	P00167 620 2	P00167 620 2	P00167 620 2
Specification	GQ MS 158	GQ MS 193	GQ MS 193	GQ MS 193
Strength (N)	3115	2670	2670	2670
Weight (gm/m)	10.3	8.7	8.7	8.7
Width (mm)	12.5	8.0	8.0	8.0
Reserve Factor	1.0	0.9	0.9	1.0
Rigging Lines				
Specification	DTD 5620 CA106	DTD 5620 CA106	DTD 5620 CA106	DTD 5620 CA106
Strength (N)	3100	3100	3100	3100
Weight (m/kg)	101.0	101.0	101.0	101.0
Reserve Factor	0.9	1.0	0.9	1.1
Vent Lines				
Specification	DTD 5620 CA106	-	DTD 5620 CA106	-
Strength (N)	3100	-	3100	-
Weight (m/kg)	101.0	-	101.0	-
Reserve Factor	0.9	-	0.9	-
Construction Details				
Vent Area (% of So)	1.5	-	1.5	1.5
Gore Width at Vent (m)	0.11	-	0.12	0.11
Maximum Gore Width (m)	0.93	-	0.68	0.93
Gore Height (m)	2.19	-	1.80	2.26
Cone Angle (degrees)	20	-	-	20
Number of Panels per Gore	2	-	3	2
Arm Span (m)	-	6.60	-	-
Arm Width (m)	-	1.89	-	-

<u>Table 5.6 (continued)</u>

	Possible Computer Solutions			Manufacturer's Proposal
Type of Canopy	Conical	3.5:1 Cruciform	Extended Skirt 10% Flat	Conical
Material Weights (kg)				
Fabric	1.137	2.018	0.817	0.994
Skirt Band	0.156	-	0.096	-
Vent Band	0.021	-	0.018	-
Reinforcing	0.018	-	0.118	-
Tapes	0.401	0.546	0.343	0.625
Lines	0.804	1.076	0.812	0.804
Vent Lines	0.081	-	0.080	-
Total	2.62	3.64	2.28	2.42
Volume (m^3)	0.008	0.011	0.007	0.008
Packing Density (kg/m^3)	320	320	320	320
Stability	Calculated Unstable	Assumed Stable	Calculated Unstable	-

Table 5.6 (continued)

5.4 A Parachute Canopy for the Plessey Marine SSQ 954 Sonobuoy

A design proposal based on either a 3:1 arm ratio cruciform canopy or a square parasheet canopy has been specified for the Plessey SSQ sonobuoy[39]. A summary of the specification is given in table 5.7.

Using this data two different computer solutions have been formulated. The output is listed in table 5.8.

(i) Cruciform Parachute. The stressing method used in the manufacturer's proposal is different from that used in the computer program. The required fabric strength has been halved because the proposal assumed that the fabric was double thickness in the crown area. Single thickness fabric was assumed in the computer model, and hence a stronger (and heavier) fabric is used. The required tape strength in the computer solution is about half that given in the proposal. This is because the tape used for parachute reinforcing has also been used for the rigging lines on the proposed cruciform canopy. A similar tape material has been used in the computer program and the proposal. However cord has been used for the lines in the computer prediction, hence the program gives a higher weight and volume than the manufacturer's proposal. Adding in the weight of the parachute sock the total mass is 0.039kg and the volume 111cm^3, which is within the allowable of 117cm^3. The cruciform parachute is known to be highly statically stable and its ease of manufacture makes it suitable for these smaller sized parachutes.

(ii) Hemispherical Parachute. A lower opening load was calculated for this parachute because the mass ratio method was used, hemispherical parachute data not being available for the Pflanz method. The same materials were used for this canopy as for the proposed cruciform canopy. The total parachute weight is 0.043kg,and the volume 123cm^3. This is greater than the allowable. Pressure packing to 367kg/m^3 would be required to enable the volume of this canopy to be within the requirements. The parachute is stable with a small vent (1% of S_0). The same weight and volume problem would occur if other solid cloth circular parachutes (flat, conical, extended skirt, etc.) were to be used. However this canopy is so small (less than 1/2 metre diameter) that manufacture of a shaped, hemispherical or extended skirt, parachute would be very difficult.

Store Mass	8.16 kg
Rate of Descent	37.2 m/s
Deployment Speed	250 knots
Maximum Volume	117.1 cm^3
Parachute Sock Weight	0.003 kg

Table 5.7 Specification for the Plessey SSQ 954 Canopy

	Possible Computer Solution		Manufacturer's Proposal	
Type of Canopy	3:1 Cruciform	Hemispherical	3:1 Cruciform	Square Parasheet
Main Dimensions				
Nominal Area (m^2)	0.13	0.15	0.13	0.14
Nominal Diameter (m)	0.41	0.44	0.41	0.43
Drag Coefficient	0.70	0.62	0.70	0.66
Line Length (m)	0.49	0.44	-	-
Number of Lines	8	8	-	-
Opening Load (N)	1647	968	1647	1647
Opening Load Method	Pflanz	Mass Ratio	Pflanz	Pflanz
Strength Calculations				
Safety Factor	1.9	1.9	1.9	1.9
Fabric Strength ((N/mm)*50)	539.72	100.71	270.00	259.25
Skirt Band Strength (N)	-	219.20	-	-
Vent Band Strength (N)	-	219.20	-	-
Tape Strength (N)	367.82	219.20	843.00	843.00
Rigging Line Strength (N)	408.69	243.56	-	-
Vent Line Strength (N)	-	243.56	-	-
Fabric				
Part Number	P00115 155 5	P00115 303 4	P00115 303 4	P00115 303 4
Specification	BSF 118/556A	MIL C 7020 Type 1	MIL C 7020 Type 1	MIL C 7020 Type 1
Strength ((N/mm)*50)	510	370	370	370
Weight (gm/m^2)	50	37	37	37

Table 5.8 Design Solution for the Plessey SSO 954 Canopy

	Possible Computer Solution		Manufacturer's Proposal	
Type of Canopy	3:1 Cruciform	Hemispherical	3:1 Cruciform	Square Parasheet
Fabric (continued)				
Width (m)	0.920	0.95	0.95	0.95
Porosity at 10 in H_2O ($ft^3/ft^2 sec$)	10	11.0	11.0	11.0
Porosity at 1/2 in H_2O ($ft^3/ft^2 sec$)	-	1.3	1.3	1.3
Reserve Factor	0.9	3.7	1.4	1.4
Skirt Band				
Part Number	-	P00167 050 7	-	-
Specification	-	MIL T 5038 Type 3	-	-
Strength (N)	-	890	-	-
Weight (gm/m)	-	3.7	-	-
Width (mm)	-	9.5	-	-
Reserve Factor	-	4.1	-	-
Vent Band				
Part Number	-	P00167 050 7	-	-
Specification	-	MIL T 5038 Type 3	-	-
Strength (N)	-	890	-	-
Weight (gm/m)	-	3.7	-	-
Width (mm)	-	9.5	-	-
Reserve Factor	-	4.1	-	-
Tape				
Part Number	P00167 050 7	P00167 050 7	-	-
Specification	MIL T 5038 Type 3	MIL T 5038 Type 3	R 807	R 807
Strength (N)	890	890	890	890
Weight (gm/m)	3.7	3.7	3.7	3.7
Width (mm)	9.5	9.5	-	-
Reserve Factor	2.4	4.1	1.1	1.1

Table 5.8 (continued)

	Possible Computer Solution		Manufacturer's Proposal	
Type of Canopy	3:1 Cruciform	Hemispherical	3:1 Cruciform	Square Parasheet
Rigging Lines				
Part No.	P00107 213 1 .	P00107 213 1	-	-
Specification	DTD 5620 SB603	DTD 5620 SB603	-	-
Strength (N)	670	670	-	-
Weight (m/kg)	588.2	588.2	-	-
Reserve Factor	1.6	2.8	-	-
Vent Lines				
Part No.	-	P00107 213 1	-	-
Specification	-	DTD 5620 SB603	-	-
Strength (N)	-	670	-	-
Weight (m/kg)	-	588.2	-	-
Reserve Factor	-	2.8	-	-
Construction Details				
Vent Area (% of So)	-	1	-	-
Gore Width at Vent (m)	-	0.01	-	-
Maximum Gore Width (m)	-	0.10	-	-
Gore Height (m)	-	0.23	-	-
Number of Panels per Gore	-	1	-	-
Arm Span (m)	0.49	-	-	-
Arm Width (m)	0.16	-	-	-
Material Weights (kg)				
Fabric	0.011	0.010	0.006	0.005
Skirt Band	-	0.004	-	-
Vent Band	-	0.001	-	-
Tapes	0.016	0.013	0.023	0.014

Table 5.8 (continued)

	Possible Computer Solution		Manufacturer's Proposal	
Type of Canopy	3:1 Cruciform	Hemispherical	3:1 Cruciform	Square Parasheet
Material Weights (kg) (continued)				
Lines	0.009	0.009	-	-
Vent Lines	-	0.003	-	-
Total	0.036	0.040	0.029	0.019
Volume (cm^3)	102	114	83	54
Packing Density (kg/m^3)	350	350	350	350
Stability	Assumed Stable	Calculated Stable	Assumed Stable	Assumed Stable

Table 5.8 (continued)

5.5 A Parachute System Incorporating a Cluster of Three Canopies

The design of clusters of parachutes is demonstrated by an example in the Recovery Systems Design Guide[9]. The specification taken from this report is given in table 5.9. The computer solution formulated from these input data is listed in table 5.10.

Cluster data are available for two types of parachute: ringsail and flat circular. In reference 9 ringsail parachutes have been used: the computer model uses flat circular canopies because in reference 9 some ringsail data are extrapolated from curves. This procedure is known to be very unreliable and therefore these ringsail data are unavailable to the computer model in its present form. The results from the computer model and reference 9 are similar, in both cases a system of three parachutes of approximately 50m nominal diameter has been calculated. The overall C_D in the computer model is less than that in reference 9, and the area is 22% greater, because a different parachute type has been used. The opening load in the computer program is 30% greater than that in reference 9. Overstrength fabric and lines have been used in this cluster example because the parachutes will not open together, if two of the canopies open at once, a load of 1.5 times the load calculated by the program would be encountered, hence a minimum reserve factor of 1.50 is recommended for the fabric and lines. The total weight of the three parachutes is 995kg, which makes the weight of one equal to 332kg. Although quite heavy, two people at least are required to lift it, this is much more manageable than a single parachute of over 900kg, which would require a mechanical device to move it. The calculated parachute reliability is 0.9837 with a 90% confidence coefficient, assuming none of the parachutes can fail. If one canopy was allowed to fail the reliability becomes 0.9949, and if two canopies were allowed to fail the reliability is greater than 0.9999, again with a confidence of 90%.

Weight	50000lbs
Maximum Rate of Descent	30 ft/sec
Altitude at Deployment	Sea Level
Deployment Velocity	68 m/s

Table 5.9 System C Specification

type of parachute - flat circular (ringsail)	
input data	
rate of descent =	9.14 m/s
store size =	1.00 m*m
store weight =	222490.8 n
store cd =	0.70
deployment velocity =	68.00 m/s
deployment altitude =	0.0 m
deployment attitude =	-90.00 degrees
allowable mass =	1000.00 kg
allowable volume =	5.0000 m*m*m
maximum shock load =	150000.0 n
output	
main dimensions	
number of canopies =	3 (3)
overall cd =	0.67 (0.92)
area of each chute =	2159.06 m*m (1767.85)
diameter of each chute =	52.43 m (47.55)
length of lines (le) =	89.00 m (80.84)
length of lines (lc) =	90.81 m
number of lines =	172
opening load =	123151 n (93444)
structural strength calculations	
safety factor used =	2.0
strength of fabric =	113.28 n/mm*50
strength of vent band =	1313.86 n
strength of skirt band =	1313.86 n
strength of tapes =	1313.86 n
strength of lines =	1459.84 n
strength of vent lines =	1459.84 n
fabric	
material properties	
part no. =	p00115 325 5
specification =	gq ms 330
material strength =	950.0 n/mm*50
material weight =	85.0 gm/m*m
material width =	1.22 m
porosity at 10 in h2o =	3.0 ft^3/ft^2sec
porosity at 1/2 in h2o =	0.2 ft^3/ft^2sec
reserve factor =	8.4
skirt band, vent band and radial tape	
material properties	
part no. =	s99167 598 6
specification =	mil t 6134 type2
strength =	1334.0 n
weight =	4.5 gm/m
width =	25.4 mm
reserve factor =	1.0

Table 5.10 Computer Solution for a Cluster of Parachutes
Figures in brackets are those for the comparable system formulated in reference 9.

lines and vent lines
material properties part no. = p00107 191 5 specification = dtd 5620 ca105 strength = 2450.0 n weight = 140.9 m/kg reserve factor = 1.7
construction details
vent area = 4.50 % of constructed area gore width at vent = 0.20 m maximum gore width = 0.96 m gore height = 20.65 m number of panels per gore = 18
total weight of each component
weight of fabric = 198.665 kg weight of vent line = 7.034 kg weight of vent band = 0.158 kg weight of skirt band = 0.742 kg weight of tapes = 16.143 kg weight of lines = 108.885 kg total weight = 994.85 kg volume = 3.1089 m*m*m
parachute is stable
reliability = 0.9837 at 90% confidence, with no canopies being allowed to fail

Table 5.10 (continued)

5.6 Reefed Airborne Forces Parachute

After consultation with the Royal Aircraft Establishment, Farnborough, a specification for an airborne forces parachute has been obtained and is listed in table 5.11. A parachute used for this application is the flat circular Irvin PX 1 Mk.4. These input data have been modified to demonstrate the reefing routine contained in the parachute design program. The allowable load was changed from 12g maximum acceleration (about 18000N) to 10000N. The deployment altitude was raised from sea level to 600m in order for the 'Lingard' opening load method to be used. The output is listed in table 5.12. Airbourne forces parachutes are never reefed, but this data is a good example for use in this way with the parachute design program.

The computer prediction is similar to the Irvin PX 1 Mk.4 parachute which also has a constructed diameter of 9.75m and uses 32 rigging lines. As can be seen from table 5.12 the calculated opening load (unreefed) is 11332N. One stage of reefing, making the initial canopy diameter 0.84m increasing to 9.75m shortly after opening, is required to reduce the load to 10000N. The calculated weight of the canopy is 11.4% greater than the allowable weight given in the specification. The weight could be reduced if a lighter weight fabric was available to the program. The PX 1 Mk.4 canopy has two different types of fabric in its construction: the program is unable to simulate this, instead it uses a large vent to stabilise the parachute.

Maximum Weight	330 lbs
Maximum Rate of Descent	7 m/sec
Maximum Opening Load	12 g
Maximum Deployment Speed	140 knots
Deployment Altitude	Ground Level
Permitted Range of Oscillations	$\pm 15^{o}$
Maximum Canopy Weight	10 lbs
Bulk as low as possible	
Nylon materials	
Not prone to blown peripheries	
Non driving canopy	
Shelf life greater than ten years	

Table 5.11 Airborne Forces Parachute Specification

type of parachute - flat circular	
input data	
rate of descent =	7.00 m/s
store size =	0.50 m*m
store weight =	1481.3 n
store cd =	0.70
deployment velocity =	72.07 m/s
deployment altitude =	600.0 m
deployment attitude =	-90.00 degrees
allowable mass =	4.54 kg
allowable volume =	0.0500 m*m*m
maximum shock load =	10000.0 n
output	
main dimensions	
nominal area =	74.73 m*m
nominal diameter =	9.75 m
cd =	0.70
line length =	9.75 m
number of lines =	32
opening load calculated using lingard method =	11332.4 n
reefing details	
opening load =	10000.0 n
reefed diameter =	1.03 m
reefed area =	0.84 m*m
length of reefing line =	3.24 m
Structural strength of each component	
safety factor used =	3.1
strength of fabric =	76.64 n/mm*50
strength of vent band =	923.62 n
strength of vent line =	1026.24 n
strength of reefing line =	573.49 n
strength of lines =	1026.24 n
strength of tapes =	923.62 n
strength of skirt band =	923.62 n
fabric	
material properties	
part no =	-
specification =	gq ms 309
material strength =	400.0 n/mm*50
material weight =	39.0 gm/m*m
porosity at 10 in h2o =	0.0 ft^3/ft^2sec
porosity at 1/2 in h2o =	0.0 ft^3/ft^2sec
reserve factor =	5.2

Table 5.12 Design Solution for a Reefed Airborne Forces Parachute

skirt band, vent band and radial tapes	
material properties	
part no. =	s99167 598 6
specification =	mil t 6134 type2
strength =	1334.0 n
weight =	4.5 gm/m
width =	25.4 mm
reserve factor =	1.4
lines and vent lines	
material properties	
part no. =	p00107 261 0
specification =	mil c 5040 type2
strength =	1779.0 n
weight =	320.0 m/kg
reserve factor =	1.7
reefing line	
material properties	
part no. =	p00107 213 1
specification =	dtd 5620 sb603
strength =	670.0 n
weight =	588.2 m/kg
reserve factor =	1.2
construction details	
vent area =	4.50 % of constructed area
gore width at vent =	0.20 m
maximum gore width =	0.96 m
gore height =	3.84 m
number of panels per gore =	4
total weight of each component	
weight of fabric =	3.180 kg
weight of skirt band =	0.139 kg
weight of vent band =	0.030 kg
weight of tapes =	0.584 kg
weight of lines =	0.995 kg
weight of vent line =	0.124 kg
weight of reefing line =	0.006 kg
total weight =	5.06 kg
volume =	0.0158 m*m*m
packing density =	320.0 kg/m*m*m
parachute is stable	
reliability =	0.9949 at 90% confidence

Table 5.12 (continued)

5.7 Staging of an Airborne Forces Parachute

The airborne forces parachute data listed in table 5.11 were modified again to demonstrate staging. The deployment speed was raised to 110m/s, as there is an option to use the staging routine in the program at deployment speeds of greater than 100m/s, the deployment altitude is increased to 600m for consistency with the reefing example, and the allowable shock load changed to 12000N (below the original requirement of 12g maximum acceleration). The computer output is listed in table 5.13.

As shown in the output a 0.66m diameter drogue is required for this system. Spring loaded auxiliary parachutes of this size are often used in these systems. The opening load is given as 11000N to avoid the reefing routine. The same materials are used as for the reefed airborne forces parachute example.

type of parachute - flat circular	
input data	
rate of descent =	7.00 m/s
store size =	0.50 m*m
store weight =	1481.3 n
store cd =	0.70
deployment velocity =	110.00 m/s
deployment altitude =	600.0 m
deployment attitude =	-90.00 degrees
allowable mass =	4.54 kg
allowable volume =	0.0500 m*m*m
maximum shock load =	12000.0 n
output	
nominal area =	74.73 m*m
nominal diameter =	9.75 m
cd =	0.70
line length =	9.75 m
number of lines =	32
staging details	
drogue nominal area =	0.34 m*m
drogue nominal diameter =	0.66 m
trailing distance =	5.59 m
opening load =	11000.0 n
structural strength calculations	
safety factor used =	3.1
strength of fabric =	84.30 n/mm*50
strength of vent band =	1015.98 n
strength of vent line =	1128.86 n
strength of lines =	1128.86 n
strength of tapes =	1015.98 n
strength of skirt band =	1015.98 n
fabric	
material properties	
part no. =	-
specification =	gq ms 309
material strength =	400.0 n/mm*50
material weight =	39.0 gm/m*m
porosity at 10 in h2o =	0.0 ft/sec
porosity at 1/2 in h2o =	0.0 ft/sec
reserve factor =	4.7
skirt band, vent band and radial tapes	
material properties	
part no. =	s99167 598 6
specification =	mil t 6134 type2
strength =	1334.0 n
weight =	4.5 gm/m
width =	25.4 mm
reserve factor =	1.3

Table 5.13 Design Solution for Staging of an Airborne Forces Parachute

rigging lines and vent lines
material properties
part no. = p00107 261 0
specification = mil c 5040 type2
strength = 1779.0 n
weight = 320.0 m/kg
reserve factor = 1.6
construction details
vent area = 4.50 % of constructed area
gore width at vent = 0.20 m
maximum gore width = 0.96 m
gore height = 3.84 m
number of panels per gore = 4
total weight of each component
weight of fabric = 3.180 kg
weight of vent line = 0.124 kg
weight of vent band = 0.030 kg
weight of skirt band = 0.139 kg
weight of tapes = 0.584 kg
weight of lines = 0.995 kg
total weight = 5.05 kg
volume = 0.0158 m*m*m
packing density = 320.0 kg/m*m*m
parachute is stable
reliability = 0.9949 at 90% confidence

Table 5.13 (continued)

5.8 Ribbon Parachute

The ribbon parachute routines in the program have been demonstrated by some data for a small (20kg store) parachute. The input data and output for this example is in table 5.14.

High factors are used in the stressing of the skirt and vent bands, this is usual for ribbon parachutes. The canopy is stable due to its high geometric porosity, 28.2%. This geometric porosity can be lowered by including more horizontal ribbons and vertical tapes. For instance if thirty horizontal ribbons are used the geometric porosity becomes 8%. The weight of this parachute is very high considering it is so small. The reason for this is that heavy ribbons are used because no light weight, wide (2 in or 50mm) ribbons are available to the program at present. In general ribbon parachutes are used for aircraft and motor vehicle deceleration applications, in which case the parachute weight is less important than in descent applications.

type of parachute - flat ribbon
input data
rate of descent = 7.00 m/s store size = 0.10 m*m store weight = 196.2 n store cd = 0.70 deployment velocity = 50.00 m/s deployment altitude = 200.0 m deployment attitude = -90.00 degrees allowable mass = 20.00 kg allowable volume = 0.0500 m*m*m maximum shock load = 20000.0 n
output
main dimensions
nominal area = 9.34 m*m nominal diameter = 3.45 m cd = 0.70 line length = 3.45 m
number of lines = 12
opening load calculated using pflanz method = 1536.7 n
structural strength calculations
safety factor used = 2.3 strength of horiz ribbons = 171.93 n strength of vert tapes = 78.15 n strength of tapes = 158.17 n strength of skirt band = 390.74 n strength of vent band = 709.59 n strength of lines = 312.59 n strength of vent line = 312.59 n
skirt band
material properties part no. = p00167 750 9 specification = gq ms 132 strength = 535.0 n weight = 3.0 gm/m width = 15.0 mm reserve factor = 1.4
horizontal ribbon and vertical tape
material properties part no. = b01103 009 2 specification = bsf 124/224 strength = 1461.0 n weight = 79.0 gm/m width = 44.5 mm reserve factor = 8.5

Table 5.14 Ribbon Parachute Design Solution

radial tape	
material properties	
part no. =	s99167 598 6
specification =	mil t 6134 type2
strength =	1334.0 n
weight =	4.5 gm/m
width =	25.4 mm
reserve factor =	8.4
vent band	
material properties	
part no. =	p00167 050 7
specification =	mil t 5038 type3
strength =	890.0 n
weight =	3.7 gm/m
width =	9.5 mm
reserve factor =	1.3
rigging lines and vent lines	
material properties	
part no. =	p00107 213 1
specification =	dtd 5620 sb603
strength =	670.0 n
weight =	588.2 m/kg
reserve factor =	2.1
construction details	
vent area =	1.00 % of constructed area
gore width at vent =	0.12 m
maximum gore width =	0.91 m
gore height =	1.53 m
number of horiz ribbons =	20
number of vertical tapes =	5
gap width =	0.034 m
geometric porosity =	28.2 percent
total weight of each component	
weight of horiz ribbon =	10.066 kg
weight of vert. tapes =	5.310 kg
weight of radial tapes =	0.097 kg
weight of vent band =	0.006 kg
weight of skirt band =	0.033 kg
weight of lines =	0.074 kg
weight of vent line =	0.008 kg
total weight =	15.59 kg
volume =	0.0487 m*m*m
packing density =	320.0 kg/m*m*m
parachute is stable	

<u>Table 5.14 (continued)</u>

5.9 6.2m Flying Diameter Aeroconical Gliding Parachute

Requirements for a gliding emergency escape personnel parachute have been taken from liason with the co-operating body and reference 40. These data are listed in table 5.15. The output for a computer solution from these requirements, together with the manufacturer's proposals are listed in table 5.16.

A parachute generally used for this application is the G.Q. Aeroconical 6.2m flying diameter canopy. The parachute calculated by the parachute design program is similar to but smaller than this system, its nominal diameter of 8.24m represents a flying diameter of only 5.4m. Opening load and structural strength data determined by the program are similar to those for the 6.2m Aeroconical canopy.

The construction of the 6.2m Aeroconical parachute is very complicated. There are six fabric panels per gore, three large outer panels and three small inner ones. Three different materials are used for these panels. In the computer model the three small panels are assumed to be one large one, giving four panels in all, and the material used for all four panels is impression N8726 fabric. The same tape is used in both the computer solution and the GQ Aeroconical 6.2m canopy for the vent and skirt bands. A different tape from that in the 6.2m canopy is used for the radials in the computer solution and different cord for the vent lines.

Both the parachute calculated by the program and the 6.2m Aeroconical are stable due to the cut-outs incorporated in their canopies.

Rate of Descent	6.5 m/s
Maximum Horizontal Velocity	4.55 m/s
Maximum Store Weight	140 kg
Maximum Deployment Speed	154 m/s
Maximum Deployment Altitude	1829 m

Table 5.15 Design Requirements for an Emergency Escape Parachute

type of parachute - aeroconical	
input data	
rate of descent =	6.50 m/s
store size =	0.50 m*m
store weight =	1373.4 n
store cd =	0.70
deployment velocity =	154.00 m/s
deployment altitude =	1829.0 m
deployment attitude =	-90.00 degrees
allowable mass =	20.00 kg
allowable volume =	0.0500 m*m*m
cone angle =	20.00 degrees
maximum shock load =	50000.0 n
horizontal velocity =	4.55 m/s
output	
main dimensions	
nominal area =	53.29 m*m
nominal diameter =	8.24 m
cr =	0.80
line length =	8.24 m
number of lines = 20 (20)	
opening load calculated using lingard method = 31051.3 n	
structural strength calculations	
safety factor used =	2.0
strength of fabric =	181.81 n/mm*50
strength of vent band =	4934.09 n
strength of vent line =	3289.39 n
strength of lines =	3289.39 n
strength of tapes =	2960.46 n
strength of skirt band =	4934.09 n
fabric	
material properties	
part no. =	-
specification =	n8726 (bsf 126/254, gq ms 502 (b1), and gq ms 330)
material strength =	400.0 n/mm*50
material weight =	40.0 gm/m*m
material width =	1.42 m
reserve factor =	2.2
skirt band and vent band	
material properties	
part no. =	p00168 959 8
specification =	gq ms 317 (gq ms 317)
strength =	5800.0 n
weight =	17.0 gm/m.
width =	25.0 mm
reserve factor =	1.2

Table 5.16 Design Solution for a Gliding Aeroconical Parachute
figures in brackets refer to the GQ 6.2m Aeroconical canopy

tape	
material properties	
part no. =	p00169 591 3 (p00167 550 7)
specification =	mil t 5038 type2 (gq ms 124)
strength =	4003.0 n (1780.0)
weight =	12.4 gm/m (5.2)
width =	25.0 mm (12.0)
reserve factor =	1.4
rigging lines	
material properties	
part no. =	p00107 226 2
specification =	dtd 5620 sc711 (dtd 5620 sc711)
strength =	3350.0 n
weight =	111.1 m/kg
reserve factor =	1.0
vent line	
material properties	
part no. =	p00107 226 2 (p00107 247 4)
specification =	dtd 5620 sc711 (dtd 5620 cc311)
strength =	3350.0 n (2000.0)
weight =	111.1 m/kg (222.2)
reserve factor =	1.0
construction details	
vent area =	0.80 % of nominal area (0.80)
gore width at vent =	0.11 m
maximum gore width =	1.26 m
gore height =	3.86 m
number of panels per gore =	4 (6)
no. of largest panel out =	2 (2)
no. of panel 2 missing =	0 (0)
no. of panel 3 missing =	0 (0)
no. of panel 4 missing =	0 (0)
total weight of each component	
weight of fabric =	2.292 kg
weight of vent line =	0.105 kg
weight of vent band =	0.042 kg
weight of skirt band =	0.431 kg
weight of tapes =	1.016 kg
weight of lines =	1.519 kg
total weight =	5.40 kg
volume =	0.0169 m*m*m
packing density =	320.0 kg/m*m*m
reliability =	0.9949 at 90% confidence

Table 5.16 (continued)
figures in brackets refer to the GQ 6.2m Aeroconical canopy

5.10 Discussion of Results and Conclusions

A number of conclusions can be drawn from the results presented in this chapter:

(i) The parachute design program is consistent between applications and types of parachute.

In the manufacturer's proposals (references 36 to 39), different stressing methods have been used for the same type of parachute and hence inconsistency arises. For example the calculation of line strength should include the angle the lines make with the vertical. This angle is not known at the time of maximum load and so the worst case should be assumed - the canopy is fully inflated. This angle is generally about 20^{o} and if omitted from the calculation the line strength becomes 6.4% too low. The strengths of the canopy reinforcing tapes and vent lines are factored from the line strength so this angle is very important.

The safety factors built into the program are 2.3 for un-manned applications and 3.1 for manned applications. They have been taken from the Recovery Systems Design Guide. The user has the opportunity to alter these factors if he wishes. In the four, all un-manned, applications studied in the proposals (references 36 to 39), three different safety factors were used. Three of these proposed parachutes are cruciform, and for each of these a different safety factor has been used. This again will cause inconsistencies which can be avoided if the parachute design program is employed.

(ii)The program allows comparisons of standard design methods.

Three methods are used in the program for the calculation of opening load, the user is able to choose which he requires. Ideally the load obtained from each method is compared and the highest value taken as the opening load. However some of the methods are more reliable than others. The 'Lingard' method is the most accurate and should be used whenever possible, although when a cruciform parachute load is being calculated using this method, the empirical data used may not correspond to a similar cruciform to the one being designed. This problem is often solved by changing the dimensionless inflation time, τ_o, value used, as this value is known to depend on the canopy mass. The canopy size is an important factor in the initial opening of the parachute. Additionally as shown in the mail dropping parachute example, section 5.3, loads obtained for a cruciform canopy can differ by up to 50% depending on the load calculation method used. Hence results obtained from the more approximate load calculation methods may not be reliable in some cases. Generally however, all three load calculation methods (Lingard, Pflanz and mass ratio) used in the "paradesign" program produce consistent results, as can be seen from the outputs.

(iii)Comparison of types of parachute.

The program demonstrates the improved drag efficiency of the conical, extended skirt and hemispherical parachutes over the flat circular parachute. Although still used for some applications this type is obsolescent. Irvin are replacing their PX1 Mk4 airborne forces flat circular mentioned in section 5.5, by a cruciform parachute.

The program allows a number of different types of parachute to be compared for use in the same application in a reasonably short time.

(iv) More data is required for the program.

The program can call on data for seventy different materials. However this is insufficient. The ribbon parachute is excessively heavy, 15.6kg compared to 2.62kg for a conical parachute which is twice as large (section 5.3). This is due to no wide, light weight ribbons being available to the program. The porosity data in both U.K and U.S. units (section 3.15) are required for some fabrics so that a stability calculation can be made when they are used.

If the cluster, reefing, staging and ribbon parachute routines contained in the program are to be used extensively more data must be obtained to check these calculations. At present, due to a lack of suitable input data, no comparison with manufacturer's current specifications has been made.

(v) The program is of limited use for complicated parachutes.

For structurally complex canopies such as the 6.2m flying diameter Aeroconical, the design program gives solutions which only approximate to the manufacturer's (table 5.15). The final design of this canopy has been formulated after many hours of design trials and many modifications. To represent all this work in a single computer program will be very difficult. However the parachute design program can be used as an initial design attempt.

The program has allowed design methods and canopy types to be compared, it has also exposed manufacturer's inconsistencies. It has demonstrated the limits of empirical design methods when they are applied to certain canopy shapes. But most importantly it is a time-saving device. Preparing a proposal such as references 36 to 39 will take on average a week's work by a parachute design engineer. The program can give answers to a design specification in half an hour. Then the numerical output from the program has to be included in a written proposal. Tables 5.10, 5.12, 5.13, 5.14 and 5.16 are data files generated by the parachute design program which have been transferred to the word processor used to write this report. Using a similar technique the writing of design proposals can be further speeded up.

The program is limited by canopy design complexity as has been shown in the gliding aeroconical example (section 5.9). For highly complex parachutes such as the 6.2m flying diameter aeroconical canopy it can be used in the preliminary design stages as a check, and it will give approximations to the weight and volume.

The main area in which the program can be expanded is in the materials section. The data tables can be added to so that a greater choice is available to the user. Expansion of the program in other areas which require more variables and lines of Pascal code will be difficult because in its present form (about 7000 lines of code and 300 variables) the program is stretching the Sirius computer to its limits.

Chapter 6

Recommendations for Further Work

Improvement and alteration of the parachute design program presented in this thesis will be essential in order to keep it up to date. So, as new design methods and types of parachute are introduced they must be incorporated into the program whenever possible. One immediate way of improving the program is by expansion of the materials data tables (tables D.1 to D.3). Expansion and alteration of these tables is easily done as explained in section 4.4.1.

The program does not save the results from the trajectory calculation, section 3.8.4, so that they can be printed out if required. There are various reasons for this, including the difficulty of input/output operations in Pascal, and the slowness of this particular calculation which would become even worse if a file had to be created and the results sent to it. However a trajectory calculation is often required when designing a parachute system and a method of printing trajectory output is needed.

Section 4.4.2 has shown that in its present state the program is stretching the UCSD system used on the Sirius computer to its limit. One way to improve this situation would be to transfer the program to the language Modula-2[41]. This language, introduced in 1979, is an improved version of Pascal, suitable for large software development (i.e. long programs similar to "paradesign"). Using this language interaction between different program units becomes easier. The input and output routines in Pascal have been improved on and interaction between the program and a printer and plotter will be possible. Modula-2 is more suitable for scientific programs than Pascal, an exponential operation is included. Some problems may be encountered in transferring the parachute design program to Modula-2, but in the long run this is the recommended course of action.

Chapter 7

Conclusions

In addition to the detailed conclusions stated in section 5.10 the following general conclusions are drawn.

(i) The parachute design program "paradesign" is a valuable time-saving device for a parachute design engineer.

(ii) The results presented in this thesis, obtained using the design program, have been shown to compare well with manufacturer´s proposals. Some manufacturer´s inconsistencies have been exposed.

(iii) Expansion of the program is essential to keep it up to date. More materials data are required, this can easily be obtained. Improved interaction with a printer and plotter, in order to enhance presentation of data generated by the program to the user, is required. However expansion is limited by the operating system used. In order to use the program to its full potential transferring it from Pascal to the new computer language Modula-2 is recommended.

(iv) In its present state the program is of limited use for complex parachutes, but improvement in this area, subject to the limits of (iii) above, is always possible.

References

1.Johns,T.F. Parachute Design, R&M 2402, HMSO, 1946.

2.Brown,W.D. Parachutes, Sir Isaac Pitman and Sons, London, 1951.

3.Solt,G.A.Jr. Performance of and Design Criteria for Deployable Aerodynamic Decelerators, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio, ASD-TR-67-799, December 1963.

4.Ibrahim,S.K. An Overview of Munition Decelerator Tchnology with Recent Applications at Honeywell, AIAA 8th Aerodynamic Decelerator and Balloon Technology Conference, Hyannis, 1984.

5.Pepper,W.B. and Maydew,R.C. Aerodynamic Decelerators - An Engineering Review, AIAA Journal of Aircraft Vol.8, No.1, January 1971.

6.Dennis,D.R. Recent Advances in Parachute Technology, R.Ae.S. Aeronautical Journal, November 1983.

7.DeWeese,J.H. and McCarty,R.E. Parachute Design and Performance Databank, Air Force Flight Dynamics Laboratory, AFFDL-TR-74-45, January 1975.

8.Bozack,J. Computerized Parachute Data Base System, Aerosystems Department, Naval Weapons Center, China Lake, California, Survival and Flight Equipment Association, 20th Symposium, Las Vegas, 1982.

9.Ewing,E.W., Bixby,H.W. and Knacke,T.W. Recovery Systems Design Guide, Irvin Industries Inc., California, Technical Report AFFDL-TR-78-151, December 1978.

10.Pinnel,W.R. Materials and Design Criteria for Kevlar-29 Ribbon Parachutes, Air Force Wright Aeronautical Laboratories, Wright Patterson Air Force Base, Ohio, April 1982.

11.Knacke,T.W. Parachute Systems Technology, Lecture Notes From Helmut G. Heinrich Short Course on Parachute Systems Technology, University of Minnesota, July 1982. (unpublished)

12.Clark,R. et al UCSD p-system and UCSD Pascal. A Product for Mini- and Micro- Computers Version IV.0. User's Manual, SofTech Microsystems Inc., San Diego, Second Edition, January 1981.

13.Bowles,K.L. Beginner's Guide for the UCSD Pascal System, Byte Books, 70 Main Street, Peterborough, 1980.

14.Chadderton,T. Technical Report on Balloon Trials on a Cruciform Parachute for the X-RAE2 RPV, GQ Defence Equipment Ltd., Woking, Surrey, GQ TR 85503, March 1985.

15.Houghton,E.L. and Brock,A.E. Aerodynamics for Engineering Students, Edward Arnold Ltd., London.

16.O'Hara,F. Notes on the Opening Behaviour and the Opening Forces of Parachutes, Journal of the Royal Aeronautical Society, Vol. 53, 1949.

17.Heinrich,H.G. A Linearised Theory of Parachute Opening Dynamics, University of Minnesota, Minneapolis, Aeronautical Journal, Vol 76, 1972..

18.Roberts,B.W. and Reddy,K.R. A Discussion of Parachute Inflation Theories, American Institute of Aeronautics and Astronautics, 5th Decelerator Systems Conference, Albuquerque, New Mexico, November 1975.

19.Purvis,J.W. Theoretical Analysis of Parachute Inflation Including Fluid Kinetics, Journal of Aircraft, Vol. 19, No. 4, April 1982.

20.Lingard,J.S. A Semi-empirical Theory to Predict the Load-time History of an Inflating Parachute, Royal Aircraft Establishment Technical Report 79141, November 1979.

21.Wolf,D. A Simplified Model of Parachute Inflation, Sandia Laboratories, Albuquerque, New Mexico, 1973.

22.Pflanz,E. Retarding Forces During Unfolding of Cargo Parachutes, ATI 20126, United States Air Force Translation of German Report FGZ 331, September 1943.

23.Scheubel,F.N. Notes on the Opening Shock of a Parachute, Foreign Exploitation Section, Intelligence (T-2), Progress Report IRE-65, 1946.

24.Schatzle,P.R. and Curry,W.H. Flight-Simulation of a Vehicle With a Two Stage Parachute System, Sandia Laboratories Albuquerque, New Mexico, 1979.

25.Houmard,J.E. Stress Analysis of the Viking Parachute, AIAA Paper no. 73-444, AIAA Decelerator Systems Conference, Palm Springs, California, May 1973.

26.Mullins,W.M. and Reynolds,D.T. Stress Analysis of Spacecraft Parachutes Using Finite Elements and Large Deformation Theory, AIAA Paper no. 70-1195, AIAA Decelerator Systems Conference, Dayton, Ohio, September 1970.

27.Garrard,W.L. and Muramoto,K.K. Use of CANO for Stress Analysis in Ribbon Parachutes, Department of Aerospace Engineering and Mechanics, University of Minnesota, Minneapolis, 1982.

28.Konicke,T.A. and Garrard,W.L. Stress Measurements in a Ribbon Parachute Canopy During Inflation and at Steady State, AIAA Paper no. 81-1944, AIAA Aerodynamic Decelerator and Balloon Technology Conference, San Diego, California, 1981.

29.Wu,K.-Y. A Method of Stress and Shape Calculation of a Ribbon Parachute, University of Minnesota, Minneapolis, 1984.

30.Payne,P.R. The Theory of Fabric Porosity as Applied to Parachutes in Incompressible Flow, Payne Inc., Annapolis, USA, Aeronautical Quarterly, Vol 29, August 1978.

31.Tory,C. and Ayres,R. Computer Model of a Fully Deployed Parachute, AIAA Journal of Aircraft, Vol 14, No. 7, July 1977.

32.Yavuz,T. Aerodynamics of Parachutes and Like Bodies in Unsteady Motion, PhD Thesis, University of Leicester, 1982.

33.Jailer,R.W. et al. Analysis of Heavy-Duty Parachute Reliability, WADD Technical Report 60-200, American Power Jet Company, Ridgefield, New Jersey, June 1960.

34.Resnikoff and Lieberman Tables of the Non-Central t-Distribution, Stanford University Press, Stanford, California, 1957.

35.Higgins,D. Designing Structured Programs, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1983.

36.Stephens,D. Proposal for X-RAE2 R.P.V. Parachute Recovery System, GQ Defence Equipment Limited, Woking, Surrey, November 1984.

37.Stephens,D. Technical Proposal for A.E.L. Sparrowhawk and Snipe Mk. 3 R.P.V. Parachute Recovery System, GQ Defence Equipment Limited, Woking, Surrey, GQ TR 85511, April 1985.

38.Stephens,D. Technical Proposal for Stokvis La Bree Mail Dropping Parachute System, GQ Defence Equipment Limited, Woking, Surrey, GQ TR 85522, September 1985.

39.Hirst,D. Proposal for a Parachute for the Plessey SSQ 954 Sonobody, GQ Defence Equipment Limited, Woking, Surrey, GQ TR 86502, January 1986.

40.Lingard,J.S. Design Study for GQ Experimental Parachute Type 3000, GQ Defence Equipment Limited, Woking, Surrey, July 1983.

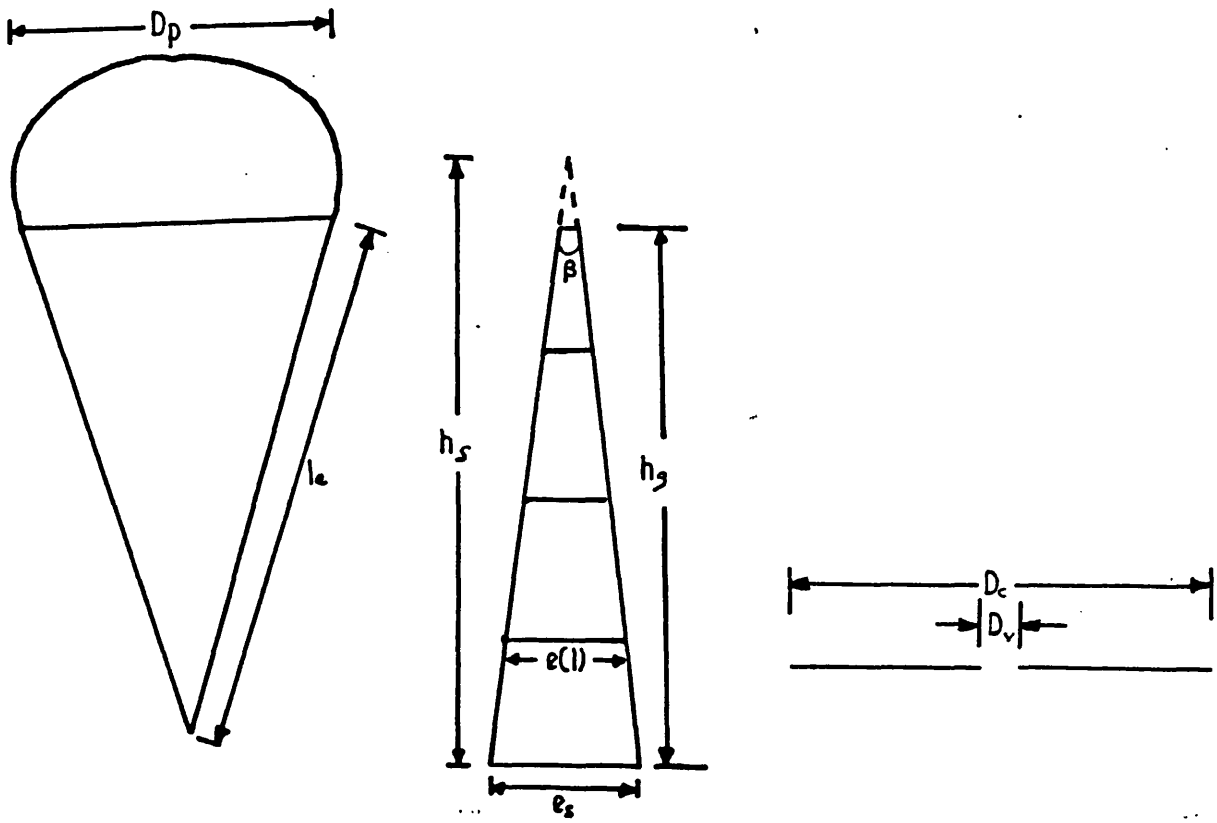
41.Knepley,E. and Platt,R. Modula-2 Programming, Reston Publishing Company Inc., Reston, Virginia, 1985.

