CRANFIELD INSTITUTE OF TECHNOLOGY

DEPARTMENT OF AIRCRAFT DESIGN

PhD Thesis

Academic Year 1987

N.A.D. Murphy

Analytical wing weight prediction/estimation using computer based design techniques.

Supervisor: A.J. Morris

1987

This thesis is submitted for the degree of PhD.

Appendices

A WEIGHTS User Manual

B D.B.M.S. Manual

C Example of WEIGHTS Output

1. INTRODUCTION

Every pilot knows that the size and position of masses in an aircraft has a fundamental effect on its performance. The layman would probably intuitively sense this, except perhaps the jet-set holiday makers who insist on carrying baggage up to and beyond their allocation! It is a popular belief that the aircraft industry is at the forefront of technology, and in many fields this might be true. This belief is fostered by regular statements by the news media that "such and such a plane was designed by a computer". It comes as a surprise then, to find that the methods used for predicting aircraft weights today were developed in the forties and there have only been half hearted attempts at making use of the engineers' newest best friend - The Digital Computer!

The traditional methods are empirical and rely on experience gained from past projects. This was good enough when new aircraft were produced every year, when new designs were often developments of previous aircraft, and were part of a prototype development program. Things have changed and these methods are potentially inaccurate when applied by the inexperienced engineer to new aircraft based on radical concepts. This danger is increased now that the gestation period for new aircraft is typically a decade and the degree of innovation included is higher. For example, the formulae developed for a fighter aircraft like the phantom F-4 would yield extremely dubious results for a new generation swept forward wing, canard fighter, constructed with modern composite materials.

Now for some more surprises, 60% of an aeroplane's program cost is determined at the initial stages of design and the structure of an aircraft accounts for up to 55% of the cost of a 200 aircraft program. Compare this with the 3% share that the initial design stage itself costs. Mistakes made during this stage are dramatically costly as evidenced by the experience of the USA in developing their supersonic transport programs. There the weight predictions were only a few percent on the optimistic side but this meant that the resulting design had a very restricted range. Clearly the problem is that a small error can be very costly or even catastrophic and there appears to be no way around it.

An accurate computer based method of aircraft weight prediction which does not rely on a database of existing aircraft would help solve this problem and that, is what this project is about. Such a computer program for use specifically on wing-type structures has been developed.

It is structured around traditional computer based tools such as finite element structural modelling, automatic design optimisation and artificial intelligence.

The aim of this work is to test the feasibility of a computer based weight estimation tool which can be used to solve two basic problems, the first being this problem of guessing an aircraft weight whilst it is still a market analyst's dream. The second problem being the rapid calculation of the weight of an aircraft which is on the drawing board.

Not all people faced with the first basic problem would be engineers. They could be airlines trying to judge the worth of a manufacturer's claims or a manufacturer trying to size up the competition or even rival airforces sizing up each others' aircraft. The information needed by the computer at this stage would be things like physical size, materials, design speed and so on. The computer then goes through a complete design excercise in which it designs the component from first principles. The weight it provides as a prediction is the result of many design optimisation cycles similar to what an engineer would have to go through when actually designing the component. Note that there is a fundamental assumption made here and it is; the design process used to design the component in question has the same efficiency as the one used to obtain the prediction.

The second basic problem faced by design engineers is that it often takes as long to work out how heavy a component is as to design it. There is also a "Catch 22" situation here, since it's impossible to fine tune the efficiency of a structure without knowing its weight. This situation often stops the designer from being too innovative since exploration of alternative designs takes time which is often in short supply.

Using this program the engineer can 'describe' his/her ideas to a computer and see very quickly how the different ideas compare and gain a better understanding of the problem. And even if after trying out several ideas no improvement is achieved, won't it be nice to know you've been doing it right all these years?

The program itself is installed in the college's D.E.C. VAX11/750 computer and it's primary job is to make other programs "talk" to one another. This is because the methods it uses are based on traditional computer techniques for which there are existing programs. It would have been wasteful to ignore these programs and develop completely new ones. Because of the design of the program, what it can do is not as impressive as what it could do with a little extra effort. For example, it might be possible to make it "talk" to a stock taking and accounting program so that it may derive completely theoretically based cost estimates as well as weight estimates.

As not everybody can afford the hardware to run this program, there will always be room for the tradition methods which can be run successfully on a manual analogue computer (slide rule) costing a few pounds. And as always the engineers' pragmatic approach would mean these methods would be used as backups since every engineer knows that, larger and faster computers are good at making bigger, better and more mistakes!

To summarize, the aim of the project was to show that a purely analytical wing weight estimation computer program based on traditional computer aided design tools is feasible. A prototype program was constructed and tested. Early test results are promising and indicate the feasibility and validity of this approach. Many more tests and much more development work is needed to improve this technique to a "production" standard.

The aim of this document is to present a self contained package on the topic of modern weight prediction and estimation. It contains a discussion of techniques currently in use including a typical derivation of each technique with the emphasis on how these techniques should be used and adjusted.

Before the proposed computerised techniques is discussed a brief description is given of each

specialised field which are brought together by the new technique.

Following the description of the computerised technique, program phylosophy and architecture are discussed leading on to a sample analysis of a wing box. Conclusion of this analysis are then provided.

2. CURRENT PRACTICE and PROPOSALS for NEW METHODS

Some mention has been made of methods used today for predicting and estimating weights of aircraft and their components. To understand how these methods work and why they have limitations it is necessary to look at some representative examples more closely.

In this chapter we do just that, breaking down the existing methods into three classes; pure statistical; hybrid analytical - statistical and pure analytical methods. In each dissection the main assumptions and formulations are given and where possible, to increase the understanding and usefulness of these methods, ways of adjusting them for novel designs are given. Finally the pros and cons of each method are summarised. The aim here was to produce a user manual as opposed to a report.

Note: Most of the methods described are of the component build up type, where each major component is considered in isolation and the weights are summed to give an overall aircraft weight. When this occurs only the details about wing weight have been considered.

2.1 Empirical Methods

These methods are usually purely statistical by nature but it is possible to have experimentally based methods for determining structural weight. Instead of analysing a database of real aircraft information it would be possible to analyse a set of experimental (either practical or f.e. modelled ones) to obtain useful methods. As the ideas behind this sort of method are so simple only a brief account of one is given here.

2.1.1 WAATS - A Statistical Based Prediction Method

This has been chosen because of its simplicity and because it is typical of methods of this type. In addition it is fairly recent and has been coded for use on computers by Glatt.

WAATS stands for Weights Analysis of Advanced Transportation Systems and is fully described by Glatt.

2.1.2 Database Manipulation

The basis of the technique is the collection and analysis of relevant information about a particular component for as many aircraft as possible. The data collection is in itself a difficult task, but once obtained, there are often problems in deciphering the information.

For example, in our case the component of interest is the wing. AT first sight the process seems deceptively simple! We collect values of wing weights for a hundred different aircraft so we can start doing clever things like least square curve fits on them. However, there are often problems when defining the weight of a component. Do the control surfaces and all the ancillaries count as part of the wing weight? Probably not, so we ignore all those things like flaps, actuators, hydraulics and electrics. Having gone through this distillation process, which is bound to be painful, we still have problems.

Our first problem concerns the definition of the wing itself. Some wings go straight through the fuselage and so perhaps thats simple. Some wings break at the fuselage, well surely that's cheating, since in this case breaking the wing leaves a gap which might not be considered as part of the wing and so leads to a lighter wing weight but the overall solution might be heavier. In some cases like

delta wings with root mounted engines it's often hard to decide where the wing ends and the fuselage begins.

This is one reason why it is necessary to classify the data that has been collected into groups. Groups are usually classed by utility (which determines the loading) and by configuration (which covers details such as continuous or broken wings).

2.1.3 Formulae Derivation

Notation

W = weight

A = Empirical Coefficient

B = Empirical exponent

X = combination of parameters (e.g. span, sweep etc)

Units are Imperial (lbs, ft)

WTO = Gross weight

N = Ult. load factor

ST = Structural span

AREA = Gross wing area

T_R = Root thickness

WLAND = landing weight

Having carried out the distillation and spectographic process on the data it is plotted on a graph of suitable scale - often log-log scale, and assumptions about its behaviour are made. In this example installed weight is used.

In order to illustrate the procedure we assume that component weight behaviour follows the power law

$$W = A.X^B$$

The important parameters effecting weight are decided and the following general relationship assumed.

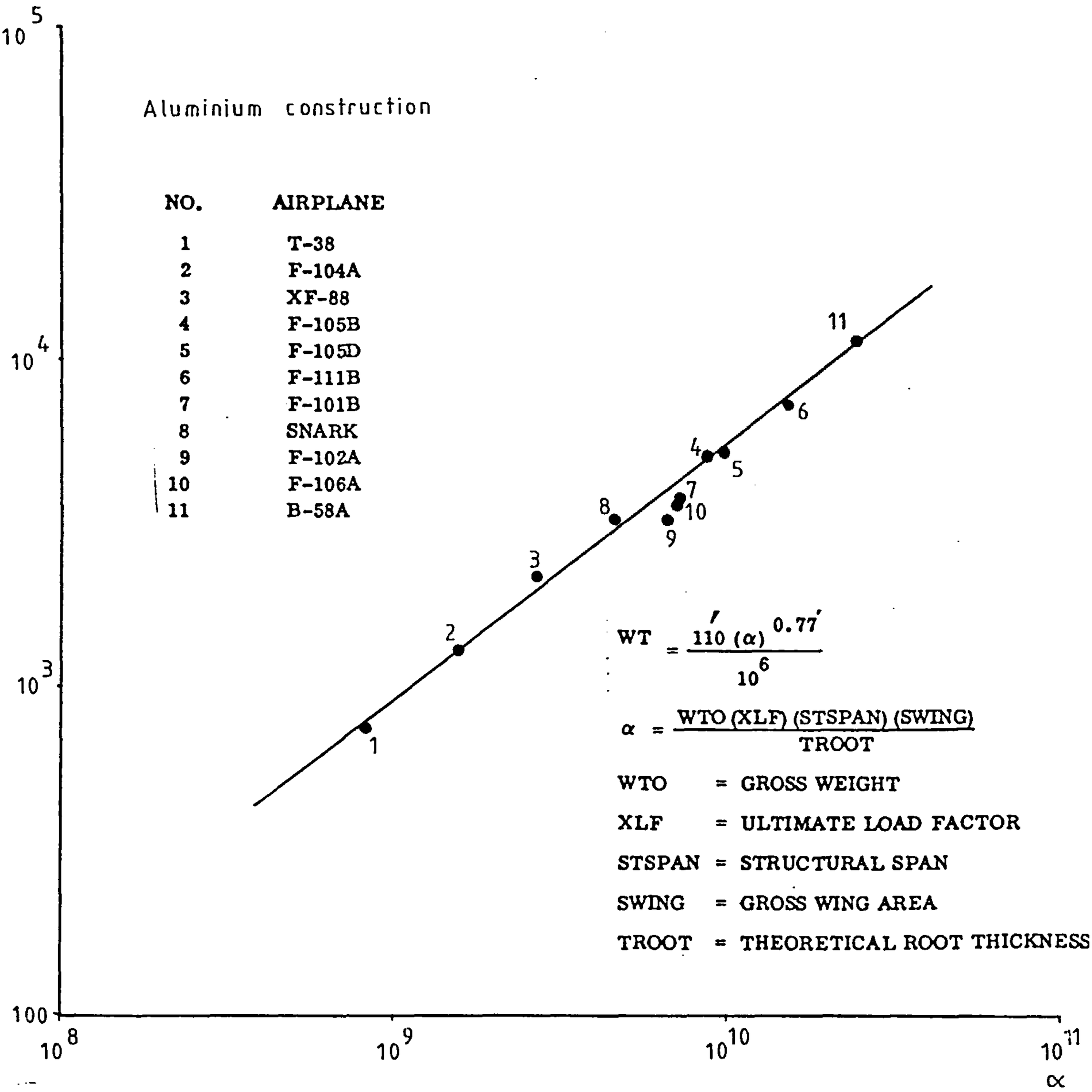


FIG. 2·1 Wing weight for high speed aircraft

$$W = A_1 \left(\frac{WTO \times N \times ST \times AREA}{T_R}\right)^{B_1} + A_2 \times AREA + A_3 + A_4 \left(\frac{WLAND \times N \times ST \times AREA}{T_R}\right)^{B_4}$$

Note that in some cases landing and take-off weight are the same or related so the equation would simplify.

A typical database and its matching line is shown in fig 2.1 reproduced from Glatt. The case illustrated applies to high speed aircraft of aluminium construction and matching gives the relationship

$$W = 110 \left(\frac{WTO \times N \times ST \times AREA}{T_R}\right)^{0.77} \times 10^{-6}$$

If it is assumed that we are dealing with high speed aircraft, high temperature construction then the relationship becomes,

$$W = 2905 \left(\frac{WTO \times N \times ST \times AREA}{T_R}\right)^{0.608} \times 10^{-6}$$

Whilst for Low to moderately swept wings

$$W = 1624 \left(\frac{WLAND \times N \times ST \times AREA}{T_R \times 10^9}\right)^{0.584}$$

The method for finding the values of coefficients and exponents A and B is simple. A least square curve fit or similar is carried out and a "line" through the data is found. (Note graph plotting is not essential). The values from two points on the "line" are inserted into the following equations.

$$B = \frac{\log (W_2 / W_1)}{\log (X_2 / X_1)}$$

and

$$A = \frac{W_i}{X_i^{\,B}}$$

2.1.4 Adjustments

It is possible to adjust the results slightly but it amounts to nothing more than applying a known factor. For example, suppose it has been discovered that the use of FRP leads to a 20% savings on weight. The line described by the equation we have been using would shift down by that much amount. To apply this shift one makes use of the following equation,

$$A_{new} = \frac{(W_{new} \ @ \ X)}{A_{old} \ (W_{old} \ @ \ X)}$$

B does not change provided the gradient of the line does not alter, otherwise the complete derivation is necessary.

This adjustment is of limited use.

2.1.5 Utility

This method typical of many in its class is very simple to apply and is useful for very early studies but amounts to little more than scaling of existing information. It has the drawback of having no theoretical basis and should the design fall outside the scope of the classes for which it has been derived large errors are likely. As things stand errors of around 15% are the norm.

2.2 Analytical - Historical Approach

This is the most common and currently the most useful method of weight estimation and is likely to remain so for some time. Typically, its formulation consists of two main stages, namely;

- The structural analysis of an idealised structure to obtain the form of the prediction equation. In other words the effect of the design parameters selected are calculated using engineering logic.

- The empirical comparison of the theoretical equation with actual weights of existing aircraft or structures. This empirical analysis adjusts, but does not change the form of, the theoretical equation to give the most accurate answers possible for a given group of results.

To make this clearer, some actual derivations of such equations would be necessary. Two examples follow, Lewis and St John (experimental - analytical) and St Johns (statistical - analytical) contain the full derivations, hence only a summary is given here. However, these references do not suggest ways in which these techniques could be manipulated for novel designs so we will place emphasis on the assumptions that were made and how they may be changed to cope with novel designs.

2.2.1 Statistical - Analytical Prediction Method for Overall Wing Weight

2.2.1.1 Structural Analysis

This method is taken from R.S. St. John and is a very simple example of this type of technique. During the structural analysis stage of its derivation the wing is idealised as two planks as shown in fig 2.2.

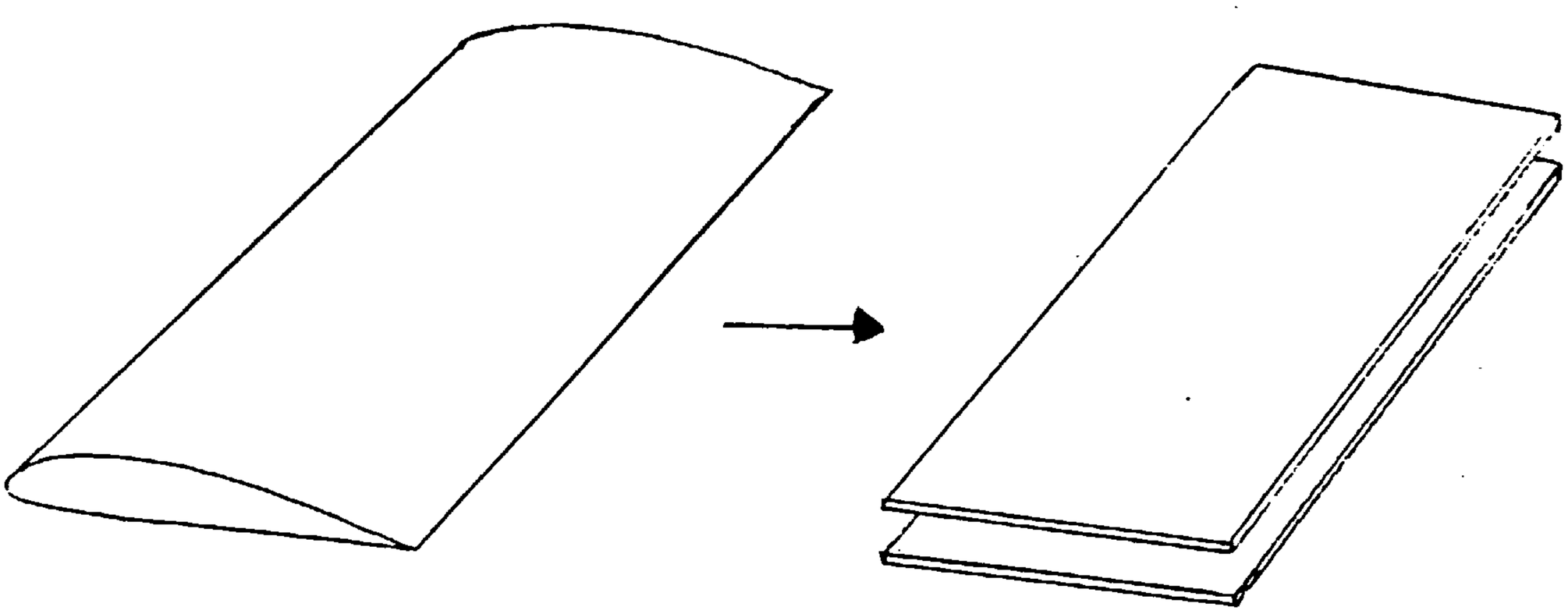


Fig 2.2 Structural Idealisation of a Wing

The engineering logic in this case is the engineers bending equation:-

$$\sigma = \frac{My}{I}$$

Which as a function of design parameters most easily available at the early stages of design is :-

$$\sigma = f\left[\frac{NW_{DES}\ (1+\lambda)^2\ AR\ \sqrt{AR\ S_W}}{\cos \Lambda_{50c}\ (t/C)_R\ S_W t_W}\right]$$

By sizing t for a particular material failure stress σ the volume of the idealised structure is calculated and hence its weight found and because; $W_W = \rho t_w s_w$ and σ = constant we get the function

$$W_W = f\left[\frac{K_{PL}\ NW_{DES}\ (1+\lambda)^2\ AR\ \sqrt{AR\ S_W}}{\cos \Lambda_{.50c}\ (t/C)_R}\right]$$

Where K_{PL} = 2.25 in this case, is an empirical constant added to take account of delta wing designs.

2.2.1.2 Statistical Analysis

The statistical analysis then begins. Parameters for existing designs are inserted into this equation and the results are compared with actual weights. Typically the comparison is done on a log log scale and a least squares curve fit for a straight line is performed.

2.2.1.3 Results

In this case the analysis leads to the following equations.

a) For Bombers and Transports:-

$$W_W = 17.792\left[\frac{K_{PL}\ NW_{DES}\ x\ 10^{-6}}{\cos \Lambda_{.50c}\ (t/C)_R}\right]^{.7388}(1+\lambda)^{1.4776}\ (S_W)^{.3694}(A_R)^{1.1082}$$

b) For U.S.N. Fighters and Attack Aircraft:-

$$W_W = 30.236 \left[\frac{K_{PL}\ NW_{DES}\ x\ 10^{-6}}{\cos \Lambda_{.50C}\ (t/C)_R} \right]^{.6840} (1+\lambda)^{1.3680} (S_W)^{.3420} (A_R)^{1.026}$$

c) For USAF Fighters

$$W_W = 19.405 \left[\frac{K_{PL}\ NW_{DES}\ x\ 10^{-6}}{\cos \Lambda_{.50C}\ (t/C)_R} \right]^{.7031} (1+\lambda)^{1.4062} (S_W)^{.3516} (A_R)^{1.0547}$$

(Units are Imperial).

This technique claims an accuracy of 10% based on the aircraft that were used in the statistical part of the analysis. i.e. that was the best accuracy obtainable during the curve fit. There is no guarantee that any new design would fall within this range, but if the method of construction, design and range of parameters are compatible with the assumptions and database used this accuracy is likely to be achieved.

2.2.1.4 Adjustments

This is a very simple example indeed. Many similar but more complex analyses exist. It is possible to refine the equations by making the structural idealisation more complex, say idealize it as a box with stiffners - but then the equations would be limited to that type of construction. The statistical analysis could be based on more refined groups of aircraft leading to a more accurate but more restricted equation.

2.2.2 Empirical - Analytical Method for Strucural Weight

We turn now to one of the more sophisticated methods of this class which is taken from Lewis and St. John. As with the previous example, the following text pays more attention to the assumptions and their implications then the actual derivation which is presented fully by Lewis and St. John.

As this is a more complicated analysis, it is worth collecting the assumptions together at the beginning in order to make the discussion on how the analysis can be adapted more easily followed.

The two stages of structural and empirical analysis are carried out for major components of a wing box, in this case the skins, and the shear webs. The skeleton of this approach is as follows:-

1) Analyse compression structure.

2) Match tension structure to compression analysis.

3) Fatigue analysis of tension structure.

4) Spar and rib analysis.

A further degree of specialisation splits the analysis into one for multicell boxes and one for stringer reinforced single cell boxes.

The empirical side of this analysis is covered by experiments conducted on structural panels.

2.2.2.1 Assumptions and Notation

A_{comp} = Compressive skin area

A_{TEN} = Tension skin area

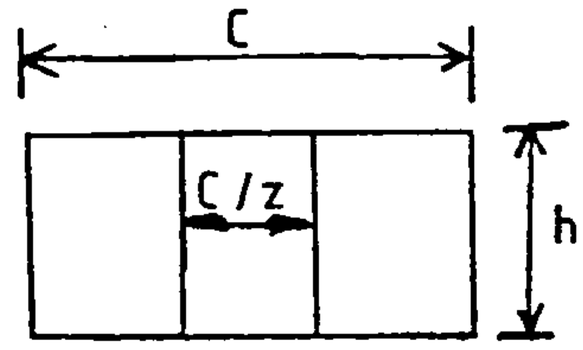
A_{SHR} = Shear web area

a = structural efficiency

A_{comp} = compression section area

b = web spacing $= \frac{C}{Z}$

β = stress concentration



C = box chord

E = Youngs modulus

E_t = tangent modulus

f_{AL} = allowable compressive strength

f_{AP} = applied stress

f_{APS} = applied shear stress

f_{CR} = theoretical buckling stress

f_{CY} = material yield strength

f_{ID} = ideal stress (ignoring buckling)

f_{INDEX} = fatigue index. fatigue life of any material normalised to that of aluminium 7075-t6

f_{MAX} = maximum allowed limit manouver load

f_{SU} = ultimate shear stress

f_{tu} = ultimate tensile stress

H = struct box depth

h_{RED} = effective box depth for tension side

h_w = web depth

$$K = \frac{5\pi^2}{12(1-\nu)^2}$$

K_{sb} = shear buckling factor

L = rib spacing

$$L' = \frac{L}{Y^{1/2}} = \text{effective column length}$$

M = applied B.M. - ultimate

$\eta = E_T/E$ = plasticity reduction factor

N_z = ultimate manouver load factor

N_{zlim} = maximum limit manouver load factor

q = chordwise load = $\frac{M}{hC}$

V = applied shear load

R = Cyclic stress ratio

r = plasticity factor

γ = end fixity coeff

t_s = skin thickness

t_{s1} = tension skin thickness

t_u = compressive section effective thickness

t_w = web thickness

X = $\frac{M}{CH^2}$ = load index or = $\frac{V}{H_w^2}$ for shear

X_1 = load index at yield stress

ie X where $f_{ID} = F_{CY}$ (see fig 2.4 and fig 2.5)

X_2 = X where F_{AL} = constant (see fig 2.4 and fig 2.5)

$\bar{Y}$ = centroid of tension section relative to N.A.