Appendix A

Types of Parachute

In this appendix each type of parachute used in the parachute design program is considered. Canopy weight calculations are given and the input data required for the "paradesign" computer program is listed.

A.1 Type 1 - Flat Circular Parachute



inflated profile

gore layout

construction schematic

Figure A.1 Flat Circular Canopy Configuration

gore angle β = 360/N degrees

A.1.1 Canopy Weight Calculations

From reference 9,

gore height $h_s = (S_0/(N \tan(\beta/2))^{1/2}$ (A.1)(1006)

and maximum gore width:

$$e_s = 2\ h_s \tan(\beta/2) \quad \text{(A.2)(1007)}$$

The vent constructed area:

$$S_v = v_x . S_0/100$$ (A.3)(1042,1045,1049,1201,1225,1250,1293, 1613,1654,1681,1770,1797)

where v_x is the vent area as a percentage of S_0. Gore height from the skirt to the vent:

$$h_g = h_s - (S_v/(N \tan(\beta/2))^{1/2} \quad \text{(A.4)(1202,1046)}$$

If NP is the number of fabric panels per gore then h_g/NP + 0.05 is the maximum panel width for use in fabric selection (section 3.11.1), and the gore width at the largest panel:

gore width $e(1) = e_s - 2\ h_g \tan(\beta/2)/NP$ (A.5)

To allow for sewing 25mm is added to all fabric dimensions. Therefore the area of the largest panel in the gore:

$$\text{area}(1) = ((e(1) + e_s + 0.1)/2)((h_g/NP) + 0.05) \quad \text{(A.6)(1204)}$$

if NP is not equal to 1 then:

$$e(2) = e(1) - 2\ h_g \tan(\beta/2)/NP \quad \text{(A.7)}$$

and $\text{area}(2) = ((e(2) + e(1) + 0.1)/2)(h_g/NP + 0.05)$ (A.8) (1206) equations A.7 and A.8 are repeated to give gore widths to e(NP) and fabric panel areas to area(NP). Then the total fabric area (m^2):

$$TOF = N \sum_{X=1}^{NP} area(x) \qquad (A.9)(1207,1623)$$

WTF is the fabric material weight (gm/m^2), the total fabric weight (kg):

$$WF = TOF.WTF/1000 \qquad (A.10)(1209,1624)$$

200mm is added to the length of each rigging line and each radial tape for attachment purposes. And if WTL is the rigging line cord weight (m/kg), the total line length (m) and weight (kg):

$$TOL = Z(l_e + 0.2) \qquad (A.11)(1210,1238,1268,1625,1644,1662,1736,1758,1782,1815,1837)$$

$$WL = TOL/WTL \qquad (A.12)(1211,1239,1269,1626,1645,1663,1737,1759,1783,1816,1838)$$

If WTT is the tape weight (gm/m), the total tape length (m) and weight (kg):

$$TOT = N((h_g/\cos(\beta/2)) + 0.2) \qquad (A.13)$$

$$WT = TOT.WTT/1000 \qquad (A.14)(1213,1665,1739,1818)$$

200mm is also added to the lengths of the skirt band, vent band, and reinforcing band. RPN is the number of the fabric panel below the reinforcing band. If WTR is the reinforcing material weight (gm/m), length (m) and weight (kg) of the reinforcing:

$$TR = N.e(RPN) + 0.2 \qquad (A.15)(1217,1632)$$

$$WR = TR.WTR/1000 \qquad (A.16)(1218,1633)$$

WTVB is the weight of the vent band material (gm/m). Total vent band length (m) and weight (kg):

$$TVB = N.e(NP) + 0.2 \qquad (A.17)(1219)$$

$$WVB = TVB.WTVB/1000 \qquad (A.18)(1220)$$

WTSB is the skirt band material weight (gm/m). Total skirt band length (m) and weight (kg):

$$TSB = N.e_S + 0.2 \quad (A.19)(1221,1246,1276,1636,1669,1743,1766,1790,1823)$$

$$WSB = TSB.WTSB/1000 \quad (A.20)(1222,1247,1277,1637,1670,1744,1767,1791,1824)$$

200mm is also added to each vent line, for attachment purposes. WTVL is the vent line material weight (m/kg), total vent line length (m) and weight (kg):

$$TOVL = N\ ((h_S - h_g)/\cos(\beta/2)) + 0.2) \qquad (A.21)$$

$$WVL = TOVL/WTVL \qquad (A.22)(1224,1672,1747,1826)$$

Total parachute weight, WTOT, is the sum of the component weights:

$$WTOT = WF + WL + WT + WR + WVB + WSB + WVL \quad (A.23)(1640,1794,1827)$$

A.1.2 Inputs Required for the Computer Program

The inputs required are:

(i) v_x, the vent area as a percentage of the total canopy area, S_0.

(ii) NP, the number of fabric panels per gore.

A.2 Type 2 - Conical Parachute

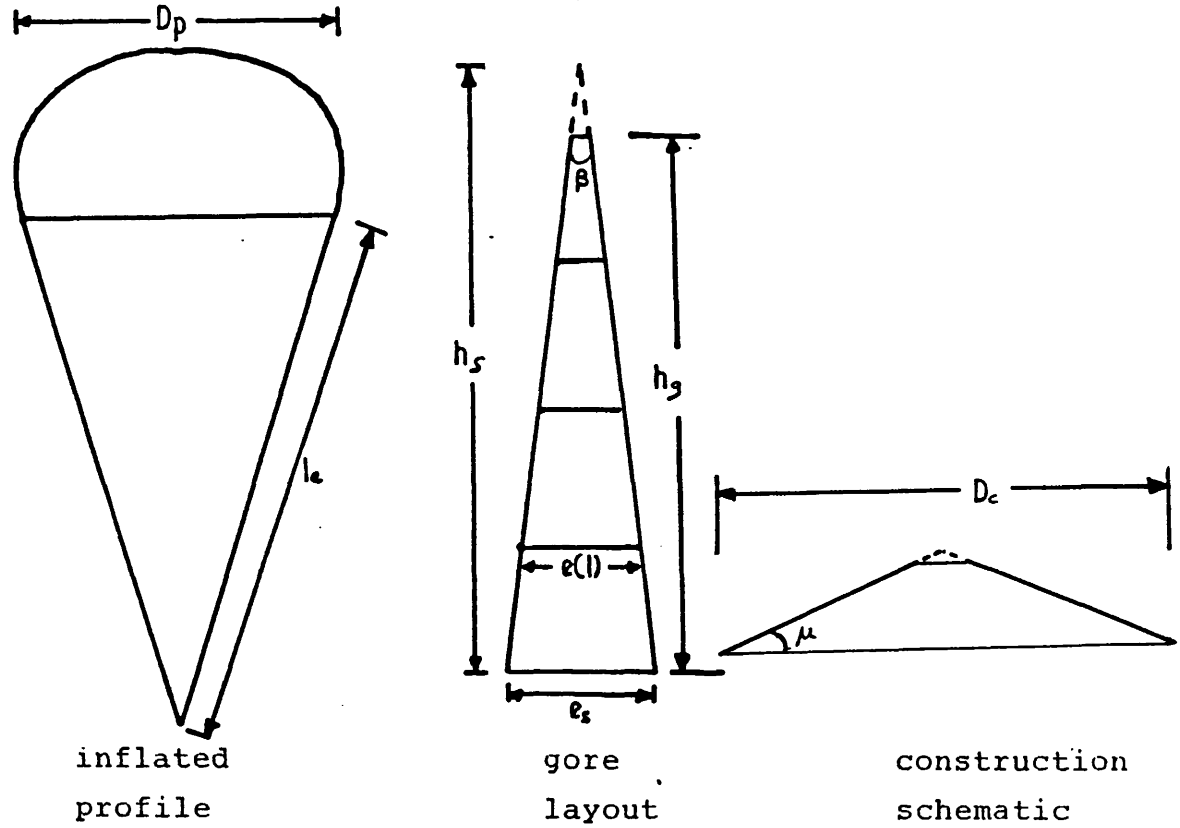


Figure A.2 Conical Parachute Configuration

μ is the canopy constructed cone angle. From reference 9:

gore angle $\beta = 2 \sin^{-1}(\sin(180/N) \cos\mu)$ (A.24)(1005,1651)

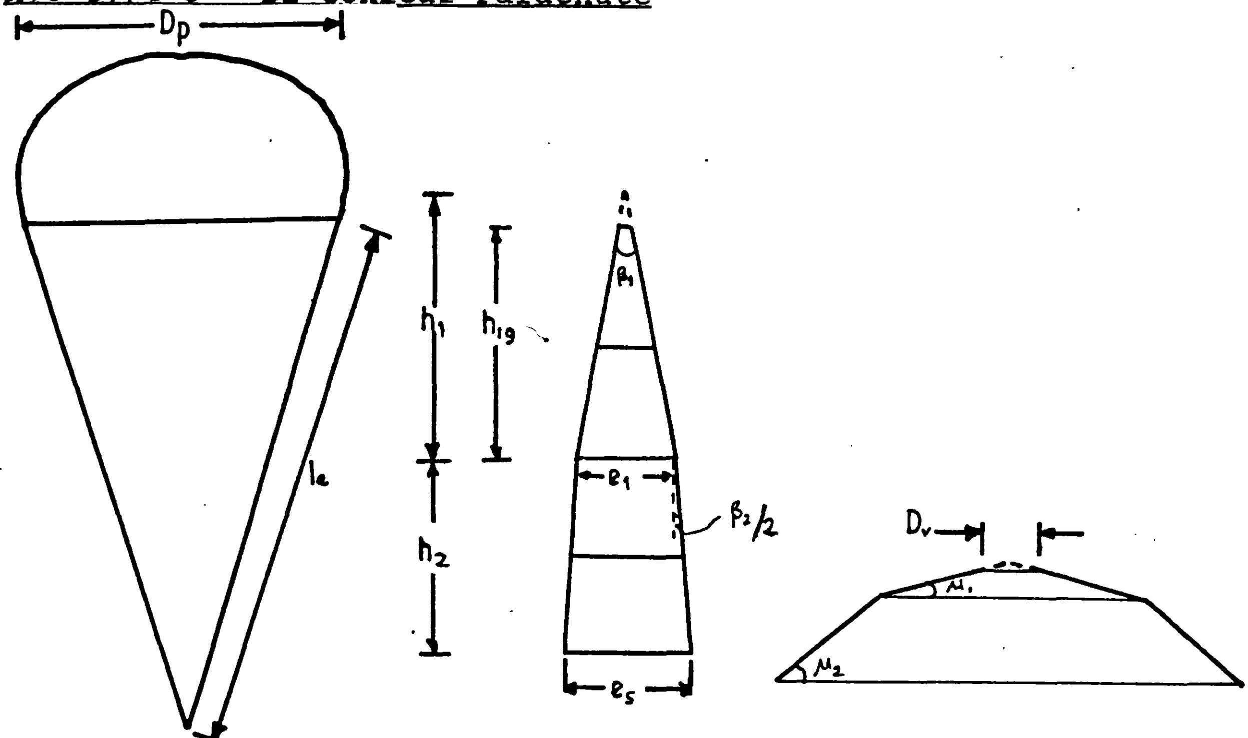
A.2.1 Parachute Weight

This is calculated as for the flat circular parachute using equations A.1 to A.23.

A.2.2 Inputs Required for the Computer Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.
(ii) NP, the number of fabric panels per gore.
(iii) μ, the canopy constructed cone angle.

A.3 Type 3 - Bi-Conical Parachute



inflated shape

gore layout

construction schematic

Figure A.3 Bi-Conical Parachute Configuration

key:
μ_1 and μ_2 ($>\mu_1$): the constructed cone angles,
h_1/h_2 ($=k$): the ratio of the gore heights.

A.3.1 Canopy Weight Calculation

Gore angles:

$$\beta_1 = 2 \sin^{-1}(\sin(180/N) \cos\mu_1) \quad \text{(A.25)(1009)}$$

$$\beta_2 = 2 \sin^{-1}(\sin(180/N) \cos\mu_2) \quad \text{(A.26)(1010)}$$

and the gore heights:

$$h_2 = (S_0/(N(k^2 \tan(\beta_1/2) + \tan(\beta_2/2) + 2 k \tan(\beta_1/2))))^{1/2} \quad \text{(A.27)(1011)}$$

$$h_1 = k\ h_2 \tag{A.28)(1012}$$

maximum gore width:

$$e_S = 2\ h_1\ \tan(\beta_1/2) + 2\ h_2\ \tan(\beta_2/2) \tag{A.29)(1013}$$

The vent constructed area is calculated from equation A.3. Then the gore height up to the vent:

$$h_{1g} = h_1 - (S_V/(N\ \tan(\beta_1/2)))^{1/2} \tag{A.30)(1226,1251}$$

gore width $e_1 = 2\ h_1\ \tan(\beta_1/2)$ (A.31)(1227,1252)

NLP is the number of fabric panels in the lower part of the gore and NUP the number of panels in the upper part of the gore. Lower gore width and the largest panel area, including sewing allowance:

$$e_B(1) = e_S - 2\ h_2\ \tan(\beta_2/2)/NLP \tag{A.32)(1228}$$

$$area_B(1) = ((e_B(1) + e_S + 0.1)/2)(h_2/NLP + 0.05) \tag{A.33)(1229}$$

If NLP is not equal to 1 then:

$$e_B(2) = e_B(1) - 2\ h_2\ \tan(\beta_2/2)/NLP \tag{A.34)(1230}$$

$$area_B(2) = ((e_B(2) + e_B(1) + 0.1)/2)(h_2/NLP + 0.05) \tag{A.35)(1231}$$

Equations A.34 and A.35 are repeated to $e_B(NLP)$ and $area_B(NLP)$. Similarly for the upper part of the gore, width and panel area:

$$e_T(1) = e_1 - 2\ h_{1g}\ \tan(\beta_1/2)/NUP \tag{A.36)(1232,1262,1295}$$

$$area_T(1) = ((e_T(1) + e_1 + 0.1)/2)(h_{1g}/NUP + 0.05) \tag{A.37)(1233,1263,1296}$$

If NUP, the number of upper gore panels is not equal to 1 then:

$$e_T(2) = e_T(1) - 2\ h_{1g}\ \tan(\beta_1/2)/NUP \quad (A.38)(1234,1264,1297)$$

$$area_T(2) = ((e_T(2) + e_T(1) + 0.1)/2)(h_{1g}/NUP + 0.05) \quad (A.39)(1235,1265,1298)$$

equations A.38 and A.39 are repeated to e_T(NUP) and $area_T$(NUP). The total fabric area (m^2):

$$TOF = \sum_{X=1}^{NLP} area_B(x) + \sum_{X=1}^{NUP} area_T(x) \quad (A.40)(1236,1299)$$

WTF is the fabric material weight (gm/m^2), so the fabric weight (kg):

$$WF = TOF.WTF/1000 \quad (A.41)(1237,1300)$$

Rigging line and skirt band weights are taken from equations A.11, A.12, A.19 and A.20.

The radial tape material weight is WTT (gm/m), so the total length (m) and weight (kg) of the tapes:

$$TOT = N(h_{1g}/\cos(\beta_1/2) + h_2/\cos(\beta_2/2) + 0.2) \quad (A.42)(1240)$$

$$WT = TOT.WTT/1000 \quad (A.43)(1241)$$

The canopy reinforcing band (if present) is assumed to be at width e_1 of the gore, the join between the wide and narrow parts. If WTR is the reinforcing tape material weight, (gm/m), then the total reinforcing length (m) and weight (kg):

$$TR = N.e_1 + 0.2 \quad (A.44)(1242,1603)$$

$$WR = TR.WTR/1000 \quad (A.45)(1243,1604)$$

WTVB is the vent band material weight (gm/m). Vent band length (m) and weight (kg):

$$TVB = N.e_T(NUP) + 0.2 \qquad (A.46)(1244,1274,1607)$$

$$WVB = TVB.WTVB/1000 \qquad (A.47)(1245,1275,1608)$$

WTVL is the weight of the vent line (m/kg). Vent line length (m) and weight (kg):

$$TOVL = N((h_1 - h_{1g})/\cos(\beta_1/2) + 0.2) \qquad (A.48)(1248,1278,1611)$$

$$WVL = TOVL/WTVL \qquad (A.49)(1249,1279,1612)$$

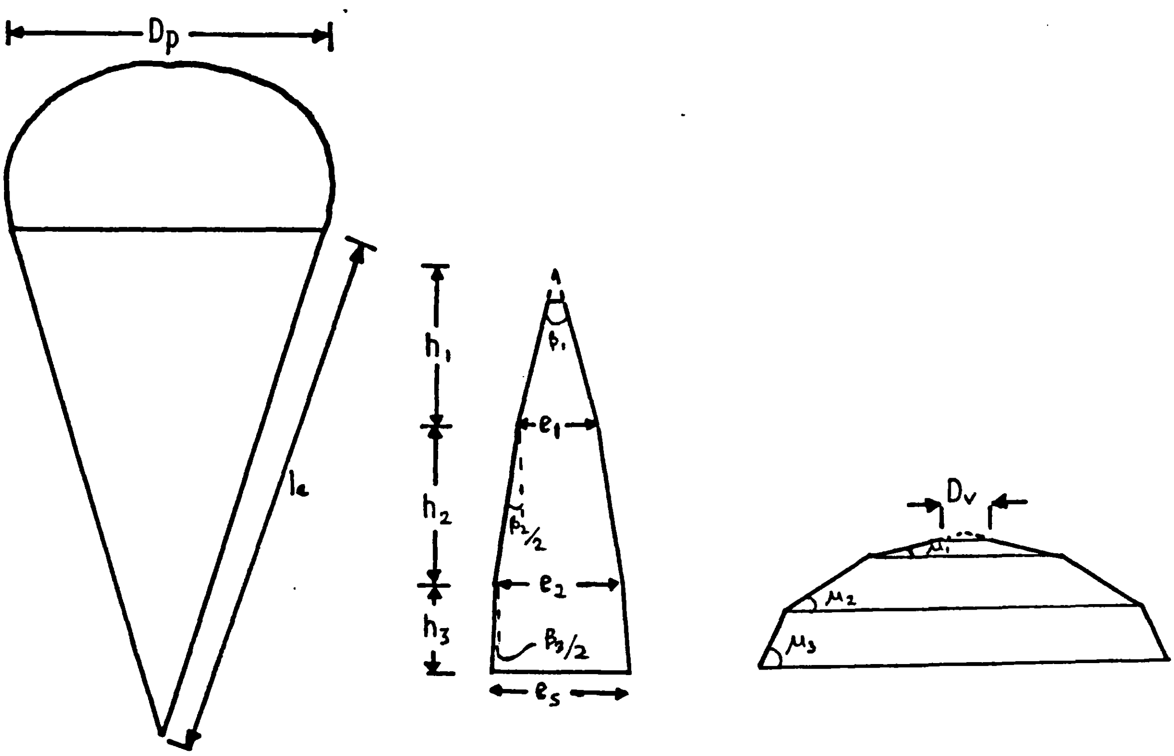
The total canopy weight can now be calculated using equation A.23.

A.3.2 Inputs Required for the Computer Program

The inputs required for the bi-conical parachute are:

(i) v_x, vent area as a percentage of total canopy area, S_0.
(ii) Canopy constructed cone angles, μ_1 and μ_2.
(iii) The gore height ratio, h_1/h_2 (=k).
(iv) NUP and NLP, the number of upper and lower fabric panels per gore.

A.4 Type 4 - Tri-Conical Parachute



inflated profile | gore layout | construction schematic

Figure A.4 Tri-Conical Parachute Configuration

key:

$\mu_1,\ \mu_2,\ \mu_3$ - constructed cone angles.

$k_1 = h_1/h_2,\ k_2 = h_3/h_2$ - gore height ratios.

$\beta_1,\ \beta_2,\ \beta_3$ - gore angles.

A.4.1 Canopy Weight Calculation

Gore angles:

$$\beta_1 = 2\sin^{-1}(\sin(180/N)\cos\mu_1) \qquad (A.50)(1015)$$

$$\beta_2 = 2\sin^{-1}(\sin(180/N)\cos\mu_2) \qquad (A.51)(1016)$$

$$\beta_3 = 2\sin^{-1}(\sin(180/N)\cos\mu_3) \qquad (A.52)(1017)$$

gore heights:

$$h_2 = (S_0/(N(\tan(\beta_1/2)\ (k_1^2 + 2\ k_1 + 2\ k_1\ k_2) + \tan(\beta_2/2)\ (1 + 2\ k_2) + \tan(\beta_3/2)\ k_2^2)))^{1/2} \quad (A.53)(1018)$$

$$h_1 = k_1\ h_2 \quad (A.54)(1019)$$

$$h_3 = k_2\ h_2 \quad (A.55)(1020)$$

and maximum gore width:

$$e_S = 2\ h_1\ \tan(\beta_1/2) + 2\ h_2\ \tan(\beta_2/2) + 2\ h_3\ \tan(\beta_3/2) \quad (A.56)(1021)$$

The vent constructed area is calculated from equation A.3. Gore height h_{1g} and width e_1 from equations A.30 and A.31. Gore width:

$$e_2 = 2\ h_2\ \tan(\beta_2/2) + e_1 \quad (A.57)(1253)$$

The gore is split into three parts and: NLP - number of lower panels, NMP - number of middle panels, NUP - number of upper panels. Lower gore widths and panel areas, including sewing allowance, are calculated as follows:

$$e_B(1) = e_S - 2\ h_3\ \tan(\beta_3/2)/NLP \quad (A.58)(1254)$$

$$area_B(1) = ((e_B(1) + e_S + 0.1)/2)(h_3/NLP + 0.05) \quad (A.59)(1255)$$

if NLP is not equal to 1,

$$e_B(2) = e_B(1) - 2\ h_3\ \tan(\beta_3/2)/NLP \quad (A.60)(1256)$$

$$area_B(2) = ((e_B(2) + e_B(1) + 0.1)/2)(h_3/NLP + 0.05) \quad (A.61)(1257)$$

A.60 and A.61 are repeated to e_B(NLP) and $area_B$(NLP). Middle gore widths and fabric areas:

$$e_M(1) = e_2 - 2\ h_2\ \tan(\beta_2/2)/NMP \qquad (A.62)(1258)$$

$$area_M(1) = ((e_M(1) + e_S + 0.1)/2)(h_2/NMP + 0.05) \qquad (A.63)(1259)$$

if NMP is not equal to 1 then:

$$e_M(2) = e_M(1) - 2\ h_2\ \tan(\beta_2/2)/NMP \qquad (A.64)(1260)$$

$$area_M(2) = ((e_M(2) + e_M(1) + 0.1)/2)(h_2/NMP + 0.05) \qquad (A.65)(1261)$$

A.64 and A.65 are repeated to e_M(NMP) and $area_M$(NMP). The top gore widths and panel areas are calculated as for the Bi-conical parachute, equations A.36 to A.39. and if WTF is the fabric material weight (gm/m^2), the total fabric area (m^2) and weight (kg) is:

$$TOF = N\ \left(\sum_{X=1}^{NLP} area_B(x) + \sum_{X=1}^{NMP} area_M(x) + \sum_{X=1}^{NUP} area_T(x)\right) \qquad (A.66)(1266)$$

$$WF = TOF.WTF/1000 \qquad (A.67)(1267)$$

The weights of the rigging lines, vent band, vent lines, and skirt band are calculated as for the bi-conical parachute, section A.3.2. If WTT is the weight of the radial tapes (gm/m), then length (m), including a sewing allowance, and weight (kg) of the tapes:

$$TOT = N\ (h_{1g}/\cos(\beta_1/2) + h_2/\cos(\beta_2/2) + h_3/\cos(\beta_3/2) + 0.2) \qquad (A.68)(1270)$$

$$WT = TOT.WTT/1000 \qquad (A.69)(1271)$$

The canopy reinforcing is assumed to lie on gore widths e_1

and e_2, WTR is the weight (gm/m) of the reinforcing material, so total reinforcing length (m) and weight (kg):

$$TR = N\ (e_1 + e_2) + 0.4 \qquad (A.70)(1272)$$

$$WR = TR.WTR/1000 \qquad (A.71)(1273)$$

The total parachute weight is taken from equation A.23.

A.4.2 Inputs Required for the Parachute Design Program

The inputs required for the tri-conical parachute are:

(i) v_x, the vent area as a percentage of total canopy area, S_0.

(ii) Constructed cone angles μ_1, μ_2, and μ_3.

(iii) Gore height ratios h_1/h_2 $(= k_1)$ and h_3/h_2 $(= k_2)$.

(iv) NUP, NMP and NLP, the number of fabric panels in each part of the gore.

A.5 Type 5 - Extended Skirt 10% Flat Parachute

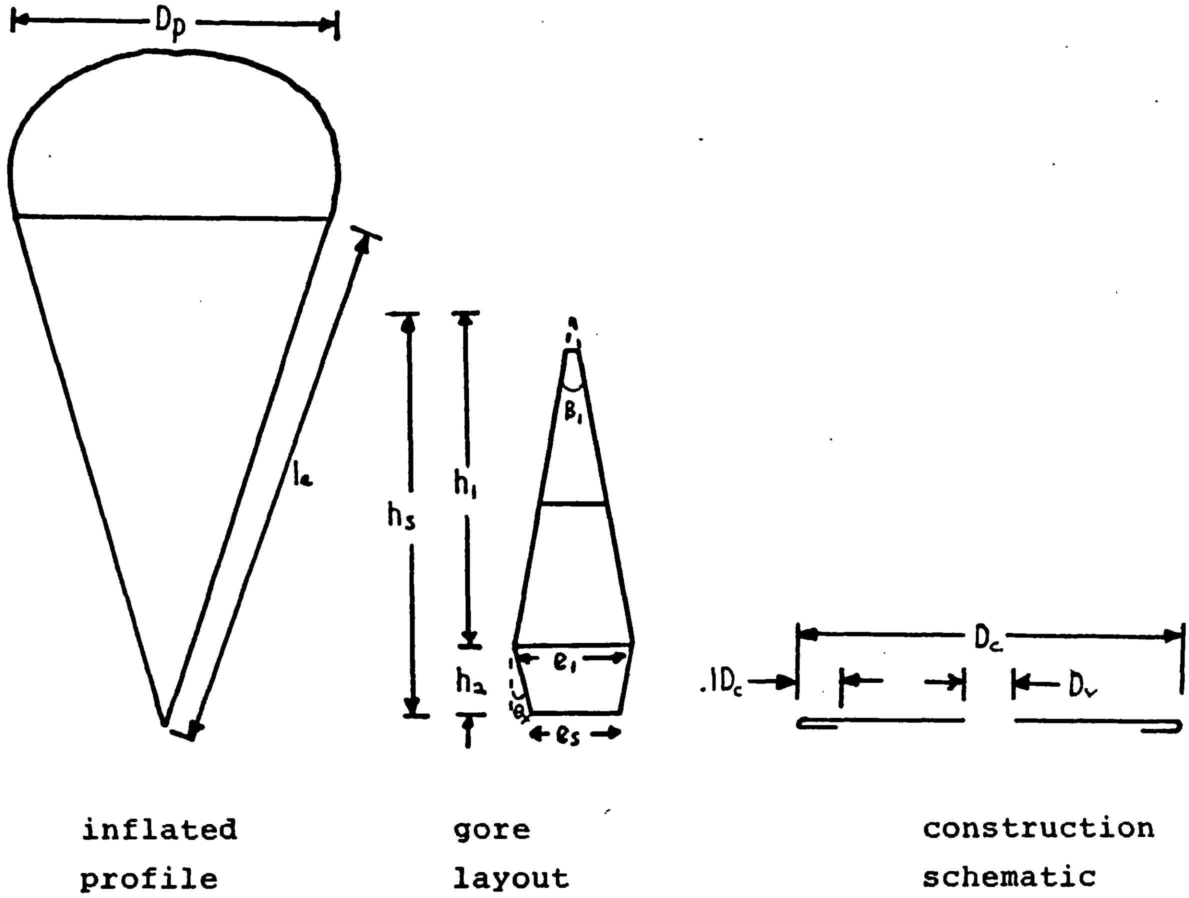


Figure A.5 Extended Skirt Canopy Configuration

gore angle: $\beta_1 = 360/N$ degrees

A.5.1 Canopy Weight Calculation

From reference 9

gore height $h_1 = 0.858(S_0/(N \tan(\beta_1/2)))^{1/2}$ (A.72)

maximum gore width $e_1 = 2\ h_1 \tan(\beta_1/2)$ (A.73)

$e_s = 0.81\ e_1$ (A.74)(1281)

$h_2 = 0.2\ h_1$ (A.75)(1282)

$$\theta_x = \tan^{-1}((e_1 - e_S)/(2\ h_2)) \qquad (A.76)(1283)$$

NLP is the number of lower panels, i.e. the number of fabric panels in the extended skirt. NUP is the number of upper gore panels. In the skirt, gore widths and areas are calculated as follows:

$$e_B(1) = e_S + 2\ h_2\ \tan\theta_x/NLP \qquad (A.77)(1290)$$

$$area_B(1) = ((e_B(1) + e_S + 0.1)/2)(h_2/NLP + 0.05) \qquad (A.78)(1290a)$$

if NLP is not equal to 1 then:

$$e_B(2) = e_B(1) + 2\ h_2\ \tan\theta_x/NLP \qquad (A.79)(1291)$$

$$area_B(2) = ((e_B(2) + e_B(1) + 0.1)/2)(h_2/NLP + 0.05) \qquad (A.80)(1292)$$

A.79 and A.80 are repeated to $e_B(NLP)$ and $area_B(NLP)$. The vent area, S_V, and gore height to the vent, h_{1g}, are obtained from equations A.3 and A.30. Then the upper fabric areas and gore widths, plus the total fabric area and weight, are obtained using the same method as that used for the bi-conical parachute: equations A.36 to A.41.

The weights of the reinforcing, rigging lines, vent band, skirt band, and vent lines are obtained from equations A.44, A.45, A11, A12, A46, A47, A19, A20, A48 and A.49. The weight of the radial tapes is WTT gm/m, tape length (m) and weight (kg):

$$TOT = N(\ h_{1g}/\cos(\beta_1/2) + h_2/\cos\theta_x + 0.2) \qquad (A.81)$$

$$WT = TOT.WTT/1000 \qquad (A.82)(1602)$$

finally, the total canopy weight is obtained from equation A.23.

A.5.2 Inputs Required for the "paradesign" Computer Program

(i) v_x, the vent area as a percentage of total canopy area, S_0.

(ii) NUP and NLP, the number of upper gore panels and the number of lower (skirt) panels.

A.6 Type 6 - Extended Skirt 14.3% Full Parachute

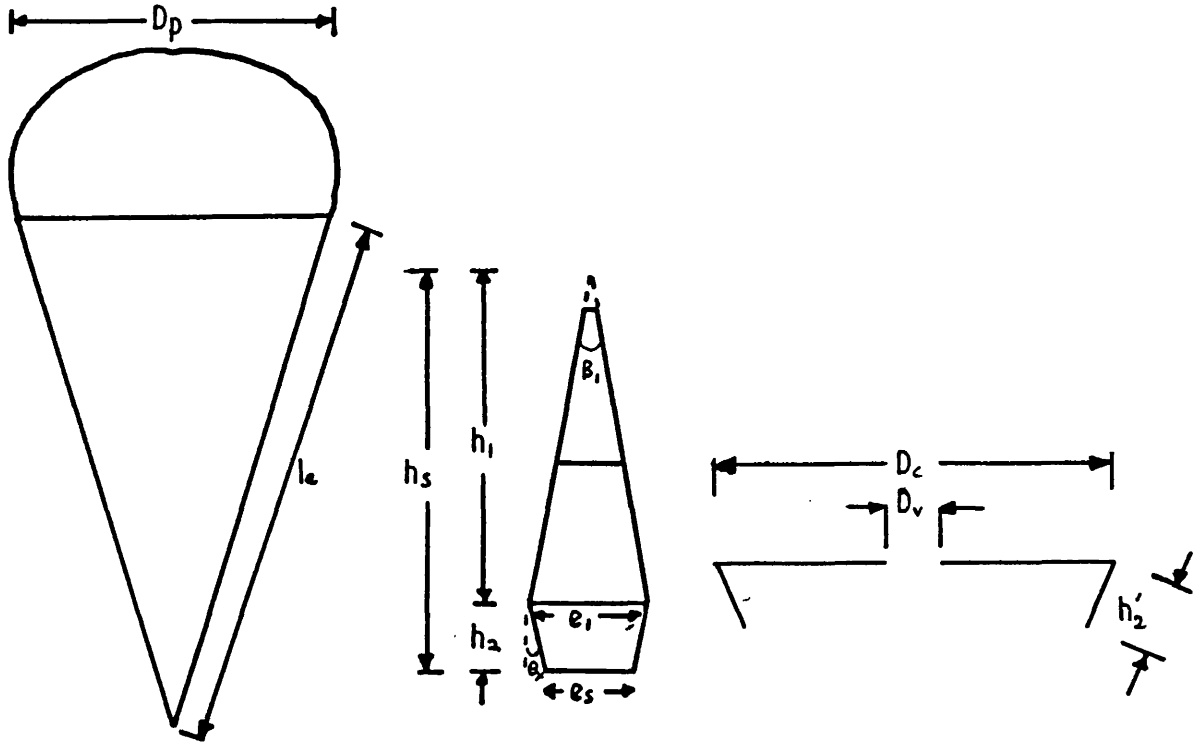


Figure A.6 Full Extended Skirt Canopy Configuration

gore angle: β_1 = 360/N degrees

A.6.1 Canopy Weight Calculation

From reference 9, the maximum gore width:

$$e_1 = 0.81\ D \sin(180/N) \qquad (A.83)(1026)$$

Also from reference 9:

$$h_1 = 0.405\ D \cos(\beta_1/2) \qquad (A.84)$$

$$e_S = e_1\ l_e/(l_e + 0.116\ D) \qquad (A.85)(1286)$$

$$h_2' = 0.116\ D \qquad (A.86)(1287)$$

$$\theta_x = \sin^{-1}((e_1 - e_S)/(2\ h_2')) \qquad (A.87)(1288)$$

$$h_2 = h_2' \cos\theta_X \qquad (A.88)(1289)$$

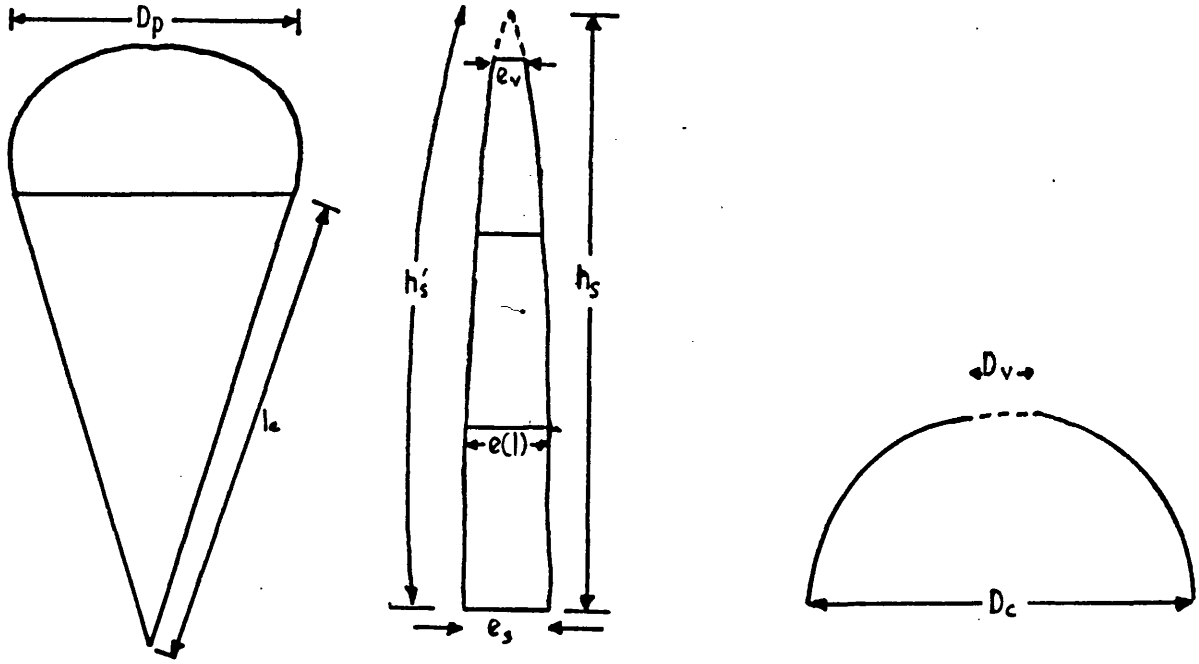
The canopy weight calculation is now the same as that for the flat extended skirt parachute, section A.5.2, starting with equation A.77.

A.6.2 Inputs Required for the Computer Program

(i) v_x, the vent area as a percentage of S_0, the total canopy area.

(ii) NLP and NUP, the number of lower (skirt) panels and the number of upper gore panels.

A.7 Type 7 - Hemispherical Parachute



inflated profile | gore details | construction schematic

Figure A.7 Hemispherical Parachute Configuration

A.7.1 Canopy Weight Calculation

From reference 9, the maximum gore width:

$$e_s = 0.7\ D\ \pi/N \qquad (A.89)(1028)$$

The canopy vent area is calculated using equation A.3. Then the vent diameter, D_v, and the gore width at the vent, e_v, are calculated as follows:

$$D_v = (2\ S_v/\pi)^{1/2} \qquad (A.90)(1050,1614,1682)$$

$$e_v = \pi\ D_v/N \qquad (A.91)(1615,1683)$$

and the gore heights:

$$h_s' = \pi\ 0.7\ D/4 \qquad (A.92)$$

$$h_g' = h_s' - D_v/2 \qquad (A.93)(1618)$$

If NP is the number of fabric panels per gore, then the gore width at, and the area of, the largest panel, including sewing allowance, can be approximated as:

$$e(1) = (e_v - e_s)/NP + e_s \qquad (A.94)(1619)$$

$$area(1) = (e_s + e(1) + 0.1)(h_g'/NP + 0.05)/2 \qquad (A.95)(1621)$$

if NP is not equal to 1 then the gore widths and the areas of all panels in the gore (e(2) to e(NP), area(2) to area(NP)) can be calculated from equations A.96 and A.97, where n is the panel number.

$$e(n) = (e_v - e_s)\ n/NP + e_s \qquad (A.96)(1620)$$

$$area(n) = (e(n-1) + e(n) + 0.1)(h_g'/NP + 0.05)/2 \qquad (A.97)(1622)$$

The total fabric area and weight is calculated as for the flat circular parachute, equations A.9 and A.10. Also the weight of the skirt band, reinforcing band, and rigging lines are calculated as for the flat circular parachute (equations A.19, A.20, A.15, A.16, A.11 and A.12).

WTT is the radial tape weight (gm/m). Total length (m) and weight (kg) of the radial tapes:

$$TOT = N\ (h_g' + 0.2) \qquad (A.98)(1627)$$

$$WT = TOT.WTT/1000 \qquad (A.99)(1628)$$

WTVB is the vent band weight (gm/m). The total length (m) and weight (kg) of the vent band:

$$TVB = N.e_v + 0.2 \qquad (A.100)(1634)$$

$$WVB = TVB.WTVB/1000 \qquad (A.101)(1635)$$

WTVL is the weight of the vent line material (m/kg). The total length (m) and weight (kg) of the vent line is:

$$TOVL = N\ (D_V/2 + 0.2) \qquad (A.102)(1638)$$

$$WVL = TOVL/WTVL \qquad (A.103)(1639)$$

The total canopy weight is the sum of the weights of its components, equation A.23.

A.7.2 Inputs Required for the "paradesign" Computer Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.

(ii) NP, the number of fabric panels per gore.

A.8 Type 11 - Cruciform Parachute

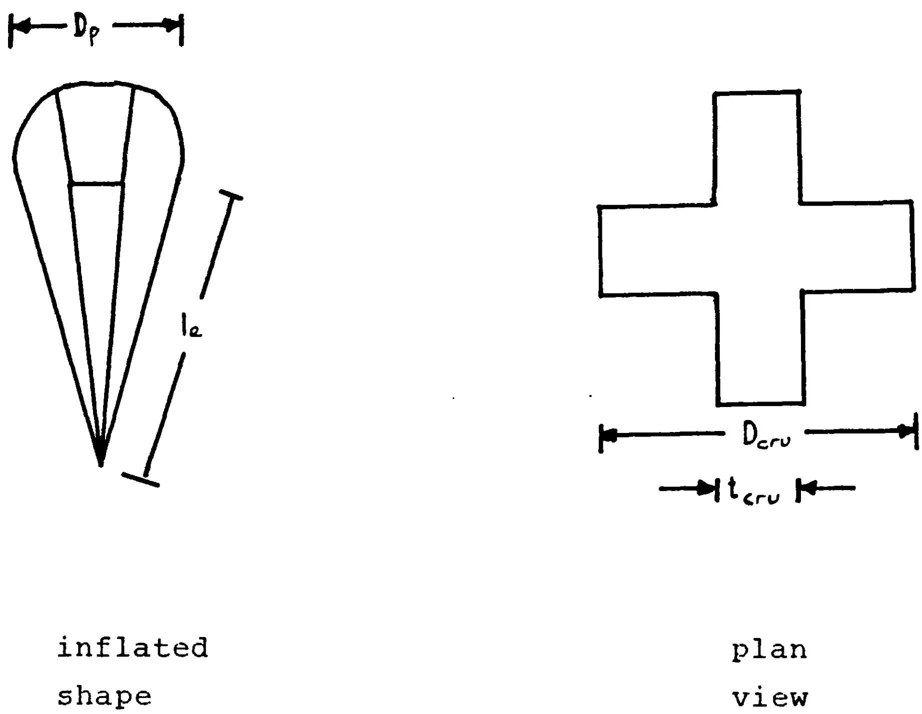


Figure A.8 Cruciform Canopy Configuration

A.8.1 Canopy Weight Calculation

The maximum gore width (for use in canopy fabric selection, section 3.11.1) is:

$$e_s = t_{cru}/N \qquad (A.104)(1031)$$

The weight (gm/m^2) of the canopy fabric is WTF. The total fabric area (m^2) and weight (kg) is:

$$TOF = 4\ N\ (e_s + 0.05)((d_{cru} - N\ e_s)/2 + 0.05) + N\ (e_s + 0.05)\ N\ (e_s + 0.05) \qquad (A.105)(1642)$$

$$WF = TOF.WTF/1000 \qquad (A.106)(1643)$$

The weight of the rigging lines, WL, is calculated using the same method as used for the flat circular parachute, section A.1.1, equations A.11 and A.12. The tape material weight (gm/m) is WTT, the total tape length, including sewing allowance, and weight:

$$TOT = Z/2\ (d_{cru} + 0.2) + 4\ (N\ e_s + 0.2) \qquad (A.107)(1646)$$

$$WT = TOT.WTT/1000 \qquad (A.108)(1647)$$

The total parachute weight:

$$WTOT = WT + WL + WF \qquad (A.109)(1648)$$

As in many of the previous calculations the cruciform canopy weight is simple to determine, compared with the other canopy types available.

A.8.2 Inputs Required for the Parachute Design Program

(i) The cruciform canopy arm ratio, d_{cru}/t_{cru}.

A.9 Type 12 - Flat Ribbon Parachute

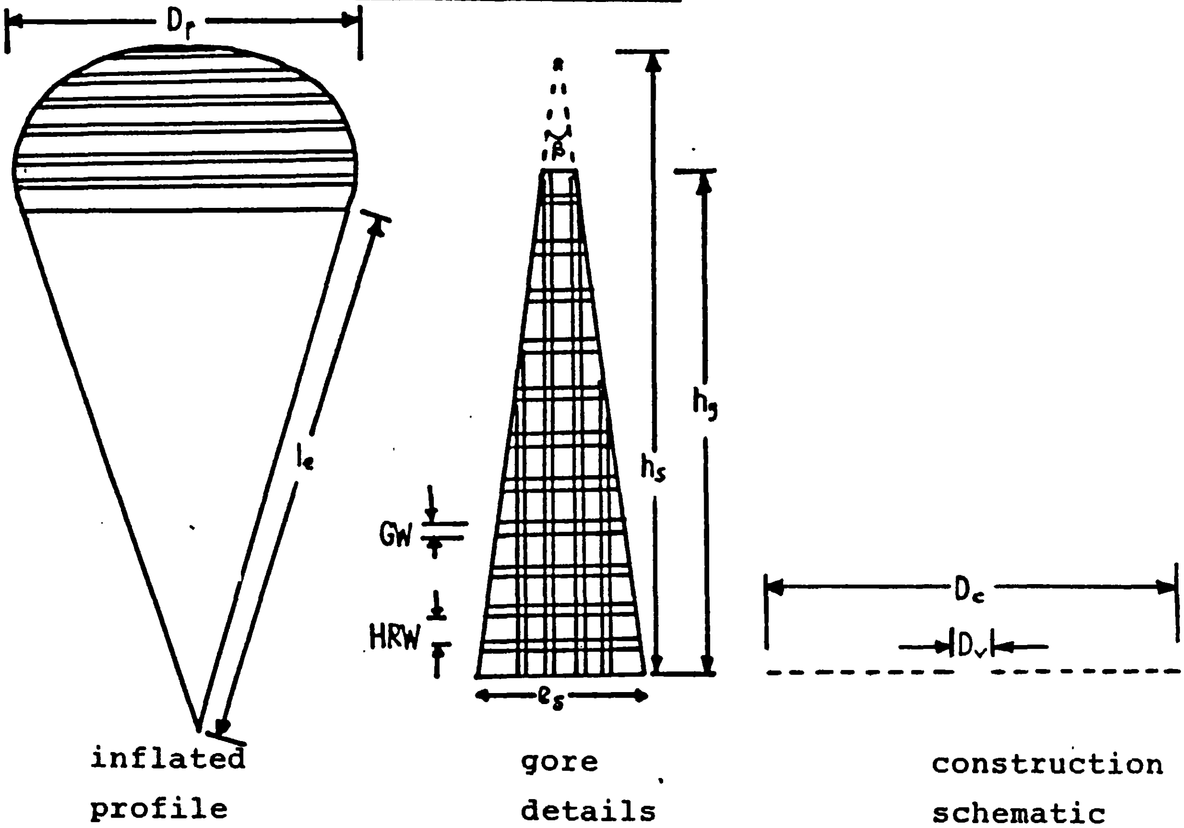


Figure A.9 Flat Ribbon Parachute Configuration

Key to Figure A.9

h_s, h_g	- gore heights
HRW	- horizontal ribbon width
GW	- gap width
NVT	- number of vertical tapes
NHR	- number of horizontal ribbons
e_s	- maximum gore width
e(1), e(2), ..., e(NHR-1)	- horizontal ribbon widths
β	- gore angle = $360/N^{o}$

A.9.1 Canopy Weight Calculation

The maximum gore width, e_s, is calculated as for the flat circular parachute, section A.1.1, equations A.1 and A.2. The vent constructed area, and the gore height, h_g, are calculated using equations A.3 and A.4. Then the ribbon widths:

$$e(1) = e_s - 2\ (GW + HRW)\ \tan(\beta/2) \qquad (A.110)(1658)$$

$$e(2) = e(1) - 2\ (GW + HRW)\ \tan(\beta/2) \qquad (A.111)(1659)$$

equation A.111 is repeated up to e(NHR-1). The total horizontal ribbon length (m) and weight (kg), taking WTHR as the horizontal ribbon material weight (g/m) are:

$$THR = N.e_s + 0.2 + \sum_{x=1}^{(NHR-1)} (N.e(x) + 0.2) \qquad (A.112)(1660)$$

$$WHR = THR.WTHR/1000 \qquad (A.113)(1661)$$

The weights of the rigging lines, radial tapes, skirt band and vent lines are calculated as for the flat circular parachute, section A.1.3, equations A.11 to A.14 and A.19 to A.22.

NVT is the number of vertical tapes per gore, and TVT is their total length. If NVT=0 TVT=0. If NVT=1 then TVT=h_g+0.2 (1667,1740a). If NVT is greater than 1 and odd then:

$$TVT = N\ (h_g + |(NVT/2)|.h_g + NVT\ 0.2) \qquad (A.114)\ (1667a,1741)$$

and if NVT is even then:

$$TVT = \sum_{x=1}^{|NVT/2|} 2\ (2 \times h_g/(NVT + 1) + 0.2) \qquad (A.115)(1667b,1741a)$$

WTVT is the vertical tape material weight (g/m), weight of vertical tapes (kg):

$$WVT = TVT.WTVT/1000 \qquad (A.116)(1668,1742)$$

WTVB is the weight of the vent band material (gm/m), the vent band length and weight:

$$TVB = N.e(NHR-1) + 0.2 \qquad (A.117)(1673)$$

$$WVB = TVB.WTVB/1000 \qquad (A.118)(1674)$$

total ribbon parachute weight:

$$WTOT = WVB + WVT + WHR + WL + WT + WSB + WVL \qquad (A.119)(1725)$$

A.9.2 Inputs Required for the Parachute Design Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.

(ii) NHR, the number of horizontal ribbons.

(iii) NVT, the number of vertical tapes per gore.

A.10 Type 13 - Conical Ribbon Parachute

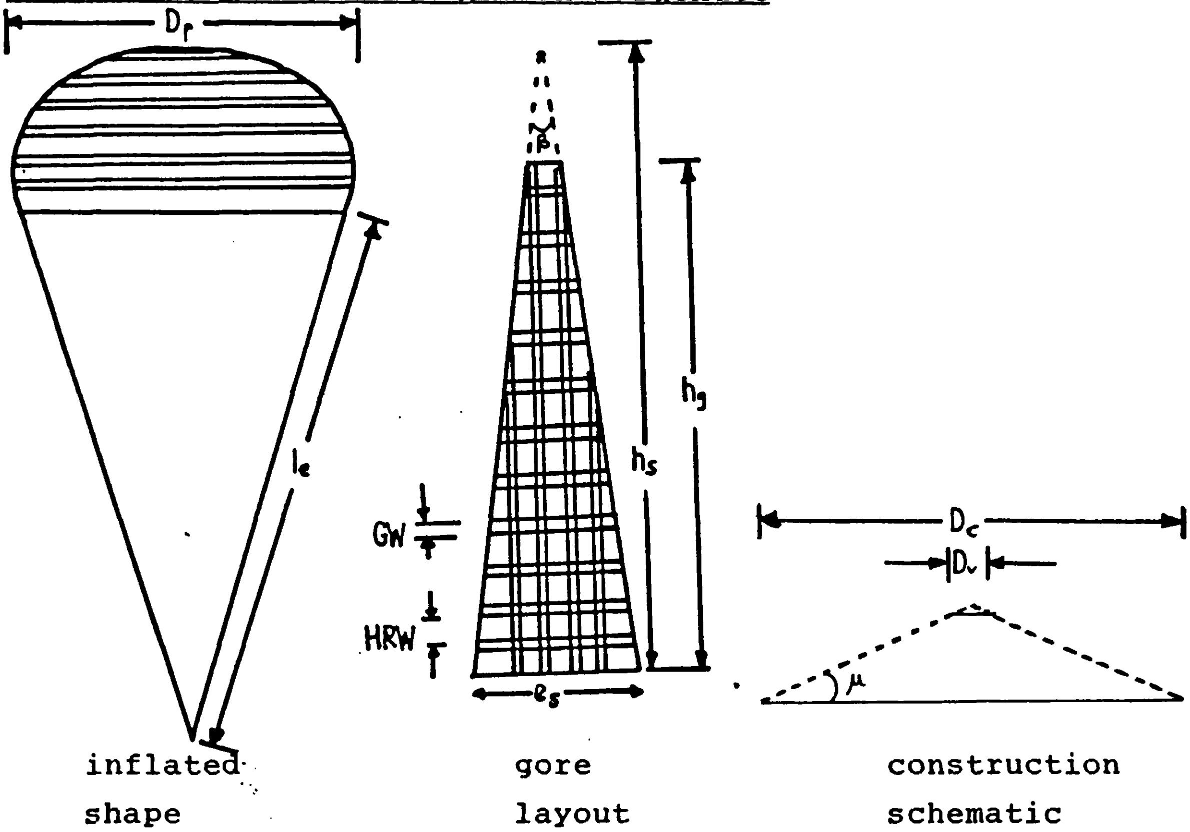


Figure A.10 Conical Ribbon Parachute Configuration
(see figure A.9 for a key to ribbon parachutes)

μ is the canopy constructed cone angle. Gore angle β is calculated as for the conical parachute, equation A.24.

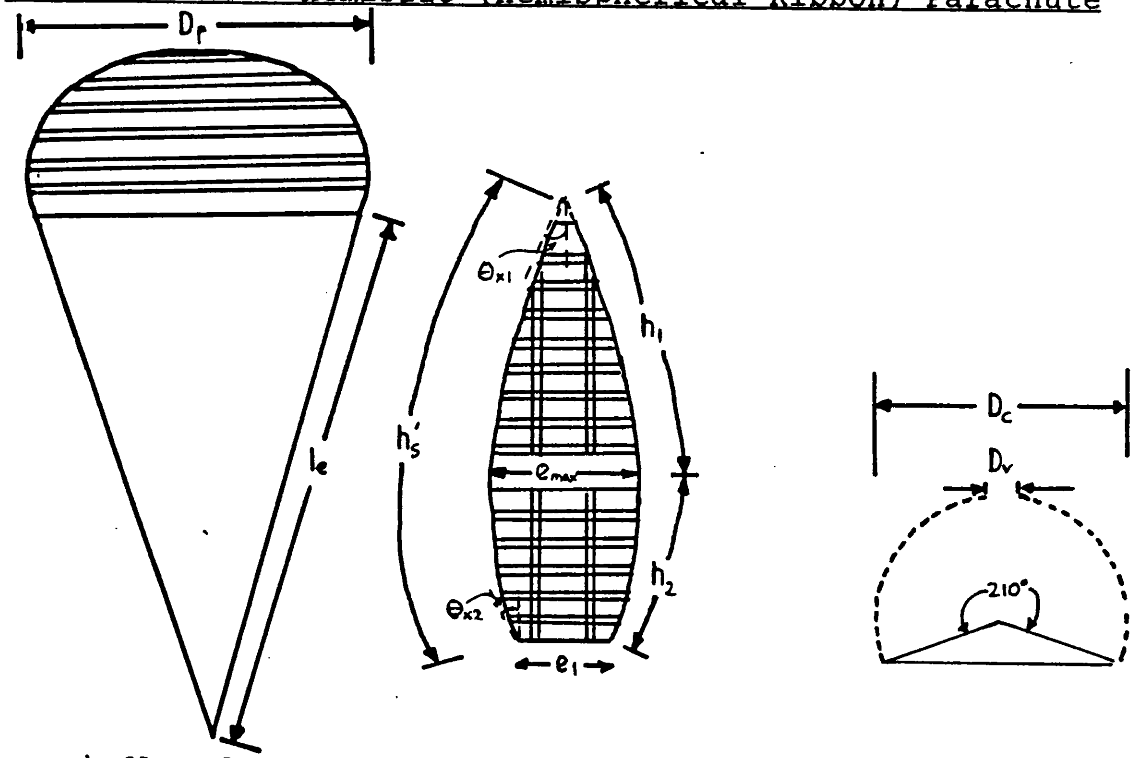
A.10.1 Canopy Weight Calculation

The canopy weight calculation is the same as for the flat ribbon parachute, section A.9.1.

A.10.2 Inputs Required for the "paradesign" Parachute Design Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.
(ii) NHR, the number of horizontal ribbons.
(iii) NVT, the number of vertical tapes per gore.
(iv) μ, the canopy constructed cone angle.

A.11 Type 15 - Hemisflo (Hemispherical Ribbon) Parachute



inflated shape

gore details

construction schematic

Figure A.11 Hemisflo Parachute Configuration

A.11.1 Canopy Weight Calculation

Reference 9 gives:

constructed diameter $D_c = (360\ S_0/(210\ \pi))^{1/2}$ (A.120)(1675)

$$e_{max} = \pi\ D_c/N \quad \text{(A.121)(1676)}$$

and it can be deduced that:

$$e_1 = \pi\ D_c\ \sin 75/N \quad \text{(A.122)}$$

$$h'_S = 0.9163\ D_c \quad \text{(A.123)(1678)}$$

$$h_1 = 6\ h'_S/7 \quad \text{(A.124)(1679)}$$

$$h_2 = h_s'/7 \quad (A.125)(1680)$$

$$\theta_{x2} = \sin^{-1}((e_{max} - e_1)/2\ h_2) \quad (A.126)(1685)$$

$$\theta_{x1} = \sin^{-1}(e_{max}/(2\ h_1)) \quad (A.127)(1686)$$

The gore width at the vent, and the vent diameter can be calculated as for the hemispherical parachute, section A.7.2, equations A.3, A.90 and A.91. the gore height:

$$h_g = h_1 - D_V/2 \quad (A.128)(1684)$$

NHR is the number of horizontal ribbons. Starting at the bottom of the gore, ribbon widths:

$$e(1) = e_1 + 2\ HRW\ \tan\theta_{x2} \quad (A.129)(1695)$$

$$e(2) = e(1) + 2(HRW + GW)\ \tan\theta_{x2} \quad (A.130)$$

Equation A.130 is repeated up to e(n) which is equal to e_{max}.

For the upper gore:

$$zz = h_2\ \cos\theta_{x2} - n(HRW + (n-1)\ GW) \quad (A.131)$$

$$e(n+1) = e_{max} - 2(GW - zz)\ \tan\theta x_1 \quad (A.132)$$

$$e(n+2) = e(n+1) - 2(HRW - zz)\ \tan\theta_{x1} \quad (A.133)$$

Equation A.133 is repeated to e(NHR). WTHR is the weight (gm/m) of the horizontal ribbons. The total horizontal ribbon length (m) and weight (kg):

$$THR = \sum_{x=1}^{NHR} (N\ e(x) + 0.2) \quad (A.134)$$

$$WHR = THR.WTHR/1000 \quad (A.135)(1705)$$

The weight of the lines is calculated as for the flat circular parachute, equations A.11 and A.12. The weights of the vent band and the vent lines are calculated as for the hemispherical parachute, equations A.100 to A.103. The skirt band material weight (gm/m) is WTSB. The total length (m) and weight (kg) of the skirt band:

$$TSB = N.e_S + 0.2 \qquad (A.136)(1719)$$

$$WSB = TSB.WTSB/1000 \qquad (A.137)(1720)$$

WTT is the weight of the radial tapes (gm/m), the total tape length (m) and weight (kg):

$$TOT = N\ (h_g + h_2 + 0.2) \qquad (A.138)(1708)$$

$$WT = TOT.WTT/1000 \qquad (A.139)(1709)$$

NVT is the number of vertical tapes per gore, and WTVT is the vertical tape material weight (gm/m).

$$h'_g = h_g\ \cos\theta_{x1} + h_2\ \cos\theta_{x2} + 0.2 \qquad (A.140)$$

If NVT is 1 then TVT, the total length of the vertical tapes:

$$TVT = N\ (h'_g + NVT\ 0.2) \qquad (A.141)$$

If NVT is greater than 1 and odd then:

$$TVT = N\ (h'_g + |(NVT/2)|\ h'_g + NVT\ 0.2) \qquad (A.142)(1712)$$

If NVT is even then:

$$TVT = N\Big(\sum_{x=1}^{|NVT/2|} (2\ (x\ 2\ h'_g)/(NVT + 1)\Big) + NVT.0.2 \qquad (A.143)(1713)$$

and: $WVT = TVT.WTVT/1000$ (A.144)(1718)

The total parachute weight is calculated from equation A.119.

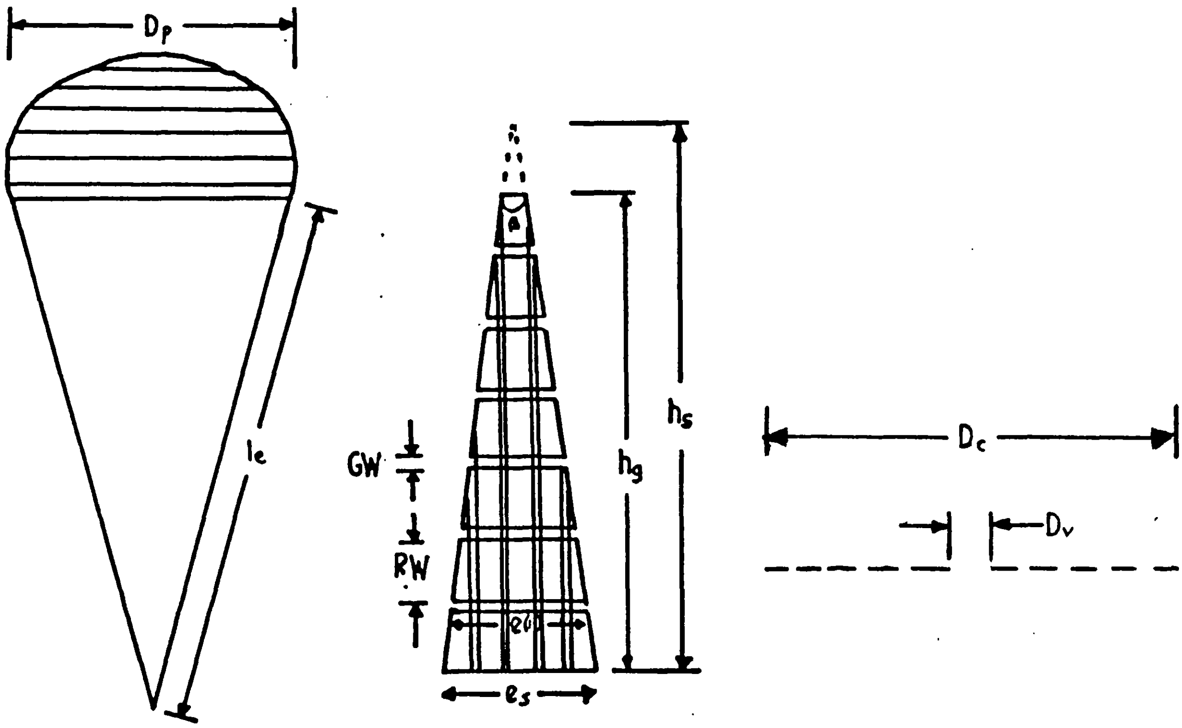
A.11.2 Inputs Required for the Parachute Design Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.

(ii) NHR, the number of horizontal ribbons.

(iii) NVT, the number of vertical tapes per gore.

A.12 Type 16a - Flat Ringslot Parachute



key

β - gore angle (= 360/N degrees)

e_s - maximum gore width

RW - ring width

GW - gap width

NR - number of rings

D_c - constructed diameter

D_p - flying diameter

Figure A.12 Flat Ringslot Parachute Configuration

A.12.1 Canopy Weight Calculation

Using the methods of section A.1.1, flat circular parachute weight calculation, gore heights h_s and h_g and the maximum gore width, e_s are calculated (equations A.1 to A.4). The ring width:

$$RW = (h_g - (NR - 1)GW)/NR \qquad (A.145)(1043,1047)$$

Gore widths and ring areas, including sewing allowance:

$$e(1) = e_s - 2\ RW \tan(\beta/2) \qquad (A.146)$$

$$area(1) = (e(1) + e_s + 0.1)(RW + 0.05)/2 \qquad (A.147)(1730)$$

$$e(2) = e(1) - 2\ GW \tan(\beta/2) \qquad (A.148)$$

$$e(3) = e(2) - 2\ RW \tan(\beta/2) \qquad (A.149)$$

$$area(2) = (e(3) + e(2) + 0.1)(RW + 0.05)/2 \qquad (A.150)(1733)$$

Equations A.148, A.149 and A.150 are repeated to e(2 NR - 2), e(2 NR - 1) and area(NR). WTF is the weight of fabric used for the rings (gm/m^2). The total area (m^2) and weight (kg) of the rings is:

$$TOF = N\ (\sum_{x=1}^{NR} area(x)) \qquad (A.151)(1734,1756)$$

$$WF = TOF.WTF/1000 \qquad (A.152)(1735,1757)$$

The weights of the rigging lines, radial tapes, skirt band and vent lines are calculated as for the flat circular parachute, section A.1.1, equations A.11 - A.14, A.19 - A.22. The weight of the vertical tapes is calculated as for the ribbon parachute, section A.9.1, equations A.114 to A.116. WTVB is the vent band weight (gm/m). The total length (m) and weight (kg) of the vent band:

$$TVB = N.e(2\ NR - 1) + 0.2 \qquad (A.153)(1745)$$

$$WVB = TVB.WTVB/1000 \qquad (A.154)(1746)$$

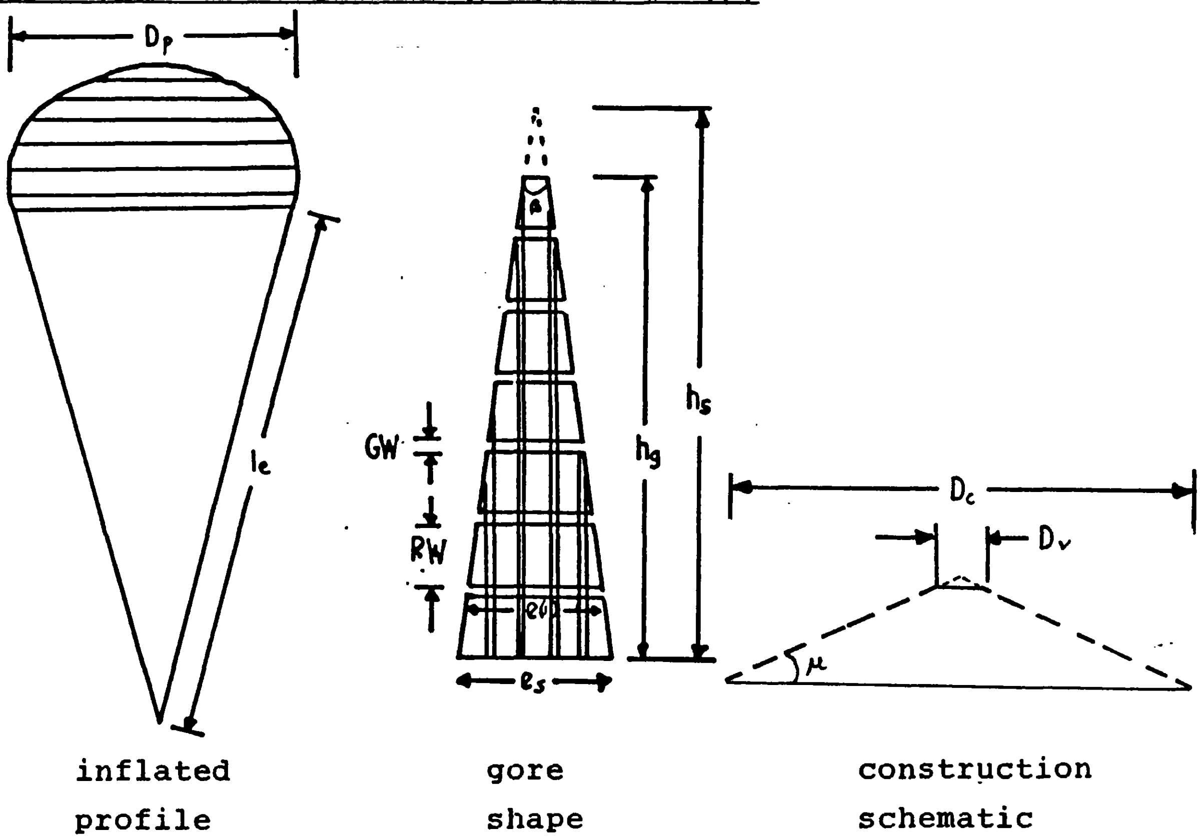
Finally the total canopy weight:

$$WTOT = WVB + WSB + WVT + WT + WL + WF + WVL \qquad (A.155)(1749)$$

A.12.2 Inputs Required for the "paradesign" Computer Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.
(ii) NR, the number of rings in the canopy.

A.13 Type 16b - Conical Ringslot Canopy



key as in figure A.12, μ is the canopy constructed angle.

Figure A.13 Conical Ringslot Parachute Configuration gore angle $\beta = 2\ \sin^{-1}(\sin(180/N)\ \cos\mu)$ (A.156)

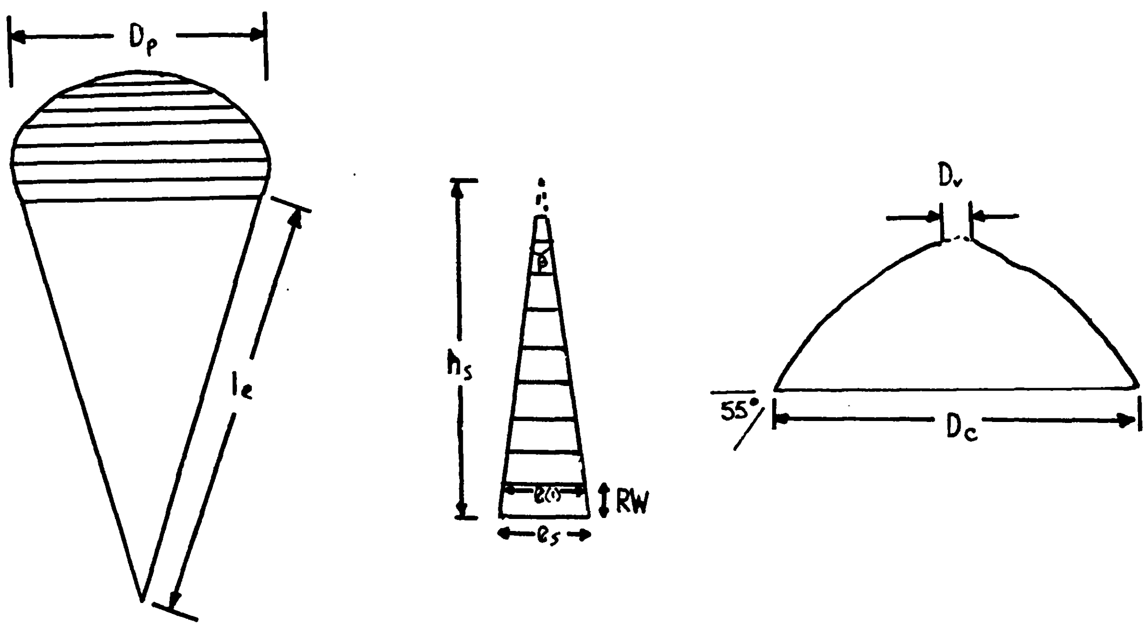
A.13.1 Canopy Weight Calculation

This calculation for the conical ringslot parachute is the same as that in section A.12.1 for the flat ringslot parachute.

A.13.2 Inputs Required for the "paradesign" Computer Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.
(ii) NR, the number of rings in the canopy.
(iii) μ, the canopy constructed cone angle.

A.14 Type 17 - Ringsail Parachute



inflated profile   gore layout   construction schematic

Figure A.14 Ringsail Parachute Configuration

A.14.1 Parachute Weight Calculation

From reference 9,

gore height $h_s = 0.519\ D$ (A.157)

maximum gore width $e_s = 5.21(h_s/N)$ (A.158)

gore angle $\beta = 2\tan^{-1}(e_s/2h_s)$ (A.159)

The vent area and vent diameter, S_v and D_v are calculated as for the hemispherical parachute, section A.7.2. The gore height and fabric ring width:

$h_g = h_s - 0.519\ D_v$ (A.160)(1051)

$RW = h_g/NR$ (A.161)(1052)

Gore widths and fabric ring areas for the ringsail parachute:

$e(1) = e_s - 2\ RW\ \tan(\beta/2)$ (A.162)

$area(1) = ((e(1) + e_s + 0.1)/2)\ (RW + 0.05)$ (A.163)(1753)

$e(2) = e(1) - 2\ RW\ \tan(\beta/2)$ (A.164)

$area(2) = ((e(2) + e(1) + 0.1)/2)\ (RW + 0.05)$ (A.165) (1755)

Equations A.164 and A.165 are repeated to e(NR) and area(NR), where NR is the number of rings. The fabric weight is then calculated as for the flat ringslot parachute, equations A.151 and A.152. Rigging line and skirt band weights are calculated as for the flat circular parachute, equations A.11, A.12, A.19 and A.20. WTVB is the vent band material weight (gm/m). Total vent band length (m) and weight (kg):

$TVB = N.e(NR) + 0.2$ (A.166)(1762)

$$WVB = TVB.WTVB/1000 \qquad (A.167)(1763)$$

WTVT is the vertical tape material weight (gm/m). The total vertical tape length (m) and weight (kg):

$$TVT = N\ (h_g + 0.2) \qquad (A.168)(1760)$$

$$WVT = TVT.WTVT/1000 \qquad (A.169)(1761)$$

WTVL is the vent line weight (m/kg). The total vent line length (m) and weight (kg):

$$TOVL = N\ (h_S - h_g + 0.2) \qquad (A.170)(1764)$$

$$WVL = TOVL/WTVL \qquad (A.171)(1765)$$

Total canopy weight:

$$WTOT = WSB + WVL + WVB + WVT + WL + WT \qquad (A.172)(1768)$$

A.14.2 Inputs Required for the Parachute Design Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.
(ii) NR, the number of fabric rings in the canopy.

A.15 Type 18 - Disk-Gap-Band Parachute

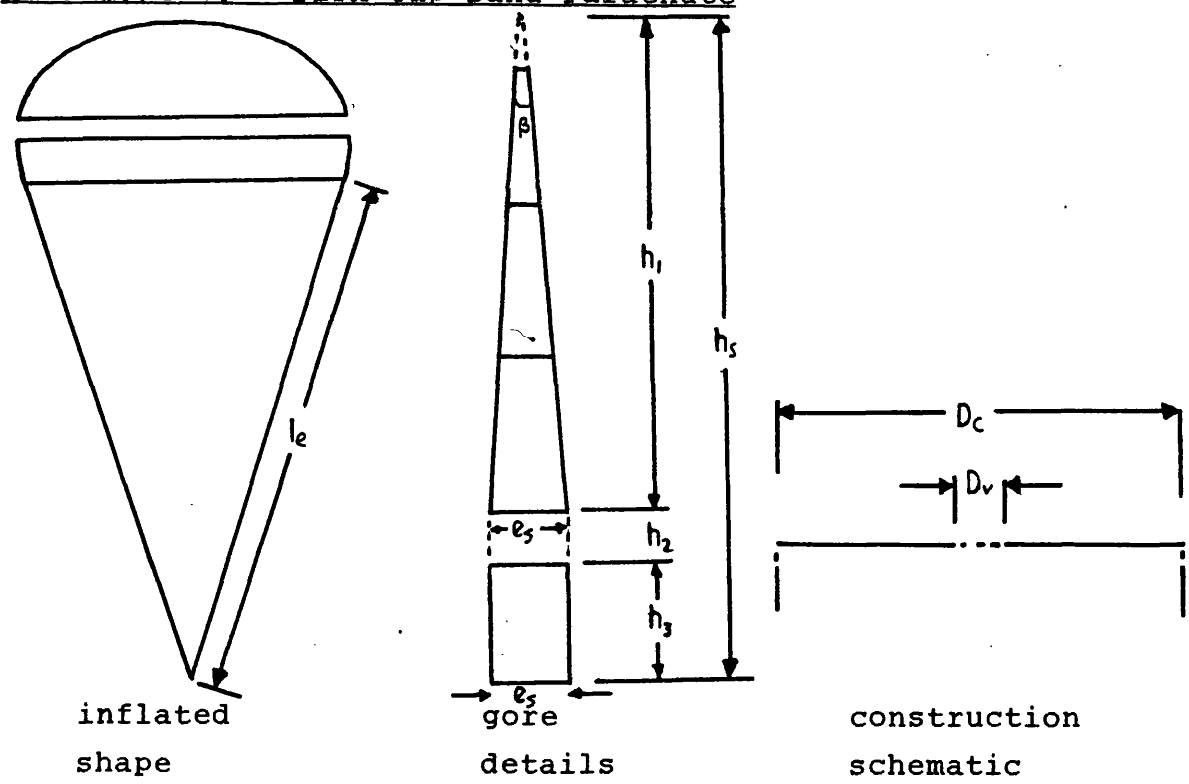


Figure A.15 Disk-Gap-Band Parachute Configuration

gore angle β = 360/N degrees

A.15.1 Canopy Weight Calculation

From reference 9

gore height $h_1 = (S_0/(1.887\ N \tan(\beta/2)))^{1/2}$ (A.173)(1036)

maximum gore width $e_S = 2\ h_1\ \tan(\beta/2)$ (A.174)(1037)

The vent area, S_V, and gore height, h_g, are calculated as for the flat circular parachute, section A.1.1, equations A.3 and A.4. Gore heights from reference 9:

$h_2 = 0.113\ h_1$ (A.175)(1772)

$h_3 = 0.33\ h_1$ (A.176)(1773)

NUP is the number of upper panels and NLP is the number of lower panels. The total area of the lower panels, including sewing allowance, is:

$$TLP = NLP\ (e_S + 0.05)(h_3/NLP + 0.05) \qquad (A.177)(1774)$$

The upper gore width and the largest upper panel area:

$$e(1) = e_S - 2\ h_g\ \tan(\beta/2)/NUP \qquad (A.178)$$

$$area(1) = (e_S + e(1) + 0.1)(h_g/NUP + 0.05)/2 \qquad (A.179)(1776)$$

if NUP is not equal to 1 then:

$$e(2) = e(1) - 2\ h_g\ \tan(\beta/2)/NUP \qquad (A.180)$$

$$area(2) = (e(1) + e(2) + 0.1)(h_g/NUP + 0.05)/2 \qquad (A.181)(1778)$$

Equations A.180 and A.181 are repeated to e(NUP) and area(NUP). The total upper fabric area:

$$TUP = \sum_{x=1}^{NUP} area(x) \qquad (A.182)(1779)$$

and if WTF is the fabric weight (gm/m^2), the total fabric area (m^2) and weight (kg):

$$TOF = N\ (TLP + TUP) \qquad (A.183)(1780)$$

$$WF = TOF.WTF/1000 \qquad (A.184)(1781)$$

The weights of the rigging lines and skirt band are calculated as for the flat circular parachute, equations A.11, A.12, A.19 and A.20. WTT is the radial tape weight (gm/m), the total length (m) and weight (kg) of the tapes:

$$TOT = N\ (h_g/\cos(\beta/2) + h_2 + h_3 + 0.2) \qquad (A.185)(1784)$$

$$WT = TOT.WTT/1000 \qquad (A.186)(1785)$$

WTR is the weight of the reinforcing (gm/m). The total reinforcing length and weight is:

$$TR = 2\ (N\ e_s + 0.2) \qquad (A.187)(1786)$$

$$WR = TR.WTR/1000 \qquad (A.188)(1787)$$

WTVB is the weight of the vent band (gm/m). The total vent band length (m) and weight (kg):

$$TVB = N.e(NP) + 0.2 \qquad (A.189)(1788)$$

$$WVB = TVB.WTVB/1000 \qquad (A.190)(1789)$$

WTVL is the vent line weight (m/kg). The total length (m) and weight (kg) of the vent lines is:

$$TOVL = N\ ((h_1 - h_g)/\cos(\beta/2) + 0.2) \qquad (A.191)(1792)$$

$$WVL = TOVL/WTVL \qquad (A.192)(1793)$$

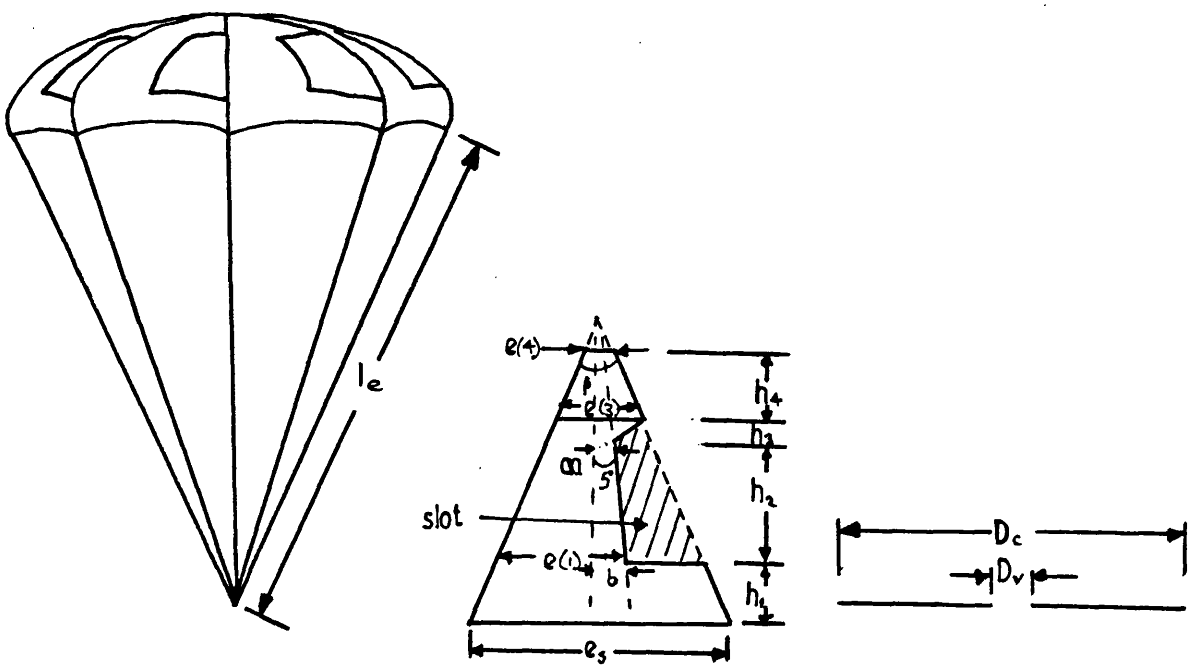
The total canopy weight is calculated using equation A.23.

A.15.2 Inputs Required for the Computer Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.

(ii) NUP and NLP, the number of upper and lower fabric panels per gore.

A.16 Type 19 - Rotafoil Parachute



inflated shape | gore profile | construction schematic

Figure A.16 Rotafoil Parachute Configuration

gore angle β = 360/N degrees

A.16.1 Canopy Weight Calculation

The maximum gore width is calculated as for the flat circular parachute equations A.1 and A.2.

The vent area, S_V, and the gore height, h_g, are known from equations A.3 and A.4. The gore heights h_2, h_3 and h_4 must be supplied. Height h_1:

$$h_1 = 0.105\ D \cos(\beta/2) \qquad \text{(A.193)}$$

Then the gore widths and areas are calculated as follows:

$$e(1) = e_S - 2\ h_1\ \tan(\beta/2) \qquad (A.194)$$

$$area(1) = (e_S + e(1) + 0.1)\ (h_1 + 0.05)/2 \qquad (A.195)(1804)$$

let $aa = (h_4 + h_3 + h_S - h_g)\ \tan 5$ (A.196)(1805)

and $b = (h_2 + h_3 + h_4 + h_S - h_g)\ \tan 5$ (A.197)(1806)

then $e(2) = (e(1) - 2\ h_2\ \tan(\beta/2))/2 + aa$ (A.198)

$$area(2) = (h_2 + 0.05)\ (3\ e(2)/2 + e(1)/4 + 0.025 + b/2) \qquad (A.199)(1808)$$

$$e(3) = e(1) - 2\ (h_2 + h_3)\ \tan(\beta/2) \qquad (A.200)$$

$$area(3) = (h_3 + 0.05)\ (e(2) - h_3\ \tan(\beta/2)/2 + 0.025 + e(3)/4 - aa/2) \qquad (A.201)$$

$$e(4) = e(3) - 2\ h_4\ \tan(\beta/2) \qquad (A.202)$$

$$area(4) = (e(4) + e(3) + 0.1)\ (h_4 + 0.05)/2 \qquad (A.203)(1812)$$

WTF is the weight of the fabric (gm/m^2), total area (m^2) and weight (kg) of the fabric:

$$TOF = \sum_{x=1}^{4} area(x) \qquad (A.204)(1813)$$

$$WF = TOF.WTF/1000 \qquad (A.205)(1814)$$

The weights of the rigging lines, radial tapes, skirt band and vent lines are calculated as for the flat circular parachute, equations A.11-A.14, A.19-A.22. WTR is the reinforcing weight (gm/m). Total reinforcing length (m) and weight (kg):

$$TR = N.e(3) + 0.2 + N\ (e(1)/2 - b + 0.2) + N(h_2/\cos 5 + 0.2) + N\ (((e(3)/2 - a)^2 + h_3^2)^{1/2} + 0.2) \qquad (A.206)(1819)$$

$$WR = TR.WTR/1000 \qquad (A.207)(1820)$$

WTVB is the vent band weight (gm/m). The total length (m) and weight (kg) of the vent band:

$$TVB = N.e(4) + 0.2 \qquad (A.208)(1821)$$

$$WVB = TVB.WTVB/1000 \qquad (A.209)(1822)$$

The total parachute weight is taken from equation A.23.

A.16.2 Inputs Required for the Computer Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.

(ii) The ratio of the gore heights, $h_2:h_3:h_4$.