Z = no of cells

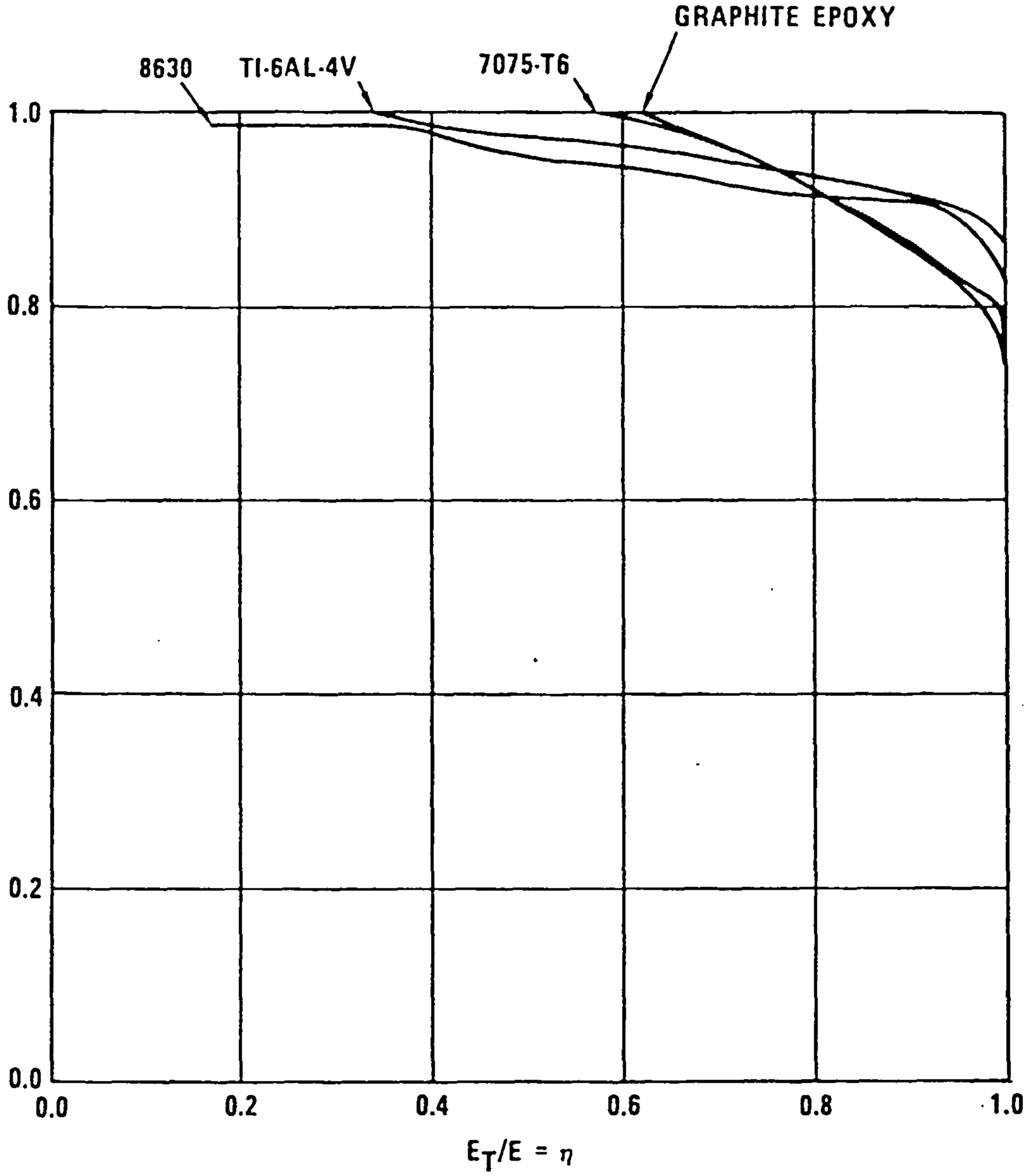


Figure 2-3. Normalized Compressive Tangent Modulus Curves

(All units are imperial (lbs, inches))

1) The web to skin thickness ratio, $\frac{t_w}{t_s} = 0.63$

2) The rib spacing to skin thickness ratio, $\frac{b}{t_s} = 25$

3) The whole analysis was based on a study of NACA conducted tests on aluminium compression panels. A study of results for several metalic and some FRP materials suggested that there might be a common relationship which can be read across to other materials.

$$\frac{f_{AP}}{f_{cy}} = fn\left(\frac{E_T}{E}\right)$$

This is shown graphically in fig 2.3. After a certain amount of manipulation via figs 2.4 and fig 2.5 the working graphs shown in fig 2.6 and fig 2.7 are obtained. (Reproduced from Lewis).

Note one set of figures apply to the multicell box and the other applies to the sheet stringer box.

4) An optimum box is assumed to have h=b.

5) An ultimate factor of 1.0 gives

$$f_{CR} = f_{AP} = f_{ID}$$

6) $\upsilon = 0.3$ for all materials

7) The effective depth of the boxes are assumed to be twice the distance of the neutral axis from the centroid

$$h_{red} = 2\bar{Y}$$

<u>The following assumptions only apply to stringer-skin analysis</u>.

8) Structural efficiency for a typical Z - stringer is used

$$a = 0.88$$

9) Elastic failure i.e. plasticity factor,

$$r = 1.0$$

10) Panel is simply supported $\gamma = 1.0$

<u>The following assumptions apply to the fatigue analysis</u>.

11) Miner and Palmgren rules apply

12) Average stress concentration factor in fighters $\beta = 3.0$

13) In fighter design manouver loads are the most important

14) For a highly manouverable fighter the cyclic stress ratio $R = 0.23$

15) All damage occurs within 65% of the average limit load i.e;

$$N_{ZLIM} = 0.65 \frac{N_z}{1.5} \Rightarrow \frac{N_{ZLIM}}{N_z} = 0.43$$

16) The load factor W is proportional to stress levels hence

$$\frac{f_{MAX}}{f_{tu}} = 0.43 \text{ also}$$

17) The fatigue index concept is a valid one. i.e. all materials fatigue in the same way

18) In the shear web analysis it is assumed that the plate behaviour is characterised by test results on aluminium

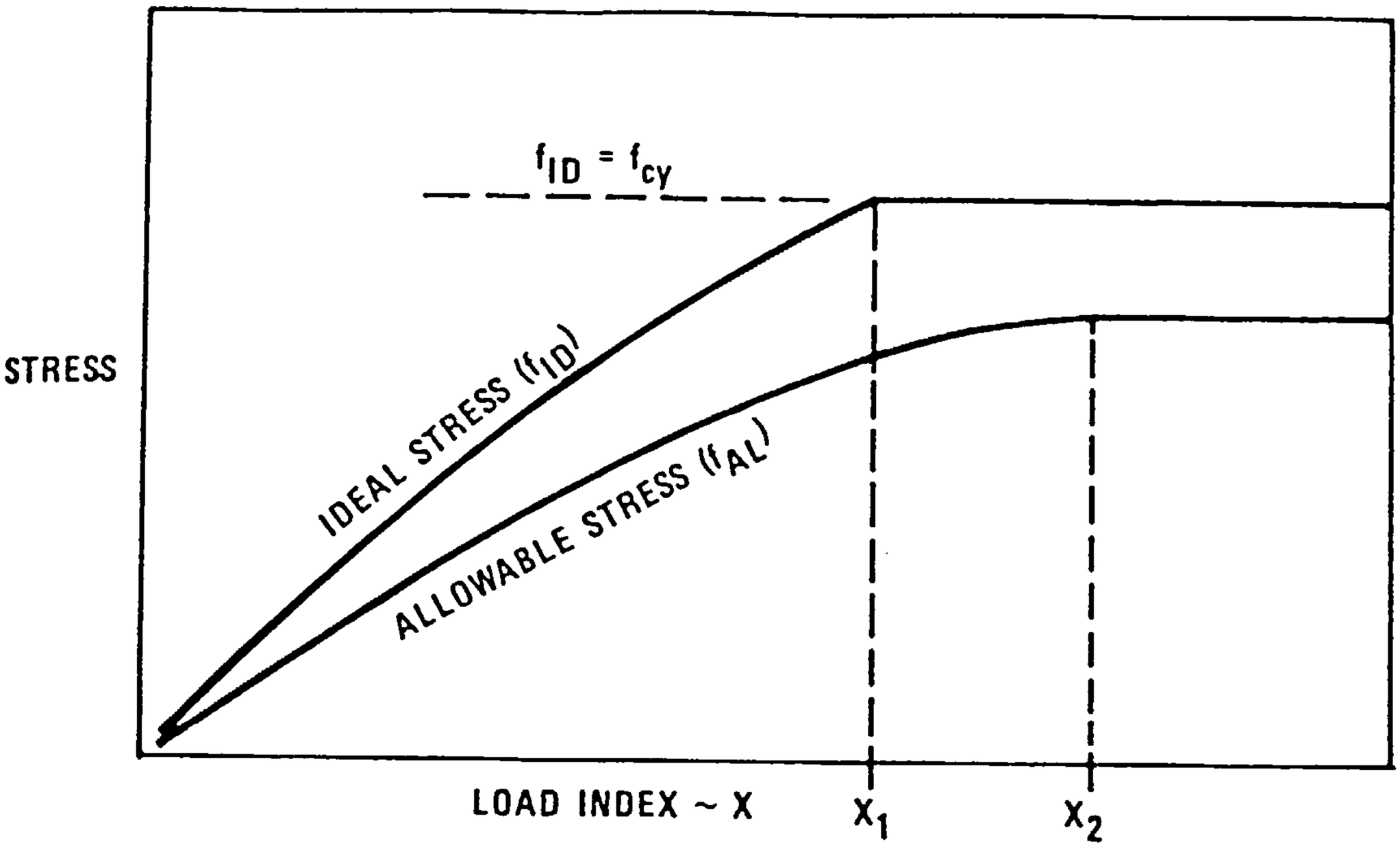


Figure 2.4. Normalized Allowable Stress Curve

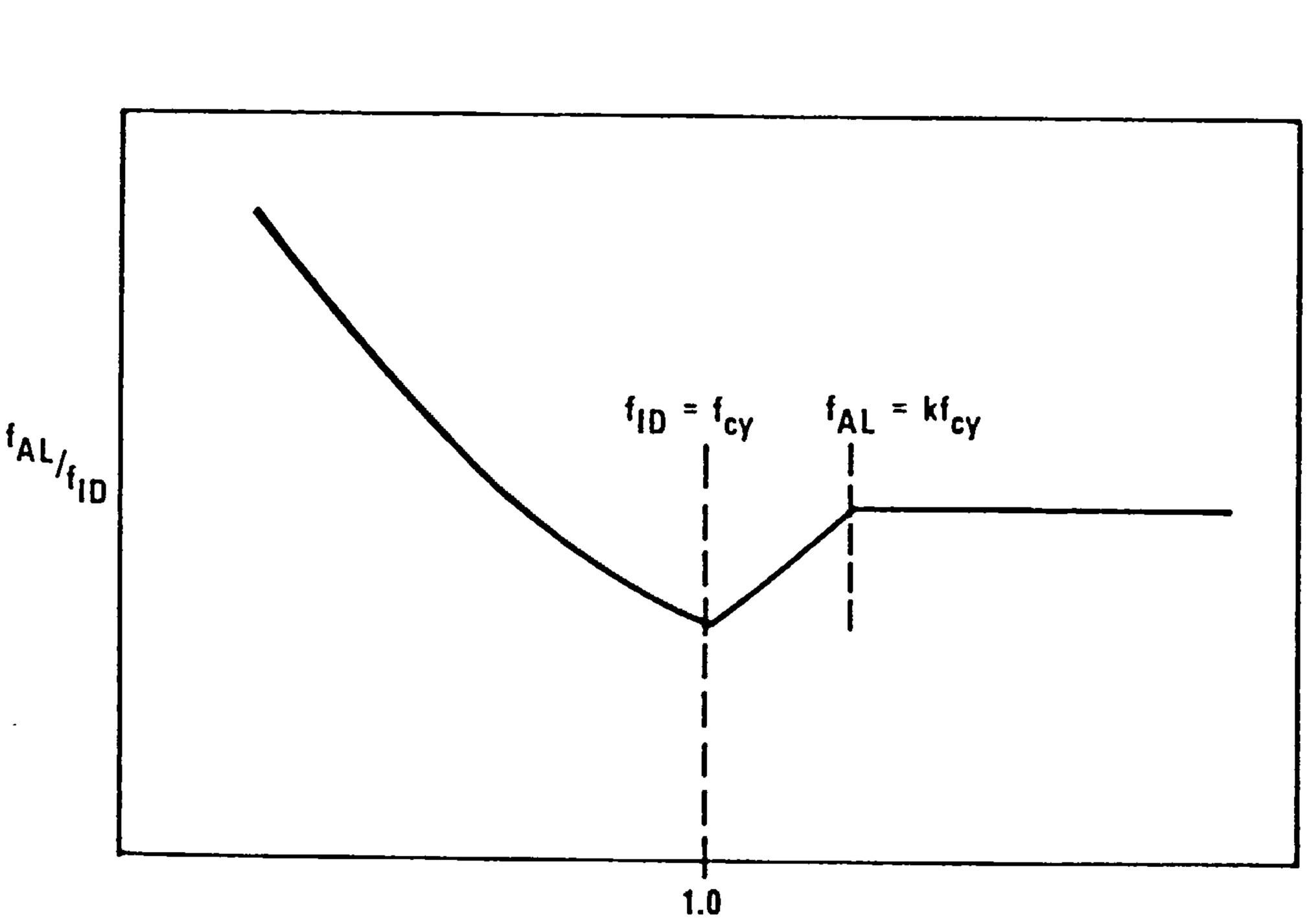


Figure 2.5. Normalized Stress Relation Curve

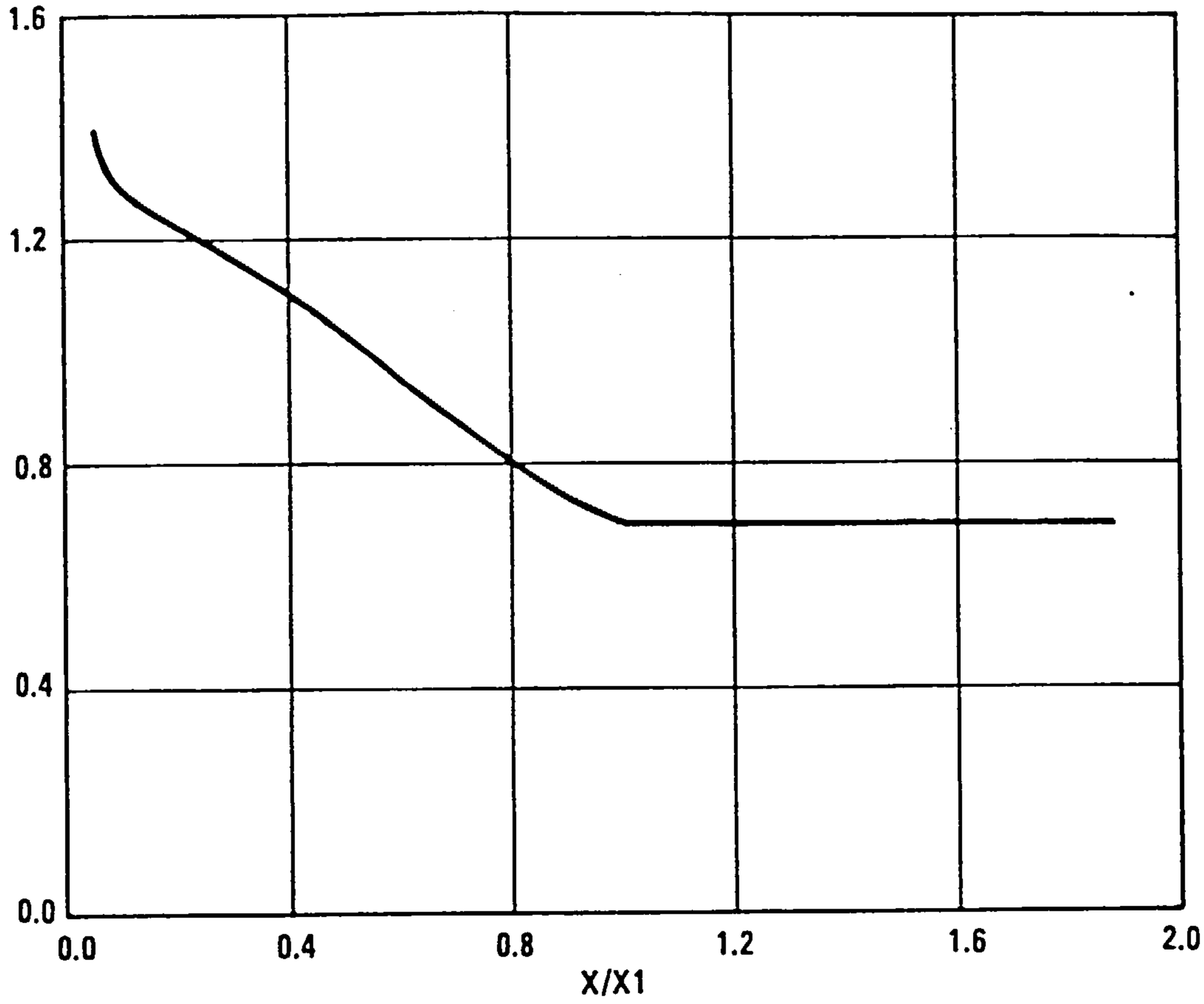


Figure 2.6. Normalized Stress Relation for Wing Multi-cell Structure

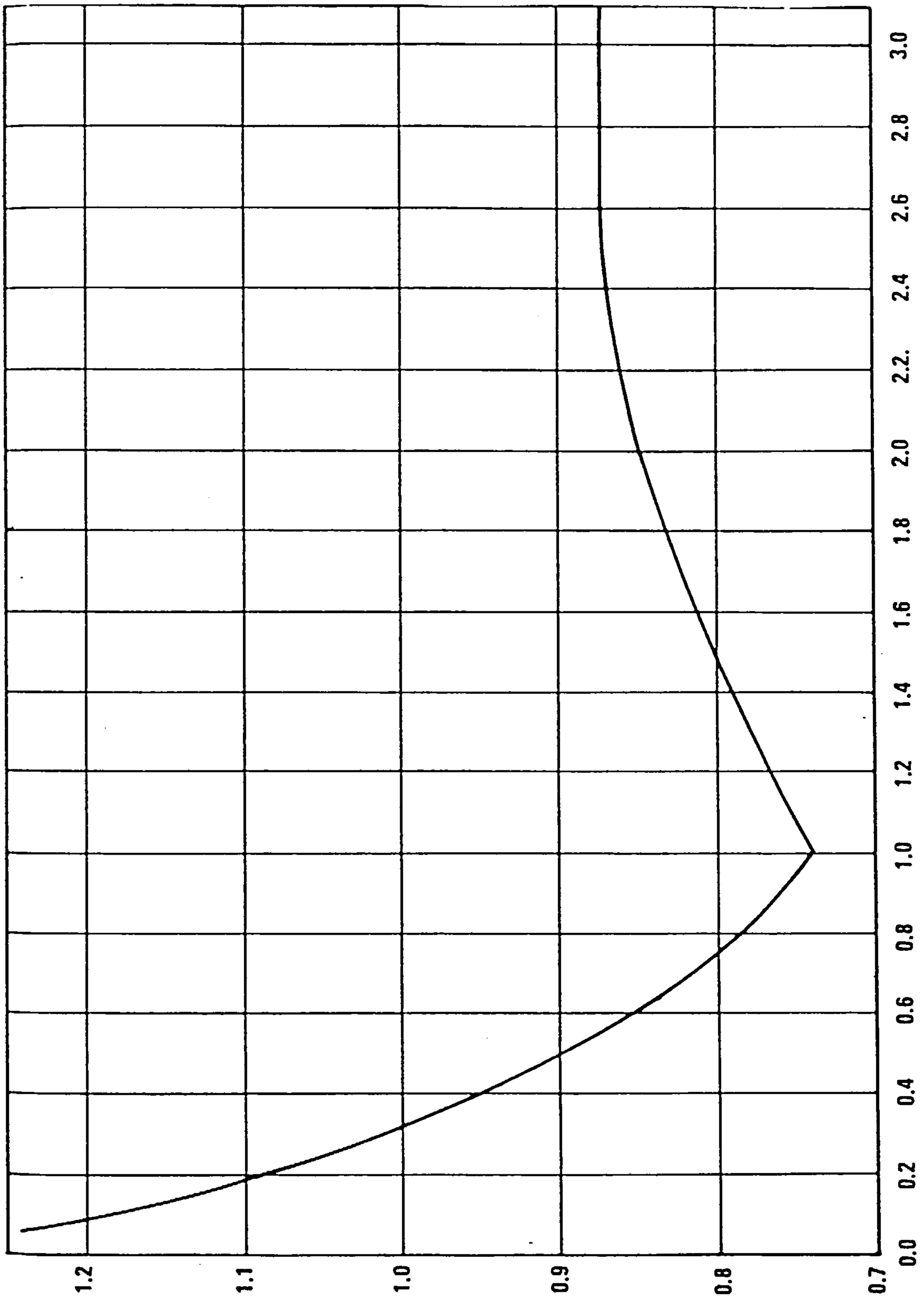


Figure 2.7 . Normalized Compressive Stress Relation for Sheet/Stringer Panels

2.2.2.2 Derivation

The object of the exercise is to find f_{AL} on which sizing is based.

a) Multi cell box compression structure

1) The theoretical equation used in this case is the elastic buckling stress equation. Assuming that for an optimum box h = b we get;

$$f_{CR} = KE \left(\frac{t}{h}\right)^2$$

or in terms of t

$$t = \left(\frac{f_{CR}\, h^2}{KE}\right)^{1/2} \qquad \text{eqn a}$$

2) The engineers theory of bending for applied stress is

$$f_{AP} = \frac{M}{hCt}$$

or in terms of t

$$t = \frac{M}{f_{AP} hC} \qquad \text{eqn b}$$

3) Matching equations (a) and (b) leads to

$$\left(\frac{f_{CR}\, h^2}{KE}\right)^{1/2} = \frac{M}{f_{AP} hC} \qquad \text{eqn c}$$

4) But assuming an ultimate factor of 1.0 we know $f_{CR} = f_{AP} = f_{ID}$ so simplifying eqn (c) leads to

$$f_{ID} = \left(\frac{M}{Ch^2}\right)^{2/3} (E)^{1/3} (K)^{1/3}$$

5) Solving for K assuming $\nu = 0.3$ leads to

$$f_{ID} = 1.653 \left(\frac{M}{Ch^2}\right)^{2/3} E^{1/3}$$

6) And as the load index defined in the notation as

$$X \frac{M}{Ch^2} \qquad \text{——— eqn d}$$

$$\Rightarrow \quad f_{ID} = 1.653\, X^{2/3}\, E^{1/3} \leq fcy \text{ —— eqn e}$$

7) By definition f_{cy} occurs when the load index reaches X_1 so finding X_1 is easy using eqn (e)

$$X_1 = 0.471 \left(\frac{f_{cy}^{\ 3}}{E}\right)^{1/2}$$

8) So all the ingredients of the analysis have been gathered;

X, X_1 F_{ID} are known and finding F_{AL} is a simple matter of looking up the graph in fig 2.6.

9) The calculation of the compressive material volume is then completed by substitution into this equation;

$$A_{comp} = \frac{M}{h\, f_{AL}} \geq \text{min gauge}$$

b) <u>Stringer skin compressive analysis</u>

The objective is the same, we have to find F_{AL} so that sizing can be carried out.

1) In this case the starting point is an equation developed by Gerard, for combined flexural instability and plate buckling.

$$f_{ID} = a(q\ r\ E/L')^{1/2}$$

2) Assuming a plasticity factor of 1 (elastic failure), and pinned ends on the panels (γ=1) and a structural efficiency of 0.88 we get

$$f_{ID} = 0.88\ X^{1/2}\ \ E^{1/2} \leq f_{cy} \qquad \text{—— eqn f}$$

Again the load index at yield X_1 can be found using equation f giving

$$X_1 = \left(\frac{f_{cy}}{0.88E^{\ 1/2}}\right)^2 \qquad \text{——— eqn g}$$

The ingredients have been collected again by finding the F_{AL} in fig 2.7 that corresponds to the values of F_{ID}, X_1 and X the compressive material can be sized as before.

c) <u>Multicell Tension analysis</u>

1) The tension material is sized by use of the following equations

$$Y = \frac{bt_{s1}\ (h/2) + t_w\ (h^2/8)}{bt_{s1} + t_w\ (h/2)} \qquad \text{——— eqn h}$$

$$t_{s1} = \frac{M}{f_{tu}\ hC} \qquad \text{——— eqn i}$$

$$A_{TEN} = \frac{M}{f_{tu}\ h_{red}} \qquad \text{——— eqn j}$$

where $h_{red} = 2Y$

2) The compression analysis equations are then applied to the tension side to check for instabilities under reverse load conditions.

d) <u>Skin - Stringer tension analysis</u>

$$A_{TEN} = \frac{M}{hf_{tu}}$$

e) Fatigue and damage analysis

Here on to the end both box types follow the same analysis using the assumptions listed earlier,

$$f_{max} = 0.43\ f_{tu}$$

$$\beta = 3.0$$

and $R = 0.23$

Employing these values the damage proportion maybe calculated using the S.N curve for 7075-T6 aluminium alloy. The damage proportion value is these multiplied by the "fatigue index" for the material being used.

The fatique index referred to here is dealt with in more detail by Lewis et al but is basically a method of scaling lives of various materials to that of the base material 7075-T6. The major assumption here is that all materials fatigue in the same way. The list of fatigue indices is reproduced in fig 2.8, note there are none for F.R.P. which fatigues in a different way.