A.17 Type 21 - Gliding Aeroconical Parachute

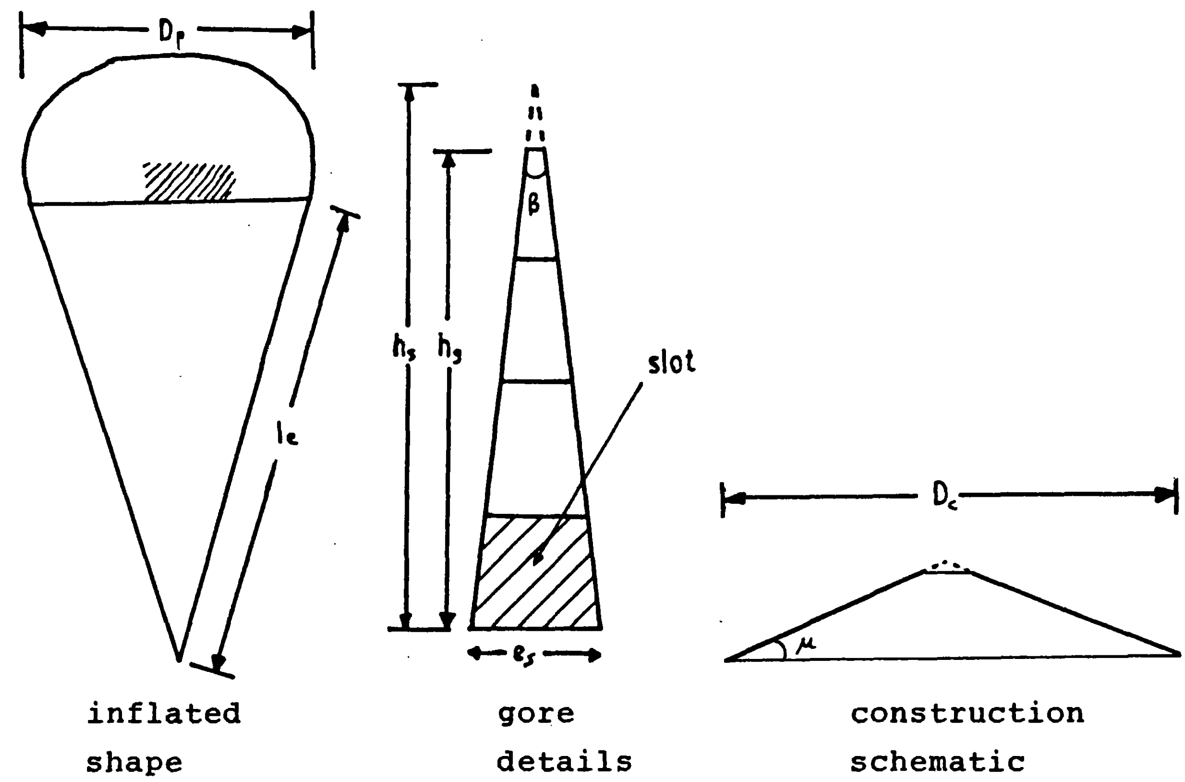


Figure A.17 Aeroconical Parachute Configuration gore angle $\beta = 2 \sin^{-1}(\sin(180/N) \cos\mu)$ (A.210)(1005)

where μ is the canopy constructed angle.

A.17.1 Canopy Weight Calculation

The fabric panel areas, area(1) to area(NP), where NP is the number of fabric panels per gore, are calculated using the same method as that for the flat circular parachute, equations A.1 to A.8. The area of the drive slots or missing panels must be taken into account in order to calculate the total fabric area for the aeroconical parachute. ONP(1) is the number of panel 1 (the largest) missing, ONP(2) is the number of panel 2 missing and so on up to ONP(NP). So the total aeroconical fabric area (m^2) is:

$$TOF = \sum_{x=1}^{NP} ((N - ONP(x))\ area(x)) \qquad (A.211)(1208)$$

The weight of the fabric, the weights of the other components in the canopy and the total canopy weight are calculated as in section A.1.1 for the flat circular parachute.

A.17.2 Inputs Required for the "paradesign" Computer Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.
(ii) NP, the number of panels per gore.
(iii) μ, the canopy constructed cone angle.
(iv) ONP(1) to ONP(NP), the number of each panel missing.

A.18 Type 22 - Le Moigne Parachute

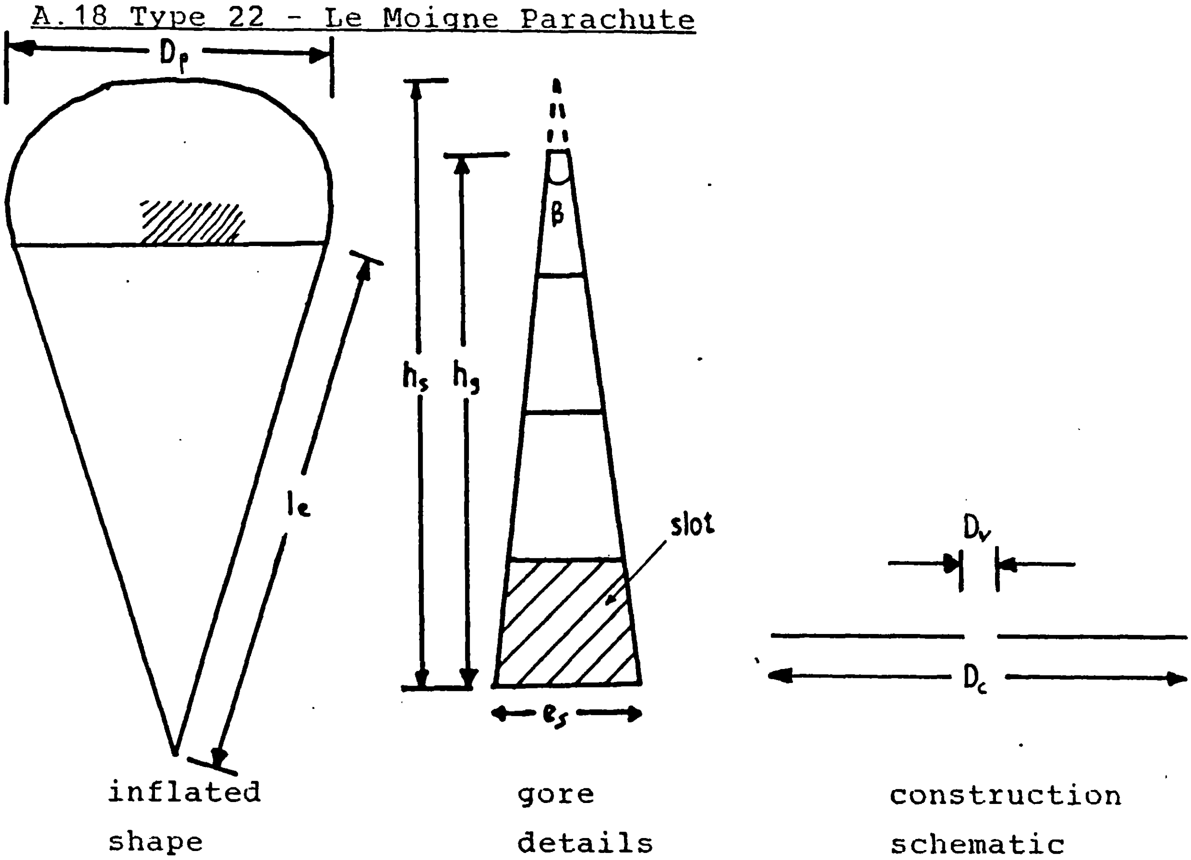


Figure A.18 Le Moigne Parachute Configuration

gore angle β = 360/N degrees.

A.18.1 Canopy Weight Calculation

The Le Moigne canopy weight is calculated using the methods employed for the aeroconical parachute, section A.17.1.

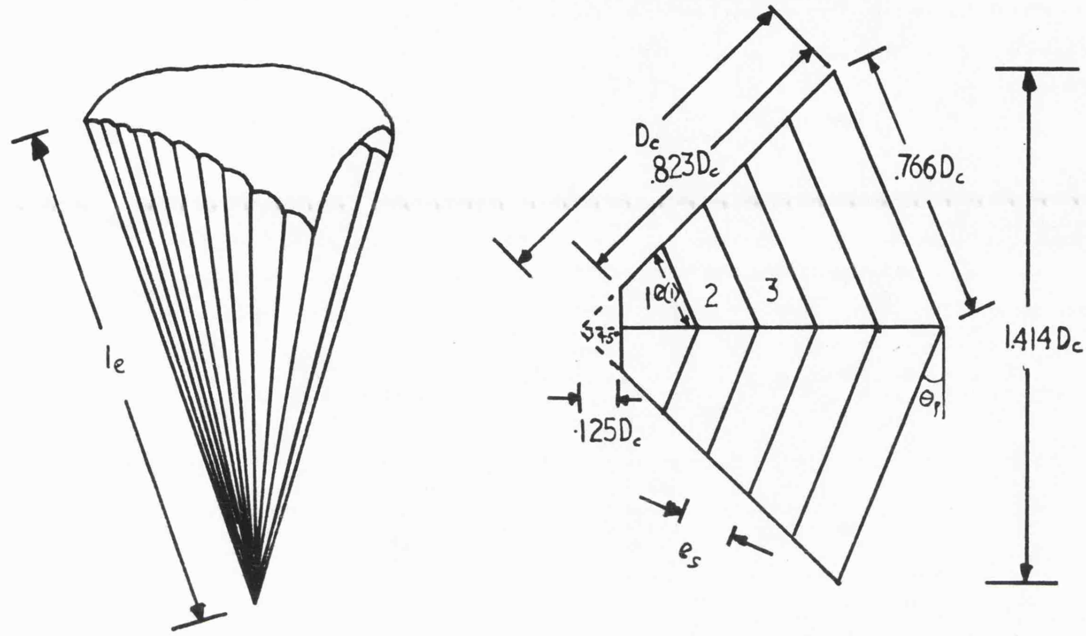
A.18.2 Inputs Required for the Parachute Design Program

(i) v_x, the vent area as a percentage of the total canopy area, S_0.

(ii) NP, the number of fabric panels per gore.

(iii) ONP(1) to ONP(NP), the number of each panel missing.

A.19 Type 23 - Parawing (Single Keel) Parachute



inflated shape

construction schematic

Figure A.19 Parawing Parachute Configuration

$$\theta_p = \cos^{-1} 0.923 \qquad (1829)$$

A.19.1 Canopy Weight Calculation

Maximum gore width $e_S = 0.823\ D_c/N$ (A.212)(1039)

Fabric areas and gore widths:

$$\text{area}(1) = (0.125\ D_c + 0.025)\ ((e_S/1.414 + 0.025)\ (e_S \tan\theta_p/1.414 + 0.025)) + 0.5\ ((e_S/1.414 + 0.025)\ ((e_S/1.414 + 0.025)\ (e_S \tan\theta_p/1.414 + 0.025)) + (0.125\ D_c \tan\theta_p + 0.025)\ (0.025 + e_S \tan\theta_p/1.414)) \qquad (A.213)(1830)$$

$$e(1) = 0.125\ D_c/\tan\theta_p + e_S/(1.414 \cos\theta_p) \qquad (A.214)(1831)$$

$$area(2) = (e(1) + e_S \tan\theta_p/2 + 0.075/2)\ (e_S + 0.05) \qquad (A.215)(1832)$$

$$e(2) = e_S \tan\theta_p + e(1) \qquad (A.216)(1833)$$

Equations A.215 and A.216 are repeated to area(N) and e(N). WTF is the fabric weight (gm/m^2).The total fabric area (m^2) and weight (kg):

$$TOF = 2 \sum_{x=1}^{N} area(x) \qquad (A.217)(1835)$$

$$WF = TOF.WTF/1000 \qquad (A.218)(1836)$$

The line length and weight are calculated as for the flat circular parachute, equations A.11 and A.12. WTT is the tape weight (gm/m). The total length (m) and weight (kg) of the tapes:

$$TOT = 2\ (0.766\ D_c + 0.2) + 0.875\ D_c + 0.2 + 2 \sum_{x=1}^{N} (e(x) + 0.2) \qquad (A.219)(1839)$$

$$WT = TOT.WTT/1000 \qquad (A.220)(1840)$$

WTSB is the skirt band weight (gm/m). The total skirt band length (m) and weight (kg):

$$TSB = 2\ (N\ e_S + 0.2) + 0.25\ D_c + 0.2 \qquad (A.221)(1841)$$

$$WSB = TSB.WTSB/1000 \qquad (A.222)(1842)$$

The total weight of the canopy:

$$WTOT = WSB + WT + WL + WF \qquad (A.223)(1843)$$

A.19.2 Inputs Required for the Computer Program

No extra input data is required for the parawing (single keel) parachute, the inputs listed in section 4.3.1 are sufficient.

Appendix B

Brief Description of the Warnier-Orr Structured Program Design Method

The Warnier-Orr Notation is based on the following principles:

(i) Abstraction - unimportant detail is omitted.

(ii) Decomposition - a large problem is broken down into a number of smaller ones.

(iii) Separation of concerns - one thing is considered at a time.

The following programming techniques are represented by the Warnier-Orr notation:

(i) Sequence.

$$A \left\{ \begin{array}{l} B \\ C \\ . \\ . \\ . \\ D \end{array} \right.$$

Figure B.1 Sequence in Warnier-Orr

Command A can be regarded as command B followed by C and finally D. The bracket can be read as "consists of". In Pascal:

```
(* A - does this *)
B;
C;
D.
```

(ii)Repetition.

$$A \begin{cases} B \\ (n) \end{cases} \qquad \text{or} \qquad A \begin{cases} B \\ (n,*)?test1 \end{cases}$$

Figure B.2 Repetition in Warnier-Orr

In the first diagram of figure B.2 command A consists of n instances of B, where n is known at run time, this is a do loop in Pascal. The second diagram in figure B.2 represents the general case, A consists of n or more instance of B depending on test1. If n is 0 then the diagram represents a while loop, while test1 is true B is repeated. If n is 1 the diagram represents a repeat loop, B is repeated until test1 is true.

(iii)Selection.

$$A \begin{cases} B \\ ?test1 \\ \oplus \\ C \end{cases}$$

$\oplus$ represents an exclusive or

Figure B.3 Selection in Warnier-Orr Notation

Command A consists of either B or C depending on test1. If test1 is true B is performed, if it is false C is performed. B and C are mutually exclusive operations. In Pascal:

```
(* A - does this *)
if test1 then B
else C.
```

Appendix C

Structure Diagram

In this appendix the Warnier-Orr structure diagram for the parachute design program is listed.

Key:
Underlined words represent separate procedures in the parachute design program (listed in appendix E).
skip means "do nothing".
The numbers in brackets refer to program equations and tables of data, see the key in figure 3.1.
⊕ represents exclusive or.
? represents if.

parachute design {
- select type
- area calculation
- cluster
- line length
- no of lines
- staging
- opening loads
- reefing
- structural strength
- select material
- landing control
- weight and volume
- cost
- stability
- reliability
- print file (optional results print out)

<u>select type</u>

- print title
- print menu1 (of types)
- input type_no
- check
  - input type_no2
    - print menu2
    - input type_no2
  - ?type_no=16
  - $\oplus$
  - skip

menu1 is table 3.1, menu2:
A - flat ringslot
B - conical ringslot

variables
type_no   :integer;
type_no2 :ch (single character input)

<u>area calculation</u>

- <u>input data</u> (as section 4.3.1)
- <u>print out input</u>
- <u>area2</u>
  - <u>a non glide</u>
  - ?type_no in [1..19]
  - $\oplus$
  - <u>a glide</u>
  - ?type_no in [21..23]
- <u>area print</u>

<u>a non glide</u>

- obtain C_D
  - input C_D
  - $\oplus$
  - calculate C_D (101 - 103,10101)
- calculate area and diameter (201-203 and 206a)

<u>a glide</u>

- obtain C_{FR}
  - input C_{FR}
  - $\oplus$
  - calculate C_{FR} (10102)
- calculate area and diameter (204-209)

variables

C_D :real; (drag coefficient)

C_{FR} :real (force coefficient, gliding parachutes)

<u>cluster</u>
- cluster 1
  - cluster 2
  - ?type_no in [1,17]
  - ⊕
  - exit(parachute_ design) (area too large)
- ?equation (301) true
- ⊕
- xcluster=false

cluster 2
- (302-315, 10301-10304)
- xcluster=true
- print cluster data

variables

xcluster :boolean

<u>line length</u>
- skip
- ?xcluster=true
- ⊕
- line length1
- ?type_no in [1-7,10-19,21,22]
- ⊕
- line length2
- ?type_no in [8,9,20,23]

line length1
- obtain l_e/D_0
  - input l_e/D_0
  - ⊕
  - l_e/D_0 from (10401)
- calculate l_e (401)
- print l_e

line length2

- obtain l_e/D_c
  - input l_e/D_c
  - ⊕
  - l_e/D_c from (10401)
- calculate l_e (402-409)
- print l_e

variables

l_e :real (line length)

l_e/D_0 :real (ratio of line length to nominal diameter)

l_e/D_c :real (ratio of line length to constructed diameter)

<u>no of lines</u>

- skip
- ?type_no in [11,23]
- ⊕
- no of lines 2
  - input Z
  - ⊕
  - calculate Z (501-506)

variables

Z :real (number of lines)

<u>staging</u>

- staging 2
  - (601-610)
  - xstaging=true
  - print staging
- ?equation (601) true
- ⊕
- xstaging=false

variables

xstaging : boolean

<u>opening loads</u>

- <u>loads cluster</u>
- ?xcluster=true
- ⊕
- loads
  - select method_no
  - calculate loads
  - print out

```
                    menu9                     print menu9
                    ?type_no in [1,11,        input method_no
                            12,15,16]
                    ⊕
                    menu10                    print menu10
                    ?type_no in [13,          input method_no
select method_no                14,21]
                    ⊕
                    menu11                    print menu11
                    ?type_no in [5,6,         input method_no
                                 9,17]
                    ⊕
                    method_no=3

                    lingard
                    ?method_no=1
                    ⊕
calculate loads     pflanz (10701,
                          701-705)
                    ?method_no=2
                    ⊕
                    mass ratio (706-
                                708)
                    ?method_no=3
```

menu9 (three methods available):
1 - Lingard
2 - Pflanz
3 - Mass ratio

menu10 (two methods available):
1 - Lingard
3 - Mass ratio

menu11 (two methods available):
2 - Pflanz
3 - Mass ratio

variables:
method_no :integer

<u>loads cluster</u>
- calculate R_M and C_x (709-710)
- select method_no
  - print menu12
  - input method_no
- calculate loads
  - (711)
  - ?method_no=1
  - ⊕
  - (711-715)
  - ?method_no=2
- print out

menu12:
1 - synchronous opening of cluster
2 - non-synchronous opening of cluster

variables
R_M : real; (mass ratio)
C_x : real (opening load factor)

<u>reefing</u>
- (801-802)
- reefing1
  - reefing2
  - ?(804) true
  - ⊕
  - xreefing=false

reefing2
- (805-824)
- check
- 825
  - skip
  - ?(824) false
  - ⊕
  - exit(parachute design) (load too high)
- xreefing=true
- print reefing

variables
xreefing :boolean

- structural strength
  - safety factor
    - input safety factor
    - ⊕
    - calculate safety factor
  - design load (901)
  - structural strength2
  - ssreef (reefing)
  - ssprint (print out)

- structural strength2
  - ss1 (solid round parachutes)
  - ?type_no in [1-7,18,19,21,22]
  - ⊕
  - ss2 (cruciform parachutes)
  - ?type_no=11
  - ⊕
  - ss3 (slotted parachutes)
  - ?type_no in [12..17]
  - ⊕
  - ss4 (parawing parachute)
  - ?type_no=23

- ss1
  - (10901, 902-905)
  - vent
    - input v_x
    - vent2
  - skirt
    - p skirt band (908)
    - ?skirt_band=true
    - ⊕
    - skip
    - ?skirt_band=false
  - reinforcing
    - (909-910)
    - ?s_band=true
    - ⊕
    - skip
    - ?s_band=false
  - (911-912)
  - fabric=true
  - tape=true
  - horiz_ribbon=false
  - vert_tape=false

```
                                  ┌ vent_band=true
                                  │ vent_line=true
                 ┌ vent3          ┤ p_vent_band (906)
                 │ ?v_x <>0       └ (907)
vent2           ─┤ (+)
                 │ no_vent        ┌ vent_band=false
                 └ ?v_x =0        └ vent_line=false
```

```
variables
horiz_ribbon    : boolean; (horizontal ribbon)
skirt_band      : boolean; (skirt band)
vent_band       : boolean; (vent band)
fabric          : boolean; (fabric)
vent_line       : boolean; (vent line)
vert_tape       : boolean; (vertical tape)
s_band          : boolean; (reinforcing band)
tape            : boolean; (radial tape)
v_x             : real;    (vent area as a percentage of
                            total canopy area, S_0)
```

```
                 ┌ (913-916)       ┌ input z
                 │ no of lines c  ─┤ (+)
                 │ (917-918)       └ calculate Z (916a)
                 │ tape=true
                 │ fabric=true
ss2             ─┤ s_band=false
                 │ skirt_band=false
                 │ vent_band=false
                 │ vent_line=false
                 │ horiz_ribbon=false
                 └ vert_tape=false
```

ss3
- lines (913-916)
- vent3
  - vent4
    - vent_band=true
    - vent_line=true
    - p vent band (906)
    - (907)
  - ⊕
  - no vent
    - vent_band=false
    - vent_line=false
- skirt
  - p skirt band (908)
  - ?skirt_band=true
  - ⊕
  - skip
  - ?skirt_band=false
- s_band=false
- horiz ribbon
- vertical tape
  - vertical tape strength (928)
  - ?type_no in [12..16]
  - ⊕
  - vertical tape strength (929)
  - ?type_no=17
- (924)
- tape=true
- vert_tape=true

horiz ribbon
- horiz ribbon2 (922-923)
- ribbon or fabric
  - fabric2
    - fabric=true
    - horiz_ribbon= false
  - ?type_no in [16,17]
  - ⊕
  - ribbon2
    - fabric=false
    - horiz_ribbon= true
  - ?type_no in [12..15]

```
                  (930-935) including Z calculation
                  fabric=true
                  tape=true
                  skirt_band=true
ss4               horiz_ribbon=false
                  vert_tape=false
                  vent_band=false
                  vent_line=false
                  s_band=false

                                                 reefing line
                                 reefing line    load 3 (936-951)
             reefing line        load 2          reef_line=true
             load 1              ?type_no in
ssreef       ?xreefing=true        [1,5,12..17]
              ⊕                   ⊕
             reef_line=          reef_line=
                 false               false

variables
reef_line   : boolean (reefing line)

                                          select fabric
                                          ?fabric=true
                     fabric1               ⊕
                                          skip
                     tapes and webs       ?fabric=false
select material      select cord
                     reserve factors
                            (1056-1065)
                     matprint
```

- select fabric
  - calculate panel width
  - calculate strength (ringslot and ringsail)
  - select fabric 2
    - select fabric from t11001
    - +
    - input fabric specification

- tapes and webs
  - vent band tape
    - material(vent_band)
    - ?vent_band=true
    - +
    - skip
  - skirt band tape
    - material(skirt_band)
    - ?skirt_band=true
    - +
    - skip
  - reinforcing tape
    - material(reinforcing)
    - ?s_band=true
    - +
    - skip
  - horizontal ribbon tape
    - material(horiz_ribbon)
    - ?horiz_ribbon=true
    - +
    - skip
  - vertical tape
    - material(vert_tape)
    - ?vert_tape=true
    - +
    - skip
  - radial tape
    - material(tape)
    - ?tape=true
    - +
    - skip

material(component)
- select tape from t11002
- $\oplus$
- input tape specification

select cord
- material cord(lines)
- vent lines cord
  - material cord(vent_line)
  - ?vent_line=true
  - $\oplus$
  - skip
- reefing line cord
  - material cord(reef_line)
  - ?reef_line=true
  - $\oplus$
  - skip

material cord(component)
- select cord from t11003
- $\oplus$
- input cord specification

landing control
- skip (not required at present)

<u>weight and volume</u>

- w+v1
- reefed w+v
  - reefed w+v2 (1845-1847)
  - ?reef_line=true
  - ⊕
  - skip
- cluster w+v
  - cluster w+v2 (1848-1849)
  - ?cluster=true
  - ⊕
  - skip
- print out w+v
- weight check
  - weight too large ?(1850) true
    - <u>select material</u>
    - ⊕
    - exit(parachute_design)
    - ⊕
    - continue
  - ⊕
  - skip
- volume check
  - volume too large ?(1851) true
    - <u>select material</u>
    - ⊕
    - exit(parachute_design)
    - ⊕
    - continue
  - ⊕
  - skip

w+v1

- w+v2 (solid cloth circular) (1201-1300, 1601-1639) ?type_no in [1..7, 21,22]
- $\oplus$
- w+v3 (cruciform) (1642-1648) ?type_no=11
- $\oplus$
- w+v4 ?type_no in [12,13,15]
  - w+v45 (ribbon) ?type_no in [12,13]
    - (1650-1674,1725)
    - porosity (1852-1862)
  - $\oplus$
  - w+v46 (hemisflo) ?type_no=15
    - (1675-1725)
    - porosity (1852-1862)
- $\oplus$
- w+v5 (ringslot) (1727-1749) ?type_no=16
- $\oplus$
- w+v6 (ringsail) (1751-1768) ?type_no=17
- $\oplus$
- w+v7 (disk gap band) (1770-1794) ?type_no=18
- $\oplus$
- w+v8 (rotafoil) (1796-1827) ?type_no=19
- $\oplus$
- w+v9 (parawing) (1829-1843) ?type_no=23

```
                    ┌calculate cost
                    │(1301-1312)                            ┌select materials
cost               ─┤print cost      ┌cost too high        ─┤ ⊕
                    │                │?(1312) true          │exit(parachute_
                    └check cost     ─┤ ⊕                    │        design
                                     └skip                  │ ⊕
                                                            └continue

                                                   ┌stability2
                         ┌stability1              ─┤?type_no in [1..7,
                         │?type_no in [1..7,       │16..19,21,22]
                         │12,13,15..19,21,22]      │ ⊕
stability               ─┤and por5<>99 and         └stability3
                         │por10<>99
                         │ ⊕
                         └skip (no data)

                                                   ┌(1401-1404)
                         ┌calculate               ─┤?(por5+por10<>0)
                         │velocity U               │ ⊕
                         │                         └u=0
stability2              ─┤(1405-1412)
(solid cloth)            │                         ┌not stable
                         └check stability         ─┤?(1413) false
                                                   │ ⊕
                                                   └stable

                         ┌(1414-1416)
stability3              ─┤                         ┌not stable
(ribbon)                 └check stability2        ─┤?(1417) false
                                                   │ ⊕
                                                   └stable

                         ┌exit(parachute_design)
not stable              ─┤ ⊕
                         └continue
```

variables

por5 : real; (porosity at 1/2 inch H_2O)

por10 : real; (porosity at 10 inches H_2O)

U : real (fluid velocity through the canopy fabric)

<u>reliability</u>
- check reliability
  - exit(reliability)
  - (no data)
  - ?reliability=false
  - ⊕
  - skip
- reliability2
  - confidence (1502-1504)
  - material (1505-1515, 11501-11503)
  - packing (1516-1518)
  - system reliability
- print R_{sys}

system reliability
- R_{sys} (1519)
- ?xcluster=false
- ⊕
- r_cluster
  - R_{sys} (1520)
  - ?NF=0
  - ⊕
  - R_{sys} (1512)

variables

R_{sys} : real; (system reliability)

NF : integer (number of parachutes in a cluster allowed to fail)

Appendix D

Data Tables Used in the "paradesign" Program

Figure D.1 shows the data table numbers and which calculation they are used in.

Table 10001, parachute types, is table 3.1 in this thesis. Tables 10101 and 10102, drag coefficient data, are tables 3.2 and 3.3. Tables 10301 to 10304, cluster data, are tables 3.4 to 3.7. Table 10401, line length to nominal diameter ratio data, is table 3.8. Table 10701, opening load data, is table 3.9. Tables 10901 and 10902, ratio of projected to nominal diameter data, are tables 3.11 and 3.12. Material data, tables 11001 to 11003, and reliability data, tables 11501 to 11503, are given below (tables D.1 to D.6).

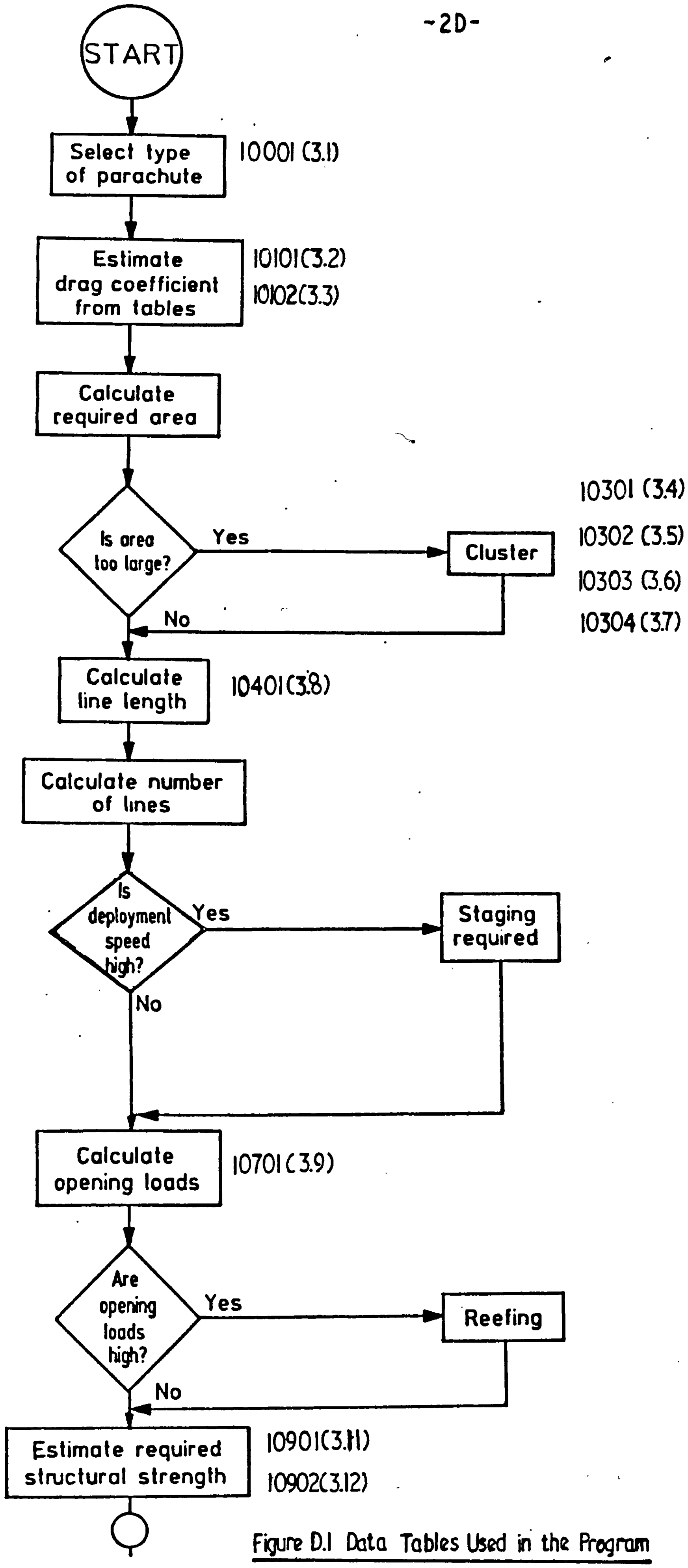


Figure D.1 Data Tables Used in the Program

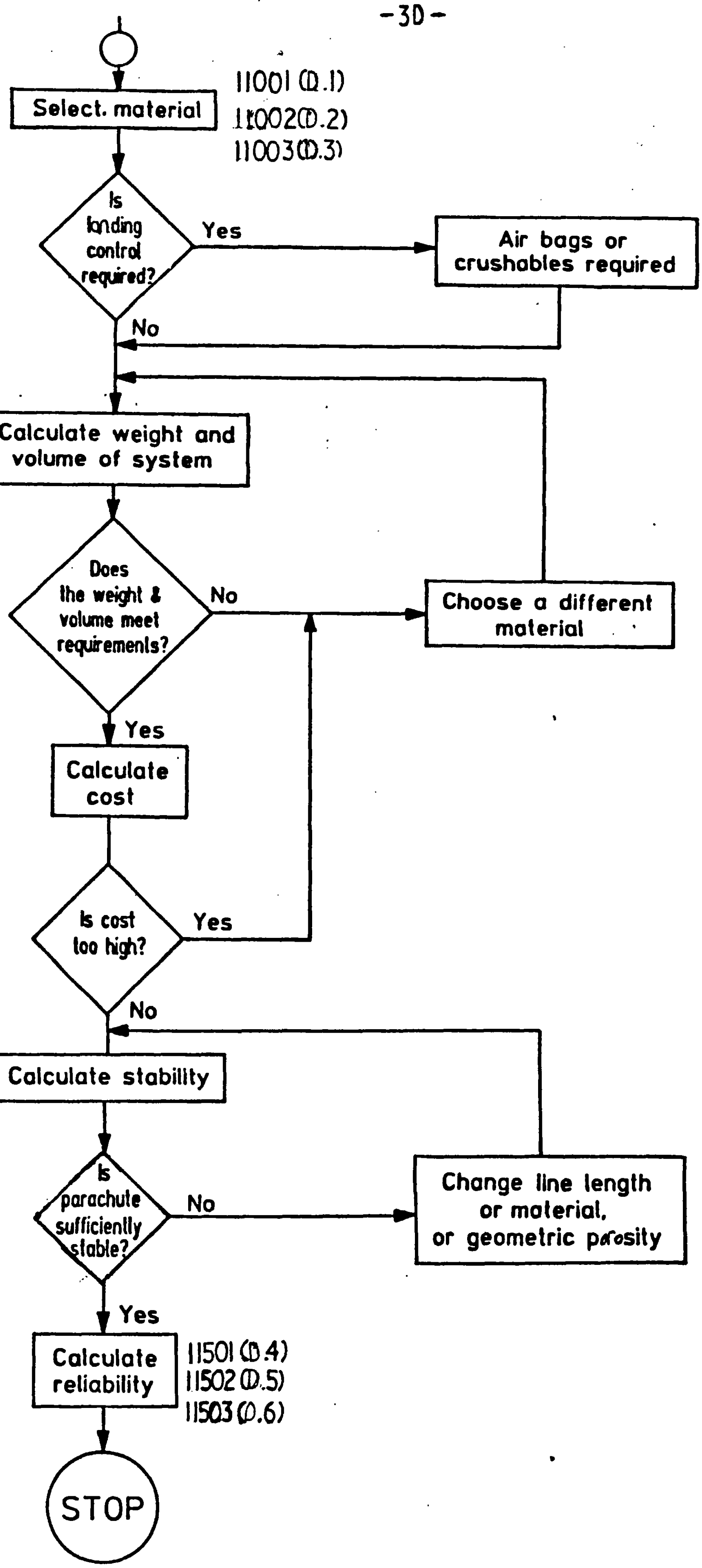


Figure D.1 (continued)

Part No.	Specification	Width (m)	Strength (N)	Weight (gm/m^2)	Porosity at 10 in H_2O (ft^3/ft^2 sec)	Porosity at 1/2 in H_2O (ft^3/ft^2 sec)
P00115 155 3	BSF 118/ 854	0.920	475	50	20	-
P00115 155 5	BSF 118/ 556A	0.920	510	50	10	-
P00115 117 1	BSF 118/ 793/4B	0.920	510	50	10	-
P00115 303 4	MIL-C-70 20 Type I	0.950	370	37	11	1.33
P00115 312 3	GQ-MS- 294	1.200	400	44	0	0
P00115 325 5	GQ-MS- 330	1.220	950	85	3	0.23
-	GQ-MS- 502 (B1)	1.170	480	54	0	0
-	N8726	1.420	400	40	-	-
-	GQ-MS- 309	1.220	400	39	0	0

Table D.1 Fabric Material Data Used in the Parachute Design Program

Part No.	Specification	Strength (N)	Weight (gm/m)	Width (mm)
B01832 009 0	IAC/S 18/350	350	1.5	16.0
Pb00167 750 9	GQ-MS-132	535	3.0	15.0
P00168 055 2	GQ-MS-115	890	3.8	18.0
P00167 050 7	MIL-T_5038 Type III	890	3.7	9.5
P00167 591 3	MIL-T-5038 Type III	1112	4.7	13.0
S99167 598 6	MIL-T-6134 Type II	1334	4.5	25.4
P02106 011 2	GQ-MS-289	1500	38.0	23.5
P00167 550 7	GQ-MS-124	1780	5.2	12.0
P00167 555 7	GQ-MS-318	1780	5.5	12.0
P00168 076 4	MIL-T-5038 Type III	1780	6.2	19.0
P00157 232 7	MIL-W-4088 Type I	2224	8.68	14.3
S99168 578 0	MIL-T-5038 Type V	2224	6.2	14.3
S99168 604 3	MIL-T-5038 Type III	2335	9.3	25.4

Table D.2 Tape and Web Material Properties

Part No.	Specification	Strength (N)	Weight (gm/m)	Width (mm)
P00157 233 5	MIL-T-6134 Type I	2335	12.4	27.0
S99167 599 4	MIL-T-5038 Type IV	2446	10.85	12.7
S99168 582 6	MIL-W-4088 Type II	2670	13.02	25.4
P00167 620 2	GQ-MS-193	2670	8.70	8.0
P00167 580 0	GQ-MS-296	3110	10.5	12.25
P00167 575 1	GQ-MS-158	3115	10.3	12.5
P00169 591 3	MIL-T-5038 Type II	4003	12.4	25.0
P00168 850 0	MIL-T-5038 TypeIV	4448	15.5	25.4
S99169 011 4	MIL-W-5625	4448	15.5	12.7
P00168 890 8	GQ-MS-131	5800	17.0	25.0
P00168 959 8	GQ-MS-317	5800	17.0	25.0
S99169 021 1	MIL-W-5625	6672	18.6	14.3
P00167 530 3	GQ-MS-252	6675	22.4	12.0
S99169 015 6	MIL-W-5625	10008	23.25	15.9

Table D.2 (continued)

Part No.	Specification	Strength (N)	Weight (gm/m)	Width (mm)
S99169 020 3	MIL-W-5625	10230	32.55	19.0
S99169 005 9	MIL-W-4088 Type XVII	11120	35.65	25.4
P00168 920 5	MIL-W-5625	17792	52.7	25.4
B02685 009 1	IAC/S 1116	17800	39.5	25.0
P00168 855 0	MIL-W-4088 Type XX	40032	100.8	25.4
B00754 009 3	GQ-MS-267	28900	90.0	42.5
P00172 382 1	GQ-MS-198	24200	89.0	50.0
P00172 425 9	GQ-MS-256	17800	72.5	48.0
B01103 009 2	BSF 124/224	1461	79.0	44.45
P00169 583 2	MIL-W-4088 Type VIII	17792	49.6	43.66
P00169 590 5	MIL-T-5038 Type II	5782	18.6	38.0
P00169 592 1	MIL-T-5038 Type IV	6672	23.3	38.0
P00171 682 4	MIL-W-4088 Type XII	5338	26.35	44.0

Table D.2 (continued)

Part No.	Specification	Strength (N)	Weight (gm/m)	Width (mm)
P00173 820 0	MIL-W-4088 Type IV	8006	37.2	76.0
S99172 900 2	MIL-T-5608	6672	67.64	50.8
-	IAC/S 15	670	2.6	15.0

Table D.2 (continued)

Part No.	Specification	Strength (N)	Weight (m/kg)
P00167 257 1	GQ-MS-307	200	180
P00107 210 7	DTD-5620-SA501	220	1666.7
P00107 180 0	DTD-5620-CA102	445	556
P00107 213 1	DTD-5620-SB603	670	588.2
P00167 185 0	DTD-5620-CA103	1350	270
P00167 261 0	MIL-C-5040 Type II	1779	320
P00107 217 3	DTD-5620-CB203	1800	181.8
P00107 247 4	DTD-5620-CC311	2000	222.2
P00107 240 8	DTD-5620-CC302	2000	217.4
P00107 191 5	DTD-5620 CA105	2450	140.9
P00167 263 6	MIL-C-5040 Type III	2450	69
P00107 205 0	DTD-5620-CB204	3100	90.9
P00107 227 0	DTD-5620-SC701	3350	111.1
P00167 226 2	DTD-5620-SC711	3350	111.1
P00107 221 2	DTD-5620-CB205	5350	58.8
P00104 385 7	DTD-5620-SB617	6650	42.0

Table D.3 Cord Material Data

Part No.	Specification	Strength (N)	Weight (m/kg)
P00107 232 7	DTD-5620-SC713	10700	35.7
-	DTD-5620-CA106	3100	101

Table D.3 (continued)

Material Specification	Number of Tests	Mean Strength (N)	Standard Deviation of Strength (N)
DTD-5620-CC302	12	2459.6	195.2
DTD-5620-SC711	25	3920.8	178.9
GQ-MS-304	12	2687.0	254.1
DTD-5620-CA105	18	2533.6	166.5
DTD-5620-CB203	25	2196.5	463.4
GQ-MS-307	50	1962.4	176.1
DTD-5620-CA103	9	1421.1	113.3
MIL-C-5040B Type I	50	538.2	17.7
MIL-C-5040B Type IA	50	622.7	33.5
MIL-C-5040B Type BII	50	1957.1	87.2
MIL-C-5040B Type III	50	2682.1	49.8
MIL-C-5040B Type IV	50	3429.4	73.4

Table D.4 Material Reliability Data

Degrees of Freedom	Line Gradient at Confidence Coefficient of:		
	90%	95%	99%
6	0.5891	0.5000	0.3786
7	0.6177	0.5300	0.3714
8	0.6364	0.5525	0.4429
9	0.6597	0.5825	0.4762
10	0.6831	0.6075	0.5000
11	0.6900	0.6200	0.5167
12	0.7116	0.6350	0.5405
13	0.7320	0.6550	0.5548
14	0.7386	0.6625	0.5667
15	0.7529	0.6760	0.5810
16	0.7642	0.6832	0.5857
17	0.7660	0.7022	0.6000
18	0.7773	0.6981	0.6071
19	0.7788	0.7082	0.6143
20	0.7887	0.7159	0.6167
21	0.7905	0.7176	0.6357
22	0.7951	0.7259	0.6452
23	0.8000	0.7234	0.6571
24	0.8172	0.7367	0.6667
29	0.8195	0.7625	0.6850
34	0.8340	0.7719	0.7018
39	0.8611	0.7949	0.7227
44	0.8653	0.7895	0.7419
49	0.8755	0.7954	0.7455

Table D.5 Gradients of the Non-Central t-Distribution Line

Degrees of Freedom	Line Constant at Confidence Coefficient of:		
	90%	95%	99%
6	-0.2775	-0.2000	-0.3171
7	-0.2548	-0.1380	0.0671
8	-0.2364	-0.0990	-0.2257
9	-0.2135	-0.1270	-0.2390
10	-0.2216	-0.1278	-0.2200
11	-0.1060	-0.1020	-0.1867
12	-0.1096	-0.0960	-0.2076
13	-0.1390	-0.1080	-0.1862
14	-0.1114	-0.0850	-0.1567
15	-0.1499	-0.0837	-0.1652
16	-0.1582	-0.0697	-0.1314
17	-0.1243	-0.0979	-0.1200
18	-0.1300	-0.0330	-0.0943
19	-0.1167	-0.0496	-0.0886
20	-0.1504	-0.0474	-0.0667
21	-0.1381	-0.0235	-0.1014
22	-0.1285	-0.0233	-0.1038
23	-0.1200	0.0157	-0.1143
24	-0.1486	-0.0020	-0.1167
29	-0.0563	-0.0252	-0.0690
34	-0.0455	0.0211	-0.0463
39	-0.0978	-0.0315	-0.0371
44	-0.0620	0.0579	-0.0526
49	-0.0866	0.0767	0.0055