If a damage proportion found is or exceeds unity then the tension material must be resized based on this fatigue criteria.

f) Shear Web analysis

1) In this case the ideal stress was taken as,

$$f_{IDS} = \frac{\Pi^2\ K_{sb}\ E}{12(1-\nu^2)} \left(\frac{t^2_w}{h_w} \right)$$

which with compatible geometry and materials leads to

$$f_{IDS} = 5.55E\ \frac{t^2}{h_w^2} \qquad \text{——— eqn k}$$

FIGURE 2.8 Fatigue Indices

MATERIAL	DERIVED LIFE (n-CYCLES)	FATIGUE INDEX (' INDEX)
7075-T6 Aluminum	11,000	1.0
2024-T6	39,000	.282
6AL-4V Titanium (160KSI-HT)	55,000	.200
4130 Steel (125KSI)	350,000	.032
4130 Steel (260KST-HT)	13,000	.846

2) Whilst the applied stress is given by

$$f_{APS} = \frac{V}{th_w} \qquad \text{——— eqn 1}$$

3) making the usual assumption that

$f_{IDS} = f_{APS}$ and combining eqn k and 1 by eliminating t we end up with

$$f_{IDS} = 1.776E^{1/3}\ X^{2/3} \leq f_{su} \qquad \text{—— eqn m}$$

where $X = \frac{V}{h_w^2}$

4) From this we obtain X_1 by going to the limit f_{su}

$$X_1 = \left(\frac{f_{su}}{1.766E^{1/3}}\right)^{3/2} \qquad \text{——— eqn n}$$

5) Again we have collected all the ingredients, X, f_{IDS} and X_1 and may read f_{AL} of fig 2.9.

6) The web material is sized using

$$A_{SHR} = \frac{V}{f_{AL}} \geq \text{min gauge} \qquad \text{——— eqn o}$$

2.2.2.3 Adjustment of this Method

Lewis continues in this vein for the fuselage and then gives some examples of how this method can be used in a design study. This method is fairly easy to use even with a calculator. It is very useful for doing a quick check on finding the benefit of using certain layouts or materials.

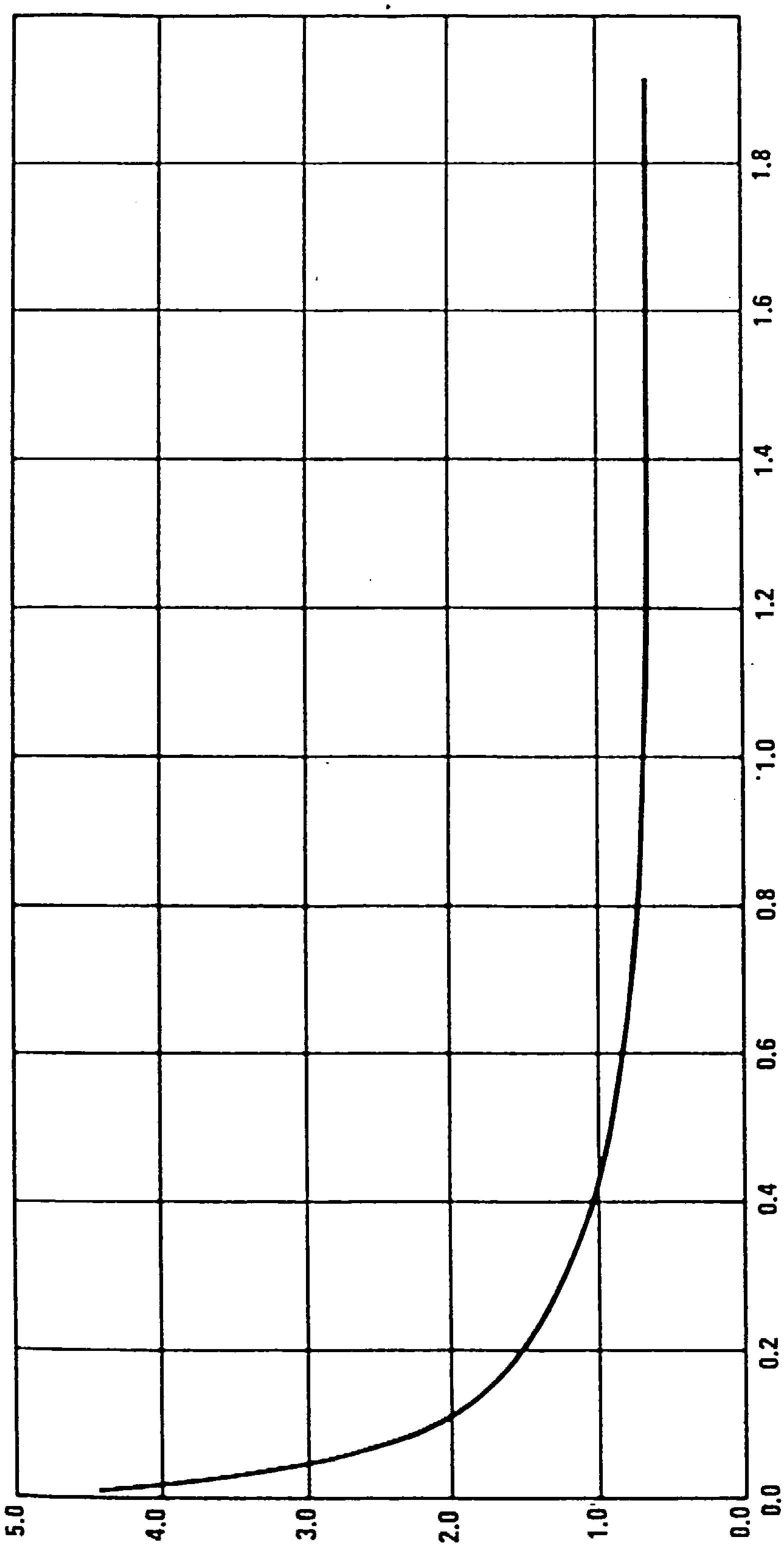


Figure 2.9 . Normalized Shear Stress Relation for Spar and Rib Webs

This method is intended for producing comparative values and care and judgement must be exercised whilst using the formulae. Using the derived formulae without understanding their derivation is a path to disaster.

It is limited to multi cell and stringer skin boxes of roughly the same geometry as specified in the first two assumptions where $\frac{t_w}{t_s} = 0.63$ and $\frac{b}{t_s} = 25$.

Should the limitations be inappropriate new tests have to be conducted and the method be readjusted. In this case perhaps an f.e. analysis might suffice, rather than expensive time consuming experiments.

The whole method leans heavily on experience gained with aluminium. Before there can be any confidence on results obtained for F.R.P. composites a composite based relation for f_{AL}/f_{ID} should be found but this

is likely to change for different layups and make a nonsense of the whole thing. In any case the fatigue analysis based on Miners rule and the tension loading would have to be scrapped.

Metalic fatigue theory has no place in F.R.P. as the failure mechanism is completely different and vastly more complicated. The assumption that the fatigue analysis should be based on the tension side is metal based. F.R.P.s are more susceptible to failure in compression as the fatigue failure is matrix dominated, the fibres being, typically, very fatigue resistant.

Perhaps the least one could do for FRP materials here is readjust the Piosons ratio.

Note also that there has been no consideration for certain engineering details such as, difficulties with joints, ribs have been ignored, and the interaction of skin panels with shear webs (the analysis was based on panels in compression.

2.2.3 Discussion of Analytical - Empirical Method

This may seem like a denouncement of these types of methods but that is not so. The aim here is to point out that it is important to apply them with care and judgement.

Even the trusty and ever useful engineers bending equation can lead the unwary into trouble. For example, if one forgets its origins and applies it to a slender beam made from a material like undirectional C.F.R.P. (which has a high undirectional modulus but, low shear modulus) to obtain "flexural" deflections, the results are likely to be smaller than the actual deflections. This is, because in this case, the shear deformation of the beam would no longer be negligble.

So these methods are very simple and quick and have a typical accurate range of about 10% if used with engineering judgement.

A worked example of this method is given in section 13.4, applied to the Cranfield A1 aircraft.

2.3 The Analytical approach

Analysis of structures purely for determining their weight is seldom done. The idea behind this approach is simple, if one analyses a structure and sizes it accurately then the weight of the structure can be determined from the amount of material used.

Actual weight estimation methods based on this principal are hard to come by in the literature. An example is given by Ritter for estimating rib weights. In this case the rib is designed from first principals and sized as accurately as possible. Only a simple geometry is assumed and the important static loads are considered and an accuracy of about 10% is claimed.

Some proposals for using f.e. techniques for this type of analysis have been put forward in the past including that by Nisbet and Hoy who also propose using engineers theory of bending as a base for simpler analyses.

2.3.1 Typical E.B.T. Based Method

The scheme proposed by Nisbet and Hoy using engineers bending theory as a basis is typical of its type.

a) Assume a value for bending and torsional stiffness.

b) Use an appropriate material failure criterion to size the structure to an ultimate load, remembering minimum gauge constraints.

c) Allow for fail safety by specifying a constraint on the ratio of cover material to overall bending material.

d) Use an envelope load case.

e) Compare new stiffness with the assumed one.

f) If they are not comparable work out the new weight and go through the resizing procedure.

g) If they are comparable then carry out a durability analysis and resize minimum gauges if necessary.

h) Estimate unmodelled structural weight to complete the picture.

Note strictly speaking this proposal was for a structural development/optimisation method rather then a weight estimation process hence the results might tend to be comparative.

2.3.2 Typical Proposed F.E. Based Method

The scheme proposed by Nisbet and Hoy for an F.E. based method of structural optimisation is summarized below. They envisaged that this process would be used during a fairly advanced stage of design. Again the intention is to produce weight values for comparision and optimisation purposes rather than for weight estimation in its own right.

a) Initial guess at element sizes.

b) Loading actions.

c) Fully stressing optimisation procedure applied.

d) Obtain fail safety by specifying ratio of cover to cap material.

e) Repeat the process till convergence.

f) Allow for unmodelled weight.

2.3.3 Discussion

These methods existing and proposed are sound and would work for any design or material though problems do exist. These problems are discussed in greater detail in a later chapter. The greatest difficulty lies in the phrase, "Allow for unmodelled weight".

2.4 Summary

The purely empirical method has the advantage of being simple to understand and being even simpler to apply. With this method the user need only plug in the basic parameters to obtain a solution. The draw back lies in the restriction on the range within which the parameters must lie in order to produce useful solutions. These methods are of little practical use to designers of novel designs whether the novelty is in the material or the configuration used.

To a certain extent this draw-back is eased by basing the form of the estimation equations on engineering logic which are empirically adjusted. Since the equations do have an engineering basis the range of the parameters would be greater and there is a greater scope for novelty. There is an extra restraint in this case however. That is, the basic assumptions (whether it be that E.B.T. is valid or the construction used is of a box type etc) must be valid.

Application of both pure empirical and semi empirical methods blindly to the general structure is asking for trouble.

The ultimate is a completely flexible theoretically based method. The more flexible the engineering logic the more flexible the method. There are problems with the amount and detail of data needed to make these methods work. Their formulation would be anything but simple and they would not be simple to use, requiring at least a fair amount of computing resources.

Before concluding this chapter it would be worth suggesting an answer to a question frequently posed about the purely analytical method. The question takes the form of, "If the actual design goes through a different optimisation process from the optimisation process used during the prediction would'nt an error occur?"

The short answer is, "Yes" but there is another factor. In practice optimisation can only go so far before the manufacturing difficulties become the limiting factor. Many optimisation methods run up against this limit before they reach an optimum, and so reach similar solutions.

The frequency at which this question about the optimisation process is asked when an analytical prediction method is being discussed is odd. It is hardly brought up during discussions of empirical methods but the same applies, it just seems to be taken for granted.

This reinforces the point made earlier about using these methods blindly.

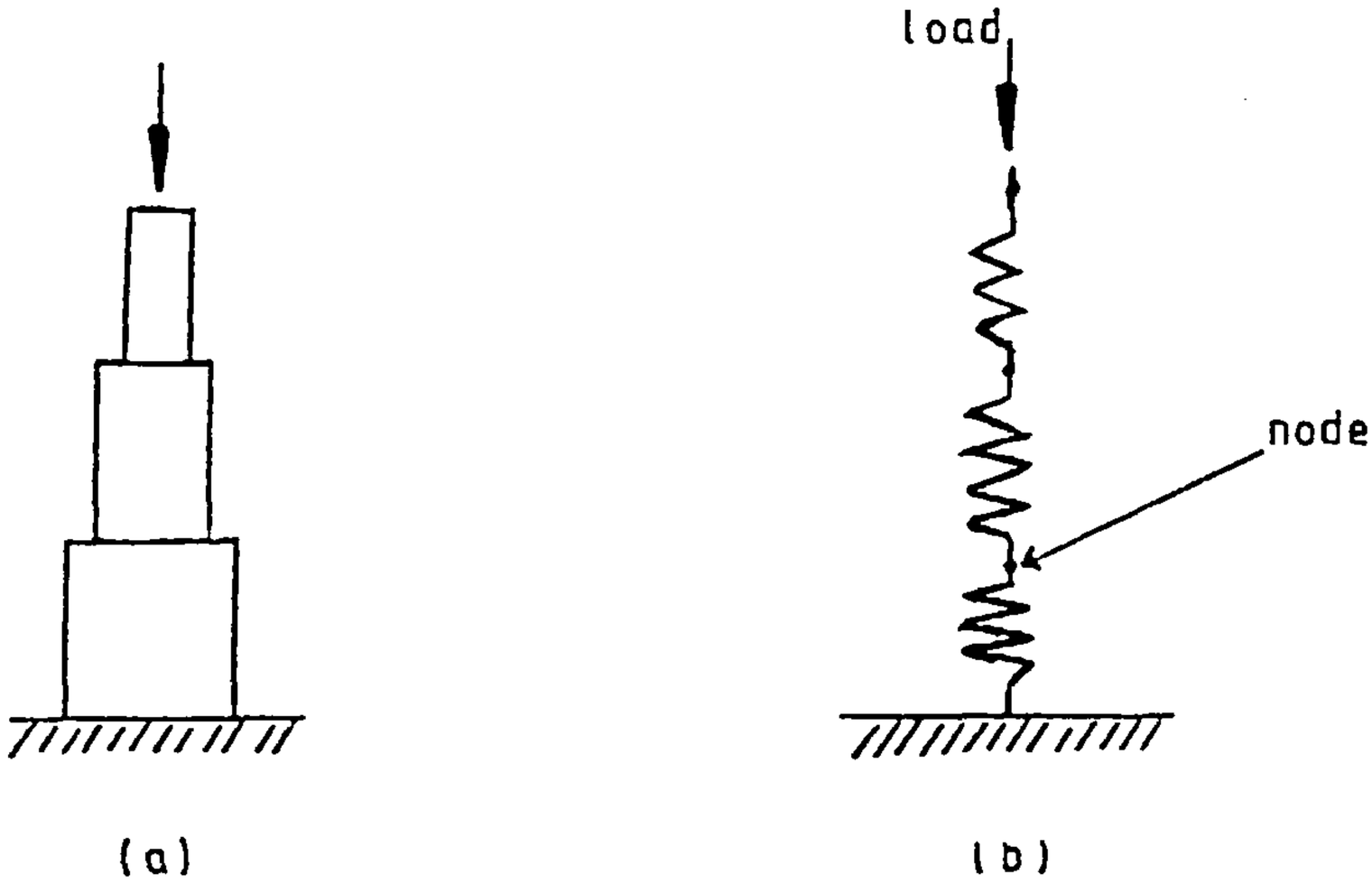
We will now move on to brief descriptions of the science and technology that forms the basis of the proposed method.

3. A Brief Description of the Finite Element Technique

The technique has been around for a long time but was neglected as a serious method for analysis of structures and field problems till the advent of the modern digital computer. Before then a proliferation of analytical and empirical methods sometimes based on incredibly crude assumptions provided the answers. In fact the finite element method (F.E.M) is very simple in concept and in its mechanisms. The reason it did not catch on earlier was in the amount of data processing and handle cranking. This made it slow and error prone when carried out manually but it was a task ideally suited for the indefatiguable electronic computer.

The published texts available on the F.E.M are extensive and numerous and the aim of the next few lines is to give the new comer a very brief description of the basic concept of the method.

The method is based on the assumption that the field or in our case structure can be broken down into small easy to understand discrete components. This is best explained using a simple example like the stepped stay rod under compression shown in fig 3.1 (a).



(a) (b)

Fig 3.1 Stepped Stay and F.E.M Idealisation

The stay could be idealised as several springs as shown in the fig 3.1 (b). The behaviour of each spring in the model is simpler than the whole stay and may be considered as the "finite element" of the analysis. The elements are connected to each other at nodes. Loads and constraints are applied to these nodes and by working out the displacements of the nodes, it is possible to find the stress fields in the elements.

This is very simplistic but that is the general idea, it is useful from the weight estimation point of view because the stiffness of the springs is usually calculated indirectly from material properties and spring geometry. Knowing the geometry and density of the springs their weights can be calculated .

4. A Brief Description of Structural Optimization

Here again there is a great proliferation of literature on the subject and so only the basic concept is considered here in a simple example. Some purists may consider the stress ratioing technique used in the WEIGHTS program not to be an optimization technique but a type of constraint matching procedure. Nevertheless, it is a genuine weight reduction technique which is very effective for strength critical designs.

Consider a bar under a tensile load as shown in fig. 4.1. The cross-sectional area of the bar is known as the design variable.

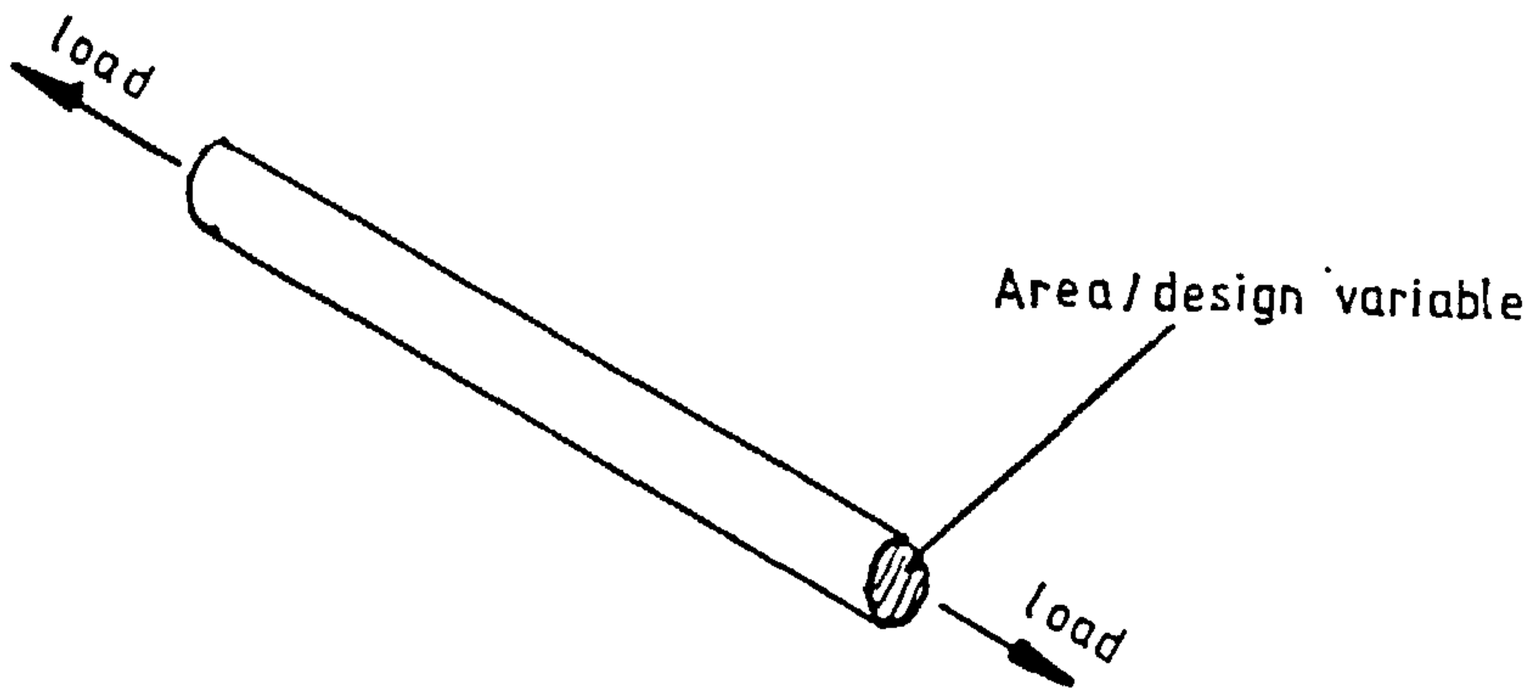


Fig 4.1. Bar Under Tensile Load

The following simple formula is applied to the cross-section area till a solution is reached.

$$A_{new} = \frac{A_{old} \times \text{Allowable stress}}{\text{Actual stress}}$$

The iterative process stops when a "design variable constraint" is met or when the optimum size is reached. A design variable constraint is a limit on the value of a design variable. In the case of the bar example there would usually be two constraints, a maximum area and a minimum area. The minimum area (minimum gauge) is determined by factors such as practical handability whilst the maximum value is often determined by available space.

Another part of the optimization terminology which will crop up later is "design variable linking". To describe this consider a membrane made from two discrete plates as shown in fig 4.2.

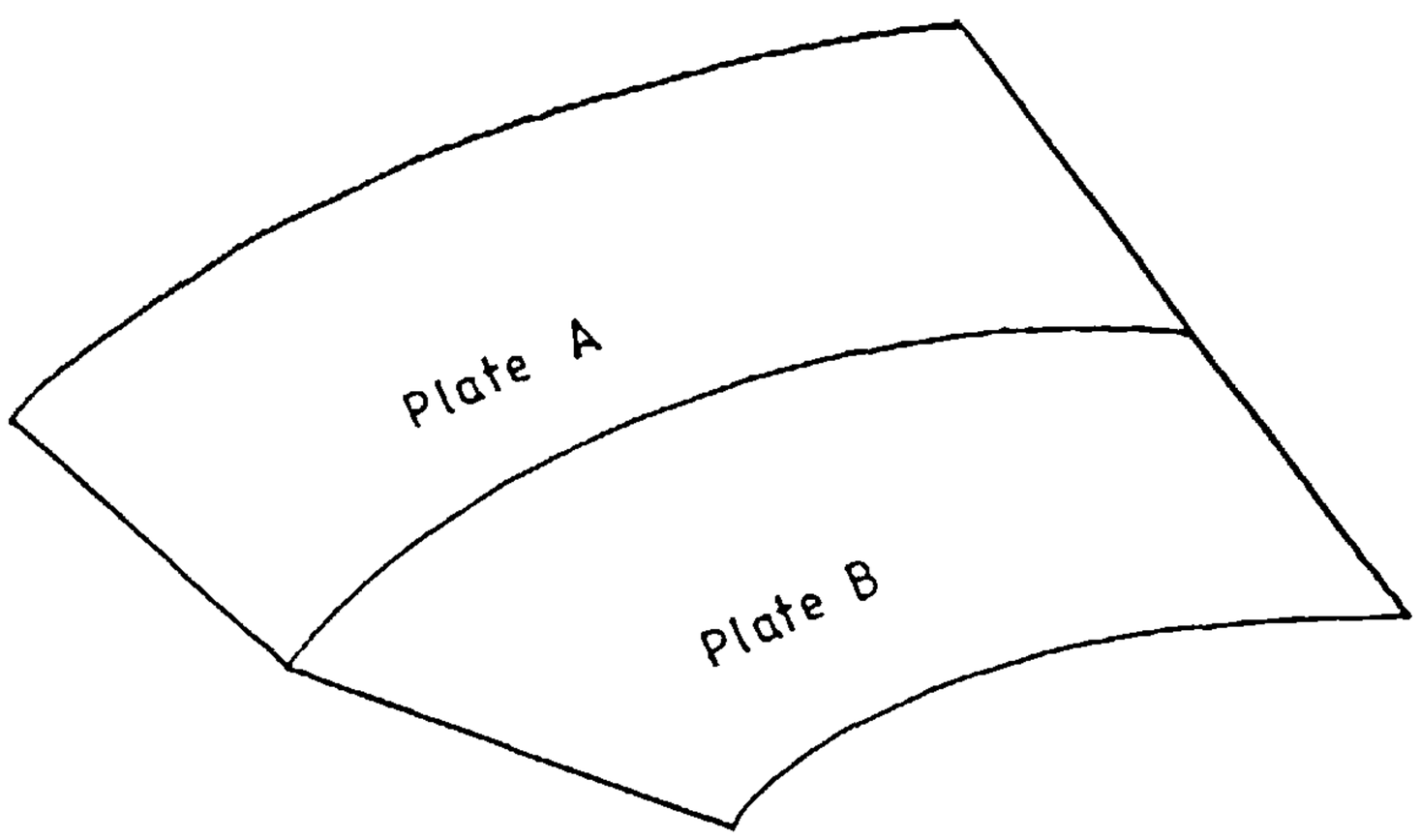


Fig 4.2 Membrane made from two plates

The finite element model may be something like the model shown in fig 4.3 with 8 elements. Clearly if the analysis is given the freedom to do so, it might be possible to end up with 8 plates of different thickness. So the design variables (thickness) of elements 1,2,3 and 4 are constrained to have the same value as in the real case, similarly for elements, 5,6,7 and 8.