Table D.6 Constants of the Non-Central t-Distribution Line

Appendix E

"paradesign" Program Listing

In this appendix the parachute design program code is listed as well as the data files used in the program. The text files are listed in the following order:

pdesign1	pages 2E-8E
pdesign2	pages 9E-11E
pdesign3	pages 12E-17E
linunit	pages 18E-25E
pjbpasinf1	pages 26E-28E
wavunit1	pages 29E-32E
wavunit2	pages 33E-38E
paradesign	pages 39E-45E
pdesignvr1	pages 46E-47E
pdesignvr2	page 48E
t10101	page 49E
t10102	page 49E
t10301	page 50E
t10302	page 50E
t10303	page 51E
t10304	page 51E
t10701	page 52E
t10901	page 52E
t10902	page 53E
t11001	page 53E
t11002	pages 54E-55E
t11003	page 56E
t11501	page 57E
t11502	page 57E
t11503	page 58E
pjblib	page 58E

```
(*     file pdesign1. this file of code contains the following design        *)
(*                    calculations:                                          *)
(*                    select type of parachute,                              *)
(*                    parachute area,                                        *)
(*                    cluster,                                               *)
(*                    line length,                                           *)
(*                    no. of lines,                                          *)
(*                    staging,                                               *)
(*                    opening loads and                                      *)
(*                    reefing.                                               *)

segment procedure continue;

begin
writeln;
writeln('     press return to continue');
readln
end;



segment function open_force(rm2:real):real;

begin
if rm2<=0.5 then
open_force:=0.7056-0.8278*rm2+0.4444*rm2*rm2                              (*  707 *)
else if (rm2>0.5) and (rm2<=3) then
open_force:=0.6179-0.5387*rm2+0.1803*rm2*rm2-2.057e-2*rm2*rm2*rm2         (*  707a*)
else if (rm2>3) and (rm2<4) then
open_force:=0.1038-1.5e-2*rm2                                             (*  707b*)
else open_force:=0.03                                                     (*  707c*)
end;


segment procedure select_type;

begin
write(chr(12));
writeln('     Parachute Design');
writeln('     ****************');
writeln;
writeln('     select type');
writeln;
writeln('       (1)  flat circular');
writeln('       (2)  conical');
writeln('       (3)  bi-conical');
writeln('       (4)  tri-conical');
writeln('       (5)  extended skirt 10% flat');
writeln('       (6)  extended skirt 14.3% full');
writeln('       (7)  hemispherical');
writeln('      (11)  cruciform');
writeln('      (12)  flat ribbon');
writeln('      (13)  conical ribbon');
writeln('      (15)  hemisflo');
writeln('      (16)  ringslot');
writeln('      (17)  ringsail');
writeln('      (18)  disc gap band');
writeln('      (19)  rotafoil');
writeln('      (21)  aeroconical');
writeln('      (22)  le moigne');
writeln('      (23)  parawing (single keel)');
cti(type_no);

if type_no=16 then
begin
write(chr(12));
writeln('     select type');
writeln;
writeln('       (a)  flat ringslot');
writeln('       (b)  conical ringslot');
readln(type_no2)
end;
end;


segment procedure input_data;

begin
write(chr(12));
writeln('     Input Data');
writeln('     **********');
writeln;
writeln('     input rate of descent (m/s)');
ctr(v);
writeln('     input store frontal area (m*m)');
ctr(ss);
writeln('     is store cd known? (y/n)');
checkyn;
if (reply1='y') or (reply1='Y') then
begin
  writeln('     input store cd');
  ctr(cds)
end
else cds:=0.7;
writeln('     input mass of store (including parachute system mass) (kg)');
ctr(ws);
ws:=ws*gravity;
writeln('     input deployment velocity (m/s)');
ctr(vd);
writeln('     input deployment altitude (m)');
ctr(a);
writeln('     deployment attitude known? (y/n)');
checkyn;
if (reply1='y') or (reply1='Y') then
begin
  writeln('     input deployment attitude (degrees) (-90=vertical) ');
  ctr(theta);
  attitude:=true
end
else attitude:=false;
writeln('     input allowable mass of parachute (kg)');
ctr(aw);
writeln('     input allowable volume of parachute (m*m*m)');
ctr(av);
writeln('     input allowable loading');
writeln('     select:');
```

```
writeln('     (1)  allowable load (n)');
writeln('     (2)  allowable loading (g) ');
cti(replylf);
while not (replylf in [1,2]) do
begin
  writeln('     error in value please re-enter');
  cti(replylf)
end;
if (replylf=1) then
begin
  writeln('     input maximum permissable shock load (n)');
  ctr(fa)
end
else if (replylf=2) then
begin
  writeln('     input maximum number of g permissible ');
  ctr(ga)
end;
if (type_no in [21,22,23]) then
begin
  writeln('     input horizontal velocity (m/s)');
  ctr(vh)
end
end;


segment procedure print_out_input;

begin
write(chr(12));
writeln('     Input Data');
writeln('     **********');
writeln;
write('     parachute - ');
case type_no of
1:writeln('flat circular');
2:writeln('conical');
3:writeln('bi-conical');
4:writeln('tri-conical');
5:writeln('extended skirt 10% flat');
6:writeln('extended skirt 14.3% full');
7:writeln('hemispherical');
11:writeln('cruciform');
12:writeln('flat ribbon');
13:writeln('conical ribbon');
15:writeln('hemisflo');
16:writeln('ringslot');
17:writeln('ringsail');
18:writeln('disc-gap-band');
19:writeln('rotafoil');
21:writeln('aeroconical');
22:writeln('le moigne');
23:writeln('parawing (single keel)')
end;
writeln('          rate of descent = ',v:4:2,' m/s');
writeln('               store size = ',ss:4:2,' m*m');
writeln('             store weight = ',ws:6:1,' n');
writeln('                 store cd = ',cds:4:2);
writeln('     deployment velocity = ',vd:4:2,' m/s');
writeln('     deployment altitude = ',a:5:1,' m');
if not attitude then theta:=-90;
writeln('     deployment attitude = ',theta:4:2,' degrees');
theta:=theta*pi/180;
writeln('          allowable mass = ',aw:5:3,' kg');
writeln('        allowable volume = ',av:5:4,' m*m*m');
if replylf=1 then writeln('       maximum shock load = ',fa:6:1,' n')
else if replylf=2 then writeln('       allowable loading = ',ga:4:2,' g');
if type_no in [21,22,23] then writeln('     horizontal velocity = ',vh:4:2,
  ' m/s');
writeln;
writeln('     input data ok? (y/n)');
checkyn;
if (reply1='n') or (reply1='N') then
begin
  input_data;
  print_out_input
end
end;


segment procedure a_non_glide;

var
x          :integer;

begin
write(chr(12));
writeln('     area non-gliding parachute');
writeln;
writeln('     select:');
reset(dfile,'b:t10101.text');
readln(dfile,value);
if value<>m10101 then exit(parachute_design);
readln(dfile,type2);
readln(dfile,cdh,cdl);
x:=2;
while type2<>type_no do
begin
  if x=(m10101+1) then exit(parachute_design);
  x:=x+1;
  readln(dfile,type2);
  readln(dfile,cdh,cdl)
end;
close(dfile);
if v<5.18 then cd:=cdh                                   (*  102 *)
else if v>9.14 then cd:=cdl                              (*  103 *)
else cd:=cdh-((cdh-cdl)*(v-5.18)/3.96);                  (*  101 *)
writeln('     (1) use cd of ',cd:4:2);
writeln('     (2) input alternative cd');
checkch;
if choice1=2 then
begin
  writeln('     input cd');
  ctr(cd)
end;
rho:=1.225*exp(4.256*ln(1-2.2605e-5*a));                 (*  201 *)
```

```
writeln;
if a<>0 then
begin
  writeln('      use sea level air density ?  (y/n)');
  checkyn;
  if reply1 in ['y','Y'] then rho:=1.225
end;
so:=(2*ws)/(rho*v*v*cd);                                    (*  202 *)
writeln;
writeln('      is the store an rpv ? (y/n)');
checkyn;
if reply1 in ['y','Y'] then so:=so-cds*ss/cd;               (*  206a*)
d:=sqrt(4*so/pi)                                            (*  203 *)
end;


segment procedure a_glide;

var
x          :integer;


begin
write(chr(12));
writeln('      area calculation gliding parachute');
writeln;
writeln('      select:');
reset(dfile,'b:t10102.text');
readln(dfile,value);
if value<>m10102 then exit(a_glide);
readln(dfile,type2);
readln(dfile,cfr);
x:=2;
while type2<>type_no do
begin
  if x=(m10102+1) then exit(parachute_design);
  x:=x+1;
  readln(dfile,type2);
  readln(dfile,cfr)
end;
close(dfile);
writeln('      (1) use cr of ',cfr:4:2);
writeln('      (2) input alternative cr');
checkch;
if choice1=2 then
begin
  writeln('      input cr');
  ctr(cfr)
end;
rho:=1.225*exp(4.256*ln(1-2.2605e-5*a));                    (*  201 *)
if a<>0 then
begin
  writeln;
  writeln('      take air density at sea level ?  (y/n)');
  checkyn;
  if reply1 in ['y','Y'] then rho:=1.225
end;
if vh=0 then theta1:=pi/2
else theta1:=arctan(v/vh);                                  (*  204 *)
vt:=(v/sin(theta1));                                        (*  205 *)
so:=(2*ws)/(rho*vt*vt*cfr);                                 (*  206 *)
writeln;
writeln('      is the store an rpv ?  (y/n)');
checkyn;
if reply1 in ['y','Y'] then so:=so-cds*ss/cd;               (*  206a*)
d:=sqrt(4*so/pi);                                           (*  207 *)
cd:=cfr;                                                    (*  208 *)
v:=vt                                                       (*  209 *)
end;


segment procedure area2;

begin
if type_no in [1..19] then a_non_glide
else if type_no in [21..23] then a_glide
else exit(parachute_design)
end;


segment procedure area_print;

begin
write(chr(12));
if type_no in [1..19] then writeln('      cd = ',cd:4:2)
else if type_no in [21..23] then writeln('      cr = ',cfr:4:2);
writeln;
writeln('         nominal area = ',so:4:2,' m*m');
writeln('      nominal diameter = ',d:4:2,' m');
continue
end;


segment procedure area_calculation;

begin
input_data;
print_out_input;
area2;
area_print
end;


segment procedure cfile(mfile:integer;
                fname:string;
          var xcdc_cd:real);

var
etacx    :integer;
x        :integer;

begin
reset(dfile,fname);
readln(dfile,value);
if value<>mfile then exit(parachute_design);
```

```
readln(dfile,etacx,xcdc_cd);
x:=2;
while etacx<>etac do
begin
  if x=(mfile+1) then exit(parachute_design);
  x:=x+1;
  readln(dfile,etacx,xcdc_cd)
end;
close(dfile)
end;


segment procedure cluster;

begin
if so>2787 then                                                     (*  301 *)
begin
  if not ((type_no=1) or (type_no=17)) then exit(parachute_design);
   (* area too large *)
  etac:=trunc((so/2787))+1;                                         (*  302 *)
  cdseach:=cd*so/etac;                                              (*  303 *)
  if type_no=17 then
  begin
    see:=so/etac;                                                   (*  304 *)
    d:=sqrt(4*see/pi)                                               (*  305 *)
  end
  if type_no=1 then
  begin
    if v<7.62 then cfile(m10301,'b:t10301.text',cdc_cd)
    else cfile(m10302,'b:t10302.text',cdc_cd)
  end
  else if type_no=17 then
  begin
    if d<9.3 then cfile(m10303,'b:t10303.text',cdc_cd)
    else cfile(m10304,'b:t10304.text',cdc_cd)
  end;
  lc_do:=sqrt(etac);                                                 (* 306 *)
  le_do:=(98/100)*lc_do;                                             (* 307 *)
  if type_no=1 then
  cdo_cprimedo:=0.8874-0.3316*le_do+0.7155*le_do*le_do-0.27*le_do*le_do*le_do;
                                                                     (* 308 *)
  if type_no=17 then
  cdo_cprimedo:=0.2519+0.9830*le_do-0.1732*le_do*le_do-0.0655*le_do*le_do*
                 le_do;
                                                                     (* 309 *)
  cdfinal:=cd*cdc_cd*cdo_cprimedo;                                   (* 310 *)
  se:=cdseach/cdfinal;                                               (* 311 *)
  d:=sqrt(4*se/pi);                                                  (* 312 *)
  le:=le_do*d;                                                       (* 313 *)
  lc:=lc_do*d;                                                       (* 314 *)
  ac:=lc-le;                                                         (* 315 *)
  xcluster:=true
  write(chr(12));
  writeln('     cluster');
  writeln('     *******');
  writeln;
  writeln('        number of parachutes = ',etac);
  writeln('                 overall cd = ',cdfinal:4:2);
  writeln('           area of each chute = ',se:4:2,' m*m');
  writeln('       diameter of each chute = ',d:4:2,' m');
  writeln('         length of lines (le) = ',le:4:2,' m');
  writeln('         length of lines (lc) = ',lc:4:2,' m');
  continue
end
else xcluster:=false
end;


segment procedure llfile;

var
x     :integer;


begin
reset(dfile,'b:t10401.text');
readln(dfile,value);
if value<>m10401 then exit(parachute_design);
readln(dfile,type2);
readln(dfile,le_do,le_dc);
x:=2;
while type2<>type_no do
begin
  if x=(m10401+1) then exit(parachute_design);
  x:=x+1;
  readln(dfile,type2);
  readln(dfile,le_do,le_dc)
end;
close(dfile)
end;


segment procedure line_length;

begin
write(chr(12));
writeln('     line length');
writeln('     ***********');
writeln;
if type_no in [1..7,10..19,21,22] then
begin
  llfile;
  writeln('     select:');
  writeln('     (1)  use le/do of ',le_do:4:2);
  writeln('     (2)  input alternative le/do ');
  checkch;
  if choice1=2 then
  begin
    writeln('     input le/do');
    ctr(le_do)
  end;
  le:=le_do*d;                                                        (*  401 *)
  write(chr(12));
  writeln('     le/do = ',le_do:4:2);
  writeln;
  writeln('     line length = ',le:4:2,' m')
```

```
end
else if type_no in [8,9,20,23] then
begin
  llfile;
  writeln('     select:');
  writeln('     (1)  use le/dc of ',le_dc:4:2);
  writeln('     (2)  input alternative le/dc ');
  checkch;
  if choice1=2 then
  begin
    writeln('     input le/dc');
    ctr(le_dc)
  end;
  case type_no of
  8:dc:=0.63*d;                                                   (*  402 *)
  9:dc:=0.66*d;                                                   (*  404 *)
  20:dc:=1.9*d;                                                   (*  406 *)
  23:dc:=sqrt(so/0.692)                                           (*  408 *)
  end;
  le:=le_dc*dc;                                                   (*  405 *)
  write(chr(12));
  writeln('     le/dc = ',le_dc:4:2);
  writeln;
  writeln('     line length = ',le:4:2,' m')
end;
writeln;
writeln('     select:');
writeln('     (1) continue');
writeln('     (2) re-calculate line length');
writeln('     (3) exit');
checkch2;
if choice1=2 then line_length;
if choice1=3 then exit(parachute_design)
end;


segment procedure no_of_lines;

begin
write(chr(12));
writeln('     number of lines calculation');
writeln;
writeln('     select:');
idia:=trunc(d*3.280833);                                          (*  501 *)
if ((idia mod 4)=0) or ((idia mod 6)=0) then z:=idia              (*  502 *)
else if (((idia+1) mod 4)=0) or (((idia+1) mod 6)=0) then z:=idia+1  (*  503 *)
else if (((idia+2) mod 4)=0) or (((idia+2) mod 6)=0) then z:=idia+2  (*  504 *)
else z:=idia-1;                                                   (*  505 *)
if z<12 then z:=12;                                               (*  506 *)
writeln('     (1) use ',z,' lines');
writeln('     (2) input alternative number of lines');
checkch;
if choice1=2 then
begin
  writeln('     input number of lines');
  cti(z)
end
end;
```

```
segment procedure staging;

begin
if vd>100 then                                                    (*  601 *)
begin
  write(chr(12));
  writeln('     velocity is greater than 100 m/s');
  writeln;
  writeln('     staging? (y/n)');
  checkyn;
  if (reply1='n') or (reply1='N') then exit(staging);
  if replylf=2 then fa:=ga*ws;                                    (*  602 *)
  rm:=(exp(1.5*ln(cd*so))*rho*gravity)/ws;                        (*  604 *)
  cx:=open_force(rm);
  qm:=fa/(cd*so*cx);                                              (*  606 *)
  cdsd:=((1.15*ws)/qm)-cds*ss;                                    (*  607 *)
  if cdsd<0 then exit(staging);                                   (*  607a*)
  sod:=cdsd/0.5;                                                  (*  608 *)
  dod:=sqrt(sod*4/pi);                                            (*  609 *)
  it:=7*sqrt(4*ss/pi);                                            (*  610 *)
  xstaging:=true
  write(chr(12));
  writeln('     staging required');
  writeln;
  writeln('          drogue constructed area = ',sod:4:2,' m*m');
  writeln('     drogue constructed diameter = ',dod:4:2,' m');
  writeln('               trailing distance = ',it:4:2,' m');
  continue
end
else xstaging:=false
end;


segment procedure loads_cluster;

begin
rm:=exp(1.5*ln(cd*so))*rho*gravity/ws;                            (*  709 *)
cx:=open_force(rm);
writeln('     select');
writeln('     (1)  synchronous opening of cluster');
writeln('     (2)  non-synchronous opening of cluster');
checkch;
f:=cd*so*0.5*rho*vd*vd*cx/etac;                                   (*  711 *)
if choice1=2 then
begin
  rml:=rm/etac;                                                   (*  712 *)
  cxl:=open_force(rml);
  cy:=cxl/cx;                                                     (*  714 *)
  f:=f*cy                                                         (*  715 *)
end
write(chr(12));
write('     load for ');
if choice1=2 then write('non-');
writeln('synchronous opening of cluster = ',f:5:1,' n');
continue
end;
```

```
segment procedure lingard;

begin
initialise;
run;
f:=max_tension
end;


segment procedure opening_loads;

var
x      :integer;

begin
write(chr(12));
writeln('      Opening Loads Calculation');
writeln('      *************************');
writeln;
writeln('      select');
writeln('      (1) input opening load');
writeln('      (2) load calculated by program');
checkch;
if choice1=1 then
begin
  writeln('      input opening load (n)');
  ctr(f);
  exit(opening_loads)
end;
if xcluster=true then loads_cluster
else
begin
  if type_no in [1,11,12,15,16] then
  begin
    writeln('      select method');
    writeln('      (1)  lingard');
    writeln('      (2)  pflanz');
    writeln('      (3)  mass ratio');
    cti(method_no);
    while not (method_no in [1..3]) do
    begin
      writeln('      error in value, please re-enter');
      cti(method_no)
    end
  end
  else if type_no in [13,14,21] then
  begin
    writeln('      select method');
    writeln('      (1)  lingard');
    writeln('      (3)  mass ratio');
    cti(method_no);
    while not (method_no in [1,3] ) do
    begin
      writeln('      error in value, please re-enter');
      cti(method_no)
    end
  end
  else if type_no in [5,6,9,17] then
begin
  writeln('      select method');
  writeln('      (2)  pflanz');
  writeln('      (3)  mass ratio');
  cti(method_no);
  while not (method_no in [2,3] ) do
  begin
    writeln('      error in value, please re-enter');
    cti(method_no)
  end
end
else method_no:=3;
if method_no=1 then lingard
else if method_no=2 then
begin
  reset(dfile,'b:t10701.text');
  readln(dfile,value);
  if value<>m10701 then exit(opening_loads);
  readln(dfile,type2);
  readln(dfile,eta1,eta2,cx);
  x:=2;
  while type2<>type_no do
  begin
    if x=(m10701+1) then exit(parachute_design);
    x:=x+1;
    readln(dfile,type2);
    readln(dfile,eta1,eta2,cx)
  end;
  close(dfile);
  tfil:=d*eta2/vd;                                                  (*  701 *)
  al:=(2*ws)/(cd*so*rho*gravity*vd*tfil);                           (*  702 *)
  if eta1=1 then
  begin
    if al<=1 then
    x1:=exp(0.5075*ln(al))/2.163                                    (*  703a )
    else if (al>1) and (al<30) then
    x1:=exp((0.3975-0.115*ln(al)/ln(10))*ln(al))/2.1742             (*  703a*)
    else x1:=1                                                      (*  703b*)
  end;
  if eta1=2 then
  begin
    if al<=0.316 then
    x1:=exp(0.733*ln(al))/1.637                                     (*  704 *)
    else if (al>0.316) and (al<=10) then
    x1:=exp((0.5067-0.2667*ln(al)/ln(10))*ln(al))/1.779             (*  704a*)
    else x1:=1                                                      (*  704b*)
  end;
  f:=cd*so*rho*vd*vd*cx*x1*0.5                                      (*  705 *)
end
begin
  rm:=exp(1.5*ln(cd*so))*rho*gravity/ws;                            (*  706 *)
  cx:=open_force(rm);
  f:=cx*0.5*rho*vd*vd*cd*so                                         (*  708 *)
end;
write(chr(12));
write('      opening load calculated using ');
case method_no of
1:write('lingard');
```

```
  2:write('pflanz');
  3:write('mass ratio')
  end;
  writeln(' method = ',f:5:1,' n');
  writeln;
  writeln('     select');
  writeln('     (1) continue');
  writeln('     (2) exit');
  writeln('     (3) re-calculate load');
  checkch2;
  if choice1=2 then exit(parachute_design);
  if choice1=3 then opening_loads
end
end;

segment procedure reefing;

begin
gx:=f/ws;                                                  (*  801 *)
if replylf=1 then ga:=fa/ws;                               (*  802 *)
if gx>ga then                                              (*  804 *)
begin
  if not attitude then theta:=(-1)*pi/2;                   (*  805 *)
  fdr:=ws*(ga-sin(theta));                                 (*  806 *)
  rmdr:=exp(1.5*ln(cd*so))*rho*gravity/ws;                 (*  807 *)
  cxdr:=0.718-0.455*(ln(rmdr)/ln(10))-0.0806*(ln(rmdr)*ln(rmdr)/(ln(10)*
                                               ln(10)));   (*  808 *)
  qdr:=fdr/(cd*so*cxdr);                                   (*  809 *)
  cdsr:=1.1*ws/qdr;                                        (*  810 *)
  zeta:=cdsr/(cd*so);                                      (*  811 *)
  if type_no in [1..4,7,21,22] then
  tau:=0.0438+0.925*zeta-0.5*zeta*zeta                     (*  812 *)
  else if type_no in [5,6] then
  tau:=0.0658+0.9*zeta-0.6667*zeta*zeta                    (*  813 *)
  else if type_no=16 then
  tau:=0.0606+0.7472*zeta-0.3056*zeta*zeta                 (*  814 *)
  else if type_no in [12..15] then
  tau:=0.0133+0.8167*zeta-0.3333*zeta*zeta                 (*  815 *)
  else if type_no in [17,18] then
  tau:=0.015+1.2*zeta-2*zeta*zeta                          (*  816 *)
  else exit(parachute_design);  (* insufficient data *)
  dr:=tau*d;                                               (*  817 *)
  sr:=pi*dr*dr/4;                                          (*  818 *)
  lr:=pi*dr;                                               (*  819 *)
  rmr:=exp(1.5*ln(cdsr))*rho*gravity/ws;                   (*  820 *)
  cxr:=open_force(rmr)
  fr:=cxr*0.5*rho*vd*vd*cdsr;                              (*  822 *)
  gxr:=fr/ws;                                              (*  823 *)
  if gxr>ga then                                           (*  824 *)
  begin
    write(chr(12));
    writeln('     one stage of reefing not sufficient');
    continue;
    exit(parachute_design)  (* load too high *)
  end;
  f:=ga*ws;                                                (*  825 *)
  xreefing:=true;
  write(chr(12));
  writeln('     reefing');
  writeln('     *******');
  writeln;
  writeln('                opening load = ',f:5:1,' n');
  writeln('             reefed diameter = ',dr:4:2,' m');
  writeln('                 reefed area = ',sr:4:2,' m*m');
  writeln('     length of reefing line = ',lr:4:2,' m');
  continue
end
else xreefing:=false
end;
```

```
(*     file pdesign2.                                                *)
(*     this file contains the structural strength calculations       *)


segment procedure p_vent_band;

var
fvb        :real;

begin

write(chr(12));
writeln('      vent band');
writeln('      select:');
writeln('      (1)  use 2.27 factor for vent band strength');
writeln('      (2)  input new factor');
checkch;
if choice1=1 then vbs:=2.27*tl                          (*  906 *)
else
begin
  writeln('      input new factor');
  ctr(fvb);
  vbs:=fvb*tl
end
end;


segment procedure p_skirt_band;

var
fsb        :real;

begin
write(chr(12));
writeln('      skirt band');
writeln('      select:');
writeln('      (1)  use 1.25 factor for skirt band strength');
writeln('      (2)  input new factor');
checkch;
if choice1=1 then sbs:=1.25*tl                          (*  908 *)
else
begin
  writeln('      input new factor');
  ctr(fsb);
  sbs:=fsb*tl
end;
skirt_band:=true
end;



segment procedure ss1;

var
x       :integer;

begin
reset(dfile,'b:t10901.text');
read(dfile,value);
if value<>m10901 then exit(parachute_design);
readln(dfile,type2);
readln(dfile,dp_do);
x:=2;
while type2<>type_no do
begin
  if x (m10901-1) then exit(parachute_design);
  x:=x-1;
  readln(dfile,type2);
  readln(dfile,dp_do)
end;
close(dfile);
dp:=dp_do*d;                                            (*  902 *)
tf:=fd/(dp*pi*1000);                                    (*  903 *)
fabric:=true;
alpha:=arctan(dp/sqrt(4*le*le-dp*dp));                  (*  904 *)
tl:=fd/(z*cos(alpha));                                  (*  905 *)
writeln('      input vent area as a percentage of total canopy area (so)');
ctr(vx);
if vx=0 then
begin
  vent_band:=false;
  vent_line:=false
end
else
begin
  vent_band:=true;
  vent_line:=true;
  p_vent_band;
  vls:=1.0*tl                                           (*  907 *)
end;
writeln('      skirt band? (y/n)');
checkyn;
if (reply1='y') or (reply1='Y') then p_skirt_band
else skirt_band:=false;
writeln('      strengthening band ? (y/n)');
checkyn;
if (reply1='y') or (reply1='Y') then
begin
  writeln('      one use ? (y/n)');
  readln(reply2);
  while not (reply2 in ['y','Y','n','N'] ) do
  begin
    writeln('      error in value, please re-enter');
    readln(reply2)
  end;
  if (reply2='y') or (reply2='Y') then sb:=0.46*tl      (*  910 *)
  else sb:=0.55*tl;                                     (*  909 *)
  s_band:=true
end
else s_band:=false;
if type_no in [18,19] then tt:=tl                       (*  911 *)
else tt:=0.9*tl;                                        (*  912 *)
horiz_ribbon:=false;
vert_tape:=false;
```

```
tape:=true
end;

segment procedure ss2;

begin
write(chr(12));
writeln('     cruciform parachute');
writeln('     input arm ratio');
ctr(ar);
t:=sqrt(so/(2*ar-1));                                       (*  913 *)
dcru:=t*ar;                                                 (*  914 *)
tf:=1.7*fd/(dcru*1000);                                     (*  915 *)
alpha:=arcsin(0.69*d/(2*le));                               (*  916 *)
writeln;
writeln('     arm span = ',dcru:4:2,' m');
writeln('     arm width = ',t:4:2,' m');
writeln;
if t<1.5 then z:=12                                         (*  916a*)
else z:=4*(trunc(1.14*t+1.29)+1);                           (*  916a*)
writeln('     select:');
writeln('     (1) use ',z,' lines');
writeln('     (2) input alternative number of lines');
checkch;
if choice1=2 then
begin
  writeln('     input number of lines');
  cti(z)
end;
tl:=fd/(z*cos(alpha));                                      (*  917 *)
tt:=0.9*tl;                                                 (*  918 *)
tape:=true;
fabric:=true;
s_band:=false;
skirt_band:=false;
vent_band:=false;
vent_line:=false;
horiz_ribbon:=false;
vert_tape:=false
end;


segment procedure ss3;

var
x       :integer;

begin
reset(dfile,'b:t10902.text');
readln(dfile,value);
if value<>m10902 then exit(parachute_design);
readln(dfile,type2);
readln(dfile,dp_do);
x:=2;
while type2<>type_no do
begin
  if x=(m10902-1) then exit(parachute_design);
  x:=x-1;
  readln(dfile,type2);
  readln(dfile,dp_do)
end;
close(dfile);
dp:=dp_do*d;                                                (*  919 *)
alpha:=arctan(dp/sqrt(4*le*le-dp*dp));                      (*  920 *)
tl:=fd/(z*cos(alpha));                                      (*  921 *)
write(chr(12));
writeln('     one use ? (y/n)');
checkyn;
if (reply1='n') or (reply1='N') then hrs:=0.55*tl           (*  922 *)
else hrs:=0.46*tl;                                          (*  923 *)
fabric:=false;
horiz_ribbon:=true;
writeln('     skirt band ? (y/n)');
checkyn;
if (reply1='n') or (reply1='N') then skirt_band:=false
else p_skirt_band;
writeln('     vent ? (y/n)');
checkyn;
if (reply1='n') or (reply1='N') then
begin
  vent_band:=false;
  vent_line:=false
end
else
begin
  p_vent_band;
  vls:=1.0*tl;                                              (*  927 *)
  vent_band:=true;
  vent_line:=true
end;
if type_no in [12..16] then vts:=0.25*tl                    (*  928 *)
else vts:=0.506*tl;                                         (*  929 *)
if type_no=17 then tape:=false
else
begin
  tt:=0.506*tl;                                             (*  924 *)
  tape:=true
end;
s_band:=false;
vert_tape:=true
end;


segment procedure ss4;

begin
dc:=sqrt(so/0.692);                                         (*  930 *)
tf:=fd/(0.66*pi*dc*1000);                                   (*  931 *)
alpha:=arctan(0.66*dc/sqrt(4*le*le-0.66*0.66*dc));          (*  932 *)
n:=trunc(dc*0.823/0.864)-1;                                 (*  932a*)
z:=(2*n+1)*3;                                               (*  932b*)
tl:=fd/(z*cos(alpha));                                      (*  933 *)
tt:=0.9*tl;                                                 (*  934 *)
sbs:=1.25*tl;                                               (*  935 *)
fabric:=true;
tape:=true;
skirt_band:=true;
horiz_ribbon:=false;
```

```
vert_tape:=false;
vent_band:=false;
vent_line:=false;
s_band:=false
end;


segment procedure ssprint;

begin
write(chr(12));
writeln('     Structural strength of each component');
writeln;
writeln('              safety factor used = ',sf:3:1);
writeln;
if fabric then writeln('              strength of fabric = ',tf:4:2,' n/mm');
if vent_band then writeln('           strength of vent band = ',vbs:4:2,' n');
if vent_line then writeln('           strength of vent line = ',vls:4:2,' n');
if reef_line then writeln('        strength of reefing line = ',trl:4:2,' n');
writeln('              strength of lines = ',tl:4:2,' n');
if tape then writeln('              strength of tapes = ',tt:4:2,' n');
if skirt_band then writeln('          strength of skirt band = ',sbs:4:2,' n');
if vert_tape then writeln('      strength of vertical tapes = ',vts:4:2,' n');
if horiz_ribbon then writeln('      strength of horiz ribbons = ',hrs:4:2,' n');
if s_band then writeln('       strength of reinforcing = ',sb:4:2,' n');
if type_no=23 then writeln('                number of lines = ',z:3)
end;


segment procedure ssreef;

begin
if (xreefing=true) and (type_no in [1,5,12..17]) then
begin
  if type_no in [1,5] then
  cdp_cdpo:=0.8638-1.9077*tau+3.3048*tau*tau                  (*  936 *)
  else if type_no in [12..15] then
  cdp_cdpo:=0.328+2.862*tau-2.958*tau*tau                     (*  937 *)
  else if type_no=16 then
  cdp_cdpo:=0.46-2.866*tau-3.844*tau*tau                      (*  938 *)
  else if type_no=17 then
  begin
    if tau<0.08 then cdp_cdpo:=0.35                           (*  939 *)
    else cdp_cdpo:=2.5*tau-0.15                               (*  940 *)
  end
  else exit(ssreef);
  cdpo:=cd/(dp*dp/(d*d));                                     (*  941 *)
  cdpr:=cdp_cdpo*cdpo;                                        (*  942 *)
  spr:=cdsr/cdpr;                                             (*  943 *)
  dpr:=sqrt(4*spr/pi);                                        (*  944 *)
  hc:=pi*dpr/4;                                               (*  945 *)
  hx:=d/2-hc;                                                 (*  946 *)
  phi:=arctan(dr/sqrt(4*le*le-dr*dr));                        (*  947 *)
  psi:=arctan((dpr-dr)/sqrt(4*hx*hx-(dpr-dr)*(dpr-dr)));      (*  948 *)
  fprime_f:=((sin(psi)/cos(psi))-(sin(phi)/cos(phi)))/(2*pi); (*  949 *)
  fprime:=fdr*fprime_f;                                       (*  950 *)
  trl:=fprime*sf;                                             (*  951 *)
  reef_line:=true
end
else reef_line:=false
end;


segment procedure structural_strength;

begin
write(chr(12));
writeln('     Structural Strength Calculation');
writeln('     ********************************');
writeln;
writeln('     safety factor');
writeln('     select:');
writeln('     (1)  input safety factor');
writeln('     (2)  safety factor calculated by program');
checkch;
if choice1=1 then
begin
  writeln('     input safety factor');
  ctr(sf)
end
else if choice1=2 then
begin
  writeln('     is parachute man carrying? (y/n)');
  checkyn;
  if (reply1='y') or (reply1='Y') then sf:=3.1
  else if (reply1='n') or (reply1='N') then sf:=2.3
end;
fd:=f*sf;                                                     (* 901 *)
if type_no in [1..7,18,19,21,22] then ss1
else if type_no=11 then ss2
else if type_no in [12..17] then ss3
else if type_no=23 then ss4
else exit(parachute_design);
ssreef;
ssprint;
writeln;
writeln('     select:');
writeln('     (1) continue');
writeln('     (2) exit');
writeln('     (3) re-calculate structural strength');
checkch2;
if choice1=2 then exit(parachute_design);
if choice1=3 then structural_strength
end;
```

```
(*     file pdesign3.                                   *)
(*     this file contains the following calculations:   *)
(*     select material.                                 *)
(*     landing control.                                 *)


procedure colours;

begin
write(chr(12));
writeln('     key to colours');
writeln;
writeln('     a - royal blue');
writeln('     b - black');
writeln('     d - olive dral');
writeln('     g - grey');
writeln('     k - khaki');
writeln('     l - light blue');
writeln('     n - navy blue');
writeln('     o - orange');
writeln('     q - blue');
writeln('     r - red');
writeln('     s - sand');
writeln('     x - green');
writeln('     y - yellow');
writeln('     z - natural');
continue
end;


procedure table_fabric;


var
tab11001     :array[1..m11001] of
               record
               t1part_no    :string;
               t1spec       :string;
               t1width      :real;
               t1strength   :real;
               t1weight     :real;
               t1por5       :real;
               t1por10      :real;
               t1colours    :string
               end;

flag_fab     :array[1..m11001] of boolean;
counter      :integer;
i            :integer;
j            :integer;
kk           :integer;
l            :integer;
no_fabric    :integer;

begin
reset(dfile,'b:t11001.text');
readln(dfile,value);
if value<>m11001 then exit(parachute_design);
for i:=1 to m11001 do
begin
  with tab11001[i] do
  begin
  readln(dfile,t1part_no);
  readln(dfile,t1spec);
  readln(dfile,t1width,t1strength,t1weight,t1por10,t1por5,t1colours)
  end
end;
close(dfile);
for i:=1 to m11001 do
begin
  if (esd>tab11001[i].t1width) or (0.9*tf>tab11001[i].t1strength)
  then flag_fab[i]:=false
  else flag_fab[i]:=true
end;
write(chr(12));
write('        part no.    specification   width strength weight por10');
writeln(' por1/2 ');
write('                                              m    n/50mm  gm/m*m ft/sec');
writeln(' ft/sec');
counter:=1;
for j:=1 to m11001 do
begin
  if counter<>20 then
  begin
    if flag_fab[j] then
    begin
      with tab11001[j] do
      begin
        write('  ',counter:2,'- ',t1part_no:12,t1spec:17,t1width:6:1,
              t1strength:9:1,t1weight:7:1);
        if t1por10>98.0 then write('   -  ')
        else write(t1por10:6:2);
        if t1por5>98.0 then writeln('   -   ')
        else writeln(t1por5:7:2)
      end;
    counter:=counter+1
    end
  end
  else  writeln('     greater than 20')  (* complete when m11001>23 *)
end;
if counter=1 then exit(parachute_design);
writeln('     input fabric number (calculated strength = ',tf:4:2,' n/50mm)');
cti(no_fabric);
kk:=0;
l:=0;
while kk<>no_fabric do
begin
  l:=l+1;
  if flag_fab[l] then kk:=kk+1
```

```
end;
part_no:=tab11001[l].t1part_no;
spec:=tab11001[l].t1spec;
mtf:=tab11001[l].t1strength;
wtf:=tab11001[l].t1weight;
widef:=tab11001[l].t1width;
por10:=tab11001[l].t1por10;
por5:=tab11001[l].t1por5
end;


procedure select_fabric;


begin
  write(chr(12));
  if (type_no in [1,19,22]) or ((type_no=16) and ((type_no2='a') or
          (type_no2='A'))) then
  begin
    n:=z;                                                       (* 1001 *)
    hs:=sqrt(so/(n*sin(pi/n)/cos(pi/n)));                       (* 1002 *)
    es:=2*hs*sin(pi/n)/cos(pi/n)                                (* 1003 *)
  end;
  if (type_no in [2,21]) or ((type_no=16) and ((type_no2='b') or
          (type_no2='B'))) then
  begin
    writeln('     input cone angle mu (degrees)');
    ctr(mu);
    n:=z;                                                       (* 1004 *)
    beta:=2*arcsin(sin(pi/n)*cos(mu*pi/180));                   (* 1005 *)
    hs:=sqrt(so*cos(beta/2)/(n*sin(beta/2)));                   (* 1006 *)
    es:=2*hs*sin(beta/2)/cos(beta/2);                           (* 1007 *)
  end;
  if type_no=3 then
  begin
    n:=z;                                                       (* 1008 *)
    writeln('     input upper cone angle mu1 (degrees)');
    ctr(mu1);
    writeln('     input lower cone angle mu2 (degrees)');
    ctr(mu2);
    writeln('     input ratio h1/h2');
    ctr(k);
    beta1:=2*arcsin(sin(pi/n)*cos(mu1*pi/180));                 (* 1009 *)
    beta2:=2*arcsin(sin(pi/n)*cos(mu2*pi/180));                 (* 1010 *)
    h2:=sqrt(so/(n*(k*k*sin(beta1/2)/cos(beta1/2)+sin(beta2/2)/cos(beta2/2)+
          2*k*sin(beta1/2)/cos(beta1/2))));                     (* 1011 *)
    h1:=k*h2;                                                   (* 1012 *)
    es:=2*h1*sin(beta1/2)/cos(beta1/2)+2*h2*sin(beta2/2)/cos(beta2/2)
                                                                (* 1013 *)
  end;
  if type_no=4 then
  begin
    writeln('     input upper cone angle mu1 (degrees)');
    ctr(mu1);
    writeln('     input middle cone angle mu2 (degrees)');
    ctr(mu2);
    writeln('     input lower cone angle mu3 (degrees)');
    ctr(mu3);
    writeln('     input ratio h1/h2');
    ctr(k1);
    writeln('     input ratio h3/h2');
    ctr(k2);
    n:=z;                                                       (* 1014 *)
    beta1:=2*arcsin(sin(pi/n)*cos(mu1*pi/180));                 (* 1015 *)
    beta2:=2*arcsin(sin(pi/n)*cos(mu2*pi/180));                 (* 1016 *)
    beta3:=2*arcsin(sin(pi/n)*cos(mu3*pi/180));                 (* 1017 *)
    h2:=sqrt(so/(n*(sin(beta1/2)*(k1*k1-2*k1-2*k1*k2)/cos(beta1/2)+
          sin(beta2/2)*(1-2*k2)/cos(beta2/2)-
          sin(beta3/2)*k2*k2/cos(beta3/2))));                   (* 1018 *)
    h1:=k1*h2;                                                  (* 1019 *)
    h3:=k2*h2;                                                  (* 1020 *)
    es:=2*h1*sin(beta1/2)/cos(beta1/2)-2*h2*sin(beta2/2)/cos(beta2/2)-
       2*h3*sin(beta3/2)/cos(beta3/2)                           (* 1021 *)
  end;
  if type_no=5 then
  begin
    writeln('    es calc');
    n:=z;                                                       (* 1022 *)
    h1:=0.858*sqrt(so*cos(pi/n)/(n*sin(pi/n)));                 (* 1023 *)
    es:=2*h1*sin(pi/n)/cos(pi/n)                                (* 1024 *)
  end;
  if type_no=6 then
  begin
    n:=z;                                                       (* 1025 *)
    es:=0.81*d*sin(pi/n)                                        (* 1026 *)
  end;
  if type_no=7 then
  begin
    n:=z;                                                       (* 1027 *)
    es:=pi*0.7*d/n                                              (* 1028 *)
  end;
  if type_no=11 then
  begin
    n:=(z div 4)-1;                                             (* 1030 *)
    es:=t/n                                                     (* 1031 *)
  end;
  if type_no=17 then
  begin
    n:=z;                                                       (* 1032 *)
    hs:=0.519*d;                                                (* 1033 *)
    es:=3.21*hs/n                                               (* 1034 *)
  end;
  if type_no=18 then
  begin
    n:=z;                                                       (* 1035 *)
    h1:=sqrt(so*cos(pi/n)/(1.887*n*sin(pi/n)));                 (* 1036 *)
    es:=2*h1*sin(pi/n)/cos(pi/n)                                (* 1037 *)
  end;
  if type_no=23 then es:=dc*0.823/n;                            (* 1039 *)
  if (type_no=16) then
  begin
```

```
  if vent_band then
  begin
    writeln('     input vent area as a percentage of total canopy area (so)');
    ctr(vx)
  end
  else vx:=0;
  writeln('     input number of rings (maximum 20)');
  cti(nr);
  writeln('     input slot width (m)');
  ctr(sw);
  n:=z;
  sv:=vx*so/100;                                                  (* 1041 *)
  if (type_no2='a') or (type_no2='A') then
  hg:=hs-sqrt(sv*cos(pi/n)/(n*sin(pi/n)))                         (* 1042 *)
  else hg:=hs-sqrt(sv*cos(beta/2)/(n*sin(beta/2)));                (* 1046 *)
  rw:=(hg-(nr-1)*sw)/nr;                                          (* 1043 *)
  while rw>1.15 do
  begin
    writeln('     not enough rings, input a higher number');
    cti(nr);
    rw:=(hg-(nr-1)*sw)/nr                                         (* 1043 *)
  end;
  tf:=hrs/(rw*1000)                                               (* 1044 *)
end;
if type_no=17 then
begin
  if vent_band then
  begin
    writeln('     input vent area as a percentage of total canopy area (so)');
    ctr(vx)
  end
  else vx:=0;
  writeln('     input number of rings (maximum 20)');
  cti(nr);
  sv:=vx*so/100;                                                  (* 1049 *)
  dv:=sqrt(4*sv/pi);                                              (* 1050 *)
  hg:=hs-0.519*dv;                                                (* 1051 *)
  rw:=hg/nr;                                                      (* 1052 *)
  while rw>1.15 do
  begin
    writeln('     not enough rings, input a higher number');
    cti(nr);
    rw:=hg/nr                                                     (* 1052 *)
  end;
  tf:=hrs/(rw*1000)                                               (* 1053 *)
end;
if type_no in [16,17] then esd:=rw+0.05                           (* 1054a*)
else
esd:=es+0.05;                                                     (* 1054 *)