Two groups of four elements

each group containing elements

with similar thicknesses

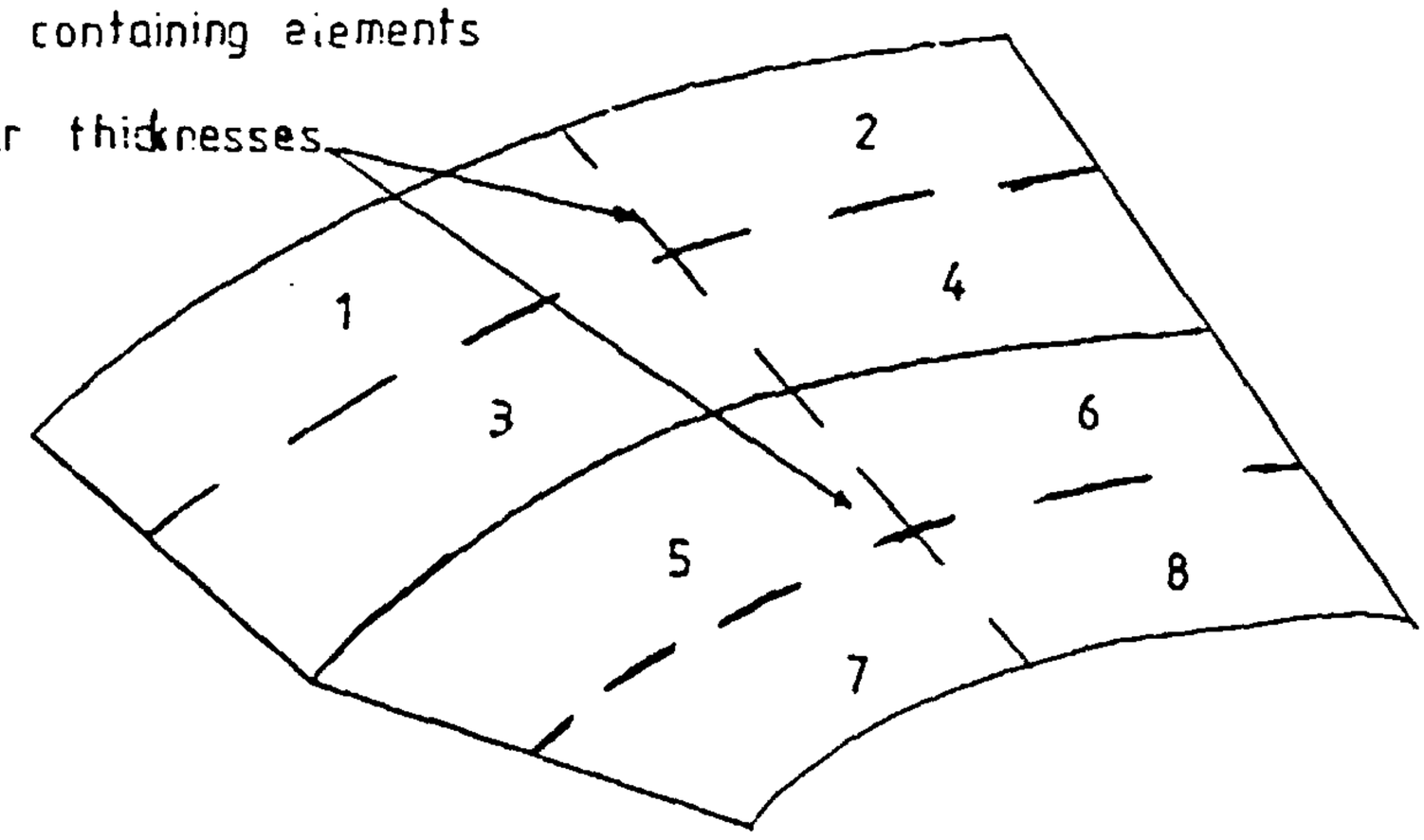


Fig 4.3 Finite Element Model of Membrane with 8 Elements

5. Stanton Jones Airload Distribution Estimation

Stanton Jones devised this method in the late forties and because swept back wings were a novelty then, the lack of empirical data led him to include as his last remark in his report,

"..... until more direct experimental evidence is available this cannot be relied upon unconditionally."

Presumably this evidence has been produced since then as his method is widely used in its original form, being as simple as the Schrenk method which is even more limited in scope.

The method is limited to straight, tapered, swept back wings at small angles of incidence in unyawed sub-sonic flight.

Other limits on parameters are;

Aspect ratio range	1.5 to 8
Sweep back range	0 to 70 degrees
Taper ratio	0 to 1.5

The equation was obtained from a parametric study of what appears to be wind tunnel data for forty different wings covering a range of aerofoil shapes. The point must be made that the method is only as good as this data and it is flexible in that it can be adjusted to new wing types given new data.

The method states that the undimensioned loading coefficient at the spanwise position n;

$$\left(\frac{cC_L}{\bar{c}\,\bar{C}_L}\right) = 1.28\,(1-n^2)^{1/2} + \begin{cases} (-6.35 + 14.13n) & n \le 0.7 \\ 4.25 - 53.8(n-0.815^2) & n \ge 0.7 \end{cases} \left[Y - 0.425\right]$$

$$Y = 0.42 + A\left[\,(4.4 + 5T)\tan S + (10.4T^{1/2} - 6.7)(1-M^2)^{1/2}\,\right]$$

$$\frac{}{10^3}$$

Where:

A = aspect ratio for wing at Mach number M

$\bar{c}$ = geometric mean chord

$\bar{C}_L$ = mean lift coefficient for whole wing

S = sweepback of the 1/4 chord line

T = taper ratio.

Y = spanwise position of centre of pressure

There are other methods available for more general planforms but they are more complex and still have their limitations. Should a further refinement of loads derivations be necessary such methods as strip theory, vortex panel methods, mach box methods and piston theory methods would be possible candidates depending on the type of planform and speed regime.

6. A Brief Description of Aeroelasticity

A very sketchy account of the concept of aeroelasticity is given here for completeness. The main concern is to do with the dynamic response of structures subject to aerodynamic loading which can be unstable. The two main categories of instability considered here are flutter (an oscillatory instability) and divergence (a monotonic instability).

Divergence occurs as the structure responds to aerodynamic loading by increasing the incidence of the wing causing a potentially unstable cycle as shown in fig 6.1.

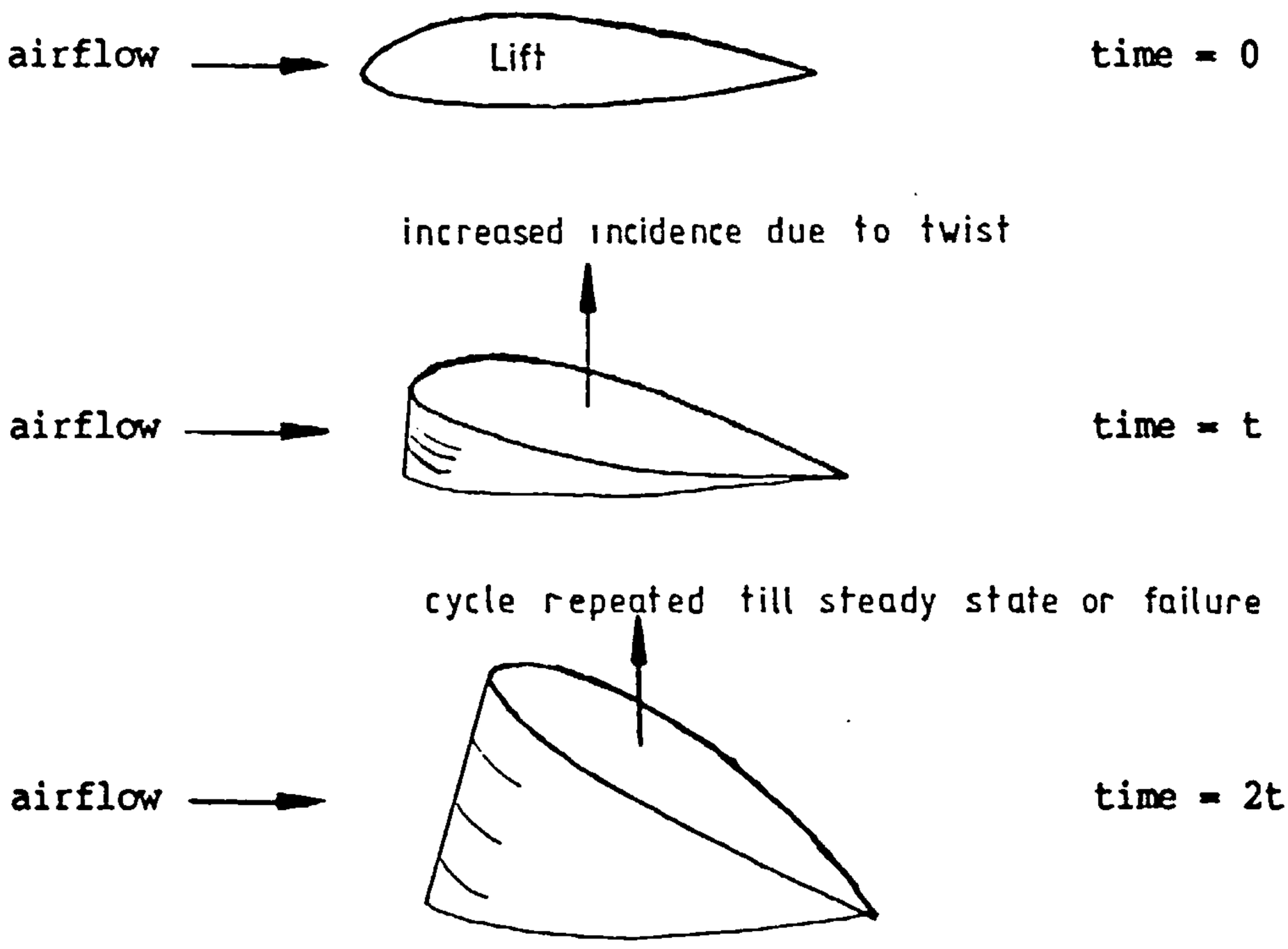


Fig 6.1 Demonstration of Divergence

In the case of flutter it is a matter of damping (both structural and aerodynamic) and the difference in phase of the structural response to aerodynamic loading. If a tuning fork is struck it vibrates at its natural frequency until the damping of its material and the damping due to the resistance of the air stops it. The energy stored in the tuning fork dissipates as heat and sound. If the fork is shaken at its natural frequency very little energy would be necessary to overcome the damping meaning that more energy is stored than dissipated - an unstable situation leading to failure.

This is what happens on a wing or control surface, the vibration energy is fed into the wing by the airstream and the damping is provided by the air. Here torsional damping is often low and is often adversly coupled with bending.

Divergence can be catered for by increasing stiffness or by configuring the structure so that incidence does not increase with aerodynamic loading. (e.g. sweep back reduces this tendancy).

Flutter is catered for by designing a structure with a high natural frequency and by decoupling the torsion and bending modes. The latter can often be achieved by mass balancing.

Nastran (an F.E. package) is capable of carrying out aeroelastic analyses (flutter and response analyses). It has several modules for calculating aerodynamic loads using a vortex panel method, mach box method, strip theory, piston theory and lifting body theory. It can calculate flutter solutions using most of the classical methods. This type of package would be connected to WEIGHTS if aeroelasticity is of overiding concern in a project and if sufficient funds are available.

7. A Brief Description of Structural Instability

This is another specialised field in which there is a copious amount of literature and this chapter is here as an introduction of the concept in a few sentences.

In some structures, such as pure monocoque structures, buckling leads to failure. In high speed structures buckling in external components leads to adverse aerodynamic effects whilst other structures (such as those in light aircraft) spend most of their operational life at least partially buckled.

Analysis of buckling can be split into two main categories; the calculation of when the buckling occurs; and the analysis of what happens after buckling occurs.

The calculation of when buckling occurs (critical load calculations) is usually based around some simple assumptions. The structure is assumed to take up a buckled shape as shown in fig 7.1.

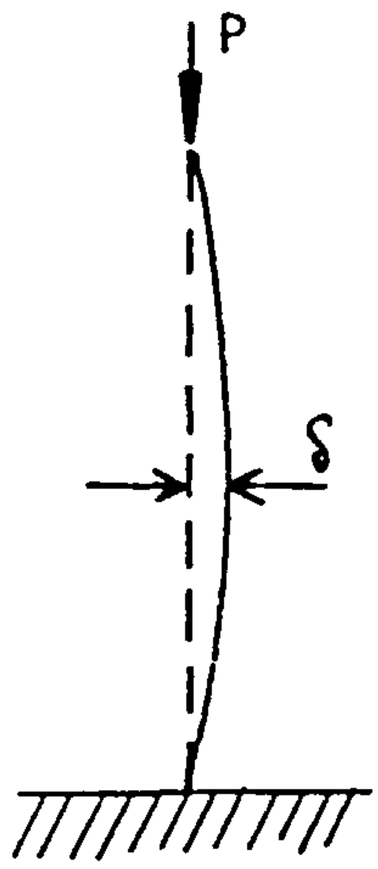


Fig 7.1 Critical load calculation of a strut.

The bending of the structure causes an internal moment which opposes the moment caused by the load and the bow (P x δ). When the point is reached where this restoring moment is overcome by the applied load, collapse occurs. This is called the critical load.

Classical calculation of what happens after this point has been reached starts to get complicated because the usual assumptions about small displacements and linearity have to be abandoned. Usually iterative techniques are employed to trace a behaviour path with increasing load or displacement.

In our case we would tend to use classical buckling theory to help size the parts of our structure where we want to avoid buckling. Then we would use the ability of finite element analysis to analyse non-linearities in materials and geometries to cope with post-buckled designs. In practice this requires a large amount of computer power.

8. A Brief Description of Structural Fatigue

We continue with our brief description of specialised topics. Fatigue is the phenomenon of cracking and fracture of materials under cyclic loading where the applied load is less than the static failure load of the material.

The mechanisms of failure depends on the material, for example metallic and fibre reinforced materials (whether they are metalic or plastic) suffer from fatigue but their mechanisms are different.

Fatigue analysis is often empirical based on many experiments on test specimens. Fatigue calculations are aimed at trying to predict two things. First the time required to produce the initial crack is estimated and then the time required for the crack system (in the case of F.R.P. materials a web of cracks and an area of damage spread rather than a single crack as in metals) to cause failure.

The calculations are imprecise and contain a large element of statistics. This statistical constituent is contained in the empirical formulae and the test data on which they were based. The tests by necessity are speeded up and this effects the results probably due to the heating effects.

But the calculations are made even more imprecise by the uncertainty in the loading conditions likely to be met. This is why most certification procedures require a factor of five applied to the fatigue life untested structures, this factor reduces with the number of tests conducted, however fatigue testing of aircraft structures is extremely costly and time consuming.

There are may other complicating effects too. Quite simply the weather can reduce fatigue lifes, hence what type of coatings one uses and operating climate are important.

9. Why Develop a F.E Approach to Weight Estimation?

Most existing methods of weight estimation are very restrictive on possible usage. Most are directed at the parametric design study stage, those that give estimates for the feasibility or preliminary design stages tend to be optimisation programs which provide weights for comparison (i.e. as a base for the optimisation procedure) rather than absolute weights.

Pure weight prediction methods tend to be restrictive in the type of weight, some give whole aircraft weights, some predict subcomponent weights (such as wing weights) and some deal with components of overall weight such as structural weight.

In any case all the methods discussed so far have some sort of empirical basis or component which implies restrictions in use. In some cases these restrictions may only apply to aircraft class types. Usually the aircraft are classed in terms of aircraft utility groups (e.g. long range jet transport) or types of design or construction. Certainly use of novel materials where specific properties differ greatly from those on which the method was based disqualify the method. This is also true of designs which lie outside the range of existing groups (e.g. the 1st forward swept aircraft). However, the expert weights engineer can (by means discussed in earlier chapters) adjust some of these methods to suit, particularly if they are semi-analytical.

The scope for use of an accurate f.e. base method of weight estimation is wide. There are two basic ways it could be used as a tool. The first would be as a design tool and the second as a design management tool.

The first class would cover the parametric study type of activity that occurs at the very early stages of the project where a blank sheet exists. This class also covers the detailed preliminary design study stage where the first detailed drawings are drafted.

In this type of activity such a method could feasibly give designers quantitative measurements of the benefits derived from alternative designs. For example the use of a different material or the placement of a subassembly, say the undercarriage of different locations.

As a design management tool it would be used as an independent monitor on the weight of a developing design giving up to date weight information as the design loop is carried out, leading to accurate load information.

Many groups may be interested in such a method of weight estimation. Customers of the aviation industry whether they be military or civil could use such a method to help develop realistic specifications for new aircraft types. The military establishments might find a use for it on their threat assessment analysis of aircraft. Civil operators might use it to evaluate new designs on offer from the aviation manufacturers. The most obvious users would be the aircraft designers themselves. It would prove particularly useful if it could be made part of their existing software or established procedures.

10. F.E. Approach to Weight Estimation

It is the finite element technique upon which this approach pivots. The technique can be applied to all the disciplines necessary to obtain the required result. These disciplines include, strength analysis, stiffness analysis, vibration analysis, thermal analysis, optimisation and aerodynamics. Most of which have been described in earlier chapters.

The finite element technique applied to structural analysis was mentioned briefly in an earlier chapter. One of the important criteria mentioned for a finite element analysis was that it should eventually converge to a solution which is similar to reality. This being so a finite element weight analysis based on structural analysis ought to be accurate where structural weight is concerned.

The problem of accounting for non-structural weight will of course remain but must be tackled if the eventual aim is to provide an overall weight estimate.

In this chapter some of the problems associated with the F.E. approach and their solutions are discussed. Before going on however it is worth glancing at the overall picture of how such a system would work. Fig 10.1 shows a general flow chart for the approach. A more detailed discussion of this is carried out in the program architecture chapter.

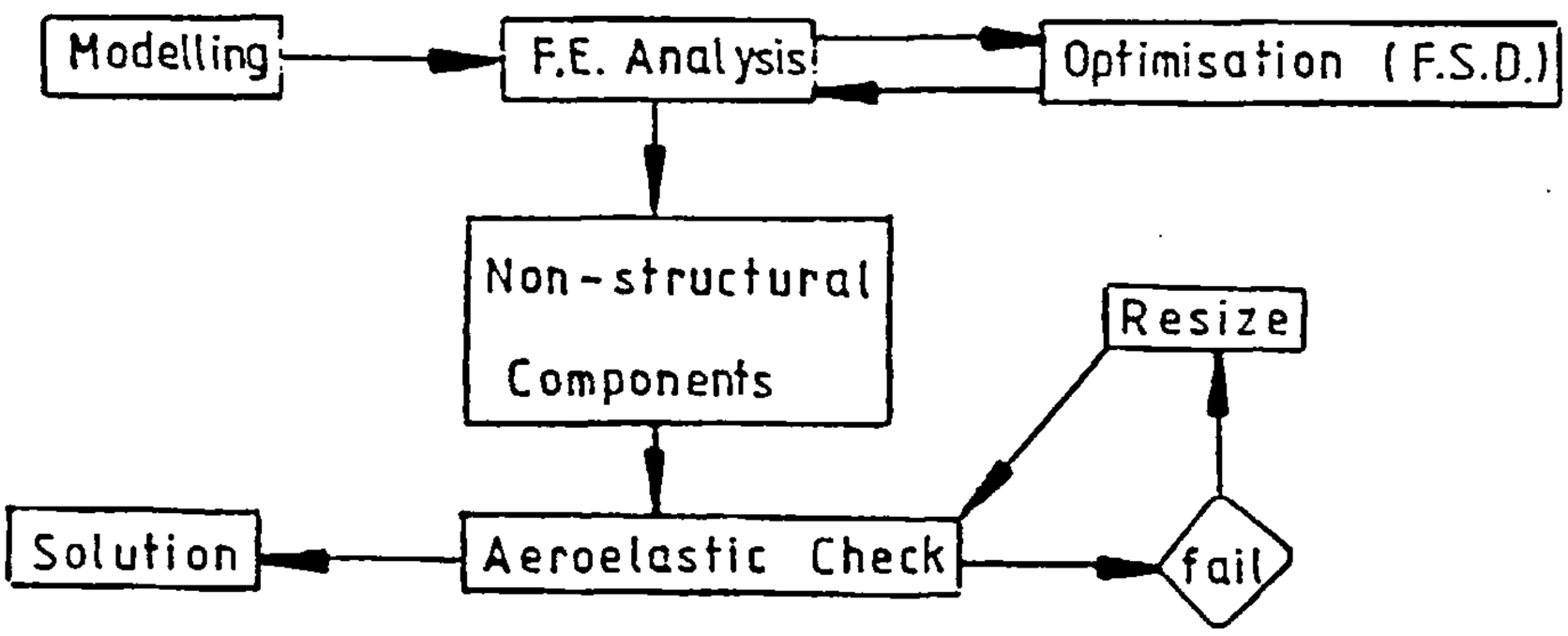


Fig 10.1 F.E. Weight Estimate Approach

10.1 F.E. Integrity and Application

If things have not been completely explained to the new comer to structural F.E analysis, analysing a structure using this technique could seem like childs play. The commonest mistake is to think of the method as a set of building blocks, and that the smaller the blocks and the more of them one uses the better the outcome.

In reality, it often requires a great deal of skill and practice to apply enough judgement during a finite element analysis to obtain a good solution. Often as much skill is required in the modelling of the structure and its loads as in the assessment of the results, before a good solution can be obtained. That is why structural analysis wing F.E. is such a lucrative business.

It has been mentioned that the "wing" was chosen as the component to be analysed because it is in general simpler to analyse this type of structure than say, a fuselage. But even so, it is so easy to go wrong. Fig 10.2 shows an analysis of a wing box using the semiloof element. This is quite a complicated element with eight-nodes, which can model curved surfaces and accounts for bending, shear and membrane stresses. Even though the correct material properties, geometry, loads and element properties were used, this analysis has gone horribly wrong due to an element mechanism coming into play.

These complex elements are often used to reduce the size of the problem and because there is often no other way of accounting for surface curvature where it may be important. Simpler elements tend to be more robust and easier to understand, but the problem still exists. Furthermore, in this, case, the problem will not go away if more elements are used. The only way in this case of irradicating this floppy behaviour was to place dummy ribs of very low stiffness in the bay, as shown in fig 10.4 to increase the nodal valency solving the floppiness probably caused due to the flatness of the surface, together with low bending stiffness leading to lack of out of plane stiffness.

Luckily for us simple 4 noded membrane elements together with 2 noded bar elements give good results when applied to wing structures. This is also fortunate in the sense that these elements are also

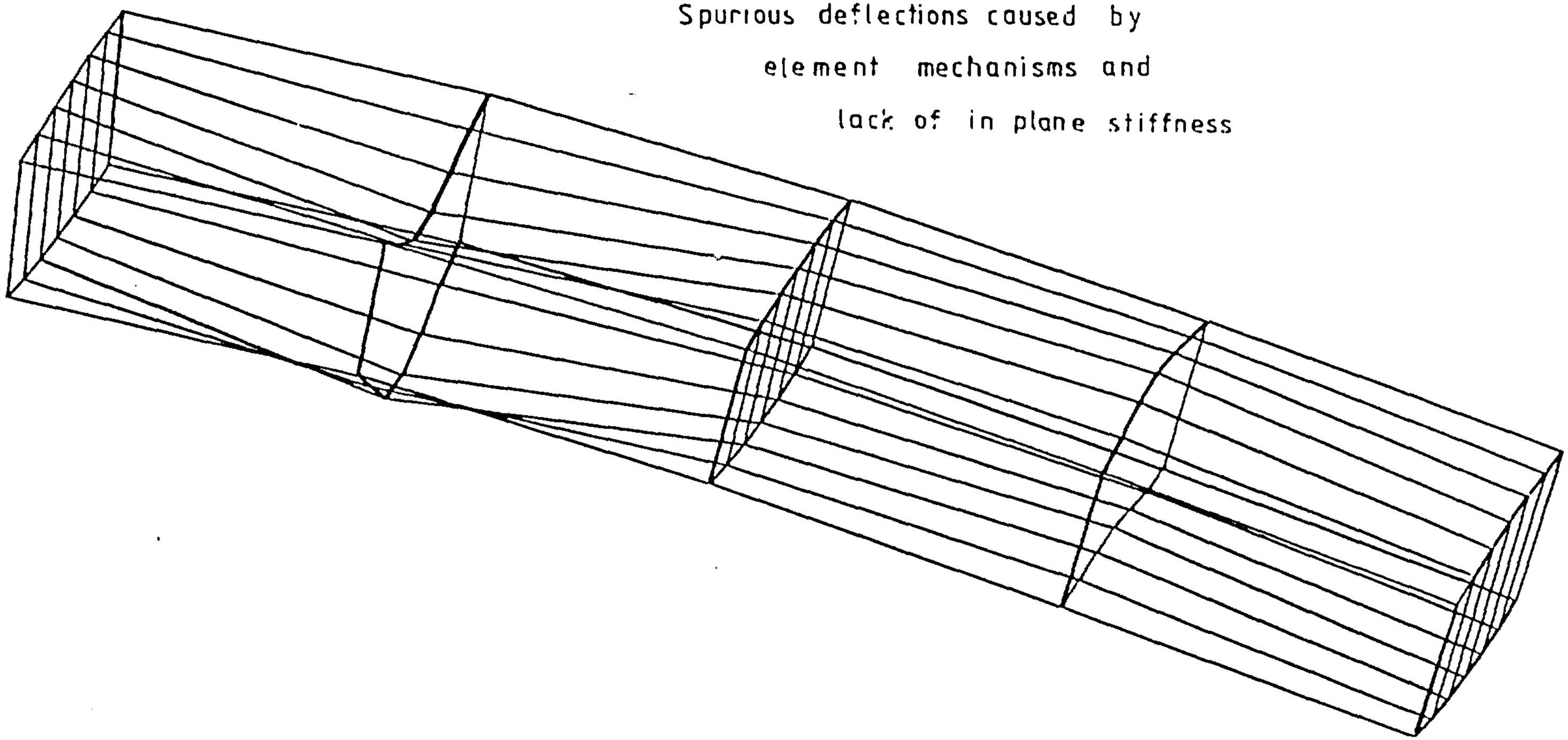


FIG. 10·2 F.E. Model of a test box

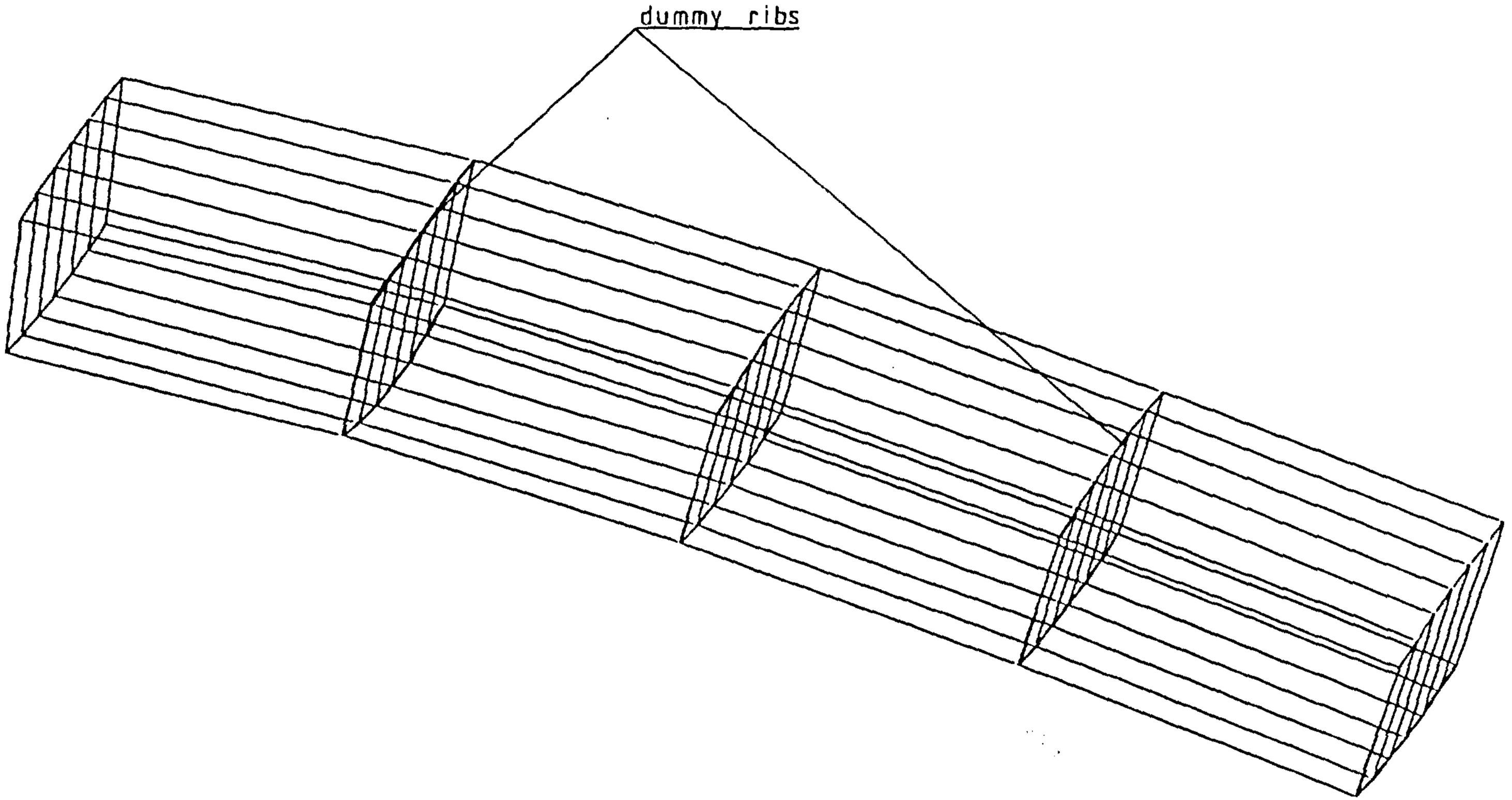


FIG. 10·4 Use of dummy elements to improve model behaviour

easier to work with in terms of optimisation (ref Lansing et al), which will be mentioned in more detail later. The reason for the correspondence with reality is these elements model very closely, what the skin and stiffners do in the real structure.

During an analysis, an engineer experienced at the task would go through many stages, the major ones are, shown in fig 10.5.

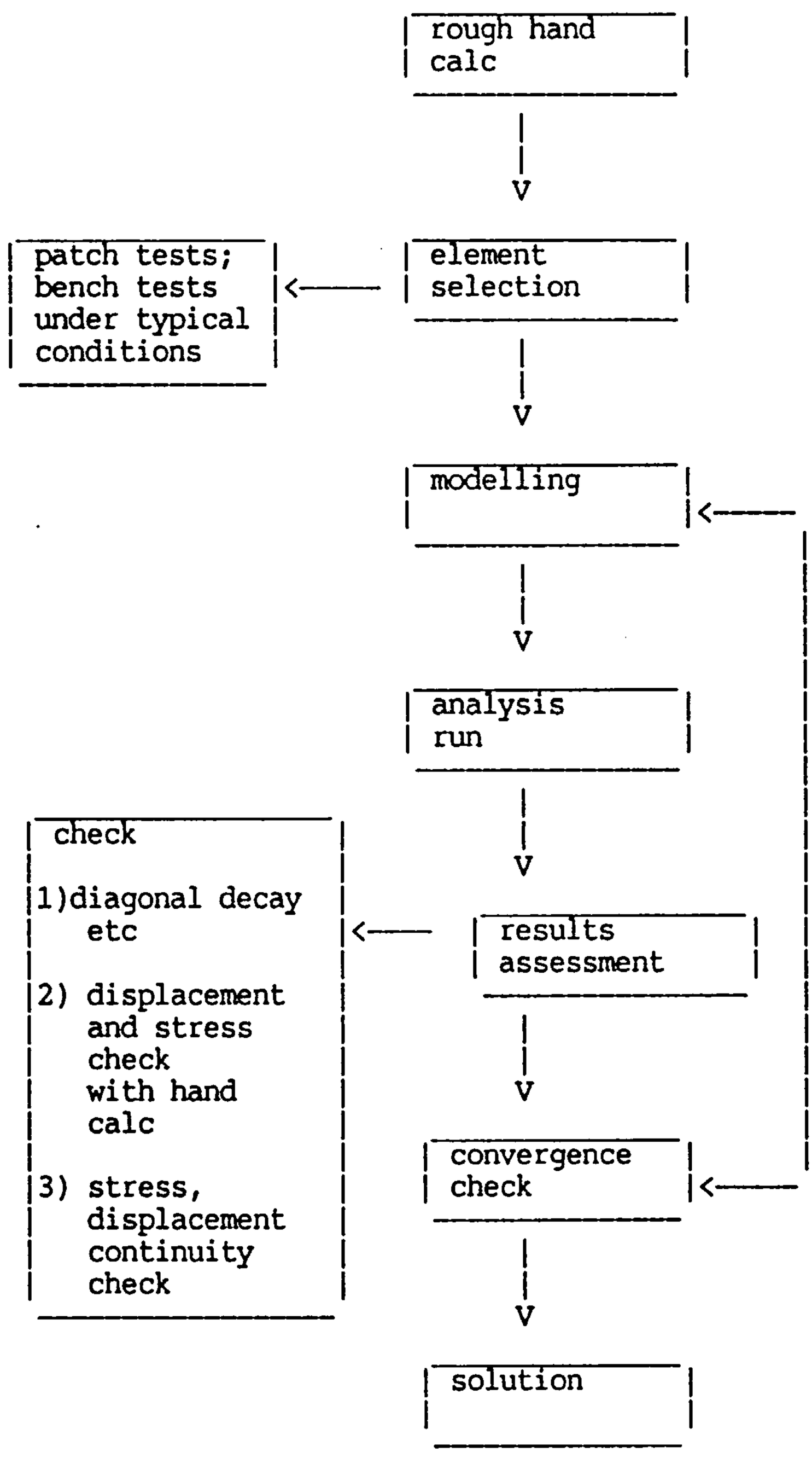


Fig 10.5 F.E. Analyses Testing and Use

The hand calculation is done since it is well known that F.E. analyses do not work unless one already knows the answer. Apart from being bad luck not to do them, hand calculations can be used later to check the analysis and are a valuable aid to modelling the structure. In addition being expensive and difficult to devise bench tests and patch tests which check the speed and accuracy of F.E. systems tend to promote a sensation of paranoia in engineers. Hence they are often left till one has a paying customer. These tests are used to evaluate which elements should be used and how they are likely to behave, and so help decide on factors such as mesh density and help in understanding the results.

Modelling is a difficult and tedious business helped along nowadays by preprocessor packages such as IDEAS, PATRAN and CADAM. Apart from modelling the structure, there are the constraints and loads to think about, more of this later.

At last the computer gets a look in, and does some numerical handle cranking before inundating the engineer with results. The engineer, pragmatic as ever takes nothing for granted and checks the F.E. packages diagnostics messages which often includes indicators of accuracy such as diagonal decay. Satisfied with this the resultant displacements and stresses at pertinent points are checked against those hand calculations. Finally the stresses and displacements are scanned for "funnies", in particular for discontinuities.

Satisfied that nothing has gone too badly wrong the results are deciphered, bearing in mind the patch test results and a solution is obtained. If there is any doubt about convergence another model with a different mesh density is run.

So it seems there are a couple of pretty big problems; firstly the F.E. method is not particularly robust; and secondly it requires a complex process to avoid pitfalls, a process that would be particularly difficult to program a computer to do. (in fact the feasibility study just such a program is being started at the college).

The solution to both problems in this case is simple. In the case of the analysis of wings, very simple membrane and post elements give good answers, and the behaviour of these elements is well understood and generally free of vices. The complex

process our expert has to go through may be done for wings in general leaving only the modelling as a variable part of the analysis.

This has been done for the package in use during development, but the process should be repeated for any other package to be connected to ensure that there are no problems.

10.2 F.E.Modelling

As mentioned earlier modelling of structures for an F.E. analysis is a difficult and tedious business. In fact the main problem with the analysis of a major structure like a wing entails juggling very many numbers, representing the structural topology, element positions and properties, material properties, loads, and constraints. (And we have not even got any answers yet!)

To make things a little less tedious and a bit more interesting most people write computer programs to generate the numbers for them if at all possible. (Especially if they had to do it manually the last time!). These programs are often very specialised, i.e. they might only generate say a particular type of model to represent a particular type of truck cab but would be useless for doing the same thing for a suspension bridge.

There are some very clever and complicated programs available for generating any model possible. Packages like PATRAN, CADAM and IDEAS are well known. They help take some of the sweat out of modelling by use of state-of-the-art interactive graphics, often requiring stand alone work stations to work efficiently as they demand lots of computing resources and slow down time share machines. The bad news is their purchase and running costs inhibit us as does the steep learning curve for a novice user before it becomes an efficient proposition.

Again this is where the choice of a wing structure comes in handy, because it is relatively simple to write a computer program to go through the entire modelling process for wings in general. The modelling process is complex but the main stages are shown in fig 10.6.

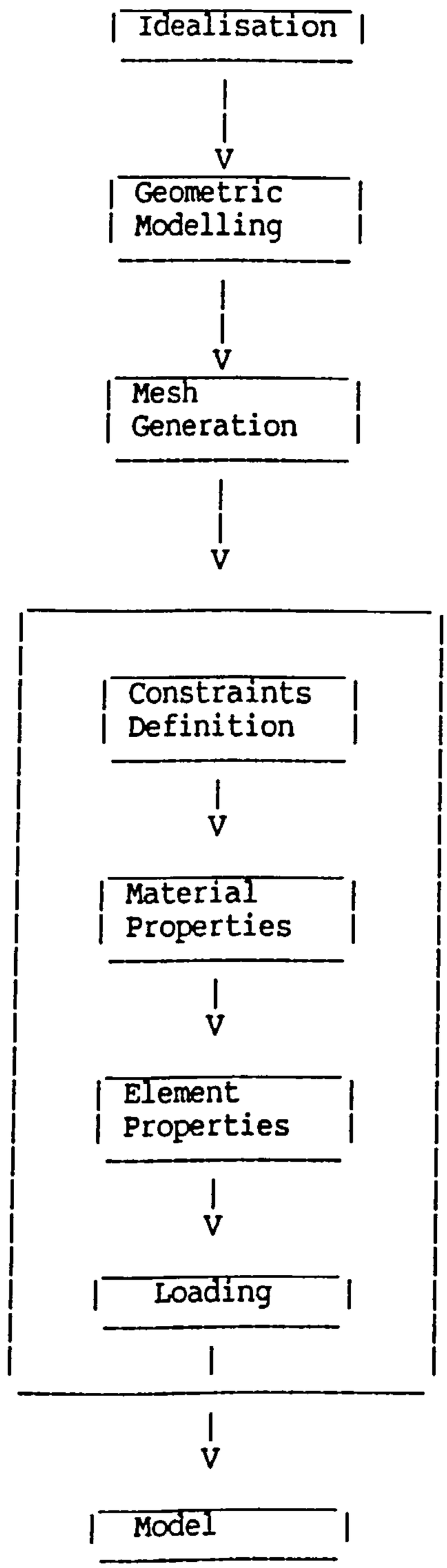


Fig 10.6 Wing Model Generation Process

The idealisation involves such things as deciding which components are relevant to the analysis and what simplifications are needed. For example non structural components are discarded at this stage along with any mechnical devices. Joints are often disregarded except for major ones such as pivots on variable geometry aircraft. Where the idealisation involves simplification care needs to be taken, for example in our case we idealise stringers as bars with no offset.

The geometric modelling involves collecting all the topological data regarding the shape and size of the structure, and is heavily connected with mesh generation. Mesh generation involves selection of suitable elements and their placement in the model. (It is often impossible to do geometric modelling without first selecting the element type to be used). In our case the selection of element types has been carried out as explained previously. This reduces the task to one of placement. Often mesh density needs to be greater in areas of high stiffness or greater geometric complexity. Overall mesh density is decided by what is needed to obtain convergence. There then follows a miscellany of tasks, such as constraints definition. Here a definition of how the structure is anchored is needed. In our case the structure is attached to the fuselage but how rigid is the fuselage and has this an important bearing on our analysis? More about this later.

The other tasks are self explanatory and loading is covered later. It should be noted that element properties (i.e. thickness and cross-sectional areas) are what we are after because this is what will be used to calculate the volume of material used and hence its weight. But to start the ball rolling we have to make a guess and define some properties so optimisation can take place. This is also covered later.

In short modelling is a house keeping nightmare which has to be removed if this approach is to be successful.

See appendix A for model generator details.

10.3 Loading

Some discussion has already been conducted on various methods of calculating aerodynamic loads, these boil down to empirical methods like the Shrenk and Stanton-Jones method, and analytical methods such as the vortex lattice, the supersonic source distribution and the aerodynamic f.e. methods.

Due to constraints on time, availability of software and computing resources it was decided that in the early stages of development, the simple empirical methods should be used. Both Shrenk and Stanton-Jones routines have been developed and are used together with an empirical formula for calculating torque. These routines give shear, moment and torque distribution for various planform at subsonic speed.

The strong intertwining of geometric modelling with mesh generation has already been discussed. There is a similar but weaker link between mesh generation and load application. For example if it is important to know the effect of a flap hinge load, it would be necessary to ensure that the application of the load is accurately represented. To do this there would have to be a way of "loading" the F.E. model at the exact position of that hinge, which means the mesh would have to be designed accordingly. This could be allowed for by giving the user the option of specifying special load positions before mesh generation occurs.

This naturally leads on to the problem of transforming the load system (aerodynamic or whatever else it maybe) to one suitable for application to the f.e. model. For example the shear distribution would have to be converted either, into a set of equivalent discrete loads to be applied to nodes or an equivalent pressure distribution over the wing surfaces. In either case, there is the added complication of ensuring that the moment distribution obtained from the transformed system is equivalent to that of the original.

Both the discrete load and pressure load transformation have been used and a brief description is included in appendix A. But to clarify the importance of this transformation its worth considering say the effect of loads on ribs.

Ribs are pretty clever things and have many general and special tasks to carry out. The earliest ribs in, cloth covered aircraft, carried out only one task. Ribs still do this task now regardless of whether the covers are aluminium or ceconite. By stopping the covers from flapping about like a flag in the wind they collect the distributed pressure loads on the skins and transmit them by bending to the spars which recieve them as shears, as shown in fig 10.7.

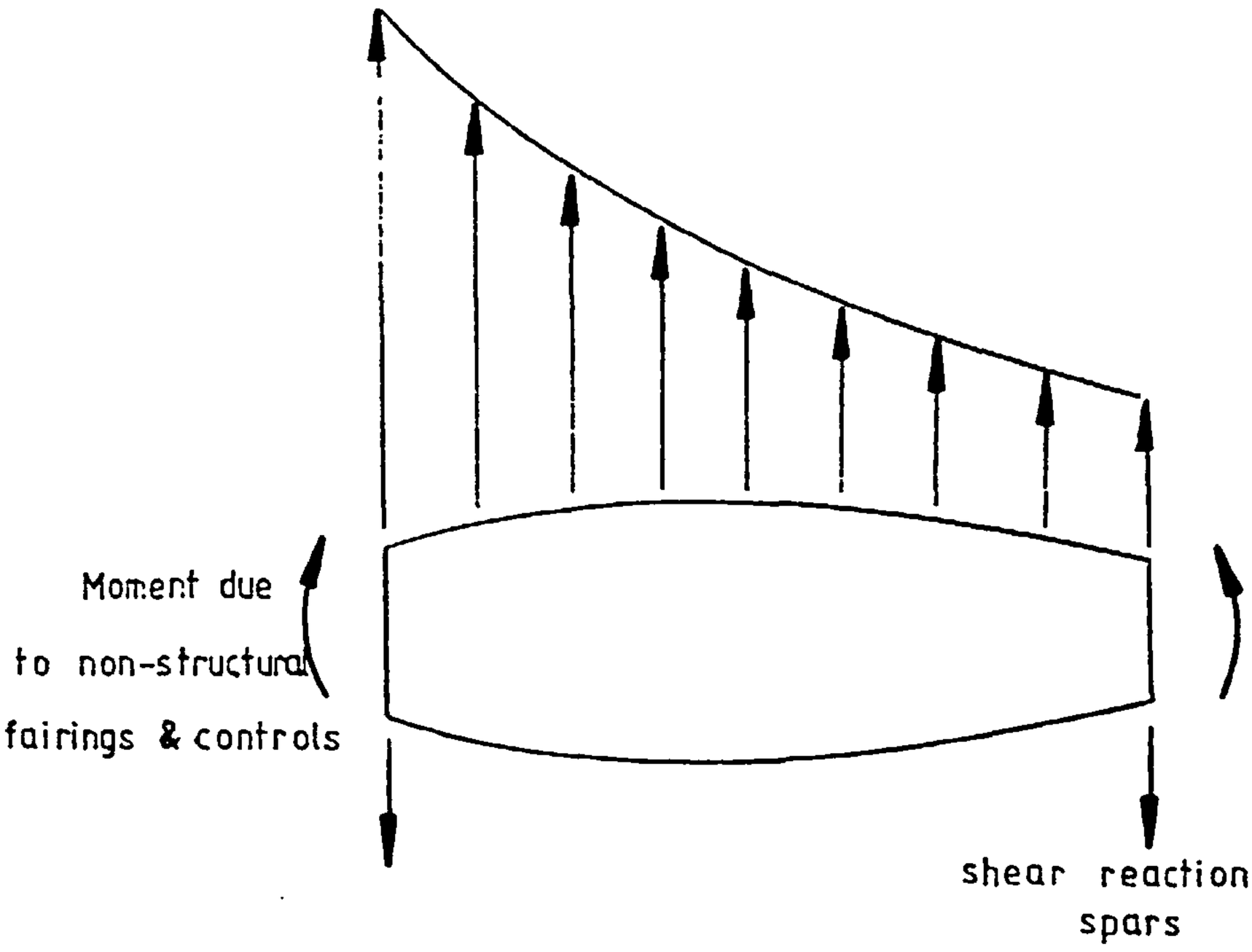


Fig 10.7 Rib behaves like a beam under distributed aerodynamic loads

Further, the rib stops the skins from folding in on themselves due to Braisier loading. This is a secondary load caused by deflexion of the wing which puts a crushing load on the ribs as shown in fig 10.8a.

Then there are the shear loads shown in fig 10.8b acting on the rib due to the action of the wing twisting under torsional loads. Then there are many other jobs our overworked rib has to do. There are hard point loads from stores, controls and flaps to carry along with pressures from fuel tanks etc.

C.R. Ritter covers the subject of rib design more throughly. Here he

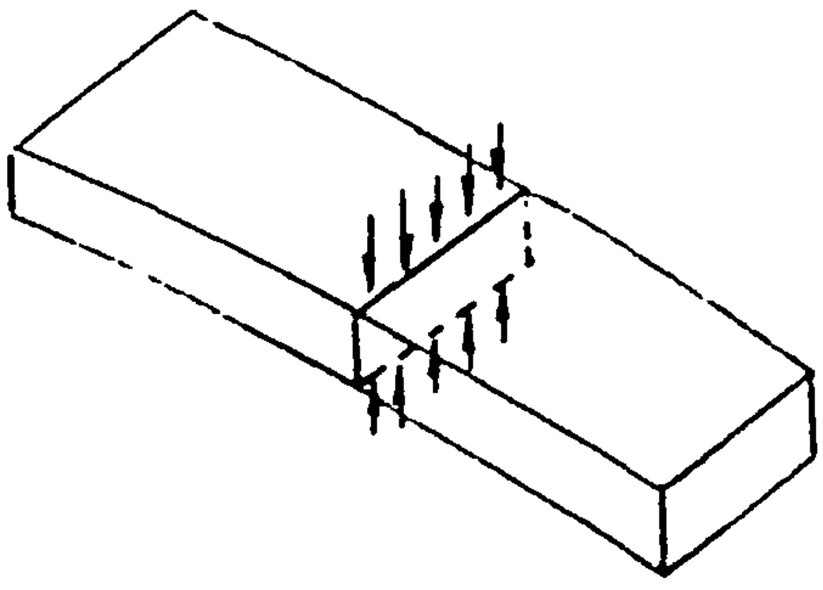


Fig 10.8a Crushing load on rib due to bending

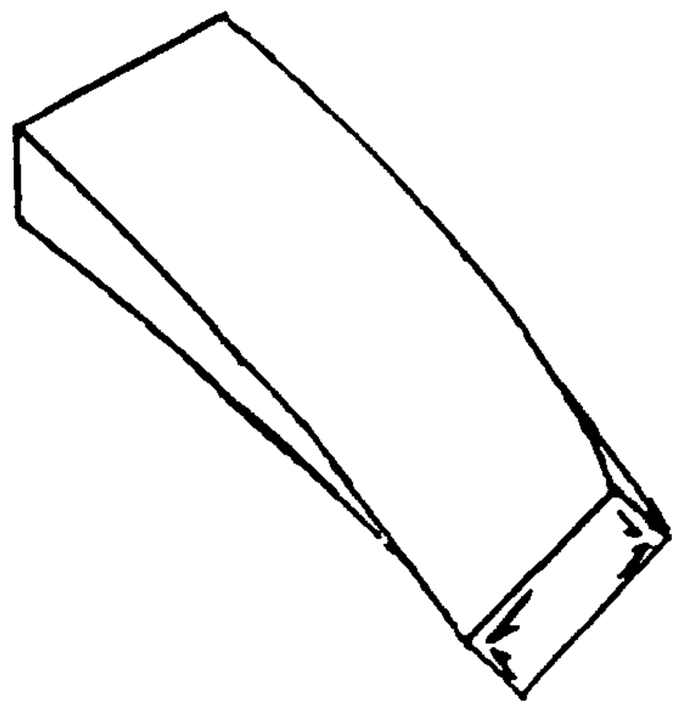
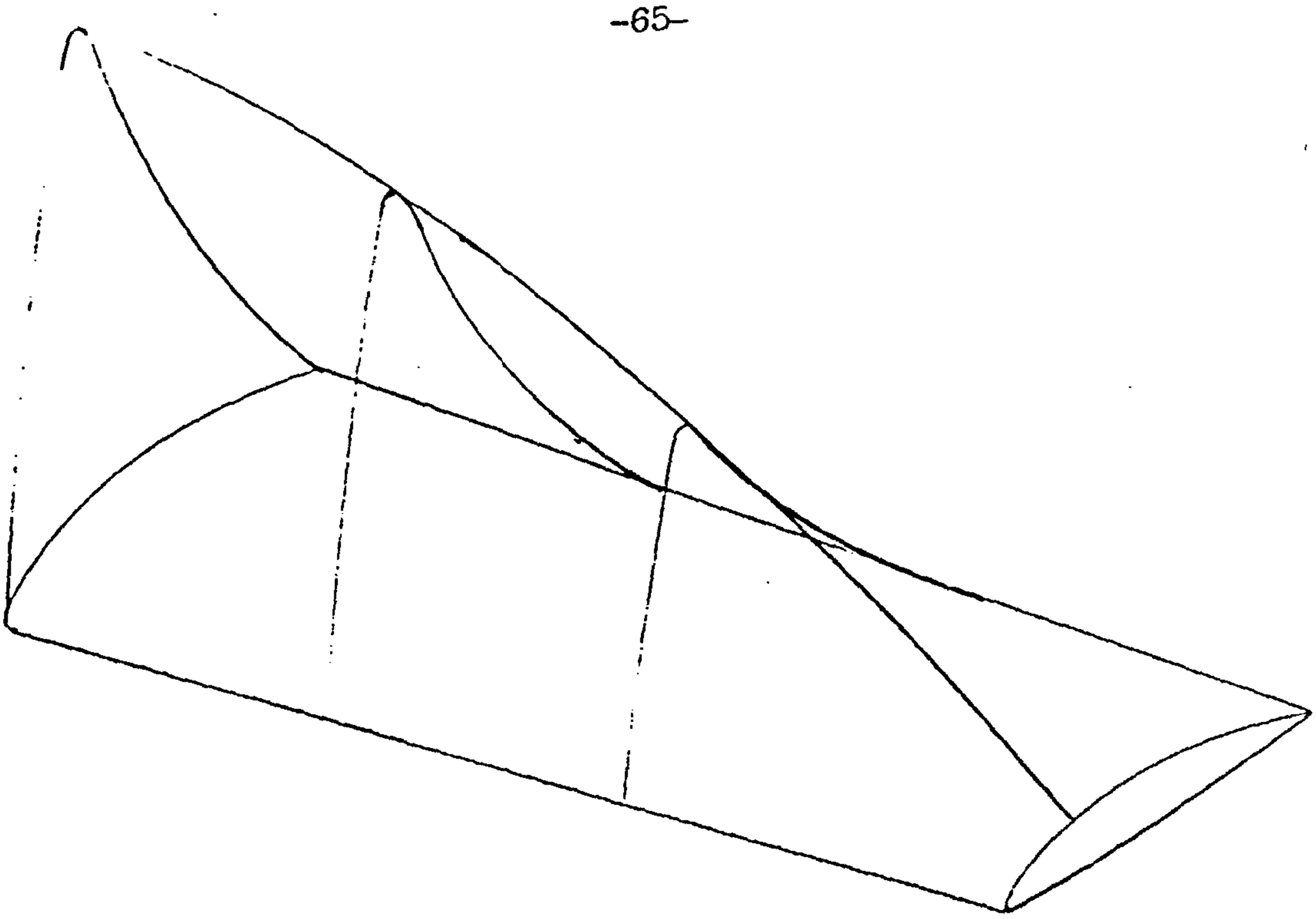


Fig 10.8 b Shear on rib due to wing torsional

has viewed rib design and loading from the point of view of weight estimation. In effect he does what we are trying to do here except without the F.E. ingredient, where the weight estimation is based on a design excercise. Despite this he has only achieved an accuracy of about 20%.