tf:=tf*50;                                                        (* 1055 *)
write(chr(12));
writeln('     fabric selection');
writeln;
writeln('     maximum gore width (+ 50mm sewing allowance) = ',
        esd:4:2,' m');
writeln('                          required material strength = ',tf:4:2,
         ' n/50mm');
```

```
  writeln;
  writeln('     select');
  writeln('     (1)  material chosen from table of embedded data');
  writeln('     (2)  material properties input');
  checkch;
  if choice1=2 then
  begin
    writeln('     input part number');
    readln(part_no);
    writeln('     input specification number');
    readln(spec);
    writeln('     input strength (n/50mm)');
    ctr(mtf);
    writeln('     input width (m)');
    ctr(widef);
    writeln('     input weight (gm/m*m)');
    ctr(wtf);
    writeln('     input porosity at 10in h2o ');
    ctr(por10);
    writeln('     input porosity at 1/2in h2o');
    ctr(por5)
  end
  else table_fabric
end;


procedure mattw(strength  :real;
                istring   :string;
           var material   :twmaterial);


var
flag_tape    :array[1..m11002] of boolean;
tab11002     :array[1..m11002] of twmaterial;
counter      :integer;
ii           :integer;
jj           :integer;
ll           :integer;
no_tape      :integer;
strong       :real;

begin
write(chr(12));
writeln('     ',istring);
writeln;
writeln('     required material strength = ',strength:5:1,' n');
writeln;
writeln('     select');
writeln('     (1)  material chosen from table of embedded data');
writeln('     (2)  material properties input');
checkch;
if choice1=2 then
begin
  with material do
  begin
```

```
      writeln('     input part no.');
      readln(twpart_no);
      writeln('     input specification number');
      readln(twspec);
      writeln('     input strength (n)');
      ctr(twstrength);
      writeln('     input weight (gm/m)');
      ctr(twweight);
      writeln('     input width (mm)');
      ctr(twwidth)
    end
  end
  else
  begin
    reset(dfile,'b:t11002.text');
    readln(dfile,value);
    if value<>m11002 then exit(parachute_design);
    for ii:=1 to m11002 do
    begin
      with tab11002[ii] do
      begin
        readln(dfile,twpart_no);
        readln(dfile,twspec);
        readln(dfile,twstrength,twweight,twwidth,twcolours)
      end
    end;
    close(dfile);
    for ii:=1 to m11002 do
    begin
      if strength<350 then strong:=strength+500
      else if strength>20000 then strong:=strength+15000
      else strong:=strength;
      if (strong*0.9>tab11002[ii].twstrength) or (strong*1.5<tab11002[ii].
                                                        twstrength)
      then flag_tape[ii]:=false
      else flag_tape[ii]:=true
    end;
    write(chr(12));
    writeln('          part no.     specification   width strength weight');
    writeln('                                        mm      n      gm/m');
    counter:=1;
    for jj:=1 to m11002 do
    begin
      if flag_tape[jj] then
      begin
        with tab11002[jj] do
        begin
          writeln('     ',counter:2,'- ',twpart_no:12,twspec:18,twwidth:6:1,
                twstrength:9:1,twweight:7:1);
          counter:=counter+1;
          if counter=20 then
          begin
            writeln('     press return for more materials');
            readln
          end
        end
      end
    end;
      if counter=1 then exit(parachute_design);
      writeln('     input tape number (calculated strength = ',strength:5:1,' n)');
      cti(no_tape);
      jj:=0;
      ll:=0;
      while jj<>no_tape do
      begin
        ll:=ll+1;
        if flag_tape[ll] then jj:=jj+1
      end;
      with tab11002[ll] do
      begin
        material.twpart_no:=twpart_no;
        material.twspec:=twspec;
        material.twstrength:=twstrength;
        material.twweight:=twweight;
        material.twwidth:=twwidth;
        material.twcolours:=twcolours
      end
    end
    end;



procedure tapes_and_webs;

begin
if vent_band then mattw(vbs,'vent band',vbmat);
if skirt_band then mattw(sbs,'skirt band',sbmat);
if s_band then mattw(sb,'strengthening band',rmat);
if horiz_ribbon then mattw(hrs,'horizontal ribbon',hrmat);
if vert_tape then mattw(vts,'vertical tape',vtmat);
if tape then mattw(tt,'tape',tmat)
end;


procedure matcord(strength2     :real;
              var  material2    :cmaterial;
                   cstring      :string);


var
flag_cord       :array[1..m11003] of boolean;
tab11003        :array[1..m11003] of cmaterial;
counter         :integer;
i               :integer;
j               :integer;
l               :integer;
no_cord         :integer;

begin
write(chr(12));
writeln('     ',cstring);
writeln;
writeln('     required strength = ',strength2:4:1,' n');
writeln('     select');
```

```
writeln('     (1)  choose material from table of embedded data');
writeln('     (2)  input material properties');
checkch;
if choice1=2 then
begin
  with material2 do
  begin
    crely:=0;
    writeln('     input part no.');
    readln(cpart_no);
    writeln('     input specification');
    readln(cspec);
    writeln('     input strength (n)');
    ctr(cstrength);
    writeln('     input weight (m/kg)');
    ctr(cweight)
  end
end
else
begin
  reset(dfile,'b:t11003.text');
  readln(dfile,value);
  if value<>m11003 then exit(parachute_design);
  for i:=1 to m11003 do
  begin
    with tab11003[i] do
    begin
      readln(dfile,cpart_no);
      readln(dfile,cspec);
      readln(dfile,cstrength,cweight,crely);
      readln(dfile,ccolours)
    end
  end;
  close(dfile);
  for i:=1 to m11003 do
  begin
    if strength2*0.9>tab11003[i].cstrength then
    flag_cord[i]:=false
    else flag_cord[i]:=true
  end;
  write(chr(12));
  writeln('              part no.       specification strength weight ');
  writeln('                                                 n       m/kg');
  counter:=1;
  for j:=1 to m11003 do
  begin
    if flag_cord[j] then
    begin
      with tab11003[j] do
      writeln('     ',counter:2,'- ',cpart_no:12,cspec:17,cstrength:9:1,
              cweight:7:1);
      counter:=counter+1;
      if counter=20 then writeln('20')
    end
  end;
  if counter=1 then exit(parachute_design);
  writeln('     input cord number (calculated strength = ',strength2:4:1,' n)');
  cti(no_cord);
  j:=0;
  i:=0;
  while j<>no_cord do
  begin
    i:=i+1;
    if flag_cord[i] then j:=j+1
  end;
  with tab11003[i] do
  begin
    material2.cpart_no:=cpart_no;
    material2.cspec:=cspec;
    material2.cstrength:=cstrength;
    material2.cweight:=cweight;
    material2.ccolours:=ccolours;
    material2.crely:=crely
  end
end
end;


procedure select_cord;

begin
if vent_line then matcord(vls,vlmat,'vent line');
if reef_line then matcord(trl,rlmat,'reefing line');
matcord(tl,lmat,'lines')
end;

procedure reserve_factors;


begin
if fabric then rff:=mtf/tf;                                   (* 1056 *)
if vent_band then rfvb:=vbmat.twstrength/vbs;                 (* 1057 *)
if skirt_band then rfsb:=sbmat.twstrength/sbs;                (* 1058 *)
if s_band then rfr:=rmat.twstrength/sb;                       (* 1059 *)
if horiz_ribbon then rfhr:=hrmat.twstrength/hrs;              (* 1060 *)
if vert_tape then rfvt:=vtmat.twstrength/vts;                 (* 1061 *)
if tape then rft:=tmat.twstrength/tt;                         (* 1062 *)
rfl:=lmat.cstrength/tl;                                       (* 1063 *)
if vent_line then rfvl:=vlmat.cstrength/vls;                  (* 1064 *)
if reef_line then rfrl:=rlmat.cstrength/trl                   (* 1065 *)
end;


procedure ptapex(  xstring   :string;
                 pmaterial   :twmaterial;
                  freserve   :real);

begin
write(chr(12));
writeln('     ',xstring);
writeln;
writeln('     material properties');
with pmaterial do
```

```
begin
  writeln('           part no. =  ',twpart_no:12);
  writeln('       specification = ',twspec:17);
  writeln('            strength = ',twstrength:4:1,' n');
  writeln('              weight = ',twweight:4:1,' gm/m');
  writeln('               width = ',twwidth:4:1,' mm');
  writeln('      reserve factor = ',freserve:3:1)
end;
continue
end;


procedure cprint( ystring  :string;
                pmaterial2 :cmaterial;
                freserve2 :real);

begin
write(chr(12));
writeln('    ',ystring);
writeln;
writeln('     material properties');
with pmaterial2 do
begin
  writeln('           part no. =  ',cpart_no:12);
  writeln('       specification = ',cspec:17);
  writeln('            strength = ',cstrength:4:1,' n');
  writeln('              weight = ',cweight:4:1,' m/kg');
  writeln('      reserve factor = ',freserve2:3:1)
end;
continue
end;



procedure matprint;

begin
if fabric then
begin
  write(chr(12));
  writeln('     fabric');
  writeln;
  writeln('     material properties');
  writeln('                        part no. =  ',part_no:12);
  writeln('                   specification = ',spec:17);
  writeln('               material strength = ',mtf:4:1,' n/mm*50');
  writeln('                  material width = ',widef:4:2,' m');
  writeln('                 material weight = ',wtf:4:1,' gm/m*m');
  if por10<98 then
  writeln('       porosity at 10 in h2o = ',por10:3:1,' ft/sec');
  if por5<98 then
  writeln('      porosity at 1/2 in h2o = ',por5:3:1,' ft/sec');
  writeln('                reserve factor = ',rff:3:1);
  continue
end;
```

```
if skirt_band then ptapex('skirt band',sbmat,rfsb);
if s_band then ptapex('strengthening band',rmat,rfr);
if horiz_ribbon then ptapex('horizontal ribbon',hrmat,rfhr);
if vert_tape then ptapex('vertical tape',vtmat,rfvt);
if tape then ptapex('tape',tmat,rft);
if vent_band then ptapex('vent band',vbmat,rfvb);
cprint('lines',lmat,rfl);
if vent_line then cprint('vent line',vlmat,rfvl);
if reef_line then cprint('reefing line',rlmat,rfrl)
end;


procedure select_material;

begin
writeln('    entering select_mat');
if (type_no in [16,17]) then
begin
  fabric:=true;
  horiz_ribbon:=false
end;
if xcluster then so:=se;
if fabric then select_fabric;
tapes_and_webs;
select_cord;
reserve_factors;
matprint
end;


procedure landing_control;

begin                          (* this calculation not used at present *)
end;
```

```
(*     file linunit.                                                              *)
(*     this file contains the following calculations:                             *)
(*     lingard inflation program (including optional staging and an               *)
(*                                   aerodynamic store).                          *)
(*     input checking.                                                            *)

unit linunit;
interface

const

pi          =3.1416;
gravity     =9.81;


type
xarray      =array[1..7] of real;

var

ws              :real;       (* weight of store                        202 *)
choice1         :integer;    (* choice in menu                             *)
reply1          :char;       (* y/n                                        *)
so              :real;       (* nominal area                           202 *)
sod             :real;       (* drogue area                            608 *)
theta           :real;       (* attitude                               805 *)
d               :real;       (* nominal diameter                       205 *)
a               :real;       (* altitude                               201 *)
attitude        :boolean;
vd              :real;       (* deployment speed                       601 *)
time            :real;
time_limit      :real;
time_step       :real;
drogue_time     :real;       (* time at detachment from store              *)
main_time       :real;       (* main parachute deployed                    *)
height          :real;
store_mass      :real;
base_area       :real;
base_dia        :real;
cg_to_tail      :real;
cp_to_tail      :real;
iyy             :real;
drogue_cd       :real;
drogue_area     :real;
drogue_line     :real;
mass_total      :real;       (* estimated mass of main and drogue          *)
constr_dia      :real;
constr_area     :real;
tau0            :real;
init_altitude   :real;
init_attitude   :real;
init_velocity   :real;
deltax_store    :real;       (* cg-cp                                       *)
cmqf            :real;       (* aerodynamic data                            *)
cmaldot         :real;       (* aerodynamic data                            *)
aldot           :real;       (* aerodynamic data                            *)
linalpha        :real;       (* angle of attack                             *)
drogue_d        :real;
u2              :real;       (* velocity                                    *)
w2              :real;       (* velocity                                    *)
gamma           :real;       (* angle of tension                            *)
total_velocity  :real;
x_tail          :real;
z_tail          :real;
x_drogue        :real;
z_drogue        :real;
linrho          :real;       (* air density                                 *)
linx2           :real;       (* displacement                                *)
linz2           :real;       (* displacement                                *)
u1              :real;
w1              :real;
tension         :real;
inflate_time    :real;
time1           :real;
time2           :real;
t1              :real;
t2              :real;
lincf           :real;
max_tension     :real;

ptype           :integer;  (* type no. of parachute                         *)
flag            :integer;


aero_data       :array[1..4,1..7] of real;  (* aerodynamic data             *)
run_values      :array[1..6,1..3] of real;
aero_coeff      :array[1..3] of real;


drogue          :boolean;
aero_store      :boolean;

procedure convert_to_integer(inp      :string;
                         var out      :real;
                         var   b      :boolean);
procedure cti(var int_val             :integer);
procedure real_convertion(inp         :string;
                      var out         :real;
                      var   b         :boolean);
procedure ctr(var real_val            :real);
procedure checkch;
procedure checkch2;
procedure checkyn;
```

```
procedure lin_input_data;
procedure print_input;
procedure init_variables;
procedure print_heading;
procedure run_data;
procedure lin_print_out;
procedure choice;
procedure initialise;
procedure drogue_separation;
procedure drogue_impulse1;
procedure tau_value;
procedure main_chute;
procedure impulses;
procedure coefficient;
procedure para;
procedure aitken(        xc          :real;
                   var   p           :real;
                         x           :xarray;
                         y           :xarray;
                         n           :integer);
procedure interpolate;
procedure derivitives;
procedure runge_kutta(no_of_equations :integer);
procedure solve_equations;
procedure run;
implementation


procedure convert_to_integer;

var
num              :real;
i                :integer;
negative         :boolean;

begin
b:=false;
inp:=concat(inp,' ');
inp[length(inp)]:=chr(0);
i:=1;
while inp[i]=' ' do i:=i-1;
if not ((inp[i] in ['0'..'9']) or (inp[i]='+') or (inp[i]='-')) then
exit(convert_to_integer);
if inp[i]='-' then
begin
  negative:=true;
  i:=i-1
end
else if inp[i]='-' then
begin
  negative:=false;
  i:=i+1
end
else negative:=false;
num:=ord(inp[i])-ord('0');
i:=i-1;
while (inp[i] in ['0'..'9']) do
begin
  num:=num*10-(ord(inp[i])-ord('0'));
  i:=i-1
end;
if (inp[i]=' ') or (inp[i]=chr(0)) then b:=true;
if negative then num:=num*(-1);
out:=num
end;


procedure cti ;

var
s2        :string;
ic        :boolean;
num2      :real;

begin
readln(s2);
convert_to_integer(s2,num2,ic);
while not ic do
begin
  writeln('error in value, please re-enter');
  readln(s2);
  convert_to_integer(s2,num2,ic)
end;
int_val:=trunc(num2)
end;  (* cti *)

procedure real_convertion;

var
i           :integer;
j           :integer;
k           :integer;
d           :integer;
ip          :real;
fp          :real;
dec         :real;
man         :real;
num         :real;
inp2        :string;
negative    :boolean;
minus       :boolean;

begin
b:=false;
inp:=concat(inp,' ');
inp[length(inp)]:=chr(0);
i:=1;
while inp[i]=' ' do i:=i-1;
if not ((inp[i] in ['0'..'9']) or (inp[i]='.') or (inp[i]='-') or
         (inp[i]='+')) then
exit(real_convertion);
if inp[i]='-' then
```

```
begin
  minus:=true;
  i:=i+1
end
else if inp[i]='-' then
begin
  minus:=false;
  i:=i+1
end
else minus:=false;
j:=i;
while inp[i] in ['0'..'9'] do i:=i+1;
if inp[j]<>'.' then
begin
  inp2:=inp;
  delete(inp2,i,(length(inp2)-i));
  delete(inp2,1,(j-1));
  convert_to_integer(inp2,ip,b)
end
else ip:=0;
if inp[i]='.' then
begin
  j:=i;
  i:=i+1;
  while inp[i] in ['0'..'9'] do i:=i+1;
  if inp[j-1] in ['0'..'9'] then
  begin
    inp2:=inp;
    delete(inp2,i,(length(inp2)-i));
    delete(inp2,1,j);
    d:=i-(j-1);
    convert_to_integer(inp2,fp,b)
  end
  else
  begin
    fp:=0;
    d:=0
  end
end
else
begin
  fp:=0;
  d:=0
end;
if (inp[i]='e') or (inp[i]='E') then
begin
  i:=i+1;
  if (inp[i]='-') then
  begin
    negative:=true;
    i:=i+1
  end
  else if (inp[i]='+') then
  begin
    negative:=false;
    i:=i+1
  end
  else if (inp[i] in ['0'..'9']) then
  negative:=false
  else exit(real_convertion);
  j:=i;
  while inp[i] in ['0'..'9'] do i:=i+1;
  if inp[j] in ['0'..'9'] then
  begin
    inp2:=inp;
    delete(inp2,i,(length(inp2)-i));
    delete(inp2,1,(j-1));
    convert_to_integer(inp2,man,b)
  end
  else man:=0
end
else man:=0;
if negative then man:=man*(-1);
dec:=1;
for k:=1 to d do dec:=dec*10;
num:=(ip+fp*(1/dec))*exp(man*ln(10));
if (inp[i]=' ') or (inp[i]=chr(0)) then
b:=true;
if minus then num:=num*(-1);
out:=num
end;



procedure ctr;

var
s        :string;
rc       :boolean;

begin
readln(s);
real_convertion(s,real_val,rc);
while not rc do
begin
  writeln(' error in value, please re-enter');
  readln(s);
  real_convertion(s,real_val,rc)
end
end;   (* ctr *)


procedure checkch;

begin
cti(choice1);
while not (choice1 in [1,2]) do
begin
  writeln('     error in value, please re-enter');
  cti(choice1)
end
end;
```

```
procedure checkch2;

begin
ctl(choice1);
while not (choice1 in [1,2,3]) do
begin
  writeln('     error in value, please re-enter');
  ctl(choice1)
end
end;

procedure checkyn;

begin
readln(reply1);
while not (reply1 in ['y','Y','n','N']) do
begin
  writeln('     error in value, please re-enter');
  readln(reply1)
end
end;

procedure lin_input_data;

var

x            :integer;


begin
write(chr(12));
writeln('     Lingard inflation program');
writeln('     **************************');
writeln;
writeln;
store_mass:=ws/gravity;
if aero_store then
begin
writeln('     input data for store');
writeln('     *********************');
writeln;
writeln('     input base area (m*m)');
ctr(base_area);
writeln('     input base diameter (m)');
ctr(base_dia);
writeln('     input distance of c.g. to tail (m)');
ctr(cg_to_tail);
writeln('     input distance of c.p. to tail (m)');
ctr(cp_to_tail);
writeln('     input 2nd moment of inertia iyy (kg*m*m)');
ctr(iyy);
for x:=1 to 7 do
begin
writeln('     input cl, cd, cm for angle of attack ',(30*x-120),' degrees');
ctr(aero_data[2,x]);
ctr(aero_data[3,x]);
ctr(aero_data[4,x]);
aero_data[1,x]:=(30*x-120)*pi/180
end

end;


if drogue then
begin

write(chr(12));
writeln('     input data for drogue');
writeln('     *********************');
writeln;
writeln('     input cd for drogue');
ctr(drogue_cd);
writeln('     input drogue line length (m)');
ctr(drogue_line);
writeln('     input drogue area (m*m)');
ctr(drogue_area)

end;

write(chr(12));
writeln('     input data for main parachute');
writeln('     ******************************');
writeln;
if drogue and aero_store then
begin
  writeln('     input estimated mass of main and drogue (kg)');
  ctr(mass_total)
end
else mass_total:=0;
constr_dia:=d;
constr_area:=so;
writeln('     select type');
writeln;
writeln('     (1)  aeroconical 6.2');
writeln('     (2)  aeroconical 5.2');
writeln('     (3)  C9');
writeln('     (4)  aeroconical 5.8');
writeln('     (5)  AIM');
writeln('     (6)  ribbon');
writeln('     (7)  cruciform');
writeln('     (8)  aeroconical (naces)');
writeln('     (9)  24m');
ctl(ptype);
```

```
writeln;
writeln;
writeln('     input maximum time (secs)');
ctr(time_limit);
writeln('     input time step (secs)');
ctr(time_step);

if drogue then
begin
writeln('     input drogue time (detachment from store ) (secs)');
ctr(drogue_time);
writeln('     input main time (from drogue deployment) (secs)');
ctr(main_time)
end

else
begin
writeln('     input parachute deployment time (secs)');
ctr(main_time)
end;

init_altitude:=a;
if attitude then init_attitude:=theta
else init_attitude:=-pi/2;
init_velocity:=vd
end;

procedure print_input;

var
x       :integer;

reply  :char;
begin
write(chr(12));
writeln('     INPUT DATA');
writeln('     **********');
writeln('     STORE');

if aero_store then
begin
writeln('     mass= ',store_mass:5:1,' kg  base area= ',base_area:4:2,
        ' m*m  base diameter= ',base_dia:4:2,' m');
writeln('     xbar=  ',cg_to_tail:5:1,' m       deltax= ',(cg_to_tail-
        cp_to_tail):4:2,' m              iyy= ',iyy:5:1,' kg*m*m');
writeln('     angle of attack    cl       cd       cm');
for x:=1 to 7 do writeln(aero_data[1,x]*180/pi:12:1,aero_data[2,x]:16:4
     ,aero_data[3,x]:9:4,aero_data[4,x]:9:4)
end

else writeln('     mass= ',store_mass:5:1,' kg');
```

```
if drogue then
begin
write('     DROGUE');
writeln('     cd= ',drogue_cd:4:2,'  area= ',drogue_area:1:2,
        ' m*m  length of lines= ',drogue_line:4:2,' m')
end;

write('     MAIN PARACHUTE  -');
case ptype of
1:writeln('     aeroconical 6.2');
2:writeln('     aeroconical 5.2');
3:writeln('     C9');
4:writeln('     aeroconical 5.8');
5:writeln('     AIM');
6:writeln('     ribbon');
7:writeln('     cruciform');
8:writeln('     aeroconical (naces)');
9:writeln('     24m')
end;
if drogue and aero_store then
writeln('     mass of parachutes= ',mass_total:4:2,' kg');
writeln('     constructed diameter= ',constr_dia:4:2,' m');
writeln('          constructed area= ',constr_area:4:2,' m*m');
if drogue then
begin
writeln('     parachutes detached from store at ',drogue_time:3:1,
       ' seconds');
writeln('     main deployed ',main_time:3:2,' seconds after drogue deployed')
end

else writeln('     parachute deployed at ',main_time:3:2,' seconds');

writeln('     INITIAL VALUES');
writeln('     altitude= ',init_altitude:5:1,' m  attitude= ',(init_attitude*
        180/pi):4:2,' deg  velocity= ',init_velocity:4:1,' m/s');
writeln;

  write('     is input data ok ? (y/n)');
  readln(reply);
  if reply='n' then
  begin
  lin_input_data;
  print_input
  end

end;

procedure init_variables;

var
x       :integer;
y       :integer;

begin
for x:=1 to 6 do
begin
for y:=1 to 3 do run_values[x,y]:=0.0
end;
```

```
if not aero_store then
begin
base_dia:=0.0;
base_area:=0.0;
cg_to_tail:=0.0;
cp_to_tail:=0.0;
iyy:=0.0
end;

deltax_store:=cg_to_tail-cp_to_tail;
run_values[3,1]:=init_attitude;
run_values[4,2]:=run_values[3,1];
cmqf:=0.0;
cma'dot:=0.0;
aldot:=0.0;
drogue_d:=0.0;
linalpha:=0.0;
time:=0.0;
run_values[1,2]:=init_velocity;
if run_values[1,2]=0.0 then run_values[1,2]:=1.0e-6;
flag:=0;
tension:=0.0;
max_tension:=0.0


end;

procedure print_heading;
begin

write(chr(12));
writeln(' time      x   altitude  velocity  alpha   theta   gamma   tension');
writeln('  sec      m       m         m/s     deg     deg     deg        n')

end;


procedure tau_value;

var
t       :real;

begin

case ptype of
1: begin
   t:=10.0/sqrt(total_velocity);
   tau0:=-total_velocity*t/8.805
   end;
2: begin
   t:=6.65/sqrt(total_velocity);
   tau0:=-total_velocity*t/7.385
   end;
3: tau0:=-10.0;
4: begin
   t:=7.8/sqrt(total_velocity);
   tau0:=-total_velocity*t/8.237
   end;
5: begin
   t:=11.5/sqrt(total_velocity);
   tau0:=-total_velocity*t/8.99
   end;
6: tau0:=-14.0;
7: tau0:=-11.8;
8: begin
   t:=10.0/sqrt(total_velocity);
   tau0:=-total_velocity*t/8.805
   end;
9: tau0:=-14.0

end

end;


procedure main_chute;
begin

tau_value;
writeln('     tau0 chosen for the canopy is ',tau0:4:2);
writeln('     do you wish to use an alternative value ? (y/n)');
checkyn;
if reply1 in ['y','Y'] then
begin
  writeln('     input new tau0');
  ctr(tau0)
end;
inflate_time:=-tau0*constr_dia/total_velocity;
time1:=time;
flag:=3

end;


procedure coefficient;
begin

time2:=time-time1;
t1:=time2/inflate_time;

case ptype of
1: begin
   if t1<=0.5 then lincf:=t1*0.074816/0.5
   else if (t1>0.5) and (t1<=0.887) then lincf:=4.202556e-1+t1*(-2.025129-
               t1*(3.364271+t1*(-1.391541)))
   else if (t1>0.887) and (t1<=0.997) then lincf:=3.854211e-1+t1*(-8.735664e-1+
               t1*(4.987718e-1))
   else if (t1>0.977) and (t1<=1.0) then lincf:=0.804-(t1-0.977)*(0.865-
               0.804)/0.023
   else if (t1>1.0) and (t1<=1.1) then lincf:=0.865-(t1-1)/0.1*0.045
   else lincf:=0.82;
```

```
    end;
2:  begin
    t2:=1-t1;
    if t2>=0.653 then lincf:=0.0553*(1-t2)/0.317
    else if (t2<0.653) and (t2>=0.110) then lincf:=0.822*exp(-4.134*t2)
    else if (t2<0.110) and (t2>=0.066) then lincf:=1.29*exp(-8.228*t2)
    else if (t2<0.066) and (t2>=0.0) then lincf:=1.005-58.78*t2*t2
    else if (t2<0.0) and (t2>=-0.1) then lincf:=1.005-t2/0.1*(1.005-0.82)
    else lincf:=0.82;
    end;
3:  begin
    t2:=1-t1;
    if t2>=0.858 then lincf:=0.018*(1-t2)/0.142
    else if (t2<0.858) and (t2>=0.636) then lincf:=3.38*exp(-6.103*t2)
    else if (t2<0.636) and (t2>=0.306) then lincf:=0.657*exp(-3.528*t2)
    else if (t2<0.306) and (t2>=0.0) then lincf:=1.739*exp(-6.716*t2)
    else if (t2<0.0) and (t2>=-0.1) then lincf:=1.739-t2/0.1*(1.739-0.82)
    else lincf:=0.82;
    end;
4:  begin
    if t1<=0.5 then lincf:=0.074816*t1/0.5
    else if (t1>0.5) and (t1<=0.887) then lincf:=4.202556e-1-t1*(-2.025129+
            t1*(3.364271+t1*(-1.391541)))
    else if (t1>0.887) and (t1<=0.977) then lincf:=1.854211e-1-t1*
            (-8.735664e-1+t1*(4.987718e-1))
    else if (t1>0.977) and (t1<=1.0) then lincf:=0.804+(t1-0.977)*(0.865-
            0.804)/0.023
    else if (t1>1.0) and (t1<=1.1) then lincf:=0.865-(t1-1)/0.1*0.045
    else lincf:=0.82;
    end;
5:  begin
    t2:=1-t1;
    if t2>=0.858 then lincf:=0.018
    else if (t2<0.858) and (t2>=0.636) then lincf:=3.38*exp(-6.103*t2)
    else if (t2<0.636) and (t2>=0.306) then lincf:=0.657*exp(-3.528*t2)
    else if (t2<0.306) and (t2>=0.0) then lincf:=1.739*exp(-6.716*t2)
    else if (t2<0.0) and (t2>=-0.1) then lincf:=1.739+t2/0.1*(1.739-0.82)
    else lincf:=0.82;
    end;
6:  begin
    if t1<=0.0 then lincf:=0.0
    else if (t1>0.0) and (t1<=1.0) then lincf:=t1*0.6
    else lincf:=0.6;
    end;
7:  begin
    t2:=1-t1;
    if t2>=0.858 then lincf:=0.018*(1-t2)/0.142
    else if (t2<0.858) and (t2>=0.636) then lincf:=3.38*exp(-6.103*t2)
    else if (t2<0.636) and (t2>=0.132) then lincf:=0.657*exp(-3.528*t2)
    else if (t2<0.132) and (t2>=0.0) then lincf:=1.0*exp(-6.716*t2)
    else if (t2<0.0) and (t2>=-0.1) then lincf:=1.0+t2/0.1*(1.0-0.7)
    else lincf:=0.7;
    end;
8:  begin
    if t1<=0.357 then lincf:=0.19*t1
    else if (t1>0.357) and (t1<=0.817) then lincf:=0.338*t1-0.05283
    else if (t1>0.817) and (t1<=1.0) then lincf:=1.6*t1-1.08388
    else if (t1>1.0) and (t1<=1.3) then lincf:=0.525-(t1-1)
    else if (t1>1.3) and (t1<2.0) then lincf:=0.225-0.475*(t1-1.3)/0.7
    else lincf:=0.8;
    end;
9:  begin
    t2:=1-t1;
    if t2>=0.858 then lincf:=(0.018*(1-t2)/0.142)*0.85
    else if (t2<0.858) and (t2>=0.636) then lincf:=(3.38*exp(-6.103*t2))*0.85
    else if (t2<0.636) and (t2>=0.306) then lincf:=(0.657*exp(-3.528*t2))*0.85
    else if (t2<0.306) and (t2>=0.0) then lincf:=(1.729*exp(-6.716*t2))*0.85
    else if (t2<0.0) and (t2>=-0.1) then lincf:=(1.739+t2/0.1*(1.739-0.82))*0.85
    else lincf:=0.82*0.85;
    end

end;  (* case *)


tension:=linrho*0.5*total_velocity*total_velocity*lincf*constr_area

end;


procedure run_data;
begin

height:=init_altitude-run_values[6,2];
u2:=run_values[1,2];
w2:=run_values[2,2]+run_values[3,2]*cg_to_tail;
gamma:=arctan(w2/u2);
total_velocity:=sqrt(run_values[1,2]*run_values[1,2]
            +(run_values[2,2]+run_values[3,2]*deltax_store)
            *(run_values[2,2]+run_values[3,2]*deltax_store));
linalpha:=arctan(run_values[2,2]/run_values[1,2]);
if run_values[1,2]<0.0 then linalpha:=pi-linalpha;
linrho:=1.225*exp(4.256*ln(1-2.2605e-5*height));
if (flag=1) and drogue then
begin
x_tail:=run_values[5,2]-cg_to_tail*cos(run_values[4,2])-linx2;
z_tail:=run_values[6,2]+cg_to_tail*sin(run_values[4,2])-linz2;
x_drogue:=u1*(time-drogue_time);
z_drogue:=w1*(time-drogue_time)+gravity*(time-drogue_time)*(time-
          drogue_time)/2.0;
drogue_d:=sqrt((x_drogue-x_tail)*(x_drogue-x_tail)-(z_drogue-z_tail)*
          (z_drogue-z_tail))
end;
```

```
if (not drogue) and (main_time=0) then
begin
  if flag=0 then main_chute;
  coefficient
end;
if flag=3 then coefficient
end;

procedure lin_print_out;
begin

writeln(time:5:1,'  ',run_values[5,2]:6:1,height:8:1,'  ',total_velocity:8:1,
        ' ',(linalpha*180/pi):7:1,' ',(run_values[3,1]*180/pi):7:1,' ',
        (gamma*180/pi):7:1,' ',tension:8:1);
if (tension>max_tension) then max_tension:=tension
end;


procedure choice;
var

reply     :integer;

begin
write(chr(12));
writeln('      INFLATION PROGRAM');
writeln;
writeln('      select');
writeln('      aerodynamic store   (1)');
writeln('      store as point mass (2)');
checkch;
if choice1=1 then aero_store:=true
else aero_store:=false;
write(chr(12));
writeln('      select');
writeln('      drogue     (1)');
writeln('      no drogue  (2)');
checkch;
if choice1=1 then drogue:=true
else drogue:=false
end;


procedure initialise;
begin
choice;
lin_input_data;
print_input;
init_variables;
run_data;
print_heading;
lin_print_out
end;

(*$I b:pjbpasinf1.text *)
```

```
(*      file pjbpasinf1.                                              *)
(*      this file contains the second part of the lingard inflation   *)
(*      program.                                                      *)


procedure drogue_separation;
begin

if aero_store then
begin
flag:=1;
u1:=u2*cos(run_values[4,2])+w2*sin(run_values[4,2]);
w1:=w2*cos(run_values[4,2])-u2*sin(run_values[4,2]);
run_values[2,2]:=(run_values[2,2]*store_mass-run_values[3,2]*
                  cg_to_tail*mass_total)/store_mass;
linx2:=run_values[5,2]-cg_to_tail*cos(run_values[4,2]);
linz2:=run_values[6,2]+cg_to_tail*sin(run_values[4,2]);
end

else
begin
flag:=2;
main_time:=main_time+time-1.0e-6
end;

store_mass:=store_mass-mass_total

end;


procedure drogue_impulse1;
begin

flag:=2;
main_time:=main_time+time-1.0e-6;
w1:=w1+gravity*(time-drogue_time);
u1:=u1*cos(run_values[4,2])-w1*cos(run_values[4,2]);
w1:=w1*cos(run_values[4,2])-u1*sin(run_values[4,2]);
run_values[1,2]:=(run_values[1,2]*store_mass+u1*mass_total)/(store_
mass+mass_total);
run_values[2,2]:=(run_values[2,2]*store_mass+w1*mass_total)/(store_
mass+mass_total);
linalpha:=arctan(run_values[2,2]/run_values[1,2]);

end;


procedure impulses;
begin

if drogue then
begin
if (time>=drogue_time) and (flag=0) then drogue_separation;
if (drogue_d>=drogue_line) and (flag=1) then drogue_impulse1;
if (time>=main_time) and (flag=2) then main_chute
end

else if (time>=main_time) and (flag=0) then main_chute
end;


procedure para;

begin

if flag=2 then
tension:=linrho*total_velocity*total_velocity*drogue_cd*
                                   drogue_area*0.5;
if flag=3 then coefficient

end;


procedure aitken;

var

value       :array[1..10,1..10] of real;

k           :integer;
m           :integer;
np1mm       :integer;
kpm         :integer;

begin
for k:=1 to n do value[1,k]:=(y[k]*(xc-x[k+1])-y[k+1]*(xc-x[k]))/
                              (x[k]-x[k+1]);
for m:=2 to n do
begin
np1mm:=n+1-m;
for k:=1 to np1mm do
begin
kpm:=k+m;
value[m,k]:=(value[(m-1),k]*(xc-x[kpm])-value[(m-1),(k+1)]*(xc-x[k]))/
            (x[k]-x[kpm])
end
end;
p:=value[n,1]

end;


procedure interpolate;

var

n             :integer;
m             :integer;
l             :integer;
gam           :xarray;
acoeff        :xarray;

begin
if aero_store then
begin
```

```
  for n:=1 to 3 do
    begin


    for m:=1 to 7 do
      begin
      gam[m]:=aero_data[1,m];
      acoeff[m]:=aero_data[(n+1),m]
      end;
    aitken(linalpha,aero_coeff[n],gam,acoeff,6)

    end

end

else for l:=1 to 3 do aero_coeff[l]:=0.0
end;


procedure derivitives;

var

qf             :real;
aeromom        :real;
nf             :real;
af             :real;
mdamp          :real;

f              :array[1..3] of real;


begin
interpolate;
run_data;

qf:=linrho*total_velocity*total_velocity/2.0;
nf:=(aero_coeff[1]*cos(linalpha)+aero_coeff[2]*sin(linalpha))*qf*base_
area;
af:=(aero_coeff[2]*cos(linalpha)-aero_coeff[1]*sin(linalpha))*qf*base_
area;
aeromom:=aero_coeff[3]*qf*base_area*base_dia;
mdamp:=-(cmqf*run_values[2,3]+cmaldot*aldot)
         *base_dia*base_dia*linrho*total_velocity*base_area/4.0;

f[1]:=-af-store_mass*gravity*sin(run_values[3,1])-tension*cos(gamma);
f[2]:=-nf+store_mass*gravity*cos(run_values[3,1])-tension*sin(gamma);
f[3]:=mdamp-nf*deltax_store-cg_to_tail*tension*sin(gamma)+aeromom;

run_values[1,3]:=f[1]/store_mass-run_values[3,2]*run_values[2,2];
run_values[2,3]:=f[2]/store_mass+run_values[3,2]*run_values[1,2];

if aero_store then run_values[3,3]:=f[3]/iyy
else run_values[3,3]:=0.0;

run_values[4,3]:=run_values[3,2];
run_values[5,3]:=run_values[1,2]*cos(run_values[3,1])+run_values[2,2]*
                 sin(run_values[3,1]);
run_values[6,3]:=run_values[2,2]*cos(run_values[3,1])-run_values[1,2]*
                 sin(run_values[3,1])

end;


procedure runge_kutta;

var

work           :real;

i              :integer;

o              :array[1..6] of real;
p              :array[1..6] of real;
z              :array[1..6,1..3] of real;

begin

derivitives;
for i:=1 to no_of_equations do
begin
o[i]:=run_values[i,1];
p[i]:=run_values[i,2];
work:=time_step*0.5*run_values[i,3];
run_values[i,1]:=o[i]+0.5*time_step*(p[i]+0.5*work);
run_values[i,2]:=p[i]+work;
z[i,1]:=work
end;

time:=time+0.5*time_step;
derivitives;

for i:=1 to no_of_equations do
begin
work:=time_step*0.5*run_values[i,3];
run_values[i,2]:=p[i]+work;
z[i,2]:=work
end;

derivitives;

for i:=1 to no_of_equations do
begin
work:=time_step*0.5*run_values[i,3];
run_values[i,1]:=o[i]+time_step*(p[i]+work);
run_values[i,2]:=p[i]+2.0*work;
z[i,3]:=work
end;

time:=time+0.5*time_step;
derivitives;

for i:=1 to no_of_equations do
begin
run_values[i,1]:=o[i]+time_step*(p[i]+(z[i,1]+z[i,2]+z[i,3])/3.0);
run_values[i,2]:=p[i]+(z[i,1]+2.0*(z[i,2]+z[i,3])+0.5*time_step*
```

```
                 run_values[i,3])/3.0
end
end;


procedure solve_equations;
begin

impulses;
para;
runge_kutta(6);
run_data

end;


procedure run;
begin

repeat
solve_equations;
lin_print_out;
until (time>=(time_limit-1.0e-6)) or (height<=0.0)

end
end.
```

```
(*      file wavunit1.                                                   *)
(*      this file contains the weight calculations for the following types of *)
(*      parachute:                                                       *)
(*      solid cloth round parachutes.                                    *)
(*      cruciform parachute.                                             *)


unit wavunit1;
interface

uses (*SU #5:linunit.code *) linunit,
     (*SU #5:wavunit2.code *) wavunit2;

procedure wav23;
procedure wav24;
procedure wav25;
procedure wav26;
procedure wav27;
procedure wav3;


implementation



procedure wav23;

var
x      :integer;

begin
if type_no in [1,22] then eta:=pi/n
else eta:=beta/2;
writeln('     input number of panels per gore (maximum 20)');
cti(np);
sv:=vx*so/100;                                                (* 1201 *)
hg:=hs-sqrt(sv*cos(eta)/(n*sin(eta)));                        (* 1202 *)
e[1]:=es-2*hg*sin(eta)/(np*cos(eta));                         (* 1203 *)
area[1]:=(e[1]+es-0.1)/2*(hg/np+0.05);                        (* 1204 *)
if np<>1 then
begin
  for x:=2 to np do
  begin
    e[x]:=e[x-1]-2*hg*sin(eta)/(np*cos(eta));                 (* 1205 *)
    area[x]:=(e[x]-e[x-1]-0.1)/2*(hg/np+0.05)                 (* 1206 *)
  end
end;
tof:=0;
if type_no in [1.2] then
begin
  for x:=1 to np do tof:=tof+area[x];                         (* 1207 *)
  tof:=tof*n                                                  (* 1207 *)
end
else
begin
  writeln('     input number of panel 1 (largest) missing');
  readln(onp[1]);
  if np<>1 then
  begin
    for x:=2 to np do
    begin
      writeln('     input number of panel '.x.' missing');
      readln(onp[x])
    end
  end;
for x:=1 to np do tof:=tof-(n-onp[x])*area[x]                 (* 1208 *)
end;
wf:=tof*wtf/1000;                                             (* 1209 *)
tol:=z*(le-0.2);                                              (* 1210 *)
wl:=tol/lmat.cweight;                                         (* 1211 *)
tot:=n*(hg/cos(eta)-0.2);                                     (* 1212 *)
wt:=tot*tmat.twweight/1000;                                   (* 1213 *)
if s_band then
begin
  if np=5 then rpn:=3                                         (* 1215 *)
  else rpn:=(np div 2)+1;                                     (* 1214 *)
  tr:=n*e[rpn]+0.2;                                           (* 1217 *)
  wr:=tr*rmat.twweight/1000                                   (* 1218 *)
end;
if vent_band then
begin
  tvb:=n*e[np]+0.2;                                           (* 1219 *)
  wvb:=tvb*vbmat.twweight/1000                                (* 1220 *)
end;
if skirt_band then
begin
  tsb:=n*es+0.2;                                              (* 1221 *)
  wsb:=tsb*sbmat.twweight/1000                                (* 1222 *)
end;
if vent_line then
begin
  tovl:=n*((hs-hg)/cos(eta)+0.2);                             (* 1223 *)
  wvl:=tovl/vlmat.cweight                                     (* 1224 *)
end
end;



procedure wav24;

var
x      :integer;

begin
writeln('     bi-conical parachute');
writeln('     input number of lower panels (maximum 10)');
cti(nlp);
writeln('     input number of upper panels (maximum 10)');
cti(nup);
sv:=vx*so/100;                                                (* 1225 *)
h1g:=h1-sqrt(sv*cos(beta1/2)/(n*sin(beta1/2)));               (* 1226 *)
e1:=2*h1*sin(beta1/2)/cos(beta1/2);                           (* 1227 *)
eb[1]:=es-2*h2*sin(beta2/2)/(nlp*cos(beta2/2));               (* 1228 *)
areab[1]:=(eb[1]-es+0.1)/2*(h2/nlp+0.05);                     (* 1229 *)
```

```
if nlp<>1 then
begin
  for x:=2 to nlp do
  begin
    eb[x]:=eb[x-1]-2*h2*sin(beta2/2)/(nlp*cos(beta2/2));                (* 1230 *)
    areab[x]:=(eb[x]+eb[x-1]+0.1)/2*(h2/nlp-0.05)                        (* 1231 *)
  end
end;
et[1]:=e1-2*h1g*sin(beta1/2)/(nup*cos(beta1/2));                         (* 1232 *)
areat[1]:=(et[1]-e1+0.1)/2*(h1g/nup+0.05);                               (* 1233 *)
if nup<>1 then
begin
  for x:=2 to nup do
  begin
    et[x]:=et[x-1]-2*h1g*sin(beta1/2)/(nup*cos(beta1/2));                (* 1234 *)
    areat[x]:=(et[x]-et[x-1]+0.1)/2*(h1g/nup-0.05)                       (* 1235 *)
  end
end;
tof:=0;
for x:=1 to nup do tof:=tof-n*areat[x];                                  (* 1236 *)
for x:=1 to nlp do tof:=tof-n*areab[x];                                  (* 1236 *)
wf:=tof*wtf/1000;                                                        (* 1237 *)
tol:=z*(le+0.2);                                                         (* 1238 *)
wl:=tol/lmat.cweight;                                                    (* 1239 *)
tot:=n*(h1g/cos(beta1/2)-h2/cos(beta2/2)+0.2);                           (* 1240 *)
wt:=tot*tmat.twweight/1000;                                              (* 1241 *)
if s_band then
begin
  tr:=n*e1+0.2;                                                          (* 1242 *)
  wr:=tr*rmat.twweight/1000                                              (* 1243 *)
end;
if vent_band then
begin
  tvb:=n*et[nup]-0.2;                                                    (* 1244 *)
  wvb:=tvb*vbmat.twweight/1000                                           (* 1245 *)
end;
if skirt_band then
begin
  tsb:=n*es+0.2;                                                         (* 1246 *)
  wsb:=tsb*sbmat.twweight/1000                                           (* 1247 *)
end;
if vent_line then
begin
  tovl:=n*((h1-h1g)/cos(beta1/2)+0.2);                                   (* 1248 *)
  wvl:=tovl/vlmat.cweight                                                (* 1249 *)
end
end;


procedure wav23;

var

x     :integer;

begin
writeln('      tri-conical parachute');
writeln;
writeln('      input number of lower panels (maximum 10)');
cti(nlp);
writeln('      input number of middle panels (maximum 10)');
cti(nmp);
writeln('      input number of upper panels (maximum 10)');
cti(nup);
sv:=vx*so/100;                                                           (* 1250 *)
h1g:=h1-sqrt(sv*cos(beta1/2)/(n*sin(beta1/2)));                          (* 1251 *)
e1:=2*h1*sin(beta1/2)/cos(beta1/2);                                      (* 1252 *)
e2:=2*h2*sin(beta2/2)/cos(beta2/2)+e1;                                   (* 1253 *)
eb[1]:=es-2*h3*sin(beta3/2)/(nlp*cos(beta3/2));                          (* 1254 *)
areab[1]:=(eb[1]-es-0.1)/2*(h3/nlp-0.05);                                (* 1255 *)
if nlp<>1 then
begin
  for x:=2 to nlp do
  begin
    eb[x]:=eb[x-1]-2*h3*sin(beta3/2)/(nlp*cos(beta3/2));                 (* 1256 *)
    areab[x]:=(eb[x]-eb[x-1]+0.1)/2*(h3/nlp-0.05)                        (* 1257 *)
  end
end;
em[1]:=e2-2*h2*sin(beta2/2)/(nmp*cos(beta2/2));                          (* 1258 *)
aream[1]:=(em[1]+e2+0.1)/2*(h2/nmp+0.05);                                (* 1259 *)
if nmp<>1 then
begin