So it seems, that in the case of skins and spar boom structure, lumping aerodynamic loads at spar-node positions would be sufficiently accurate (in fact many design models of wings do it this way) but when it comes to rib design or spar web design more care has to be taken. In our case we have allowed a distributed load with allowances for missing aerodynamic components as shown in fig 10.9 i.e. we have within the accuracy of the load calculations, tried to represent the loads as true to life as possible. The envisaged use of f.e. aerodynamics can only improve the situation.

Kinson et al go through the problem in greater detail. Note no mention of design load cases have yet been made. In the case of fighters, this would most likely be a manoeuvre case whereas in an airliner it might be several cases (for different parts of the wing), say blade failure near the engine mounts and a landing case near the root. It would be up to the engineer to exercise judgement here with the program giving useful hints.



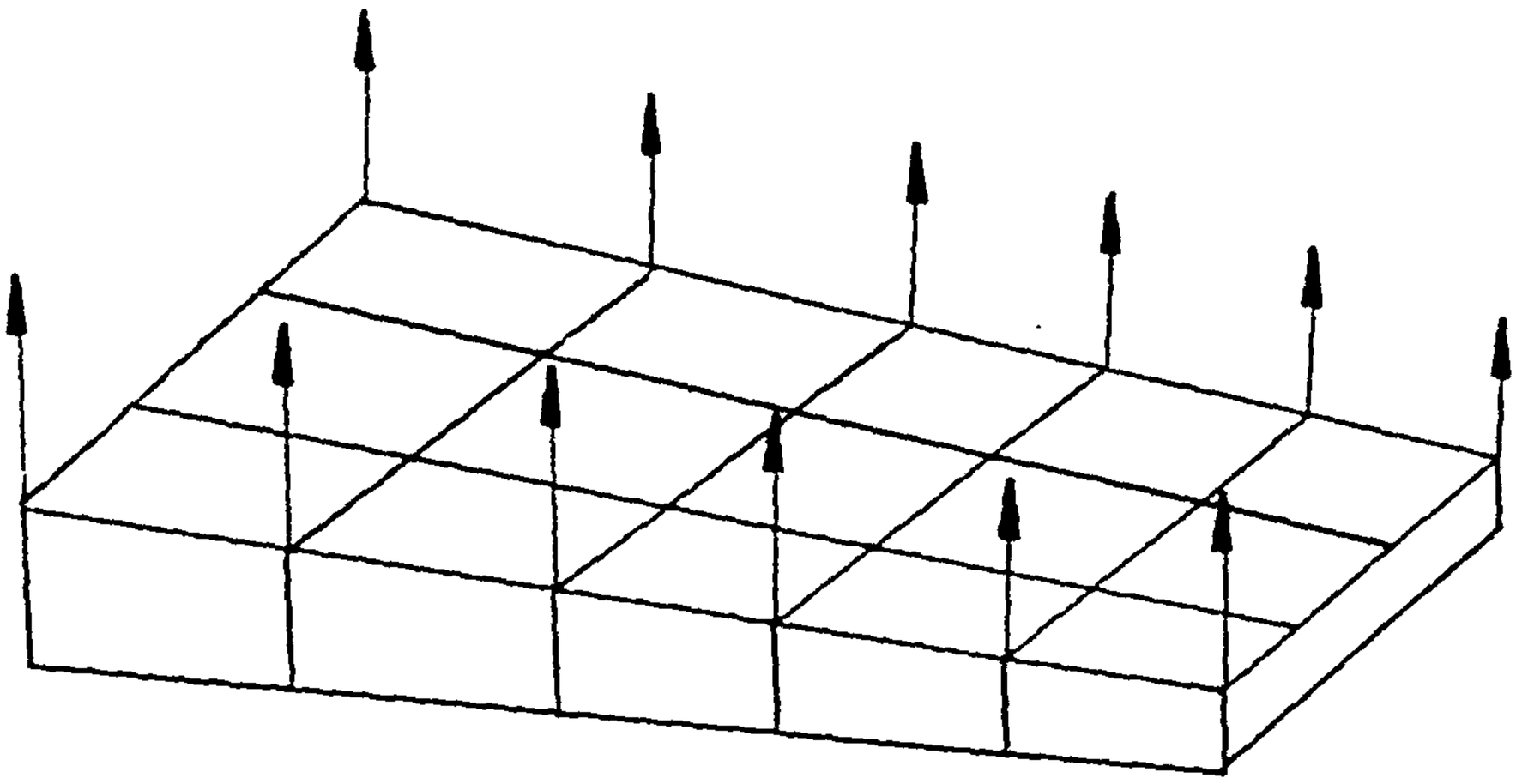


Fig 10.9 Conversion of Aerodynamic Loads onto Structural Grid

10.4 Displacement Constraints

The importance of how a structural model is anchored is highlighted when comparing the differences in the wing mounting arrangements in variable geometry aircraft with conventional ones. Note variable geometry covers variable sweep, swing wing, extensible wings (as on some gliders), variable camber etc. In one case there may be a large load carrying bearing with a small actuating link whilst the latter would have a vertical load carrying lug and a seperate drag link. There is also likely to be greater redundancy in supports in larger aircraft to avoid high load intensities.

Other factors such as attachment rib stiffness or wing design (e.g. continuous or split wing) have an important bearing on how the displacement constraints are modelled. A continuous wing is popular with designers since it avoids putting bending loads into the fuselage structure or a root joint. In this case a plane of symmetry may be defined at the wing center line and a plane of antisymmetry in antisymmetric cases. Assymmetric cases may be considered as combined symmetric and antisymmetric cases. In the case of a broken wing cantilever fixations have to be modelled.

In any case a simple "built-in" centre line rib as shown in fig 10.10 is unrealistic, whilst a more diffused fixation at the centre line and root ribs are probably more representative of the general situation.

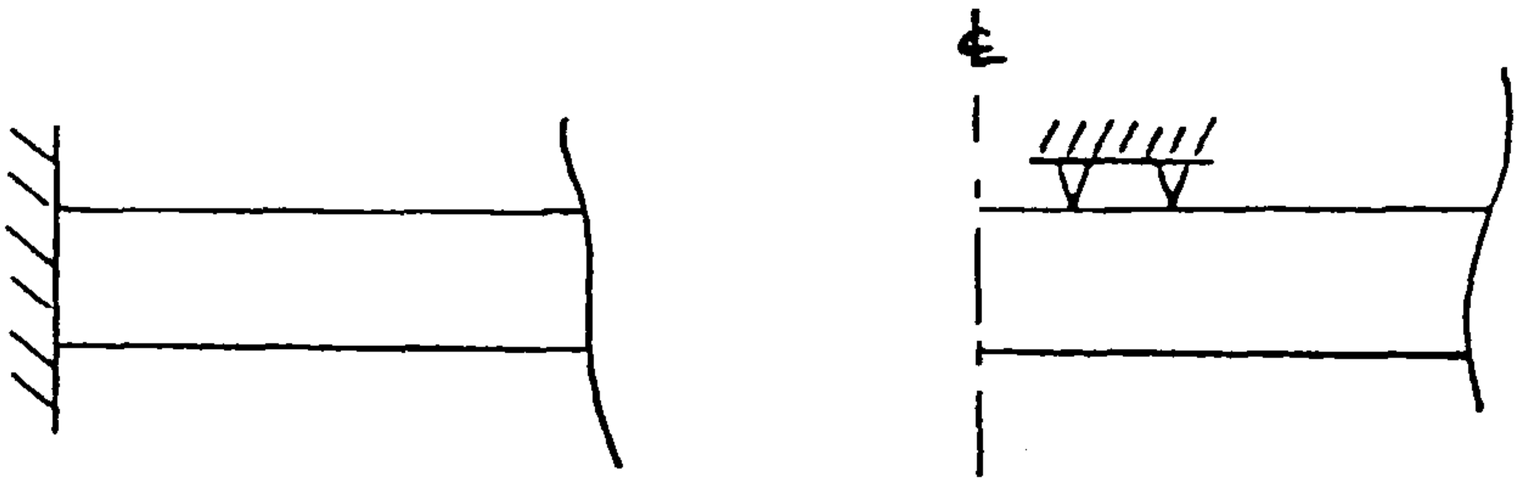


Fig 10.10 Typical Root Constraints

So in WEIGHTS the user was given complete freedom to define the constraints but a default of a built in root is provided if no user specifications are input. This means extraordinary designs may be modelled, e.g. twin fuselages, or delta wings.

10.5 Secondary Structure and Non Structural Masses

This is where things get more difficult finding a logical procedure may not be possible. Some items are pointed out here, in most cases solutions for accounting for them correctly are yet to be found.

The best place to start would be on the wing box itself where joints abound. Joints imply, cleats, doublers, lugs, bolts, pins, rivets, adhesive, welding, flanges, lips, etc. Fasteners could be accounted for either by modelling them in some simplistic form or using joint analyses such as described by Hughes, Taig, Green and Kerr. For the moment fasteners and their holes have been ignored. Doublers could be added in as lumped masses but at early stages of design it might not be obvious that they are needed. In the case of adhesives, welds and flanges a factor could be applied to the effected elements in the f.e. model. This process has been carried out and is described briefly in the appendix. Hence we account for the difference between reality and modelled structure as shown in fig 10.11.

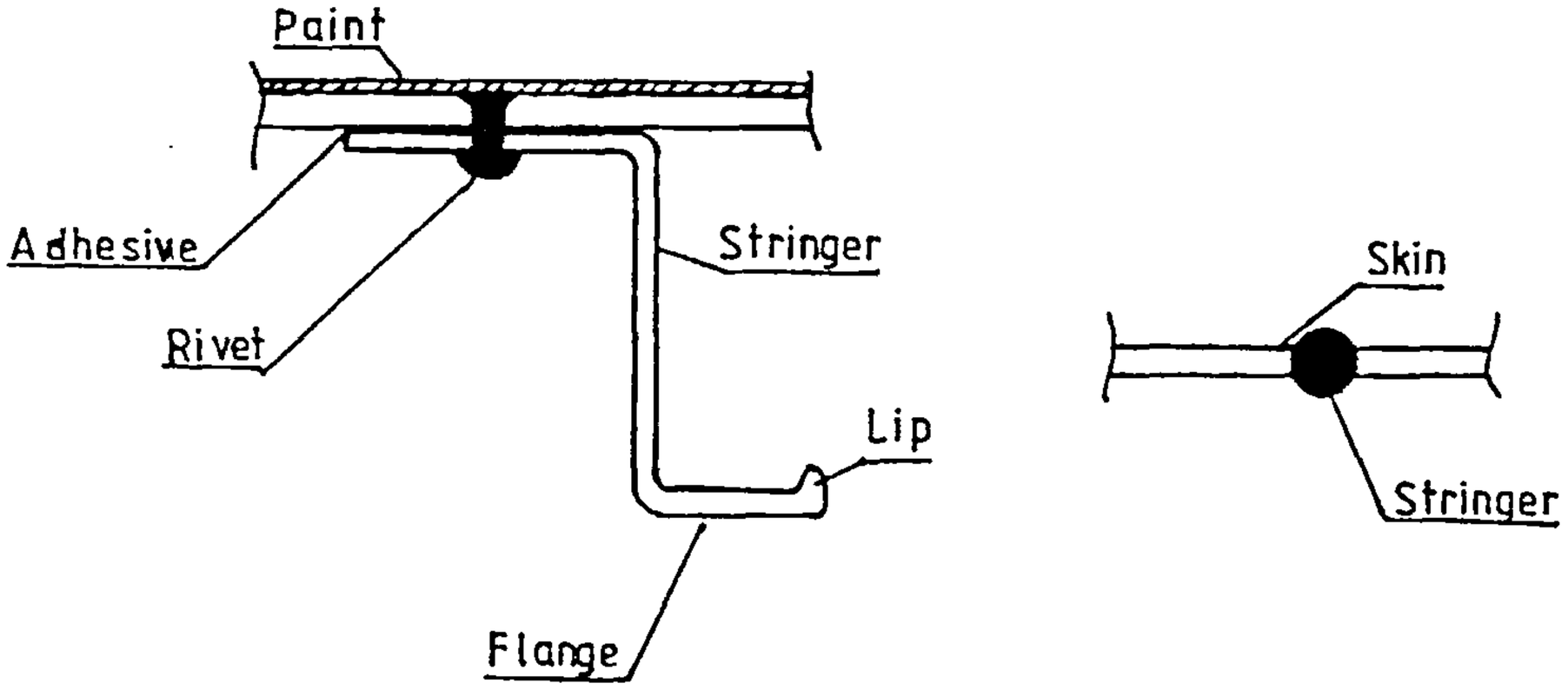


Fig 10.11 Real Structure and F.E. Idealisation

Apart from the common point layer for decorative and anti-corrosive purposes there are often other non-structural built-in items such as fuel tank sealant. The ultimate in protective layers has to be the heat resistant tiles on the space shuttle. On FRP structures, there would also be lightning strike meshes or heavy duty conductors and anti-skuff layers all part and parcel of FRP lay ups. Which leads onto composites (metalic and non metalic) which are often efficient in areas of directional loading but less

efficient in regions of high stress intensity or gradient or complex stress patterns. This situation usually occurs near joints and the typical case is the thick core sandwich at an attachment point. W Lansing et al showed that by accounting for the core behaviour doing an optimization of a structure near a joint the weight of a stabiliser increased by 100% over the optimisation which discounted problems of local core shear failure and skin buckling. There are other times when core behaviour is dominant, for example in the Quickie the wing has a solid core and in the Tiger Cub the wing consists of some aluminium tube spars with foam for everything else (the cloth covering material goes directly onto the foam), the ultimate core?

Of course in this case the foam could be considered as a fairing and the spar as the only structural component. Most aircraft have non-structural fairings, for example the D-nose, trailing edge, and wing root fairings on the airbus airliners are considered non-structural.

Fail safety may be accounted for by specifying a ratio of cover material to spar and stringer material. Then in the interests of safety and servicability there have to be inspection hatches. The covers and reinforcings for such cutouts have to be accounted for. There are cutouts for other reasons too, such as control runs, electrical looms and fuel piping.

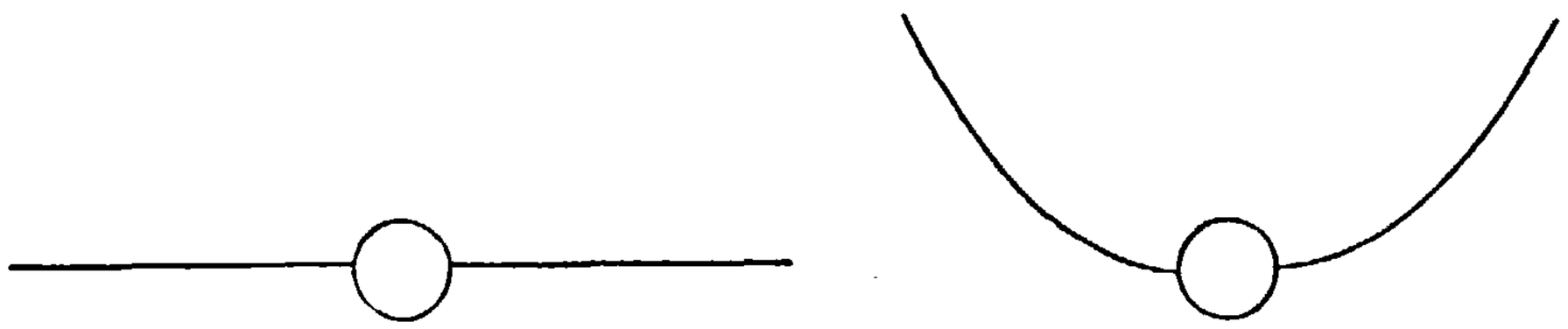
There are expendibles like fuel, weapons, weapons stores, avionics, external stores, deicing and anti icing fluid equipment, aerials, hydraulics, actuators, electrics, lights etc. In fact almost anything imaginable. The Tipsy Nipper even has a retractable step ladder and windows built into a wing. Many light twins have luggage space in the wings. In the case of some early airliners, floors and passage ways were built in the wings to allow engineers to inspect or repair engines in flight.

Often power plants and undercarraige including floatation gear find their way onto wings. Then are numerous secondary aerodynamic surfaces attached to the wing. These last two groups could theoretically be accounted for in the same way the wing structure is. They are not structural in the sense of the wing box but they are structures in their own right and so as the approach is developed it should be possible to treat say, the ailerons in the same way the wing box.

In the case of items which are definitely not structures, suitably distributed masses are needed to model them. The difficulty is knowing what they are and where to place so many items at an early design stage. The larger the masses the easier they would be to account for, for example external stores on a military aircraft would be easier to account for then its controls.

10.6 Failure/Design Criteria

Designing wings to be strong enough to take a particular design load without breaking is not necessarily enough as depicted in fig 10.12.



undeflected shape deflected shape

Fig 10.12 Strength sufficient but stiffness is not

Often other criteria lead to failure, lack of stiffness can lead to divergence failure or wing tips hitting the ground on landing etc. The Flying Flea is an example of divergence of the control surfaces which meant it was impossible to recover from a dive once a speed has been exceeded.

In some cases buckling can lead to failure or not be allowable. This is particularly true of supersonic and high subsonic designs where buckling on aerodynamic surfaces leads to high drag. On the other hand in some cases post buckled designs are the most efficient.

In most cases fatigue plays a part in aircraft design where fatigue life has to be several times greater than envisaged service life. Though fatigue is impossible to avoid, fatigue failure is avoided either by fail safe (redundant) design or safe life design philosophies. The early D.H. Comet crashes due to fatigue of the fuselage ADF, window cutout shows the importance fo this criteria.

At the higher speed ranges wing flutter as opposed to control surface flutter, becomes an important factor in wing design. Failure to account for this leads to failures such as those experienced by the Victor bomber where the tailplane suffered from this problem. At supersonic and hypersonic speeds thermodynamics comes into play. An extreme example is the space shuttle.

Some military aircraft such as the A10 Thunderbird were designed with a damage tolerance philosophy where the design still had to function normally despite severe battle damage.

Other designs (many G.A aircraft fall into this category) merely have to be strong enough to withstand handling and are so called minimum gauge designs. On the whole however, if judgement is used to adjust minimum gauges to cover fatigue, durability, and buckling the strength design philosophy would lead to a design of near minimum weight.

10.7 Optimization

As explained in an earlier chapter the optimization process carried out in this case would be a simple fully stressed design proceedure (FSD) followed by resizing for aeroelastic, stiffness, fatigue and buckling later. The reason for choosing this approach lies in, expediency, cost and overall effectiveness. For further details refer to the earlier chapter on the background to optimisation.

This approach is commonly used for structural sizing and the experience of others can be used. Lansing et al have some useful suggestions which have been followed. Firstly they suggest the use of nodal stress values as opposed to "averaged" element stresses the reason being that discontinuities in element sizes are more likely to arise by using the latter. This chess board effect is well known and can also be avoided by using a process called design variable linking where the sizes of groups of elements are made to change rather than single elements. A useful feature of this process is the

ability to obtain good f.e. representation and yet vary properties of large regions during the optimization. This occurs say when a complex component is made from one sheet of metal. In our case both nodal stresses and design variable linking are used.

Lansing et al suggest constraining the sizes of the bar elements until the membrane elements have been sized. Failure to do this, as verified by our own research leads to erratic behaviour during resizing.

They also recommend that, initially at least, no minimum gauge constraint be placed on the optimization. Only after the initial optimization has been carried out and a durability study has been carried out can minimum gauges be accurately judged. This last recommendation would be difficult to follow for a quick analysis but the option has been implemented.

This is as far as the optimisation procedure goes now but as some of the more important difficulties are overcome procedures for aeroelastic, buckling and fatigue optimization in the vein of those techniques discussed in the background chapter should be developed and included.

10.8 Making Life Easier for the User

Given two programmed procedures, if one is easy to use and all other things being equal it is likely to be more widely used and better understood and therefore produce better results, than the other program which is more difficult to use. This phenomena may be observed in any computer orientated environment where there are several packages which do the same job.

To give an example, when some new users are introduced to a computer system they are invariably given a demonstration of software which would almost certainly include several text editors. The editor that is the most user-friendly would be the most popular even though it might provide fewer facilities than a more sophisticated but more cryptic editor. User-friendly refers to a combination of ease of use, toughness (i.e. not easily crashed) and how easily it can be understood.

This is true for engineering software where cryptic

difficult to use software seems to be the par. To find popularity a program must do more than just give the correct results.

Summarizing user-friendliness we see that, firstly simplicity in use is imperative and there are many ways of achieving this. One way is to make the program commands and input easily understood so that once learnt the user controls the program with ease. This is fine for regular users but for the one-off or casual user this system has the draw-back that the learning curve has to be overcomne each time the program is used. Other paths to simplicity are possible at the expense of higher computer processing costs. Highly interactive menus together with "mice" or "wands" can make life a lot easier and in some cases make the "QWERTY" keyboard largely redundant. Its also worth noting the uses of high speed graphics and colour for this purpose.

Secondly it has to be a robust package. It should be extremely difficult or impossible for the user to "hang" the program and if it does happen some indication as to why it happened should be given. If it has a high typical elapsed time for execution, it should not be sensitive to hardware failures and should have some sort of recovery action from failures. This toughness should be extended to the documentation. It is common when using software to run up against an error for what seems to be a correct submission only to find the reason for the error hidden in some obsure section of the documentation,or worse it could be due to an error or omission in the documentation.

Thirdly it should be easily understood, and the process (if it is interactive) should be easily visualized so that the user can keep track of the current status and position of the "run". i.e. in a complex program it is easy to lose track of which data has been generated, what output is available and how far computations have progressed.

Fourth, is simply, that speed of execution should be as fast as possible, during interactive sessions. The user should not have to wait more than a second for a response to a command. In practice this is difficult to achieve, particularly on any sort of time sharing system, where response slows as the number of users and tasks increases. The solution to this problem has been found in the form of stand alone work stations, where the interactive session is carried out in what amounts to a machine

which is a computer in its own right but which has the capability to communicate rapidly with a host machine which does all the number crunching. In this project it might be possible, eventually to use a high speed micro, with graphics capability for this purpose. Though the requirement for high speed graphics would entail the use of a mini-computer work station of a high order of sophistication.

The fact remains that the success of this approach depends as much as the user interface as on the theoretical soundness of the approach.

11. Program Design Phylosophy

Since the original software requirements document was drawn up a major change has been made to the design phylosophy adopted. Apart from the practical aspects, the two most important abstract aims were to achieve "portability" and "user-friendliness".

11.1 Portability

The aim of achieving portability has been dropped as it imposed too many restrictions on achieving user-freindliness and on the technical capabilities of the program. In this case portability means the ability to install the program on more than one type of computer without modification.

It is easy to see how such an aim can cause difficulties. There are many types of computer with different memory, storage and word sizes and supporting software. Portability is achieved by aiming for a program which uses the common components of the target computers. The larger the range of target computers the smaller the commonality and the more restricted the program becomes.

For example if it was decided to include micro computers the memory limit could be as low as 16K, the programming language would probably have to be Basic and no graphics could be done since there is almost no common ground in this category between different micro computers.

11.2 The Target Computer

It was decided that the program be written for computers which are capable of serious finite element analysis. This means a machine that fits in the super-mini or mainframe category.

But this restriction was still not good enough since even in this category of machine there is not much common ground. Hence it was decided to use the full capabilities of the DEC VAX family of computers running the VMS operating system. This was because;

a) They are available to the College of Aeronautics.

b) They are extensively used in the aerospace industry in the U.K. and the U.S.A.

c) They are extensively used by the M.O.D., the sponsors of this project.

d) They are likely to be around in the foreseeable future.

11.3 The Programming Language

It is probably a fact that most engineering applications programs are written in Fortran. But then there are many subsets of Fortran, with Fortran 77 probably the most used today. The reason for this is a form of inertia and a degree of illiteracy on the part of the programmers.

Most engineers use Fortran because that is what they learnt first, it is what everyone else uses and they do not know any other language. In truth Fortran is archaic, cumbersome with its fixed format code (e.g. special uses for the first 6 columns in a program), cryptic with its numerical labelling and is difficult to follow with its implicit variable typing.

Languages such as PL/1 and Pascal are much more suitable, being less cryptic and more structured, the code is easier to follow making team development easier.

Since Pascal is widely available this was the language used. Note; though the routines written specifically for WEIGHTS were written in Pascal this did not stop it from being able to link together with existing Fortran routines.

12. PROGRAM ARCHITECTURE

The main aims achieved were, flexibility and user-friendliness. Flexibility in this case refers to the ease which it is possible to adapt the software to other uses, or, add to and modify it.

These aims and the initial software architecture were laid down in a software requirements document but changes have occurred since then and it is worth describing again here.

The basic structure of the package is shown in fig 12.1. The main components are the controller, the modules and the database. This chapter contains a description of each of these main components.

12.1 The Controller

The controller has three main tasks, the first of which is the setting up of the weights file storage environment. This is done each time the package is invoked and areas requested by the user are allocated to the current analysis. During this phase the program warns the user if there is any possibility of currupting data left behind from previous analyses and warns if there is a lack of files that are expected at this stage.

The second task is the definition of the package command language. This language is a set of words the user may use to invoke modules installed in the package. Details of these command words are stored within a module called hint which makes use of the native VAX/VMS 'HELP' facility. More details of this are given later under the heading of user friendliness.

The last task is the control of the flow of the modules invoked. Since the modules may be started in any order and run simultaneously, there is a possibility of a conflict between modules for the same data or one module might have to wait for another to produce data before it can proceed. In this case the controller acts like a signal-man on a railway network.

The first two tasks of initialisation are executed and finished in series before anything else can be done. The final task of management continues in parallel with the analysis.

The user has a choice of staying within a menu driven environment or exiting to the normal VAX/VMS operating system without terminating the analysis. Once the menu environment has been stopped, the user has access, to all the usual operating system commands as well as all the package command words.

To achieve this a combination of operating system command language, VAX system library modules and pascal was used. In summary the controller carries out the following tasks;

a) Database initialisation

b) Program command initialisation

c) Program flow control

12.2 Modules

Quite simply a module is a program which the package recognises and there are two main classes of modules. These are the resident modules and external modules which have been connected to the host WEIGHTS estimation package. The resident modules are those which have been written specifically for the project, whilst the external modules are such things as the finite element packages.

The database management system is a resident module, but it may be seperated from the other resident modules to clarify the workings of the package. This has been done in figure 12.2 where the module block represented in figure 12.1 has been split into 3 parts, the resident module block, the external module block and the database management system block. The arrows indicate the flow of data between the blocks. The database is also split into parts in this diagram but that is explained in the next sub-section.

What is not clear in the diagram is that the modules may be run simultaneously and the controller controls their flow. This has a small advantage in terms of speed of execution when used on the development computer a VAX 750 but will have greater pay offs on larger multi-processor computers, where jobs are not fighting for the same central processor resources.

As a summary we note that;

a) Modules are programs.

b) There are external and internal modules.

c) Modules may run simultaneously

d) The database management system is a module.

12.3 DBMS and Database

The schemetics in fig 12.3 show the ways in which modules are linked to the database. Figure 3A shows that early modules were connected to the database directly. This was because, at the early stages of development it was not clear what the database should contain and the impetus was on getting some sort of program working.

Later as it became clear what the database requirements were, and the database and its management system were devised. Thus all modules developed after this stage used the method, shown in fig 12.3b, of passing information to the database via the management system.

External modules have their own database and formats for input and output of data. To simplify the process of connecting these packages to the weights program the DBMS acts as a translator as shown in figure 12.3c, it simply translates the data stored in the main database into a format understood by the packages and visa versa. Each time a new package is added the instructions for this translation process have to be added to the DBMS.

It may be noted from fig 12.2 that the database is split into two parts. These parts may be used to contain the same information, only in different ways. The first method of storage is the sequential access storage method, where the files used for storing information are printable on a terminal and each type of data is stored in seperate files. This method is best for program development work where it is useful to be able to look at results of individual module runs directly. It has the disadvantage that it uses a lot of storage space and appears messy because of the number of files created. Another disadvantage is that this method can be slow because all the data has to be read sequentially and there is no way a piece of data may be extracted from a large file directly without reading all the data that precedes it.

The second method of storage uses random accessing, where all the information in the database is stored in one file and data may be accessed from that file directly without searching through it sequentially leading to faster accessing of data in large files. The disadvantage of using this method is that the data cannot be listed directly onto a printer or screen, making debugging of new modules difficult. Another disadvantage is that quite a complicated management system needs to be devised to maintain and access the database if it is to realise the advantage of speed over the simpler method. The DBMS also has to maintain the database if certain problems are to be avoided.

The DBMS has to ensure that there is sufficient room in the random access database for storage of data it needs to append, and yet make sure that once the data is added only a minimum amount of space is used. It also maintains what amounts to an index in the random access database which tells it where to search for certain types of data and how much data there is. For example if a module requires the span of the wing, the DBMS looks up the index which tells it that this type of data is stored in a chapter entitled "Basic Wing Geometry" and that the chapter starts on a certain page of the database file and that a certain amount of data is stored in that chapter. Armed with this information the DBMS then jumps directly to that chapter and starts searching for the required data. Note that because the page numbers of chapters change as data is added and deleted the index needs maintaining.

To summarize, the database is split into ;

a) A sequential access database, for use by development programmers.

b) A random access database, for other users.

And the DBMS carries out the following tasks;

a) Allows internal modules READ, WRITE and DELETE data on the database.

b) Translates foreign data to and from the database.

c) Expands and compresses the random access database.

d) Maintains the index in the random access database.

e) Transfers data between the sequential and random database files.

f) It is transparent to the user.

12.4 User-friendliness

The greatest contribution towards achieving the aim of a user-friendly piece of software in this case is the fact that most of the processes are transparent to the user. This leads to a minimum amount of instructions the user has to become familar with to achieve a result. In other words it's better to have a program which needs only a dozen words to tell it what to do than to have one which has hundreds of words and has a really comprehensive help or menu facility.

Having said that, this package has made use of the inherent facilities of VAX VMS to provide a useful help facility which gives the user information on any topic regarding the program. If developed carefully it could be used to replace the user manual.

Another module which enhances this aspect of the program is the interactive data aquistion module. This module arranges the questions and data entered by the user on the screen in an easy to understand format. Presenting as much data as possible at anytime whilst avoiding "Clutter". It uses VAX VMS software and is independent of the type of terminal device.

The user and programmer are allowed a high degree of flexibility by using this approach. The user as mentioned earlier is able to operate the software and the operating system simultaneously. The programmer is given great flexibility in connecting new packages and modules, and this is due to the design of the DBMS.

To summarize the contributors to user-friendliness are;

a) HELP facility.

b) High degree of simplicity/transparency.

c) Flexibility for the programmers due to DBMS.

d) Flexibility for the user allowing use of basic machine facilities as well as program commands.

12.5 Further Information

The original details of the software requirements are laid out in a Software Requirements Document. Complete details of the functioning and programming of the database and DBMS are contained in the Weights Database Manual contained in appendix B. Details of modules other than those invoking the database are contained in the Weights Programmers Manual. A collection of most of the references listed are contained in a "theoretical" folder available at the C.O.A.

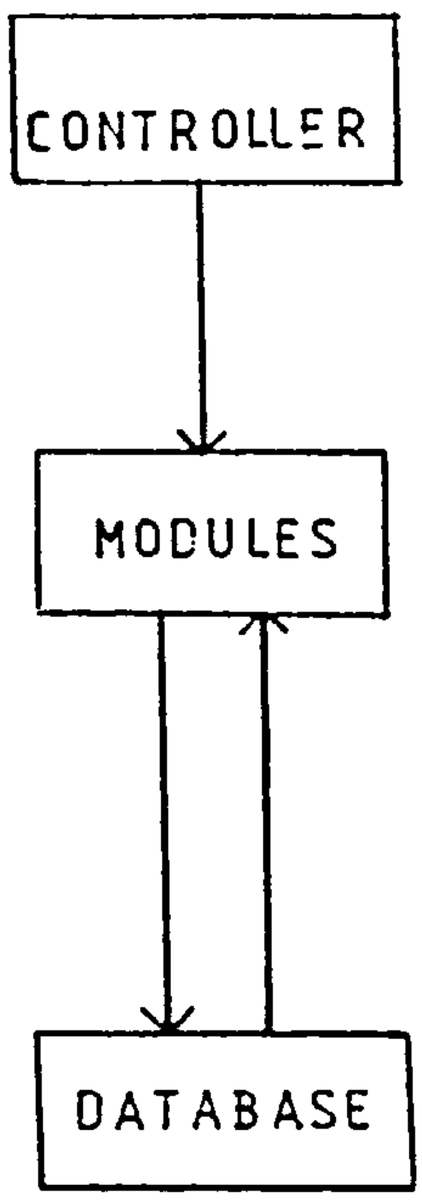


Fig. 12·1 Basic structure of WEIGHTS package

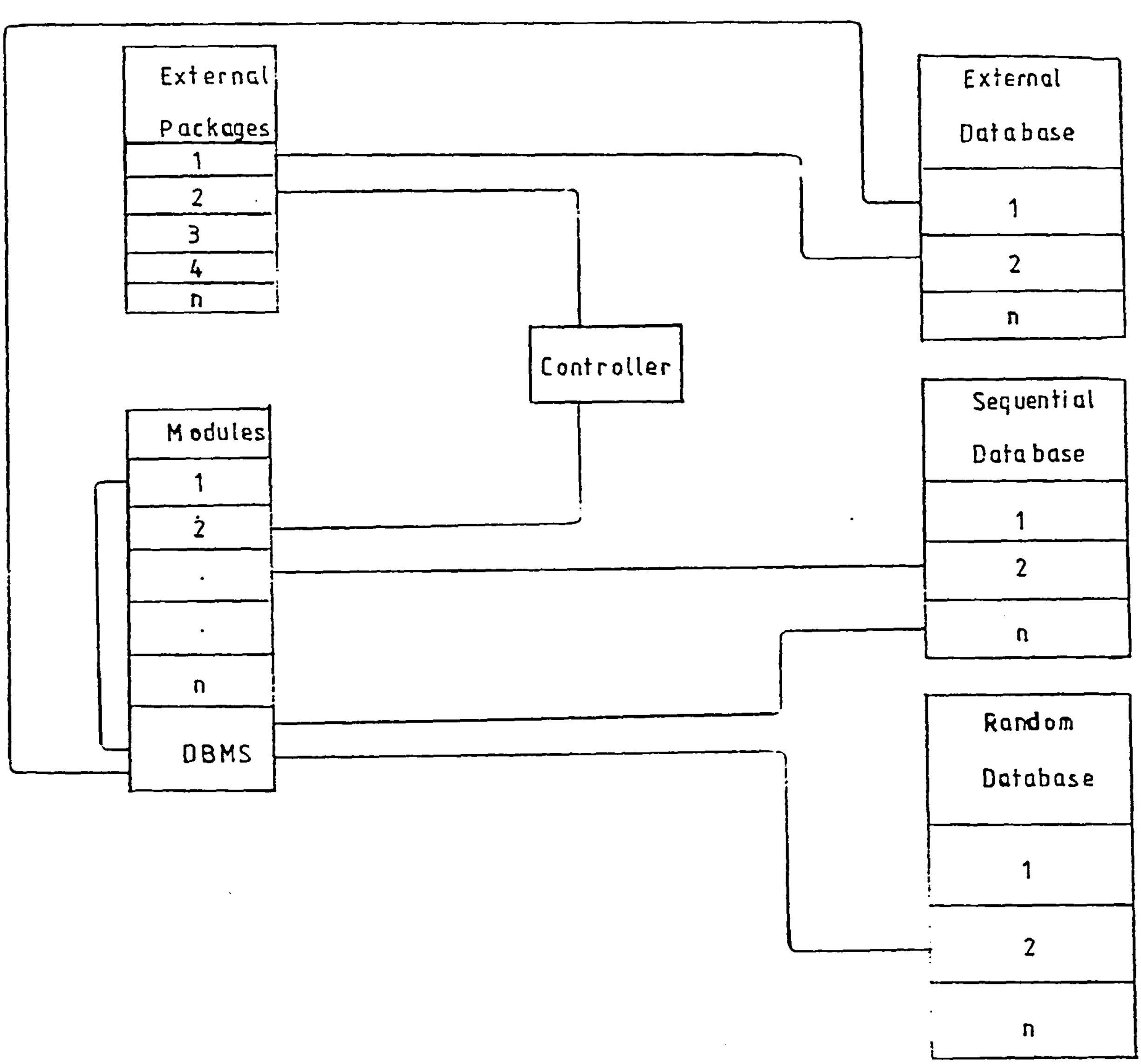
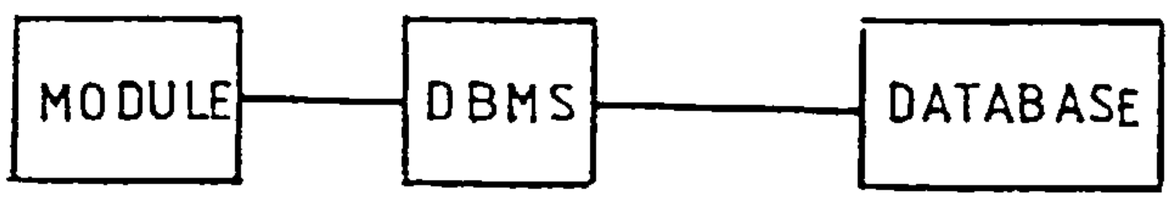


Fig.12-2 Block diagram of WEIGHTS architecture

a

b



c

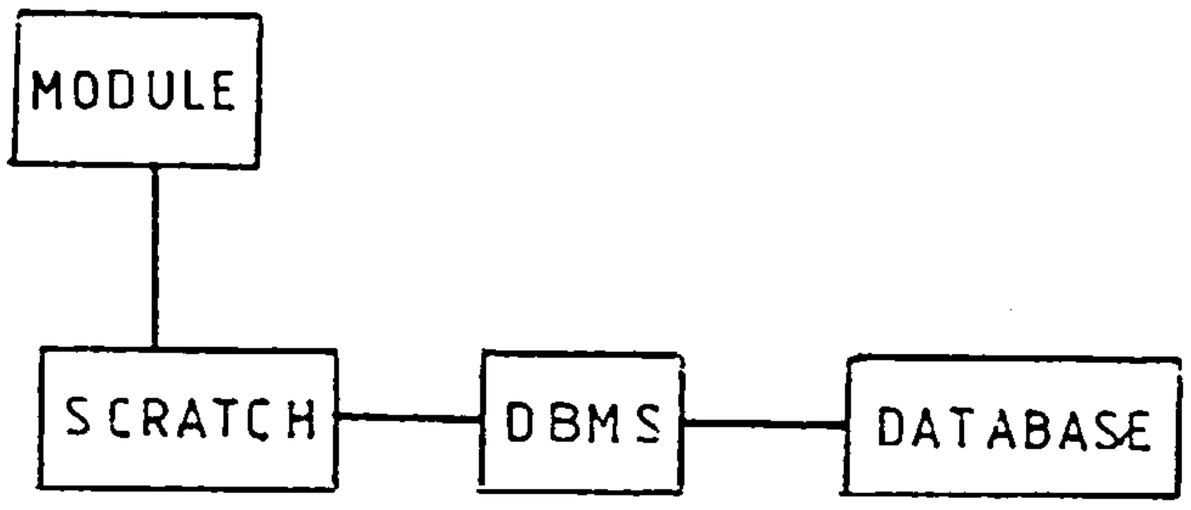


Fig. 123 Three ways of addressing the database

13. A1 Wing Box Analysis

The A1 is an aircraft designed at Cranfield to meet specifications laid down by British aerobatic pilots. It was built to be flown in aerobatic competitions with "g" limits of +9 and -6.

Since the aircraft was designed, built and is kept at Cranfield there is a wealth of information available which made it ideal for testing the WEIGHTS program.

Several weights analyses were carried out using the package with the view to investigating the effect of geometry and mesh density on the results.

These results were analysed by breaking them down into their component parts and comparing them with the existing wing. In general the results compared well and discrepancies were easily accounted for.

13.1 A1 Wing Weight Estimate

Unfortunately the components of the wing were not weighed during its construction,if so then no record of such a weighing survives. In fact until the such time that the wing is removed it is impossible to say exactly how much the wing weighs.

So an estimate of the structural weight of the wing was made from the construction drawings. The detailed weights breakdown is given in table 13.1 and the breakdown into major components is given in table 3 together with computer analysis results. Note that the weights are for a semi span.

More stringent tests where actual weight records are available for each component are available, must be carried to validate the software. This has proven impossible so far, since such data is only available from aircraft manufacturers who refuse to disclose the data due to its commercially or security sensitive nature.

Note that all the drawings of the stringer/stiffners were missing and so that part of the weight analysis was based on the initial design and stressing notes.

One semi-span only.		kg
Wing Panels		
Outer panels (top and bottom)	@ 24 SWG	1.585
Intermediate panels (top and bottom)	@ 22 SWG	5.38
Center panels (top and bottom)	@ 20 SWG	6.69
Panel sub total		13.66
Wing Ribs		
Rib 1	22 SWG	0.313
2	22 SWG	0.310
3	18 SWG (root rib)	0.555
4	22 SWG	0.307
5	22 SWG	0.287
6	20 SWG (U.C. ribs)	0.379
7	20 SWG (U.C. ribs)	0.371
8	24 SWG	0.206
9	22 SWG	0.267
10	24 SWG	0.185
11	24 SWG	0.158
12	24 SWG	0.127
13	24 SWG	0.102
14	24 SWG	0.074
15	24 SWG	0.055
16	22 SWG	0.054
Rib sub total		3.75
Rear Spar		2.197
Front Spar Web		3.11
Stringers		10
and F.Spar boom		10.03
Total Wing Weight		43

Table 13.1 Actual A1 Wing Weight Breakdown.

13.2 A1 Wing Weight Computer Based Estimation

The purpose of this exercise was multi-fold;

a) To find out whether package produced a realistic weight estimate.

b) To investigate the effect of mesh density on the results.

c) To investigate the effect of geometric descrepancies on the results.

d) To check the validity of the load modelling used.

e) Investigate the feasibility of analysing a composite material structure.

f) Investigate the effects of fixation points on results.

g) The effect of variable constraints.

To achieve these aims eight analyses were carried out. A drawing of each model is given in figs 13.1 to 13.8 and a list summarizing the main characterstics of each model is provided in table 13.2. A more detailed discussion about each model follows.

13.2.1 The A1 Wing Box Finite Element Models

At the current stage of development the package uses simple 2,3 and 4 noded bar, shell and membrane elements to model the wing. In this case, membrane and post elements were used. In general this was a good arrangement since the skin and web components behave as membranes whilst the stiffners, though present to increase the local bending stiffnesses of the skins behave as posts till buckling occurs. And since the package is only capable of coping with a strength analysis (until some more sophisticated software is plugged in) buckling and therefore local bending are unimportant making shell elements unnecessary.

The aerodynamic loads were calculated using the built-in empirical formulae which gave point loads at fifty positions along the span at the quarter chord position. These loads were then spread out along the front and rear spar using a numerical trestle tree. It was noted that, since the structural box was shorter than the aerodynamic wing the aerodyanmic

FIG. 13·1 Analysis 1, untapered with root supports, 16 ribs 8 stringers

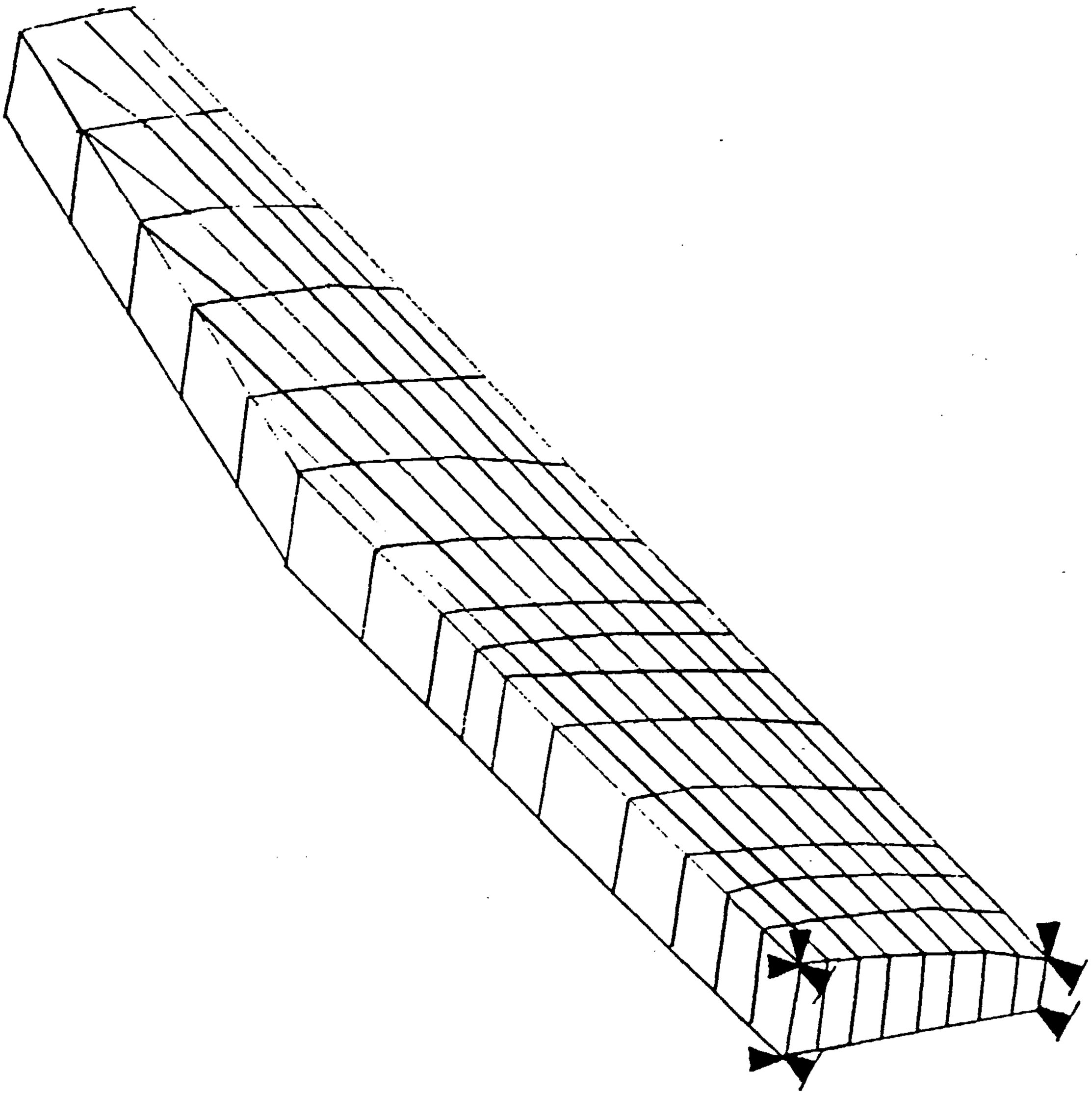


FIG. 13·8 Analysis 8, tapered with supports on rib 2, CFRP material, 10ribs 8stringe