  for x:=2 to nmp do
  begin
    em[x]:=em[x-1]-2*h2*sin(beta2/2)/(nmp*cos(beta2/2));                 (* 1260 *)
    aream[x]:=(em[x]+em[x-1]+0.1)/2*(h2/nmp+0.05)                        (* 1261 *)
  end
end;
et[1]:=e1-2*h1g*sin(beta1/2)/(nup*cos(beta1/2));                         (* 1262 *)
areat[1]:=(et[1]+e1+0.1)/2*(h1g/nup-0.05);                               (* 1263 *)
if nup<>1 then
begin
  for x:=2 to nup do
  begin
    et[x]:=et[x-1]-2*h1g*sin(beta1/2)/(nup*cos(beta1/2));                (* 1264 *)
    areat[x]:=(et[x]+et[x-1]+0.1)/2*(h1g/nup-0.05)                       (* 1265 *)
  end
end;
tof:=0;
for x:=1 to nlp do tof:=tof+areab[x];                                    (* 1266 *)
for x:=1 to nmp do tof:=tof+aream[x];                                    (* 1266 *)
for x:=1 to nup do tof:=tof+areat[x];                                    (* 1266 *)
tof:=tof*n;                                                              (* 1266 *)
wf:=tof*wtf/1000;                                                        (* 1267 *)
tol:=z*(le+0.2);                                                         (* 1268 *)
wl:=tol/lmat.cweight;                                                    (* 1269 *)
tot:=n*(h1g/cos(beta1/2)+h2/cos(beta2/2)+h3/cos(beta3/2)+0.2);           (* 1270 *)
wt:=tot*tmat.twweight/1000;                                              (* 1271 *)
if s_band then
begin
  tr:=n*(e1+e2)+0.4;                                                     (* 1272 *)
  wr:=tr*rmat.twweight/1000                                              (* 1273 *)
```

```
end;
if vent_band then
begin
  tvb:=n*et[nup]-0.2;                                      (* 1274 *)
  wvb:=tvb*vbmat.twweight/1000                             (* 1275 *)
end;
if skirt_band then
begin
  tsb:=n*es+0.2;                                           (* 1276 *)
  wsb:=tsb*sbmat.twweight/1000                             (* 1277 *)
end;
if vent_line then
begin
  tovl:=n*((h1-hlg)/cos(beta1/2)+0.2);                     (* 1278 *)
  wvl:=tovl/vlmat.cweight                                  (* 1279 *)
end
end;


procedure wav26;

var
x          :integer;


begin
e1:=es;                                                    (* 1280 *)
if type_no=5 then
begin
  writeln('      extended skirt 10% flat');
  es:=0.81*e1;                                             (* 1281 *)
  h2:=0.2*h1;                                              (* 1282 *)
  thetax:=arctan((e1-es)/(2*h2))                           (* 1283 *)
end
else
begin
  writeln('      extended skirt 14.3% full');
  h1:=0.405*d*cos(pi/n);                                   (* 1285 *)
  es:=e1*le/(le-0.116*d);                                  (* 1286 *)
  h2prime:=0.116*d;                                        (* 1287 *)
  thetax:=arcsin((e1-es)/(2*h2prime));                     (* 1288 *)
  h2:=h2prime*cos(thetax)                                  (* 1289 *)
end;
writeln('      input number of lower panels (maximum 10)');
cti(nlp);
writeln('      input number of upper panels (maximum 10)');
cti(nup);
eb[1]:=es+2*h2*sin(thetax)/(nlp*cos(thetax));              (* 1290 *)
areab[1]:=(eb[1]+es+0.1)/2*(h2/nlp+0.05);                  (* 1290a*)
if nlp<>1 then
begin
  for x:=2 to nlp do
  begin
    eb[x]:=eb[x-1]+2*h2*sin(thetax)/(nlp*cos(thetax));     (* 1291 *)
    areab[x]:=(eb[x]+eb[x-1]-0.1)/2*(h2/nlp+0.05)          (* 1292 *)
  end
end;
sv:=vx*so/100;                                             (* 1293 *)
hg:=h1-sqrt(sv*cos(pi/n)/(n*sin(pi/n)));                   (* 1294 *)
et[1]:=e1-2*hg*sin(pi/n)/(nup*cos(pi/n));                  (* 1295 *)
areat[1]:=(et[1]+e1+0.1)/2*(hg/nup+0.05);                  (* 1296 *)
if nup<>1 then
begin
  for x:=2 to nup do
  begin
    et[x]:=et[x-1]-2*hg*sin(pi/n)/(nup*cos(pi/n));         (* 1297 *)
    areat[x]:=(et[x]+et[x-1]-0.1)/2*(hg/nup+0.05)          (* 1298 *)
  end
end;
tof:=0;
for x:=1 to nlp do tof:=tof+areab[x];                      (* 1299 *)
for x:=1 to nup do tof:=tof+areat[x];                      (* 1299 *)
tof:=tof*n;                                                (* 1299 *)
wf:=tof*wtf/1000;                                          (* 1300 *)
tot:=n*(hg/cos(pi/n)+h2/cos(thetax)-0.2);                  (* 1601 *)
wt:=tot*tmat.twweight/1000;                                (* 1602 *)
if s_band then
begin
  tr:=n*e1+0.2;                                            (* 1603 *)
  wr:=tr*rmat.twweight/1000                                (* 1604 *)
end;
tol:=z*(le+0.2);                                           (* 1605 *)
wl:=tol/lmat.cweight;                                      (* 1606 *)
if vent_band then
begin
  tvb:=n*et[nup]+0.2;                                      (* 1607 *)
  wvb:=tvb*vbmat.twweight/1000                             (* 1608 *)
end;
if skirt_band then
begin
  tsb:=n*es+0.2;                                           (* 1609 *)
  wsb:=tsb*sbmat.twweight/1000                             (* 1610 *)
end;
if vent_line then
begin
  tovl:=n*((h1-hg)/cos(pi/n)+0.2);                         (* 1611 *)
  wvl:=tovl/vlmat.cweight                                  (* 1612 *)
end;
end;


procedure wav27;

var
x          :integer;

begin
sv:=vx*so/100;                                             (* 1613 *)
dv:=sqrt(2*sv/pi);                                         (* 1614 *)
cv:=pi*dv/n;                                               (* 1615 *)
dc:=sqrt(2*so/pi);                                         (* 1616 *)
hsprime:=pi*dc/4;                                          (* 1617 *)
hgprime:=hsprime-dv/2;                                     (* 1618 *)
```

```
writeln('     hemispherical parachute');
writeln;
writeln('     input number of panels per gore (maximum 20)');
cti(np);
e[1]:=(ev-es)/np-es;                                          (* 1619 *)
if np<>1 then
for x:=2 to np do e[x]:=(ev-es)*x/np+es;                      (* 1620 *)
area[1]:=(es+0.05)*(hgprime/np+0.05);                         (* 1621 *)
if np<>1 then
for x:=2 to np do area[x]:=(e[x-1]+0.05)*(hgprime/np+0.05);   (* 1622 *)
tof:=0;
for x:=1 to np do tof:=tof+area[x];                           (* 1623 *)
tof:=tof*n;                                                   (* 1623 *)
wf:=tof*wtf/1000;                                             (* 1624 *)
tol:=z*(le+0.2);                                              (* 1625 *)
wl:=tol/lmat.cweight;                                         (* 1626 *)
tot:=n*(hgprime-0.2);                                         (* 1627 *)
wt:=tot*tmat.twweight/1000;                                   (* 1628 *)
if s_band then
begin
  if np=3 then rpn:=3;                                        (* 1630 *)
  if np<>5 then rpn:=trunc(np/2)+1;                           (* 1631 *)
  tr:=n*e[rpn]-0.2;                                           (* 1632 *)
  wr:=tr*rmat.twweight/1000                                   (* 1633 *)
end;
if vent_band then
begin
  tvb:=n*ev+0.2;                                              (* 1634 *)
  wvb:=tvb*vbmat.twweight/1000                                (* 1635 *)
end;
if skirt_band then
begin
  tsb:=n*es-0.2;                                              (* 1636 *)
  wsb:=tsb*sbmat.twweight/1000                                (* 1637 *)
end;
if vent_line then
begin
  tovl:=n*(dv/2+0.2);                                         (* 1638 *)
  wvl:=tovl/vlmat.cweight                                     (* 1639 *)
end
end;



procedure wav3;

begin
tof:=4*n*(es+0.05)*((dcru-n*es)/2+0.05)+n*(es-0.05)*n*(es+0.05);   (* 1642 *)
wf:=tof*wtf/1000;                                             (* 1643 *)
tol:=z*(lc+0.2);                                              (* 1644 *)
wl:=tol/lmat.cweight;                                         (* 1645 *)
tot:=z/2*(dcru-0.2)+4*(n*es-0.2);                             (* 1646 *)
wt:=tot*tmat.twweight/1000;                                   (* 1647 *)
wtot:=wt+wl+wf                                                (* 1648 *)
end
end.
```

```
(*      file wavunit2.                                                      *)
(*      this file contains the weight and volume calculations for the      *)
(*      following types of parachute:                                       *)
(*      ringslot,                                                           *)
(*      ringsail,                                                           *)
(*      disk-gap-band,                                                      *)
(*      rotafoil,                                                           *)
(*      parawing (single keel),                                             *)
(*      ribbon.                                                             *)

unit wavunit2;
interface

uses (*$U #5:linunit.code *) linunit;


type
twmaterial  = record
              twpart_no     :string;
              twspec        :string;
              twstrength    :real;
              twweight      :real;
              twwidth       :real;
              twcolours     :string
                end;
cmaterial   = record
              cpart_no      :string;
              cspec         :string;
              cstrength     :real;
              cweight       :real;
              ccolours      :string;
              crely         :integer
                end;

var
dcru            :real;                  (* span (of cruciform)           914 *)
type_no2        :char;                  (* type_no (ringslot)                *)
mu              :real;                  (* cone angle                   1005 *)
rw              :real;                  (* ring width                   1043 *)
nr              :integer;               (* no. of rings                 1043 *)
sw              :real;                  (* slot width                   1043 *)
vert_tape       :boolean;
nvt             :integer;               (* no. vertical tapes/gore      1666 *)
tvt             :real;                  (* total vertical tape (m)      1667 *)
wvt             :real;                  (* weight vertical tape (kg)    1668 *)
vtmat           :twmaterial;            (* vertical tape material            *)
tlp             :real;                  (* area lower panels            1774 *)
tup             :real;                  (* area upper panels            1779 *)
rh4             :real;                  (* ratio h4 (type 19)           1800 *)
rh3             :real;                  (* ratio h3 (type 19)           1800 *)
rh2             :real;                  (* ratio h2 (type 19)           1800 *)
h4              :real;                  (* gore height                  1800 *)
aa              :real;                  (* length (type 19)             1805 *)
b               :real;                  (* length (type 19)             1806 *)
wtot            :real;                  (* total weight                 1640 *)
vol             :real;                  (* volume of chute              1641 *)
type_no         :integer;
eta             :real;                  (* cone angle                   1202 *)
n               :integer;               (* number of gores              932a*)
beta            :real;                  (* gore angle                   1005 *)
np              :integer;               (* number of panels             1203 *)
e               :array[1..40] of real;  (* gore widths                  1203 *)
sv              :real;                  (* vent area                    1041 *)
vx              :real;                  (* vent area as a % of so       1041 *)
hg              :real;                  (* gore height                  1042 *)
hs              :real;                  (* gore height                  1002 *)
es              :real;                  (* gore width                   1003 *)
area            :array [1..20] of real; (* gore areas                   1204 *)
tof             :real;                  (* total fabric (m*m)           1207 *)
onp             :array [1..20] of integer;(* panels missing types 21.22 1208 *)
wtf             :real;                  (* weight of fabric (gm/m*m)    1209 *)
wf              :real;                  (* weight of fabric (kg)        1209 *)
tol             :real;                  (* total line length (m)        1210 *)
z               :integer;               (* number of lines               502 *)
lmat            :cmaterial;             (* line material properties          *)
vlmat           :cmaterial;             (* vent line properties              *)
tmat            :twmaterial;            (* tape material propertiies         *)
sbmat           :twmaterial;            (* skirt band properties             *)
vbmat           :twmaterial;            (* vent band properties              *)
le              :real;                  (* line length                   313 *)
wl              :real;                  (* line weight                  1211 *)
tot             :real;                  (* total tape length            1212 *)
wt              :real;                  (* weight of tape (kg)          1213 *)
s_band          :boolean;               (* reinforcing                       *)
vent_band       :boolean;
rpn             :integer;               (* reinforcing panel number     1215 *)
tr              :real;                  (* total reinforcing (m)        1217 *)
wr              :real;                  (* reinforcing weight (kg)      1218 *)
tvb             :real;                  (* total vent band (m)          1219 *)
vent_line       :boolean;
skirt_band      :boolean;
wvb             :real;                  (* vent band weight (kg)        1220 *)
tsb             :real;                  (* total skirt band (m)         1221 *)
wsb             :real;                  (* weight of skirt band (kg)    1222 *)
tovl            :real;                  (* total vent line (m)          1223 *)
wvl             :real;                  (* weight of vent line (kg)     1224 *)
nlp             :integer;               (* number of lower panels       1228 *)
nmp             :integer;               (* number of middle panels      1258 *)
nup             :integer;               (* number of upper panels       1232 *)
hig             :real;                  (* gore height                  1226 *)
beta1           :real;                  (* cone angle (types 3.4)       1009 *)
beta2           :real;                  (* cone angle (types 3.4)       1010 *)
beta3           :real;                  (* cone angle (type 4)          1017 *)
e1              :real;                  (* gore width                   1227 *)
h1              :real;                  (* gore height                  1012 *)
eb              :array [1..10] of real; (* gore widths                  1228 *)
em              :array [1..10] of real; (* gore widths                  1258 *)
et              :array [1..10] of real; (* gore widths                  1232 *)
areab           :array [1..10] of real; (* fabric areas                 1229 *)
aream           :array [1..10] of real; (* fabric areas                 1259 *)
areat           :array [1..10] of real; (* fabric areas                 1233 *)
h2              :real;                  (* gore height                  1011 *)
e2              :real;                  (* gore width                   1253 *)
h3              :real;                  (* gore height                  1020 *)
```

```
rmat          :twmaterial;                  (* reinforcing material             *)
h2prime       :real;                        (* gore height (type 6)        1287 *)
thetax        :real;                        (* gore height (type 5,6)      1283 *)
dv            :real;                        (* vent diameter               1050 *)
ev            :real;                        (* vent width (type 7)         1615 *)
dc            :real;                        (* constructed diameter         402 *)
hsprime       :real;                        (* gore height                 1617 *)
hgprime       :real;                        (* gore height                 1618 *)
thetap        :real;                        (* gore angle type 23          1829 *)
thetax1       :real;                        (* gore angle type 15          1686 *)
thetax2       :real;                        (* gore angle type 15          1685 *)
nhr           :integer;                     (* number of horiz. ribbons    1656 *)
hrmat         :twmaterial;                  (* horiz. ribbon properties         *)
gw            :real;                        (* gap width                   1657 *)
thr           :real;                        (* total horiz. ribbon length  1660 *)
whr           :real;                        (* weight of horiz. ribbon     1661 *)
emax          :real;                        (* gore width type 15          1676 *)
zz            :real;                        (* gore height type 15         1692 *)
tvtl          :array[1..10] of real;        (* total vert. tapes lower     1711 *)
tvtu          :array[1..10] of real;        (* total vert. tapes upper     1714 *)
sg            :real;                        (* total gore area                  *)
tape_hr       :real;                        (* overlap of tapes on h.ribbon     *)
v_tape_hr     :real;                        (* overlap of vert tapes on hr      *)
str           :real;                        (* total area of ribbons            *)
lambda_g      :real;                        (* geometric porosity               *)


function arcsin(boverc:real):real;
procedure wav5;
procedure wav6;
procedure wav7;
procedure wav8;
procedure wav9;
procedure porosity;
procedure wav45;
procedure wav46;


implementation


function arcsin;
begin
arcsin:=arctan(boverc/sqrt(1-boverc*boverc))
end;


procedure wav5;

var
x          :integer;

begin
if (type_no2 in ['a','A']) then eta:=pi/n                            (* 1727 *)
else eta:=arcsin(sin(pi/n)*cos(mu));                                 (* 1728 *)
e[1]:=es-2*rw*sin(eta)/cos(eta);                                     (* 1729 *)
area[1]:=(e[1]-es+0.1)/2*(rw+0.05);                                  (* 1730 *)
for x:=2 to nr do
begin
  e[2*x-2]:=e[2*x-3]-2*sw*sin(eta)/cos(eta);                         (* 1731 *)
  e[2*x-1]:=e[2*x-2]-2*rw*sin(eta)/cos(eta);                         (* 1732 *)
  area[x]:=(e[2*x-1]+e[2*x-2]+0.1)/2*(rw+0.05)                       (* 1733 *)
end;
tof:=0;
for x:=1 to nr do tof:=tof-area[x];                                  (* 1734 *)
tof:=tof*n;                                                          (* 1734 *)
wf:=tof*wtf/1000;                                                    (* 1735 *)
tol:=z*(le+0.2);                                                     (* 1736 *)
wl:=tol/lmat.cweight;                                                (* 1737 *)
tot:=n*(hg/cos(eta)-0.2);                                            (* 1738 *)
wt:=tot*tmat.twweight/1000;                                          (* 1739 *)
if vert_tape then
begin
  writeln('     ringslot parachute');
  writeln;
  writeln('     input number of vertical tapes per gore (maximum 10)');
  cti(nvt);
  if nvt=0 then tvt:=0                                               (* 1740 *)
  else
  begin
    tvt:=0;
    if nvt in [1,3,5,7,9] then
    begin
      tvt:=tvt+hg+0.2;                                               (* 1740a*)
      if nvt>1 then for x:=1 to (nvt div 2) do tvt:=tvt+2*(x+hg/(nvt div 2)-0.2)
                                                                     (* 1741 *)
    end
    else for x:=1 to (nvt div 2) do tvt:=tvt+2*(2*x*hg/(nvt+1)+0.2); (* 1741a*)
    tvt:=tvt*n
  end;
  wvt:=tvt*vtmat.twweight/1000                                       (* 1742 *)
end;
if skirt_band then
begin
  tsb:=n*es+0.2;                                                     (* 1743 *)
  wsb:=tsb*sbmat.twweight/1000                                       (* 1744 *)
end;
if vent_band then
begin
  tvb:=n*e[2*nr-1]-0.2;                                              (* 1745 *)
  wvb:=tvb*vbmat.twweight/1000                                       (* 1746 *)
end;
if vent_line then
begin
  tovl:=n*((hs-hg)/cos(eta)-0.2);                                    (* 1747 *)
  wvl:=tovl/vlmat.cweight                                            (* 1748 *)
end;
wtot:=wvb-wsb-wvt+wt+wl-wf-wvl                                       (* 1749 *)
end;



procedure wav6;
```

```
var
x       :integer;

begin
eta:=arctan(es/(2*hs));                                      (* 1751 *)
e[1]:=es-2*rw*sin(eta)/cos(eta);                             (* 1752 *)
area[1]:=(e[1]-es-0.1)/2*(rw-0.05);                          (* 1753 *)
for x:=2 to nr do
begin
  e[x]:=e[x-1]-2*rw*sin(eta)/cos(eta);                       (* 1754 *)
  area[x]:=(e[x]-e[x-1]+0.1)/2*(rw+0.05)                     (* 1755 *)
end;
tof:=0;
for x:=1 to nr do tof:=tof+n*area[x];                        (* 1756 *)
wf:=tof*wtf/1000;                                            (* 1757 *)
tol:=z*(le+0.2);                                             (* 1758 *)
wl:=tol/lmat.cweight;                                        (* 1759 *)
if vert_tape then
begin
  tvt:=n*(hg-0.2);                                           (* 1760 *)
  wvt:=tvt*vtmat.twweight/1000                               (* 1761 *)
end;
if vent_band then
begin
  tvb:=n*e[nr]+0.2;                                          (* 1762 *)
  wvb:=tvb*vbmat.twweight/1000                               (* 1763 *)
end;
if vent_line then
begin
  tovl:=n*((hs-hg)+0.2);                                     (* 1764 *)
  wvl:=tovl/vlmat.cweight                                    (* 1765 *)
end;
if skirt_band then
begin
  tsb:=n*es+0.2;                                             (* 1766 *)
  wsb:=tsb*sbmat.twweight/1000                               (* 1767 *)
end;
wtot:=wsb-wvl+wvb+wvt+wl+wf                                  (* 1768 *)
end;


procedure wav7;

var
x          :integer;

begin
sv:=vx*so/100;                                               (* 1770 *)
hg:=h1-sqrt(sv*cos(pi/n)/(n*sin(pi/n)));                     (* 1771 *)
h2:=0.113*h1;                                                (* 1772 *)
h3:=0.33*h1;                                                 (* 1773 *)
writeln('     disc-gap-band parachute');
writeln;
writeln('     input number of lower panels (maximum 10)');
cti(nlp);
writeln('     input number of upper panels (maximum 10)');
cti(nup);
tlp:=nlp*(es+0.05)*(h3/nlp+0.05);                            (* 1774 *)
e[1]:=es-2*hg*sin(pi/n)/(nup*cos(pi/n));                     (* 1775 *)
area[1]:=(es-e[1]-0.1)/2*(hg/nup-0.05);                      (* 1776 *)
if nup<>1 then
begin
  for x:=2 to nup do
  begin
    e[x]:=e[x-1]-2*hg*sin(pi/n)/(nup*cos(pi/n));             (* 1777 *)
    area[x]:=(e[x-1]+e[x]-0.1)/2*(hg/nup+0.05)               (* 1778 *)
  end
end;
tup:=0;
for x:=1 to nup do tup:=tup+area[x];                         (* 1779 *)
tof:=n*(tup+tlp);                                            (* 1780 *)
wf:=tof*wtf/1000;                                            (* 1781 *)
tol:=z*(le+0.2);                                             (* 1782 *)
wl:=tol/lmat.cweight;                                        (* 1783 *)
tot:=n*(hg/cos(pi/n)+h2-h3+0.2);                             (* 1784 *)
wt:=tot*tmat.twweight/1000;                                  (* 1785 *)
if s_band then
begin
  tr:=2*(n*es-0.2);                                          (* 1786 *)
  wr:=tr*rmat.twweight/1000                                  (* 1787 *)
end;
if vent_band then
begin
  tvb:=n*e[nup]+0.2;                                         (* 1788 *)
  wvb:=tvb*vbmat.twweight/1000                               (* 1789 *)
end;
if skirt_band then
begin
  tsb:=n*es+0.2;                                             (* 1790 *)
  wsb:=tsb*sbmat.twweight/1000                               (* 1791 *)
end;
if vent_line then
begin
  tovl:=n*((h1-hg)/cos(pi/n)+0.2);                           (* 1792 *)
  wvl:=tovl/vlmat.cweight                                    (* 1793 *)
end;
wtot:=wvl-wsb+wvb+wr+wt+wl+wf                                (* 1794 *)
end;


procedure wav8;

var
x          :integer;

begin
writeln('     rotafoil parachute');
writeln;
writeln('     input ratio h4:h3:h2');
writeln('     input rh4');
ctr(rh4);
writeln('     input rh3');
ctr(rh3);
writeln('     input rh2');
ctr(rh2);
```

```
dc:=1.05*d;                                                   (* 1796 *)
sv:=vx/100*so;                                                (* 1797 *)
hg:=hs-sqrt(sv*cos(pi/n)/(n*sin(pi/n)));                      (* 1798 *)
h1:=0.1*dc*cos(pi/n);                                         (* 1799 *)
h4:=(hs-h1)*rh4/(rh4+rh3+rh2);                                (* 1800 *)
h3:=(hs-h1)*rh3/(rh4+rh3+rh2);                                (* 1801 *)
h2:=(hs-h1)*rh2/(rh4+rh3+rh2);                                (* 1802 *)
e[1]:=es-2*h1*sin(pi/n)/cos(pi/n);                            (* 1803 *)
area[1]:=(es+e[1]-0.1)/2*(h1+0.05);                           (* 1804 *)
aa:=(h4+h3+hs-hg)*sin(5*pi/180)/cos(5*pi/180);                (* 1805 *)
b:=(h2+h3+h4+hs-hg)*sin(5*pi/180)/cos(5*pi/180);              (* 1806 *)
e[2]:=(e[1]-2*h2*sin(pi/n)/cos(pi/n))/2+aa;                   (* 1807 *)
area[2]:=(h2+0.05)*(3*e[2]/2+e[1]/4+0.025+b/2);               (* 1808 *)
e[3]:=e[1]-2*(h2-h3)*sin(pi/n)/cos(pi/n);                     (* 1809 *)
area[3]:=(h3-0.05)*(e[2]-h3*sin(pi/n)/(2*cos(pi/n))+0.025-
          e[3]/4-aa/2);                                       (* 1810 *)
e[4]:=e[3]-2*h4*sin(pi/n)/cos(pi/n);                          (* 1811 *)
area[4]:=(e[4]+e[3]+0.1)/2*(h4+0.05);                         (* 1812 *)
tof:=n*(area[1]-area[2]-area[3]+area[4]);                     (* 1813 *)
wf:=tof*wtf/1000;                                             (* 1814 *)
tol:=z*(lc+0.2);                                              (* 1815 *)
wl:=tol/lmat.cweight;                                         (* 1816 *)
tot:=n*(hg/cos(pi/n)+0.2);                                    (* 1817 *)
wt:=tot*tmat.twweight/1000;                                   (* 1818 *)
if s_band then
begin
  tr:=n*e[3]+0.2-n*(e[1]/2-b-0.2)+n*(h2/cos(5*pi/180)+0.2)+
    n*(sqrt((e[3]/2-a)*(e[3]/2-a)+h3*h3)+0.2);                (* 1819 *)
  wr:=tr*rmat.twweight/1000                                   (* 1820 *)
end;
if vent_band then
begin
  tvb:=n*e[4]+0.2;                                            (* 1821 *)
  wvb:=tvb*vbmat.twweight/1000                                (* 1822 *)
end;
if skirt_band then
begin
  tsb:=n*es+0.2;                                              (* 1823 *)
  wsb:=tsb*sbmat.twweight/1000                                (* 1824 *)
end;
if vent_line then
begin
  tovl:=n*((hs-hg)/cos(pi/n)+0.2);                            (* 1825 *)
  wvl:=tovl/vlmat.cweight                                     (* 1826 *)
end;
wtot:=wvl-wsb+wvb-wr+wt-wl+wf                                 (* 1827 *)
end;


procedure wav9;

var
x        :integer;
```

```
begin
thetap:=0.395;                                                                 (* 1829 *)
area[1]:=(0.125*dc-0.025)*(es/sqrt(2)+0.025)-
         (0.125*dc-0.025)*(es*sin(thetap)/(sqrt(2)*cos(thetap))+0.025)+
         0.5*(0.125*dc*sin(thetap)/cos(thetap)+0.025)*(0.125*dc+0.05)+
         0.5*(es*sin(thetap)/(sqrt(2)*cos(thetap))+0.025)*(es/sqrt(2)+0.025)+
         0.5*(es/sqrt(2)-0.025)*(es/sqrt(2)+0.025);                            (* 1830 *)
e[1]:=0.125*dc/cos(thetap)+es/(sqrt(2)*cos(thetap));                           (* 1831 *)
for x:=2 to n do
begin
  area[x]:=(e[x-1]+es*sin(thetap)/(2*cos(thetap))-0.075/2)*(es+0.025);
                                                                               (* 1832 *)
  e[x]:=es*sin(thetap)/cos(thetap)+e[x-1]                                      (* 1833 *)
end;
tof:=0;
for x:=1 to n do tof:=tof+2*area[x];                                           (* 1835 *)
wf:=tof*wtf/1000;                                                              (* 1836 *)
tol:=z*(le+0.2);                                                               (* 1837 *)
wl:=tol/lmat.cweight;                                                          (* 1838 *)
tot:=2*(0.766*dc+0.2)+0.875*dc+0.2;                                            (* 1839 *)
for x:=1 to n do tot:=tot+2*(e[x]+0.2);                                        (* 1839 *)
wt:=tot*tmat.twweight/1000;                                                    (* 1840 *)
if skirt_band then
begin
  tsb:=2*(n*es+0.2)+0.25*dc+0.2;                                               (* 1841 *)
  wsb:=tsb*sbmat.twweight/1000                                                 (* 1842 *)
end;
wtot:=wsb-wt+wl+wf                                                             (* 1843 *)
end;


procedure porosity;

var
x            :integer;

begin
if type_no=15 then
sg:=n*((ev+emax)/2*hg*cos(thetax1)+(emax+e1)/2*h2*cos(thetax2))                (* 1852 *)
else
begin
  if vent_band then sg:=n*hg*(es-e[nhr-1])/2                                   (* 1853 *)
  else sg:=n*(es*hg/2)                                                         (* 1854 *)
end;
tape_hr:=hrmat.twwidth/1000*tmat.twwidth/1000*nhr*n;                           (* 1855 *)
if nvt=0 then v_tape_hr:=0                                                     (* 1856 *)
else
begin
  if nvt in [1,3,5,7,9] then
  v_tape_hr:=((nvt+1) div 2)*nhr*n*hrmat.twwidth/1000*vtmat.twwidth/1000+
      (nvt-1)*(tmat.twwidth/1000/2-hrmat.twwidth/1000)*vtmat.twwidth/1000*n
                                                                               (* 1857 *)
  else
```

```
  begin
    v_tape_hr:=0;                                                         (* 1858 *)
    for x:=1 to (nvt div 2) do v_tape_hr:=v_tape_hr+2*x;                  (* 1859 *)
    v_tape_hr:=v_tape_hr*2*nhr*n*hrmat.twwidth/1000*vtmat.twwidth/1000/(nvt+1)+
              nvt*(tmat.twwidth/(2*1000)-
              hrmat.twwidth/1000)*vtmat.twwidth/1000*n                    (* 1860 *)
  end
end;
str:=(thr-0.2*nhr)*hrmat.twwidth/1000+(tot-0.2*n)*tmat.twwidth/1000+
     (tvt-n*nvt*0.2)*vtmat.twwidth/1000-tape_hr-v_tape_hr;                (* 1861 *)
lambda_g:=(sg-str)/sg*100;                                                (* 1862 *)
if lambda_g<0 then lambda_g:=0                                            (* 1862 *)
end;


procedure wav45;

var
x         :integer;

begin
writeln('     ribbon parachute');
n:=z;
if type_no=12 then eta:=pi/n                                              (* 1650 *)
else
begin
  writeln('     input cone angle mu (degrees)');
  ctr(mu);
  mu:=mu*pi/180;
  beta:=2*arcsin(sin(pi/n)*cos(mu));                                      (* 1651 *)
  eta:=beta/2                                                             (* 1652 *)
end;
hs:=sqrt(so*cos(eta)/(n*sin(eta)));                                       (* 1653 *)
if vent_band then
begin
  writeln('     input vent area as a percentage of so');
  ctr(vx)
end
else vx:=0;
sv:=vx*so/100;                                                            (* 1654 *)
hg:=hs-sqrt(sv*cos(eta)/(n*sin(eta)));                                    (* 1655 *)
es:=2*hs*sin(eta)/cos(eta);                                               (* 1655a*)
writeln('     input number of horizontal ribbons (maximum 40)');
cti(nhr);
while (nhr*hrmat.twwidth/1000)>hg do                                      (* 1656 *)
begin
  writeln('     too many ribbons input a lower number');
  cti(nhr)
end;
gw:=(hg-nhr*hrmat.twwidth/1000)/(nhr-1);                                  (* 1657 *)
e[1]:=es-2*(gw+hrmat.twwidth/1000)*sin(eta)/cos(eta);                     (* 1658 *)
for x:=2 to (nhr-1) do
e[x]:=e[x-1]-2*(gw+hrmat.twwidth/1000)*sin(eta)/cos(eta);                 (* 1659 *)
thr:=n*es+0.2;                                                            (* 1660 *)
for x:=1 to (nhr-1) do thr:=thr+n*e[x]+0.2;                               (* 1660 *)
whr:=thr*hrmat.twweight/1000;                                             (* 1661 *)
tol:=z*(le-0.2);                                                          (* 1662 *)
wl:=tol/lmat.cweight;                                                     (* 1663 *)
tot:=n*(hg/cos(eta)+0.2);                                                 (* 1664 *)
wt:=tot*tmat.twweight/1000;                                               (* 1665 *)
writeln('     input number of vertical tapes per gore (maximum 10)');
cti(nvt);
if nvt=0 then tvt:=0                                                      (* 1666 *)
else
begin
  tvt:=0;
  if nvt in [1,3,5,7,9] then
  begin
    tvt:=tvt+hg;                                                          (* 1667 *)
    if nvt>1 then tvt:=tvt+(nvt div 2)*hg                                 (* 1667a*)
  end
  else for x:=1 to (nvt div 2) do tvt:=tvt+2*(x*2*hg/(nvt+1));            (* 1667b*)
  tvt:=n*(tvt+nvt*0.2)
end;
wvt:=tvt*vtmat.twweight/1000;                                             (* 1668 *)
if skirt_band then
begin
  tsb:=n*es+0.2;                                                          (* 1669 *)
  wsb:=tsb*sbmat.twweight/1000                                            (* 1670 *)
end;
if vent_line then
begin
  tovl:=n*((hs-hg)/cos(eta)+0.2);                                         (* 1671 *)
  wvl:=tovl/vlmat.cweight                                                 (* 1672 *)
end;
if vent_band then
begin
  tvb:=n*e[nhr-1]+0.2;                                                    (* 1673 *)
  wvb:=tvb*vbmat.twweight/1000                                            (* 1674 *)
end;
wtot:=wvl+wl+wvb+wsb+wvt+wt+whr;                                          (* 1725 *)
porosity;
writeln;
writeln('     geometric porosity = ',lambda_g:5:2,' percent');
writeln;
writeln('     select:');
writeln('     (1) continue');
writeln('     (2) re-calculate weight and volume');
checkch;
if choice1=2 then wav45
end;


procedure wav46;

var
x      :integer;
m      :integer;
xx     :real;

begin
writeln('     hemisflo parachute');
n:=z;
dc:=sqrt(360*so/(210*pi));                                                (* 1675 *)
emax:=pi*dc/n;                                                            (* 1676 *)
es:=pi*dc*sin(75*pi/180)/n;                                               (* 1677 *)
```

```
el:=es;                                                          (* 1677a*)
hsprime:=0.9136*dc;                                              (* 1678 *)
h1:=6*hsprime/7;                                                 (* 1679 *)
h2:=hsprime/7;                                                   (* 1680 *)
if vent_band then
begin
  writeln('     input vent area as a percentage of so');
  ctr(vx);
  sv:=vx*so/100;                                                 (* 1681 *)
  dv:=sqrt(2*sv/pi);                                             (* 1682 *)
  ev:=pi*dv/n;                                                   (* 1683 *)
  hg:=h1-dv/2                                                    (* 1684 *)
end
else
begin
  vx:=0;
  sv:=0;
  dv:=0;
  ev:=0;
  hg:=h1
end;
thetax2:=arcsin((emax-es)/(2*h2));                               (* 1685 *)
thetax1:=arcsin(emax/(2*h1));                                    (* 1686 *)
writeln('     input number of horizontal ribbons (maximum 40)');
cti(nhr);
while (nhr*hrmat.twwidth/1000)>(h2*cos(thetax2)-hg*cos(thetax1)) do  (* 1687 *)
begin
  writeln('     too many ribbons input a lower number');
  cti(nhr)
end;
gw:=(h2*cos(thetax2)-hg*cos(thetax1)-hrmat.twwidth/1000*nhr)/(nhr-1);(* 1688 *)
m:=0;
xx:=0;                                                           (* 1689 *)
while m<nhr do
begin
  m:=m+1;                                                        (* 1701 *)
  if (xx+hrmat.twwidth/1000)>=(h2*cos(thetax2)) then             (* 1690 *)
  begin
    e[m]:=emax;                                                  (* 1691 *)
    zz:=h2*cos(thetax2)-(m-1)*(hrmat.twwidth/1000+gw);           (* 1692 *)
    e[m+1]:=emax-2*(hrmat.twwidth/1000-zz+gw)*sin(thetax1)/
                                              cos(thetax1);      (* 1693 *)
    for x:=(m+2) to nhr do
    e[x]:=e[x-1]-2*(hrmat.twwidth/1000+gw)*sin(thetax1)/
                                            cos(thetax1);        (* 1694 *)
    m:=nhr
  end
  else
  begin
    if m=1 then e[1]:=el+2*hrmat.twwidth/1000*sin(thetax2)/
                                              cos(thetax2)       (* 1695 *)
    else e[m]:=e[m-1]+2*(hrmat.twwidth/1000+gw)*
                                  sin(thetax2)/cos(thetax2);     (* 1703 *)
    xx:=x+hrmat.twwidth/1000                                     (* 1696 *)
  end;
  if (xx+gw)>=(h2*cos(thetax2)) then                             (* 1697 *)
  begin
    zz:=h2*cos(thetax2)-m*(hrmat.twwidth/1000+(m-1)*gw);         (* 1698 *)
    e[m+1]:=emax-2*(gw-zz)*sin(thetax1)/cos(thetax1);            (* 1699 *)
    for x:=(m+2) to nhr do
    e[x]:=e[x-1]-2*(hrmat.twwidth/1000-gw)*sin(thetax1)/cos(thetax1);(* 1700 *)
    m:=nhr;
  end
  else
  xx:=xx+gw                                                      (* 1702 *)
end;
thr:=0.2;                                                        (* 1704 *)
for x:=1 to nhr do thr:=thr+n*e[x];                              (* 1704 *)
whr:=thr*hrmat.twweight/1000;                                    (* 1705 *)
tol:=z*(le+0.2);                                                 (* 1706 *)
wl:=tol/lmat.cweight;                                            (* 1707 *)
tot:=n*(hg+h2+0.2);                                              (* 1708 *)
wt:=tot*tmat.twweight/1000;                                      (* 1709 *)
writeln('     input number of vertical tapes (maximum 10)');
cti(nvt);
if nvt=0 then tvt:=0                                             (* 1710 *)
else
begin
  hgprime:=hg*cos(thetax1)+h2*cos(thetax2);
  tvt:=0;
  if nvt in [1,3,5,7,9] then
  begin
    tvt:=tvt+hgprime;                                            (* 1711 *)
    if nvt>1 then tvt:=tvt+(nvt div 2)*hgprime                   (* 1712 *)
  end
  else for x:=1 to (nvt div 2) do tvt:=tvt+2*(x*2*hgprime/(nvt+1));  (* 1713 *)
  tvt:=n*(tvt+nvt*0.2)
end;
wvt:=tvt*vtmat.twweight/1000;                                    (* 1718 *)
if skirt_band then
begin
  tsb:=n*cs+0.2;                                                 (* 1719 *)
  wsb:=tsb*sbmat.twweight/1000                                   (* 1720 *)
end;
if vent_band then
begin
  tvb:=n*ev+0.2;                                                 (* 1721 *)
  wvb:=tvb*vbmat.twweight/1000                                   (* 1722 *)
end;
if vent_line then
begin
  tovl:=n*(dv/2+0.2);                                            (* 1723 *)
  wvl:=tovl/vlmat.cweight                                        (* 1724 *)
end;
wtot:=wvl+wl+wvb+wsb+wvt+wt+whr;                                 (* 1725 *)
porosity;
writeln;
writeln('     geometric porosity = ',lambda_g:5:2,' percent');
writeln;
writeln('     select:');
writeln('     (1) continue');
writeln('     (2) re-calculate weight and volume');
checkch;
if choice1=2 then wav46
end
end.
```

```
(*     file paradesign.                                              *)
(*     this file contains the following calculations:                *)
(*     weight and volume,                                            *)
(*     cost,                                                         *)
(*     stability,                                                    *)
(*     reliability,                                                  *)
(*     print out (optional).                                         *)


program parachute_design;
uses (*$U #5:linunit.code *) linunit,
     (*$U #5:wavunit2.code *) wavunit2,
     (*$U #5:wavunit1.code *) wavunit1;

const

m10101     =21;
m10102     =5;
m10301     =4;
m10302     =4;
m10303     =3;
m10304     =2;
m10401     =23;
m10701     =9;
m10901     =11;
m10902     =6;
m11001     =9;
m11002     =43;
m11003     =18;
m11501     =12;
m11502     =24;
m11503     =24;


var

horiz_ribbon    :boolean;
rlmat           :cmaterial;   (* reefing line material              *)
reef_line       :boolean;
part_no         :string;      (* part number of fabric              *)
spec            :string;      (* specification of fabric            *)
widef           :real;        (* width of fabric (m)                *)
wrl             :real;        (* weight of reefing line (kg)   1845 *)
reply2          :char;        (* y/n                                *)
xstaging        :boolean;
xcluster        :boolean;
replylf         :integer;     (* allowable load type                *)
method_no       :integer;     (* opening load calculation           *)
value           :integer;
type2           :integer;     (* type number                        *)
dfile           :text;        (* file of data                       *)
xreefing        :boolean;
fabric          :boolean;
tape            :boolean;
d_pack          :real;        (* packing density                    *)
```

```
(*$I b:pdesignvr1.text *)
(*$I b:pdesignvr2.text *)

(*$I b:pdesign1.text *)
(*$I b:pdesign2.text *)
(*$I b:pdesign3.text *)



procedure weight_and_volume;


begin
wt:=0;
wr:=0;
wvb:=0;
wsb:=0;
wvl:=0;
wf:=0;
whr:=0;
wvt:=0;
wrl:=0;
wl:=0;
write(chr(12));
writeln('     weight and volume calculation');
writeln;
if type_no in [1..7,21,22] then
begin
  if type_no in [1,2,21,22] then wav23;
  if type_no=3 then wav24;
  if type_no=4 then wav25;
  if type_no in [5,6] then wav26;
  if type_no=7 then wav27;
  wtot:=wsb+wvb+wr+wt+wl+wf+wvl                          (* 1640 *)
end;
case type_no of
11: wav3;
16: wav5;
17: wav6;
18: wav7;
19: wav8;
23: wav9;
15: wav46
end;
if type_no in [12,13] then wav45;
writeln;
writeln('     select:');
writeln('     (1) use packing density of 320 kg/m*m*m');
writeln('     (2) input alternative packing density');
checkch;
if choice1=2 then
begin
  writeln('     input packing density (kg/m*m*m)');
  ctr(d_pack)
```

```
end
else d_pack:=320;
vol:=wtot/dpack;                                                      (* 1641 *)
if reef_line then
begin
  wrl:=lr/rlmat.cweight;                                              (* 1845 *)
  wtot:=wtot-wrl;                                                     (* 1846 *)
  vol:=wtot/dpack                                                     (* 1847 *)
end;
if xcluster then
begin
  wtot:=etac*wtot;                                                    (* 1848 *)
  vol:=etac*vol                                                       (* 1849 *)
end;
writeln('     total weight of each component');
writeln;
if fabric then       writeln('          weight of fabric = ',wf:4:2,' kg');
if vent_line then    writeln('        weight of vent line = ',wvl:4:2,' kg');
if vent_band then    writeln('        weight of vent band = ',wvb:4:2,' kg');
if skirt_band then   writeln('       weight of skirt band = ',wsb:4:2,' kg');
if s_band then       writeln('      weight of reinforcing = ',wr:4:2,' kg');
if horiz_ribbon then writeln('     weight of horiz ribbon = ',whr:4:2,' kg');
if vert_tape then    writeln('      weight of vert. tapes = ',wvt:4:2,' kg');
if tape then         writeln('           weight of tapes = ',wt:4:2,' kg');
if reef_line then    writeln('     weight of reefing line = ',wrl:4:2,' kg');
                     writeln('           weight of lines = ',wl:4:2,' kg');
                     writeln;
                     writeln('              total weight = ',wtot:4:2,' kg');
                     writeln('                    volume = ',vol:4:3,
                     ' m*m*m');
continue;
if (wtot>aw) or (vol>av) then
begin
  write(chr(12));
  if wtot>aw then                                                     (* 1850 *)
  writeln('     the weight is too large')
  else if vol>av then                                                 (* 1851 *)
  writeln('     the volume is too large');
  writeln('     select');
  writeln('     (1) return to material choice');
  writeln('     (2) continue');
  writeln('     (3) exit');
  checkch2;
  if choice1=1 then
  begin
    select_material;
    weight_and_volume
  end;
  if choice1=3 then exit(parachute_design)
end;
writeln('     leaving weight and volume')
end;
```

```
procedure matcost(costring  :string;
                  matstring :string;
              var ctcost    :real;
                  matlength :real);

var
inpcost   :real;

begin
writeln('     input cost of ',costring,' material ',matstring:15,' (#/m)');
ctr(inpcost);
ctcost:=matlength*inpcost;
cotot:=cotot+ctcost                                                   (* 1311 *)
end;


procedure cost;

begin
write(chr(12));
writeln('     cost calculation');
writeln;
cotot:=0;
if fabric then
begin
  writeln('     input cost of fabric ',spec:15,' #/m*m');
  ctr(cof);
  cf:=cof*tof;                                                        (* 1301 *)
  cotot:=cotot+cf
end;
if vent_band then matcost('vent band',vbmat.twspec,cvb,tvb);
if skirt_band then matcost('skirt band',sbmat.twspec,csb,tsb);
if s_band then matcost('reinforcing',rmat.twspec,cr,tr);
if horiz_ribbon then matcost('horizontal ribbon',hrmat.twspec,costhr,thr);
if vert_tape then matcost('vertical tapes',vtmat.twspec,cvt,tvt);
if tape then matcost('tapes',tmat.twspec,ct,tot);
matcost('lines',lmat.cspec,cl,tol);
if vent_line then matcost('vent lines',vlmat.cspec,cvl,tovl);
if reef_line then matcost('reefing line',rlmat.cspec,crl,lr);
writeln;
writeln('     total cost (of materials) = # ',cotot:4:2);
continue;
writeln('     input allowable cost (#)');
ctr(aco);
if cotot>aco then                                                     (* 1312 *)
begin
  writeln('     materials cost is too high');
  writeln('     select');
  writeln('     (1) return to materials choice');
  writeln('     (2) continue');
  writeln('     (3) exit');
  checkch2;
  if choice1=1 then
  begin
    select_material;
    weight_and_volume;
```

```
    cost
  end;
  if choice1=3 then exit(parachute_design)
end
end;


procedure not_stable;

begin
writeln('     parachute is not stable');
writeln('     select:');
writeln('     (1) exit');
writeln('     (2) continue');
checkch;
if choice1=1 then exit(parachute_design)
end;


procedure stability;

var
x        :integer;


begin
write(chr(12));
writeln('     stability calculation');
writeln;
if (type_no in [1..7,12,13,15..19,21,22] ) and (por5<>99) and (por10<>99) then
begin
  if type_no in [1..7,16..19,21,22] then
  begin
    if (por5+por10)<>0 then
    begin
      k22:=(52.0236*por5*por5-2.6012*por10*por10)/(por10*por5*(por5-por10));
                                                                      (* 1401 *)
      k11:=(2.6012-k22*por5)/(por5*por5);                             (* 1402 *)
      deltap:=0.5*rho*v*v*1.967e-3*3.2808*3.2808;                     (* 1403 *)
      u:=0.3048*(-k22+sqrt(k22*k22+4*k11*deltap))/(2*k11)             (* 1404 *)
    end
    else u:=0;
    if type_no in [1..7,17] then acu:=sv;                             (* 1405 *)
    if type_no=16 then
    begin
      acu:=sv;
      for x:=1 to (nr-1) do acu:=acu+n*sw*e[2*x-1]                    (* 1406 *)
    end;
    if type_no=18 then acu:=sv+n*h2*es;                               (* 1407 *)
    if type_no=19 then
    acu:=sv-(e[1]+e[3])/2*(h2+h3)-area[2]-area[3];                    (* 1408 *)
    if type_no in[21,22] then
    begin
      acu:=sv;                                                        (* 1409 *)
      for x:=1 to np do acu:=acu+onp[x]*area[x]                       (* 1409 *)
    end;
    ma:=pi*dp*dp/4;                                                   (* 1410 *)
    vfr:=(so-acu)*u+acu*v;                                            (* 1411 *)
    stab:=vfr/(ma*v);                                                 (* 1412 *)
    if stab>0.1 then writeln('     parachute is stable')              (* 1413 *)
    else not_stable
  end;
  if type_no in [12,13,15] then
  begin
    acu:=sv;                                                          (* 1114 *)
    for x:=1 to (nhr-1) do acu:=acu+n*gw*e[x];                        (* 1111 *)
    ma:=pi*dp*dp/4;                                                   (* 1115 *)
    stab:=acu/ma;                                                     (* 1416 *)
    if stab>0.1 then writeln('     parachute is stable')              (* 1417 *)
    else not_stable
  end
end
else writeln('     stability data not avaliable');
continue
end;



procedure reliability;

var
x              :integer;
rmat           :string;
dof2           :integer;

begin
if lmat.crely=0 then exit(reliability);
write(chr(12));
writeln('     reliability');
writeln;
if xcluster then
begin
  writeln('     input number of parachutes in cluster that can fail');
  cti(nf);
  if nf=0 then cc:=exp((1/etac)*ln(0.9))                              (* 1502 *)
  else
  begin
    nnf:=etac-nf;                                                     (* 1503 *)
    cc:=exp((1/nnf)*ln(0.9))                                          (* 1504 *)
  end
end
else cc:=0.9487;                                                      (* 1501 *)
reset(dfile,'b:t11501.text');
readln(dfile,value);
if value<>m11501 then exit(reliability);
readln(dfile,rmat);
readln(dfile,nn,xbar,s);
x:=2;
while lmat.cspec<>rmat do
begin
  if x=(m11501-1) then exit(parachute_design);
  x:=x+1;
  readln(dfile,rmat);
  readln(dfile,nn,xbar,s)
```

```
end;
close(dfile);
dof:=nn-1;                                                    (* 1505 *)
ybar:=tl/sf;                                                  (* 1506 *)
z2:=(xbar-ybar)/s;                                            (* 1507 *)
x2:=sqrt(nn/dof)*z2;                                          (* 1508 *)
reset(dfile,'b:tl1502.text');
readln(dfile,value);
if value<>ml1502 then exit(reliability);
readln(dfile,dof2,m9,m95,m99);
while dof2<>dof do readln(dfile,dof2,m9,m95,m99);
close(dfile);
if cc<0.95 then mcc:=(cc-0.9)*(m95-m9)/0.05-m9                (* 1509 *)
else mcc:=(cc-0.95)*(m99-m95)/0.04+m95;                       (* 1510 *)
reset(dfile,'b:tl1503.text');
readln(dfile,value);
if value<>ml1503 then exit(reliability);
readln(dfile,dof2,c9,c95,c99);
while dof2<>dof do readln(dfile,dof2,c9,c95,c99);
close(dfile);
if cc<0.95 then ccc:=(cc-0.9)*(c95-c9)/0.05+c9                (* 1511 *)
else ccc:=(cc-0.95)*(c99-c95)/0.04+c95;                       (* 1512 *)
kc:=mcc*x2+ccc;                                               (* 1513 *)
u_sigma:=kc;                                                  (* 1514 *)
if u_sigma>=3.87 then r:=1.0                                  (* 1514a*)
else r:=0.50438+0.47381*u_sigma-0.14993*u_sigma*u_sigma+
        0.01565*u_sigma*u_sigma*u_sigma;                      (* 1515 *)
if cc<0.98 then tv:=29.4*cc-21.5                              (* 1516 *)
else tv:=100*cc-89;                                           (* 1517 *)
rp:=1-tv/1257;                                                (* 1518 *)
if not xcluster then rsys:=r*rp                               (* 1519 *)
else
begin
  if nf=0 then rsys:=exp(etac*ln(r*rp))                       (* 1520 *)
  else rsys:=(1-exp((nf+1)*ln(1-r*rp)))*exp((etac-nf-1)*ln(r*rp))   (* 1521 *)
end;
writeln('     reliability = ',rsys:5:4,' at 90% confidence');
continue
end;

procedure cp2( ystring  :string;
               pmaterial2 :cmaterial;
                freserve2 :real);

begin
writeln(dfile);
writeln(dfile,'     ',ystring);
writeln(dfile);
writeln(dfile,'     material properties');
with pmaterial2 do
begin
  writeln(dfile,'                 part no. =  ',cpart_no:12);
  writeln(dfile,'             specification = ',cspec:17);
  writeln(dfile,'                  strength = ',cstrength:4:1,' n');
  writeln(dfile,'                    weight = ',cweight:4:1,' m/kg');
  writeln(dfile,'            reserve factor = ',freserve2:3:1)
end
end;

procedure pt2(  xstring  :string;
                  pmaterial  :twmaterial;
                   freserve  :real);

begin
writeln(dfile);
writeln(dfile,'     ',xstring);
writeln(dfile);
writeln(dfile,'     material properties');
with pmaterial do
begin
  writeln(dfile,'                 part no. =  ',twpart_no:12);
  writeln(dfile,'             specification = ',twspec:17);
  writeln(dfile,'                  strength = ',twstrength:4:1,' n');
  writeln(dfile,'                    weight = ',twweight:4:1,' gm/m');
  writeln(dfile,'                     width = ',twwidth:4:1,' mm');
  writeln(dfile,'            reserve factor = ',freserve:3:1)
end
end;


procedure print_file;

var
x                 :integer;
restitle          :string;

begin
write(chr(12));
writeln('     do you require a datafile of the results ? (y/n)');
checkyn;
if reply1 in ['n','N'] then exit(print_file);
rewrite(dfile,'b:pararesult.text');
writeln;
writeln('     input the title for the results file');
readln(restitle);
writeln(dfile,restitle:25);
writeln(dfile);
write(dfile,'        type of parachute - ');
if type_no=16 then
begin
  if type_no2 in ['a','A'] then
  writeln(dfile,'flat ringslot')
  else writeln(dfile,'conical ringslot')
end
else
```

```
case type_no of
1:writeln(dfile,'flat circular');
2:writeln(dfile,'conical');
3:writeln(dfile,'bi-conical');
4:writeln(dfile,'tri-conical');
5:writeln(dfile,'extended skirt 10% flat');
6:writeln(dfile,'extended skirt 14.3% full');
7:writeln(dfile,'hemispherical');
11:writeln(dfile,'cruciform');
12:writeln(dfile,'flat ribbon');
13:writeln(dfile,'conical ribbon');
15:writeln(dfile,'hemisflo');
17:writeln(dfile,'ringsail');
18:writeln(dfile,'disc-gap-band');
19:writeln(dfile,'rotafoil');
21:writeln(dfile,'aeroconical');
22:writeln(dfile,'le moigne');
23:writeln(dfile,'parawing (single keel)')
end;
writeln(dfile);
writeln(dfile,'                          input data');
write(dfile,'          rate of descent = ');
if type_no in [21..23] then writeln(dfile,v*sin(theta1):4:2,' m/s')
else writeln(dfile,v:4:2,' m/s');
writeln(dfile,'                store size = ',ss:4:2,' m*m');
writeln(dfile,'              store weight = ',ws:6:1,' n');
writeln(dfile,'                  store cd = ',cds:4:2);
writeln(dfile,'       deployment velocity = ',vd:4:2,' m/s');
writeln(dfile,'       deployment altitude = ',a:5:1,' m');
writeln(dfile,'       deployment attitude = ',theta*180/pi:4:2,' degrees');
writeln(dfile,'            allowable mass = ',aw:4:2,' kg');
writeln(dfile,'          allowable volume = ',av:6:4,' m*m*m');
if (type_no in [2,13,21]) or ((type_no=16) and (type_no2 in ['R','b'])) then
writeln(dfile,'                cone angle = ',mu:4:2,' degrees');
if replylf=1 then writeln(dfile,'        maximum shock load = ',fa:6:1,' n')
else if replylf=2 then writeln(dfile,'         allowable loading = ',ga:4:2,
                                                            ' g');
if type_no in [21,22,23] then writeln(dfile,'       horizontal velocity = ',
                            vh:4:2,' m/s');
writeln(dfile);
writeln(dfile,'                              output');
if xcluster then
begin
  writeln(dfile,'     a cluster of ',etac,' parachutes');
  writeln(dfile,'                overall cd = ',cdfinal:4:2);
  writeln(dfile,'        area of each chute = ',se:4:2,' m*m');
  writeln(dfile,'    diameter of each chute = ',d:4:2,' m');
  writeln(dfile,'      length of lines (le) = ',le:4:2,' m');
  writeln(dfile,'      length of lines (lc) = ',lc:4:2,' m');
end
else
begin
  writeln(dfile,'          constructed area = ',so:4:2,' m*m');
  writeln(dfile,'      constructed diameter = ',d:4:2,' m');
  if (type_no in [1..19]) then
  writeln(dfile,'                        cd = ',cd:4:2)
else
writeln(dfile,'                         cr = ',cfr:4:2);
if type_no=11 then
begin
  writeln(dfile,'                  arm ratio = ',ar:4:2);
  writeln(dfile,'                   arm span = ',dcru:4:2);
  writeln(dfile,'                  arm width = ',t:4:2)
end;
writeln(dfile);
writeln(dfile,'                line length = ',le:4:2,' m');
end;
writeln(dfile);
writeln(dfile,'          number of lines = ',z);
if xstaging then
begin
  writeln(dfile);
  writeln(dfile,'                        staging');
  writeln(dfile,'  drogue constructed area = ',sod:4:2,' m*m');
  writeln(dfile,'   drogue constr diameter = ',dod:4:2,' m');
  writeln(dfile,'        trailing distance = ',it:4:2,' m');
  writeln(dfile)
end;
writeln(dfile);
if xstaging or xcluster or (not (method_no in [1,2,3])) then
writeln(dfile,'             opening load = ',f:5:1,' n')
else
begin
  write(dfile,'     opening load calculated using ');
  case method_no of
    1:write(dfile,'lingard');
    2:write(dfile,'pflanz');
    3:write(dfile,'mass ratio')
  end;
  if xreefing then writeln(dfile,' method = ',max_tension:5:1,' n')
  else writeln(dfile,' method = ',f:5:1,' n')
end;
if xreefing then
begin
  writeln(dfile);
  writeln(dfile,'                        reefing');
  writeln(dfile);
  writeln(dfile,'             opening load = ',f:5:1,' n');
  writeln(dfile,'          reefed diameter = ',dr:4:2,' m');
  writeln(dfile,'              reefed area = ',sr:4:2,' m*m');
  writeln(dfile,'   length of reefing line = ',lr:4:2,' m')
end;
writeln(dfile);
writeln(dfile,'     Structural strength of each component');
writeln;
writeln(dfile,'       safety factor used = ',sf:3:1);
writeln(dfile);
if fabric then writeln(dfile,'       strength of fabric = ',tf:4:2,' n/mm*50');
if vent_band then writeln(dfile,'    strength of vent band = ',vbs:4:2,' n');
if vent_line then writeln(dfile,'    strength of vent line = ',vls:4:2,' n');
if reef_line then writeln(dfile,' strength of reefing line = ',trl:4:2,' n');
writeln(dfile,'        strength of lines = ',tl:4:2,' n');
```

```
if tape then writeln(dfile,'        strength of tapes = ',tt:4:2,' n');
if skirt_band then writeln(dfile,'   strength of skirt band = ',sbs:4:2,' n');
if s_band then writeln(dfile,'  strength of reinforcing = ',sb:4:2,' n');
if vert_tape then writeln(dfile,' strength of vertic tapes = ',vts:4:2,' n');
if horiz_ribbon then writeln(dfile,'strength of horiz ribbons = ',hrs:4:2,' n');
if fabric then
begin
  writeln(dfile);
  writeln(dfile,'     fabric');
  writeln(dfile);
  writeln(dfile,'     material properties');
  writeln(dfile,'               part no. =  ',part_no:12);
  writeln(dfile,'           specification = ',spec:17);
  writeln(dfile,'        material strength = ',mtf:4:1,' n/mm*50');
  writeln(dfile,'           material width = ',widef:4:1,' m');
  writeln(dfile,'          material weight = ',wtf:4:1,' gm/m*m');
  if por10<98 then
  writeln(dfile,'    porosity at 10 in h2o = ',por10:3:1,' ft/sec');
  if por5<98 then
  writeln(dfile,'   porosity at 1/2 in h2o = ',por5:3:1,' ft/sec');
  writeln(dfile,'           reserve factor = ',rff:3:1)
end;
if skirt_band then pt2('skirt band',sbmat,rfsb);
if s_band then pt2('strengthening band',rmat,rfr);
if horiz_ribbon then
begin
  pt2('horizontal ribbon',hrmat,rfhr)
end;
if vert_tape then
begin
  pt2('vertical tape',vtmat,rfvt);
end;
if tape then pt2('tape',tmat,rft);
if vent_band then pt2('vent band',vbmat,rfvb);
cp2('lines',lmat,rfl);
if vent_line then cp2('vent line',vlmat,rfvl);
if reef_line then cp2('reefing line',rlmat,rfrl);
writeln(dfile);
if type_no in [1,2,5,6,7,12,13,21,22] then
begin
  writeln(dfile,'     construction details');
  writeln(dfile);
  if vent_band then
  begin
    writeln(dfile,'                  vent area = ',vx:4:2,' % of constructed area'
             );
    write(dfile,'       gore width at vent = ');
    if type_no=7 then writeln(dfile,ev:4:2,' m')
    else if type_no in [5,6] then writeln(dfile,et[nup]:4:2,' m')
    else if type_no in [12,13] then writeln(dfile,e[nhr-1]:4:2,' m')
    else writeln(dfile,e[np]:4:2,' m')
  end;
  writeln(dfile,'       maximum gore width = ',es:4:2,' m');
  if type_no=7 then writeln(dfile,'               gore height = ',hgprime
                              :4:2,' m')
  else writeln(dfile,'              gore height = ',hg:4:2,' m');
  if type_no in [5,6] then
  writeln(dfile,'number of panels per gore =  ',(nup+nlp))
  else if type_no in [12,13] then
  begin
    writeln(dfile,'  number of horiz ribbons =  ',nhr);
    writeln(dfile,' number of vertical tapes =  ',nvt);
    writeln(dfile,'                gap width = ',gw:5:3,' m')
  end
  else writeln(dfile,'number of panels per gore =  ',np);
  if type_no in [21,22] then
  begin
    writeln(dfile,' no. of largest panel out =  ', onp[1]);
    for x:=2 to np do writeln(dfile,'   no. of panel ',x,' missing =  ',onp[x])
  end;
  writeln(dfile)
end;
writeln(dfile,'     total weight of each component');
writeln(dfile);
if fabric then writeln(dfile,'          weight of fabric = ',wf:5:3,' kg');
if vent_line then writeln(dfile,'       weight of vent line = ',wvl:5:3,' kg');
if vent_band then writeln(dfile,'       weight of vent band = ',wvb:5:3,' kg');
if skirt_band then writeln(dfile,'     weight of skirt band = ',wsb:5:3,' kg');
if s_band then writeln(dfile,'    weight of reinforcing = ',wr:5:3,' kg');
if horiz_ribbon then writeln(dfile,
                      '   weight of horiz ribbon = ',whr:5:3,' kg');
if vert_tape then writeln(dfile,'    weight of vert. tapes = ',wvt:5:3,' kg');
if tape then writeln(dfile,'           weight of tapes = ',wt:5:3,' kg');
if reef_line then writeln(dfile,'   weight of reefing line = ',wrl:5:3,' kg');
writeln(dfile,'          weight of lines = ',wl:5:3,' kg');
writeln(dfile);
writeln(dfile,'             total weight = ',wtot:4:2,' kg');
writeln(dfile,'                   volume = ',vol:5:4,' m*m*m');
writeln(dfile);
writeln(dfile,'          packing density = ',d_pack:5:1,' kg/m*m*m');
if type_no in [12,13,15] then
begin
  writeln(dfile);
  writeln(dfile,'      geometric porosity = ',lambda_g:5:1,' percent')
end;
writeln(dfile);
if stab>0.1 then writeln(dfile,'         parachute is stable');
writeln(dfile);
if lmat.crely<>0 then
begin
  writeln(dfile,'              reliability = ',rsys:5:4,
                 ' at 90% confidence');
  writeln(dfile);
  if xcluster then writeln(dfile,' no. chutes that can fail = ', nf)
end;
close(dfile,lock);
writeln;
writeln('     the results are in a file "b:pararesult.text",');
writeln('     this file can be printed out if required')
end;
```

```
begin
select_type;
area_calculation;
cluster;
if not xcluster then line_length;
if not ((type_no=11) or (type_no=23)) then no_of_lines;
staging;
opening_loads;
reefing;
structural_strength;
select_material;
landing_control;
weight_and_volume;
(* cost; *)
stability;
reliability;
print_file

end.
```

```
(*     file pdesignvrl.                                                  *)
(*     this file contains variables from a to s.                         *)

     (*   list of variables    *)

     (*   aaaaaaaaaaaaaaaa     *)

ac          :real            (* cluster length (lc-le)          315 *);
al          :real            (* constant pflanz opening         702 *);
ar          :real            (* arm ratio (type 11)             913 *);
av          :real            (* allowable volume               1851 *);
aw          :real            (* allowable weight               1850 *);
aco         :real            (* allowable cost                 1312 *);
acu         :real            (* total vent and cut out         1405 *);

     (*   bbbbbbbbbbbbbbbb     *)


     (*   cccccccccccccccc     *)

c99         :real            (* constant at 99% confidence     1512 *);
c95         :real            (* constant at 95% confidence     1511 *);
c9          :real            (* constant at 90% confidence     1511 *);
ccc         :real            (* constant at confdnce coeff     1511 *);
cc          :real            (* confidence coefficient         1501 *);
cotot       :real            (* total cost                          *);
crl         :real            (* cost of reefing lines #             *);
cvl         :real            (* cost of vent lines #                *);
cl          :real            (* cost of lines #                     *);
ct          :real            (* cost of tape #                      *);
cfr         :real            (* force coefficient             10102 *);
cdpr        :real            (* cd projected reefed             942 *);
cdp_cdpo    :real            (* ratio of cdp to cdpo            936 *);
cds         :real            (* drag coefficient of store       607 *);
cdsd        :real            (* drogue drag area                607 *);
cy          :real            (* ratio cxl/cx                    714 *);
cxl         :real            (* cx for cluster non-synchro      713 *);
cx          :real            (* opening force coefficient       605 *);
cd          :real            (* drag coefficient (nominal)      101 *);
cdo_cprimedo :real           (* drag correction for lines       308 *);
cdfinal     :real            (* final drag coefficient          310 *);
cdc_cd      :real            (* drag coefficient (cluster)    10301 *);
cdseach     :real            (* drag area each parachute        303 *);
cxdr        :real            (* cx at disreef                   808 *);
cdsr        :real            (* reefed drag area                810 *);
cxr         :real            (* cx reefed                       821 *);
cdh         :real            (* high cd                       10101 *);
cdl         :real            (* low cd                        10101 *);
cof         :real            (* cost of fabric #/m*m           1301 *);
cf          :real            (* cost of fabric #               1301 *);
cvb         :real            (* cost of vent band #                 *);
csb         :real            (* cost of skirt band #                *);
cr          :real            (* cost of reinforcing #               *);
costhr      :real            (* cost of horiz ribbon #              *);
cvt         :real            (* cost of vertical tapes #            *);
cdpo        :real            (* cd (reefing line load)          941 *);

     (*   dddddddddddddddd     *)

dr          :real            (* reefed diameter                 817 *);
dp          :real            (* projected diameter              902 *);
dod         :real            (* nominal diameter of drogue      609 *);
dp_do       :real            (* ratio of dp to do             10901 *);
dpr         :real            (* diameter projected reefed       944 *);
dof         :integer         (* degrees of freedom             1505 *);

     (*   eeeeeeeeeeeeeeee     *)

esd         :real            (* design es (es+0.05)            1054 *);

     (*   ffffffffffffffff     *)

f           :real            (* max opening load                705 *);
fa          :real            (* allowable load (n)              602 *);
fdr         :real            (* allowable opening force         806 *);
fr          :real            (* reefed opening load             822 *);
fd          :real            (* design load                     901 *);
fprime_f    :real            (* ratio of fprime to f            949 *);
fprime      :real            (* reefing line load               950 *);

     (*   gggggggggggggggg     *)

gx          :real            (* opening load factor             801 *);
gxr         :real            (* reefed opening load factor      823 *);
ga          :real            (* allowable load factor           602 *);

     (*   hhhhhhhhhhhhhhhh     *)

hrs         :real            (* horizontal ribbon strength      922 *);
hc          :real            (* inflated circumference          945 *);
hx          :real            (* non-inflated circumference      946 *);

     (*   iiiiiiiiiiiiiiii     *)

lt          :real            (* trailing distance (drogue)      610 *);
idia        :integer         (* integer part of do              501 *);

     (*   jjjjjjjjjjjjjjjj     *)

     (*   kkkkkkkkkkkkkkkk     *)

k11         :real            (* constant (stability)           1402 *);
k22         :real            (* constant (stability)           1401 *);
k           :real            (* ratio h1/h2 (type 3)           1011 *);
k1          :real            (* ratio h1/h2 (type 4)           1018 *);
k2          :real            (* ratio h3/h2 (type 4)           1018 *);
kc          :real            (* constant (reliability)         1513 *);
```

```
      (*   llllllllllllllll   *)

lc          :real           (* cluster line length          314 *);
lr          :real           (* reefing line length          819 *);
lc_do       :real           (* ratio of lc to do            306 *);
le_do       :real           (* ratio of le to do            307 *);
le_dc       :real           (* ratio of le to dc          10401 *);

      (*   mmmmmmmmmmmmmmmm   *)

mtf         :real           (* material strength (fabric)  1036 *);
ma          :real           (* mouth area                  1410 *);
mcc         :real           (* gradient at conf. coeff.    1509 *);
m9          :real           (* gradient at 90% confidence  1509 *);
m95         :real           (* gradient at 95% confidence  1509 *);
m99         :real           (* gradient at 99% confidence  1510 *);

      (*   nnnnnnnnnnnnnnnn   *)

nf          :integer        (* no. chutes that can fail    1503 *);
nnf         :integer        (* no. chutes that can't fail  1503 *);
nn          :integer        (* reliability parameter       1505 *);

      (*   oooooooooooooooo   *)

      (*   pppppppppppppppp   *)

por5        :real           (* porosity at 1/2 in. h2o     1401 *);
por10       :real           (* porosity at 10 in. h2o      1401 *);

      (*   qqqqqqqqqqqqqqqq   *)

qdr         :real           (* dynamic pressure disreef     809 *);
qm          :real           (* allowable dynamic pressure   606 *);

      (*   rrrrrrrrrrrrrrrr   *)

rmdr        :real           (* mass ratio at disreef        807 *);
rmr         :real           (* mass ratio (reefed)          820 *);
rm          :real           (* mass ratio                   604 *);
rml         :real           (* mass ratio cluster           712 *);
rff         :real           (* reserve factor fabric       1056 *);
rfvb        :real           (* reserve factor vent band    1057 *);
rfsb        :real           (* reserve factor skirt band   1058 *);
rfr         :real           (* reserve factor reinforcing  1059 *);
rfhr        :real           (* reserve factor horiz. ribb  1060 *);
rfvt        :real           (* reserve factor vert. tape   1061 *);
rft         :real           (* reserve factor tape         1062 *);
rfl         :real           (* reserve factor lines        1063 *);
rfvl        :real           (* reserve factor vent lines   1064 *);
rfrl        :real           (* reserve factor reef line    1065 *);
rp          :real           (* packing reliability         1518 *);
rsys        :real           (* system reliability          1521 *);
r           :real           (* material reliability        1515 *);

      (*   ssssssssssssssss   *)

stab        :real           (* measure of stability        1412 *);
spr         :real           (* area projected reefed        943 *);
scc         :real           (* estimated area (cluster)     304 *);
se          :real           (* area of each parachute       311 *);
sr          :real           (* reefed area                  818 *);
sf          :real           (* safety factor                901 *);
sbs         :real           (* skirt band strength          908 *);
sb          :real           (* reinforcing strength         909 *);
ss          :real           (* area of store                607 *);
s           :real           (* reliability parameter       1507 *);
```

```
(*    file pdesignvr2.                                                *)
(*    this file contains variables from a to z and greek variables.   *)

     (*   list of variables (cont)   *)

     (*   tttttttttttttttt   *)

tv          :real             (* table value (reliability)   1517 *);
tf          :real             (* fabric strength              903 *);
tl          :real             (* line strength                905 *);
tt          :real             (* tape strength                911 *);
t           :real             (* width (type11 (cruciform))   913 *);
trl         :real             (* reefing line load            951 *);
tfil        :real             (* filling time                 701 *);

     (*   uuuuuuuuuuuuuuuu   *)

u           :real             (* velocity (stability)        1404 *);
u_sigma     :real             (* reliability parameter       1514 *);

     (*   vvvvvvvvvvvvvvvv   *)

v           :real             (* rate of descent              101 *);
vh          :real             (* horizontal velocity          204 *);
vt          :real             (* total velocity               205 *);
vbs         :real             (* vent band strength           906 *);
vls         :real             (* vent line strength           907 *);
vts         :real             (* vertical tape strength       928 *);
vfr         :real             (* volumetric flow rate        1411 *);

     (*   wwwwwwwwwwwwwwww   *)


     (*   xxxxxxxxxxxxxxxx   *)

x2          :real             (* reliability variable        1508 *);
x1          :real             (* load decreasing factor       703 *);
xbar        :real             (* reliability variable        1507 *);

     (*   yyyyyyyyyyyyyyyy   *)

ybar        :real             (* load (reliability)          1506 *);

     (*   zzzzzzzzzzzzzzzz   *)

z2          :real             (* mean of loads               1507 *);

     (*   greekgreekgreekgr   *)

theta1      :real             (* arctan(vh/v)                          204 *);
eta1        :integer          (* eta pflanz method                   10701 *);
eta2        :real             (* eta for calculation of tf           10701 *);
alpha       :real             (* line angle                            904 *);
rho         :real             (* air density                           201 *);
etac        :integer          (* number of parachutes                  302 *);
tau         :real             (* dr/do reefing ratio                   812 *);
zeta        :real             (* cdsr/cdso (reefing)                   811 *);
phi         :real             (* line angle (reefed)                   947 *);
psi         :real             (* angle of radial members               948 *);
mu1         :real             (* cone angle (types 3,4)               1009 *);
mu2         :real             (* cone angle (types 3,4)               1010 *);
mu3         :real             (* cone angle (type 4)                  1017 *);
deltap      :real             (* pressure drop                        1403 *);
```

```
(*      file t10101.                                                          *)
(*      this file contains drag coefficient data for non-gliding parachutes.  *)

21
01 flat circular
0.800 0.750
02 conical
0.900 0.750
03 bi-conical
0.920 0.750
04 tri-conical
0.960 0.800
05 extended skirt 10% flat
0.870 0.780
06 extended skirt 14.3% full
0.900 0.750
07 hemispherical
0.770 0.620
08 guide surface (ribbed)
0.420 0.280
09 guide surface (ribless)
0.340 0.300
10 annular
1.000 0.950
11 cross
0.820 0.600
12 flat ribbon
0.500 0.450
13 conical ribbon
0.550 0.500
14 conical ribbon (varied porosity)
0.650 0.550
15 ribbon (hemisflo)
0.460 0.300
16 ringslot
0.650 0.560
17 ringsail
0.900 0.750
18 disk-gap-band
0.580 0.520
19 rotafoil
0.990 0.850
20 vortex ring
1.800 1.300
28 balloon (ballute)
1.200 0.510
```

```
(*      file t10102.                                                          *)
(*      this file contains force coefficient data for gliding parachutes.     *)

5
21 aeroconical
0.875
22 le moigne
0.950
23 parawing (single keel)
1.000
24 parawing (twin keel)
1.050
25 parafoil (ram air)
0.800
```

```
(*      file t10301.                                                      *)
(*      this file contains drag coefficient correction due to a cluster data  *)
(*      for a flat circular parachute at a rate of descent of less than 7.62  *)
(*                                                                   m/sec   *)

4
1 1.000
2 0.980
3 0.965
4 0.920
```

```
(*      file t10302.                                                      *)
(*      this file contains drag coefficient correction for a cluster data,    *)
(*      for a flat circular parachute with a rate of descent of greater than   *)
(*      7.62 m/sec.                                                       *)

4
1 1.000
2 0.890
3 0.840
4 0.780
```

```
(*     file t10303.                                                     *)
(*     this file contains drag coefficient correction for a cluster data.    *)
(*     for a ringsail parachute with a nominal diameter greater than 9.30 m. *)


3
1 1.000
2 0.990
3 0.960
```

```
(*     file t10304.                                                     *)
(*     this file contains drag coefficient corection for a cluster data.     *)
(*     for a ringsail parachute of a nominal diameter greater than 9.30m..   *)


2
1 1.000
2 0.930
```

```
(*      file t1070'.                                                      *)
(*      this file contains pflanz opening load data (eta1 and eta2), and the  *)
(*      opening force factor, cx, for a number of types of parachute.      *)

9
01 flat circular
2  9.0 1.80
03 extended skirt 3%
2  9.0 1.40
06 extended skirt 14.3%
2 12.0 1.40
09 guide surface (ribless)
2  5.0 1.40
11 cross
1 11.8 1.20
12 flat ribbon
1 14.0 1.05
13 ribbon (hemisflo)
1 20.0 1.15
16 ringslot
1 14.0 1.05
17 ringsail
1  7.0 1.10
```

```
(*      file t1090l.                                                      *)
(*      this file contains vales of the ratio of nominal to projected (in  *)
(*      flight) diameter for solid cloth circular parachutes.              *)

11
01 flat circular
0.66
02 conical
0.70
03 bi-conical
0.70
04 tri-conical
0.70
05 extended skirt 10% flat
0.68
06 extended skirt 14.3% full
0.68
07 hemispherical
0.66
18 disk-gap-band
0.65
19 rotafoil
0.90
21 aeroconical
0.66
22 le moigne
0.66
```

```
(*    file t10902.                                                      *)
(*    this file contains values of the ratio of nominal to projected (in *)
(*    flight) diameter for slotted canopies.                            *)

6
12 flat ribbon
0.67
13 conical ribbon
0.70
14 conical ribbon (varied porosity)
0.70
15 ribbon (hemisflo)
0.62
16 ringslot
0.68
17 ringsail
0.69
```

```
(*    file t11001.                                                      *)
(*    this file contains fabric material data.                          *)

9
p00115 155 3
bsf 118/854
0.920  473  50 20.0 99.00 wrd
p00115 155 5
bsf 118/556a
0.920  510  50 10.0 99.00 w
p00115 117 1
bsf 118/793/4b
0.920  510  50 10.0 99.00 s
p00115 303 4
mil c 7020 type1
0.930  370  37 11.0  1.33 wo
p00115 312 3
gq ms 294
1.200  400  44 00.00 0.00 rbyxwbl
p00115 325 5
gq ms 330
1.220  950  85  3.00 0.23 d
-
gq ms 502 (bl)
1.170  480  54  0.00 0.00 -
-
n8726
1.420  400  40 99.0 99.00 -
-
gq ms 309
1.220  400  39  0.00 0.00 -
```

```
(*     file t11002.                                        *)
(*     this file contains tape and web material data.      *)

43
b01832 009 0
lac's 18/350
  350   1.50 16.00 w
p00167 730 9
gq ms 132
  535   3.00 15.00 w
p00168 055 2
gq ms 115
  890   3.80 18.00 w
p00167 050 7
mil t 5038 type3
  890   3.70  9.50 w
p00167 591 3
mil t 5038 type3
 1112   4.70 13.00 wdz
s99167 598 6
mil t 6134 type2
 1334   4.50 25.40 z
p02106 011 2
gq ms 289
 1500  38.00 23.50 w
p00167 550 7
gq ms 124
 1780   5.20 12.00 wdq
p00167 555 7
gq ms 318
 1780   5.50 12.00 wd
p00168 076 4
mil t 5038 type3
 1780   6.20 19.00 drwbaz
p00157 232 7
mil w 4088 type1
 2224   8.68 14.30 bdw
s99168 378 0
mil t 5038 type5
 2224   6.20 14.30 z
s99168 604 3
mil t 5038 type3
 2335   9.30 25.40 z
p00157 233 5
mil t 6134 type1
 2335  12.40 27.00 dz
s99167 599 4
mil t 5038 type4
 2446  10.85 12.70 b
s99168 582 6
mil w 4088 type2
 2670  13.02 25.40 z
p00167 620 2
gq ms 193
 2670   8.70  8.00 wd
p00167 380 8
gq ms 296
 3110  10.50 12.25 w
p00167 575 1
gq ms 158
 3115  10.30 12.50 wbd
p00169 391 3
mil t 5038 type2
 4003  12.40 25.00 d
p00168 850 0
mil t 5038 type1
 4448  15.50 25.40 bdyrqz
s99169 011 4
mil w 5625
 4448  15.50 12.70 z
p00168 890 8
gq ms 131
 5800  17.00 25.00 wkdg
p00168 939 8
gq ms 317
 5800  17.00 25.00 wd
s99169 021 1
mil w 5625
 6672  18.60 14.30 z
p00167 330 3
gq ms 252
 6675  22.40 12.00 w
s99169 015 6
mil w 5625
10008  23.25 15.90 z
s99169 020 3
mil w 5625
10230  32.55 19.00 y
s99169 005 9
mil w 4088 type17
11120  33.65 25.40 z
p00168 920 5
mil w 5625
17792  52.70 25.40 z
b02685 009 1
lac/s 1116
17800  39.50 25.00 wd
p00168 855 0
mil w 4088 type20
40032 100.80 25.40 w
b00754 009 3
gq ms 267
28900 90.0 42.5
p00172 382 1
gq ms 198
24200 89.0 30.0
p00172 425 9
```

```
gq ms 256
17800 72.5 48.0
b01103 009 2
bsf 124/224
1461 79.0 44.45
p00169 583 2
mil w 4088 type8
17792 49.6 43.66
p00169 590 5
mil t 5038 type2
5782 18.6 38.0
p00169 592 1
mil t 5038 type4
6672 23.3 38.0
p00171 632 4
mil w 4088 type12
5338 26.35 44.0
p00173 820 0
mil w 4088 type4
8006 37.2 76.00
s99172 900 2
mil t 5608
6672 67.64 50.8
-
lac/s 15
670 2.6 15.0
```

```
(*      file 111503.                                                    *)
(*      this file contains data for the constant in the non-central     *)
(*      t-distribution curve.                                           *)


24
 6 -0.2775 -0.2000 -0.3171
 7 -0.2548 -0.1380  0.0671
 8 -0.2364 -0.0990 -0.2257
 9 -0.2135 -0.1270 -0.2390
10 -0.2216 -0.1270 -0.2200
11 -0.1060 -0.1020 -0.1867
12 -0.1096 -0.0960 -0.2076
13 -0.1390 -0.1080 -0.1862
14 -0.1114 -0.0850 -0.1567
15 -0.1499 -0.0837 -0.1652
16 -0.1582 -0.0697 -0.1314
17 -0.1243 -0.0979 -0.1200
18 -0.1300 -0.0330 -0.0943
19 -0.1167 -0.0496 -0.0886
20 -0.1504 -0.0474 -0.0667
21 -0.1381 -0.0235 -0.1014
22 -0.1285 -0.0233 -0.1038
23 -0.1200  0.0157 -0.1143
24 -0.1486 -0.0020 -0.1167
29 -0.0563 -0.0252 -0.0690
34 -0.0455  0.0211 -0.0463
39 -0.0978 -0.0315 -0.0371
44 -0.0620  0.0579 -0.0526
49 -0.0866  0.0767  0.0055
```

```
(*      file njblib.                                                    *)
(*      this file is the unit library file.                             *)

#5:linunit.code
#5:wavunit2.code
#5:wavunit1.code
```

A Computer Solution to Parachute Design Problems

by P.J.Broadbent

Abstract

In this thesis a Pascal computer program is presented which calculates a proposed design of parachute from some simple input parameters, of the type specified by a customer to a parachute company. The program reduces by a significant degree time spent by parachute engineers in the preliminary design stages.

Parachute design is a process which (in common with much engineering design) can be regarded as consisting of a number of separate calculations. The most suitable method (or methods) for each calculation were selected after a thorough investigation of parachute design techniques. The chosen methods must be sufficiently accurate and readily conform to a computer treatment. The data required by the program have been collected from various sources and are stored in a number of files on a floppy disk.

The program is applied to requirements received by a parachute company and results obtained compared with the actual parachutes designed. The program is highly interactive with the user who is able to dispute its selection of values for various parameters. Because the designer can make a rapid and objective choice between a number of methods for various calculations, the existence of this program contributes to his knowledge of the relevance of the parameters involved in, and his understanding of, parachute design. Examples of these techniques are given in the text.

Possibilities for expanding and improving the program exist in a number of areas. In some cases the data required for a particular parachute or particular design methods are not available or do not exist. Provision has been made for such data to be included in the program when they are received.