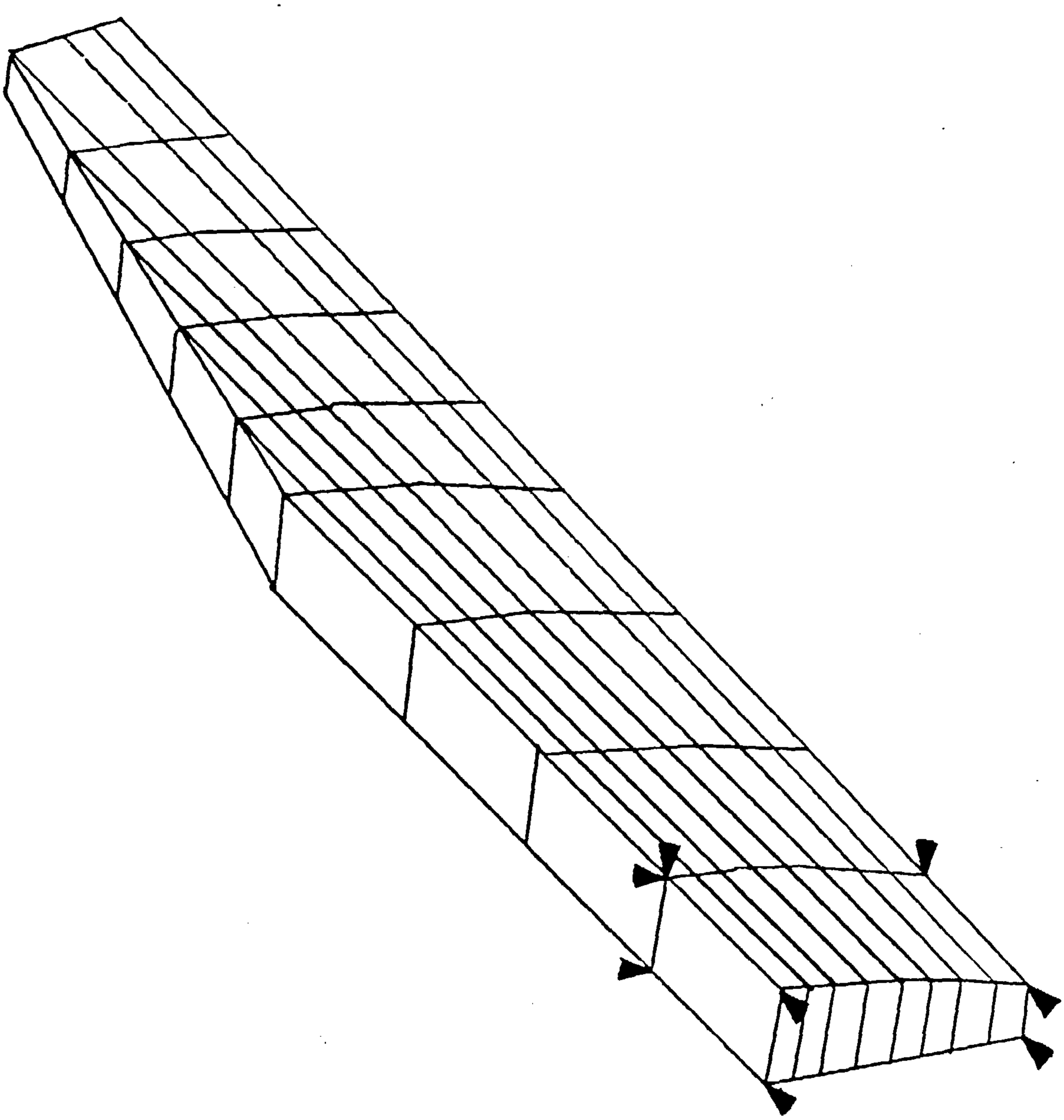


FIG. 13·7 Analysis 7, tapered with supports on rib 2, 10 ribs 8 stringers

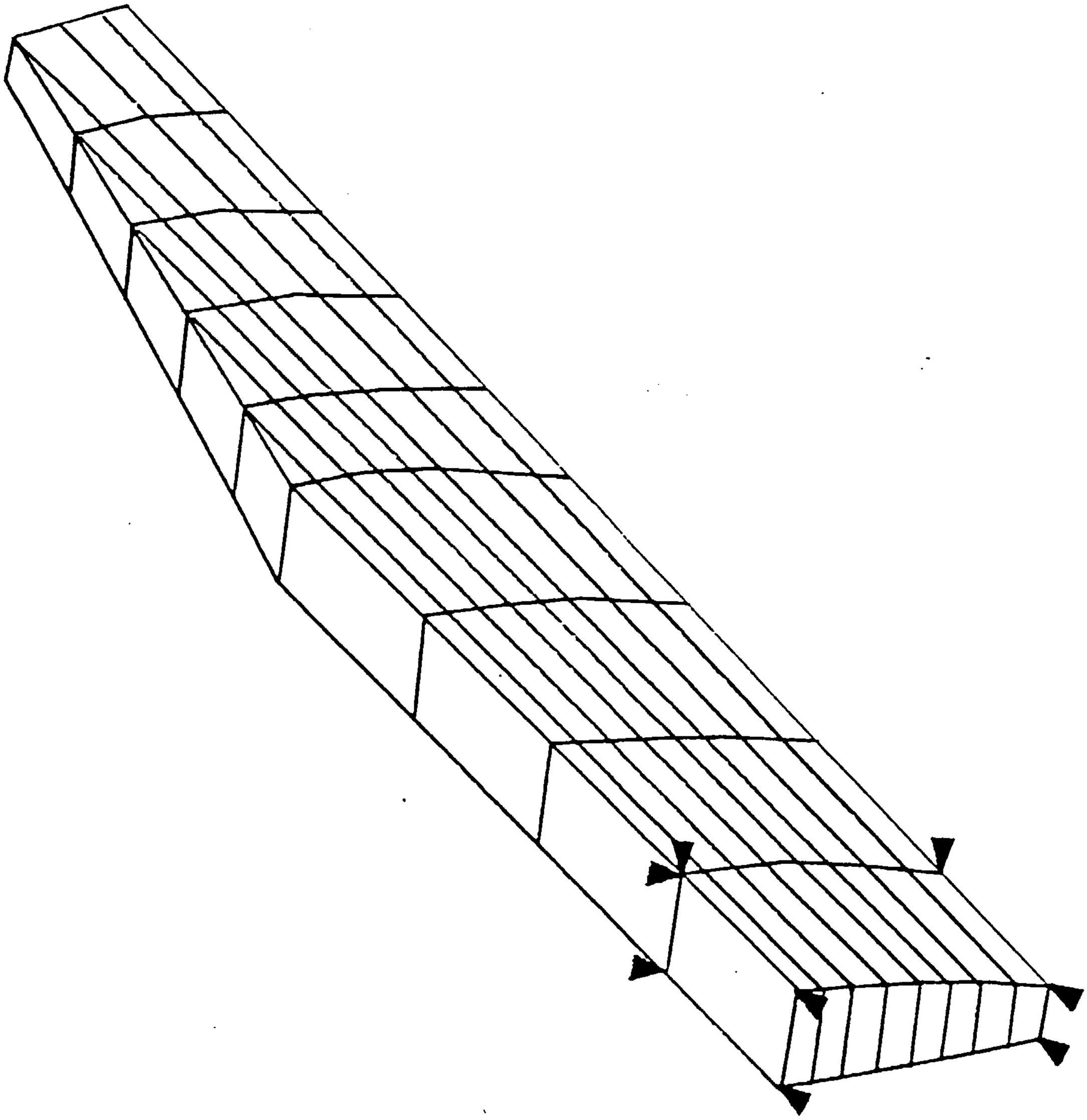


FIG. 13·6 Analysis 6, over taperered with supports on rib 3, 16 ribs 8 stringers

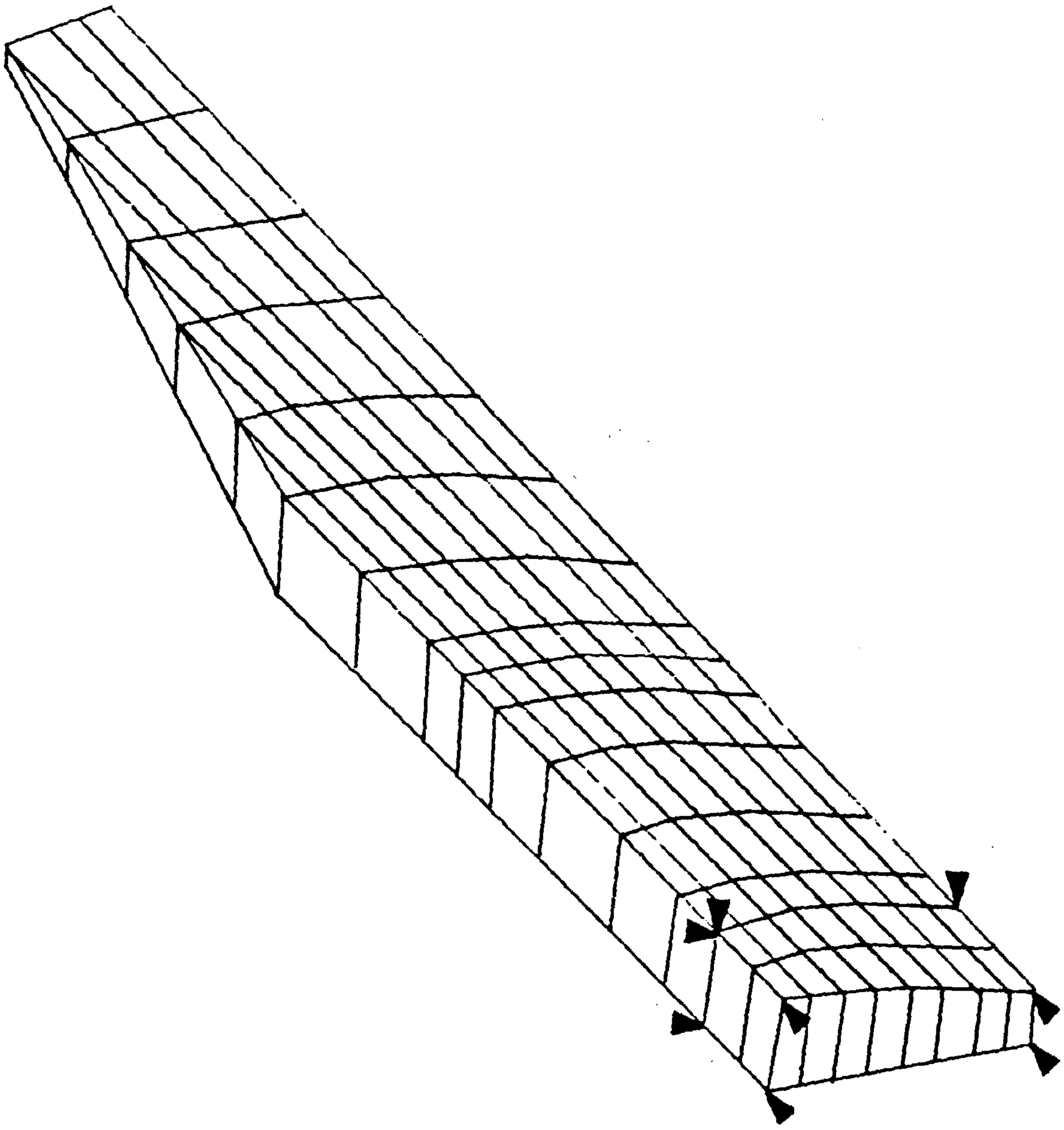


FIG. 13·5 Analysis 5, over tapered with supports on rib 2, 8 ribs 5 stringers

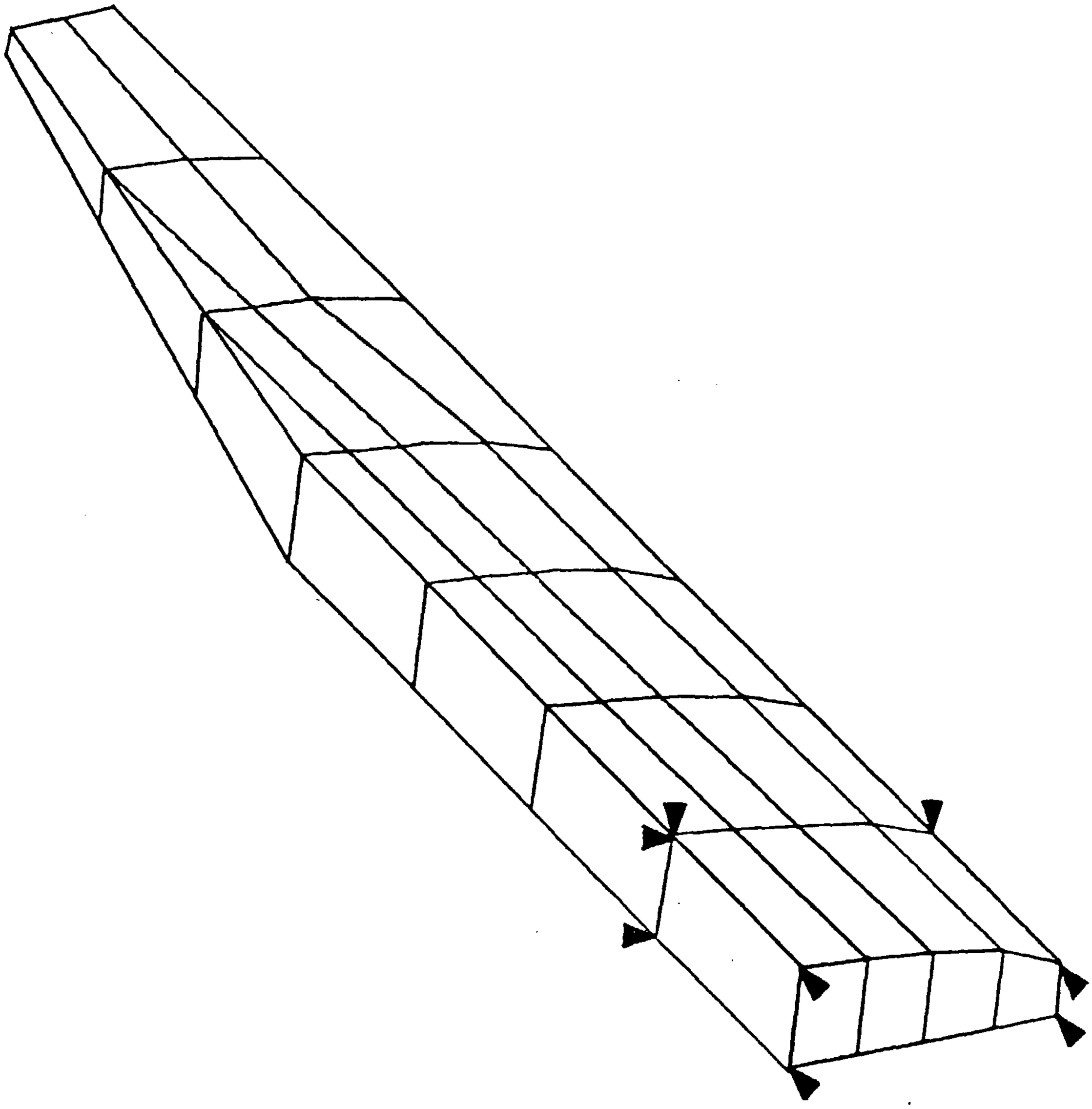


FIG. 13·4 Analysis 4, over tapered with supports on rib 2 , 16 ribs 8 stringers

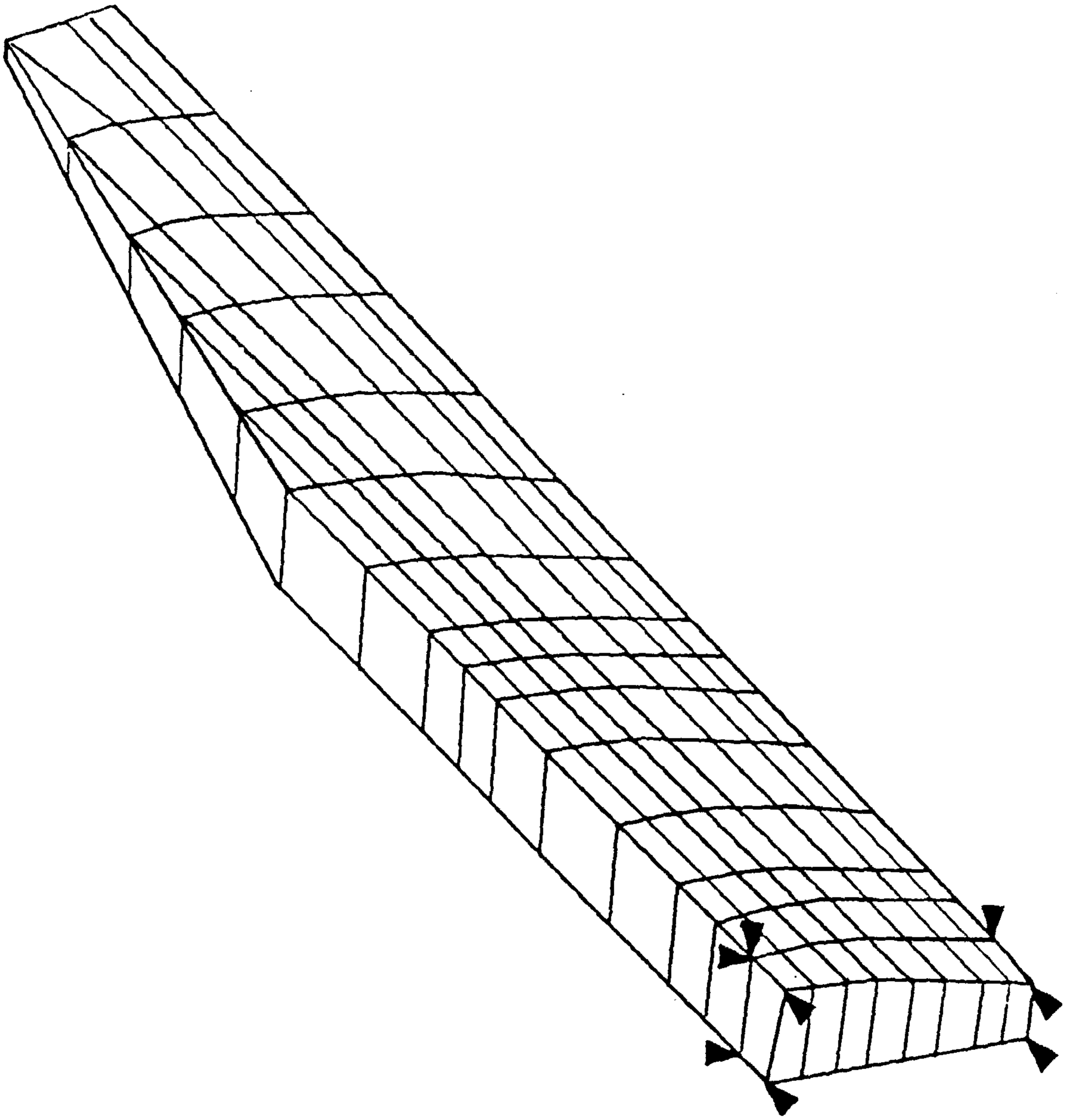


FIG. 13-3 Analysis 3, untapered with supports on rib 2, 16 ribs 8 stringers

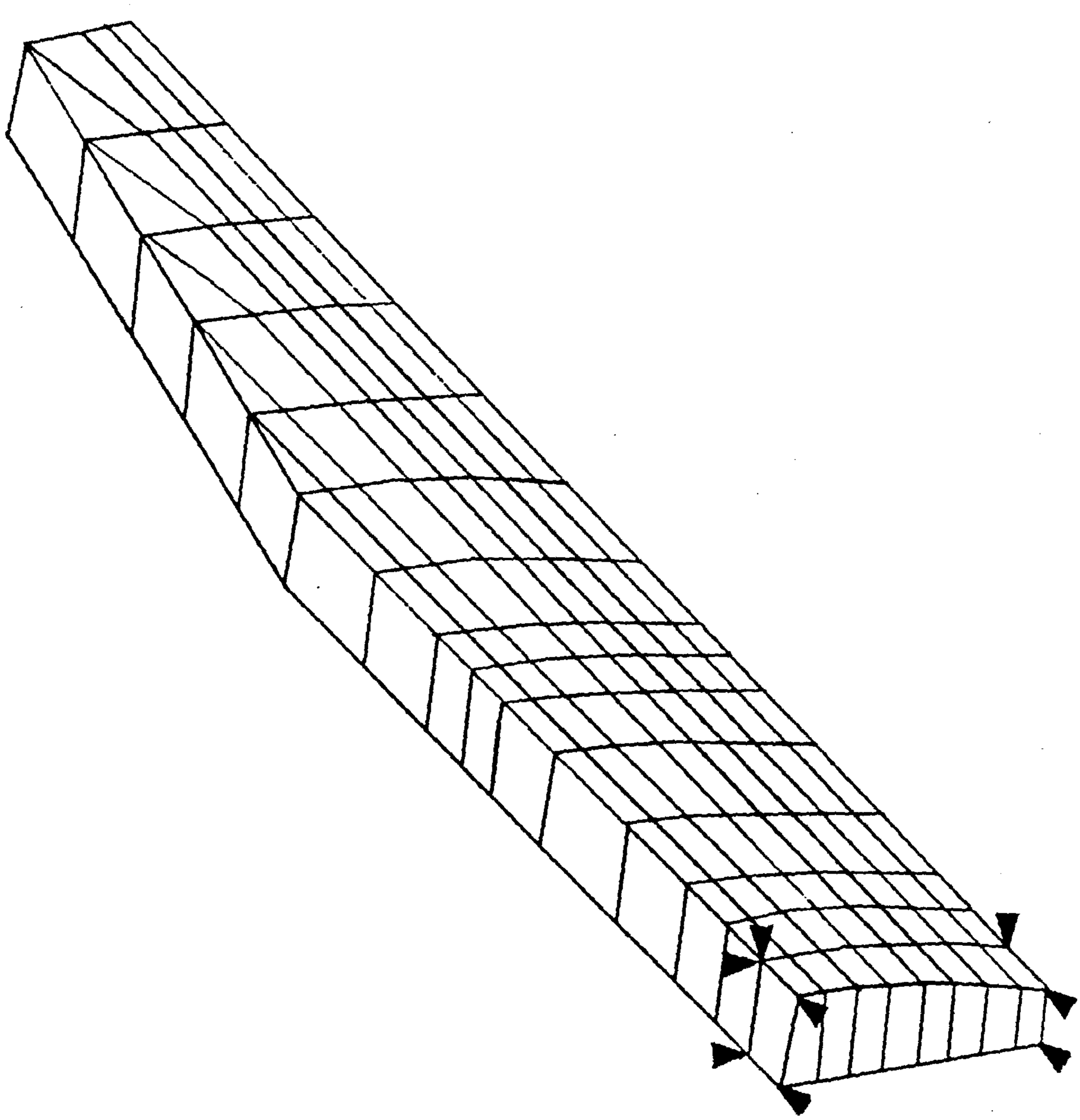
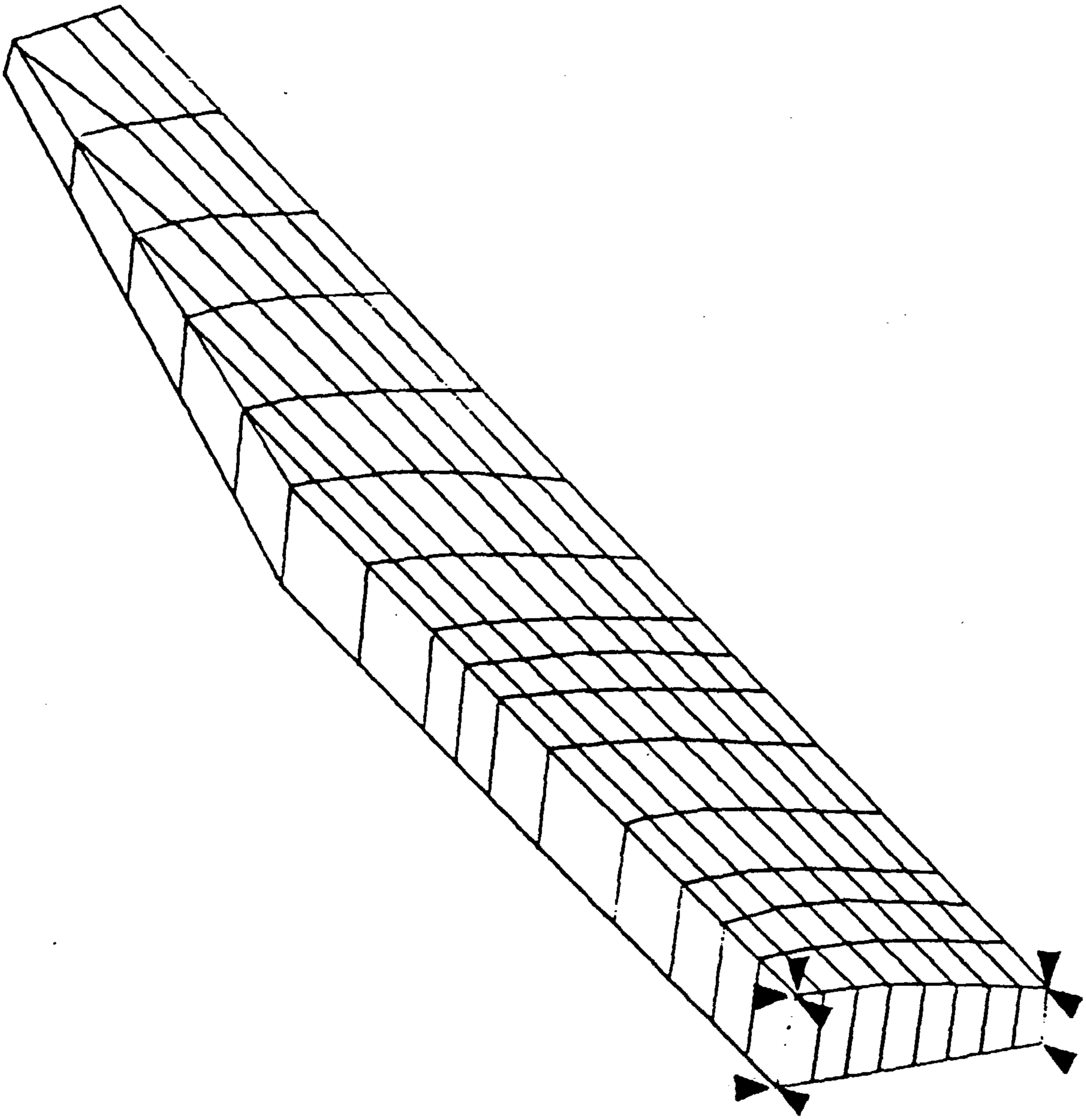


FIG. 13·2 Analysis 2, tapered with root supports, 16 ribs 8 stringers



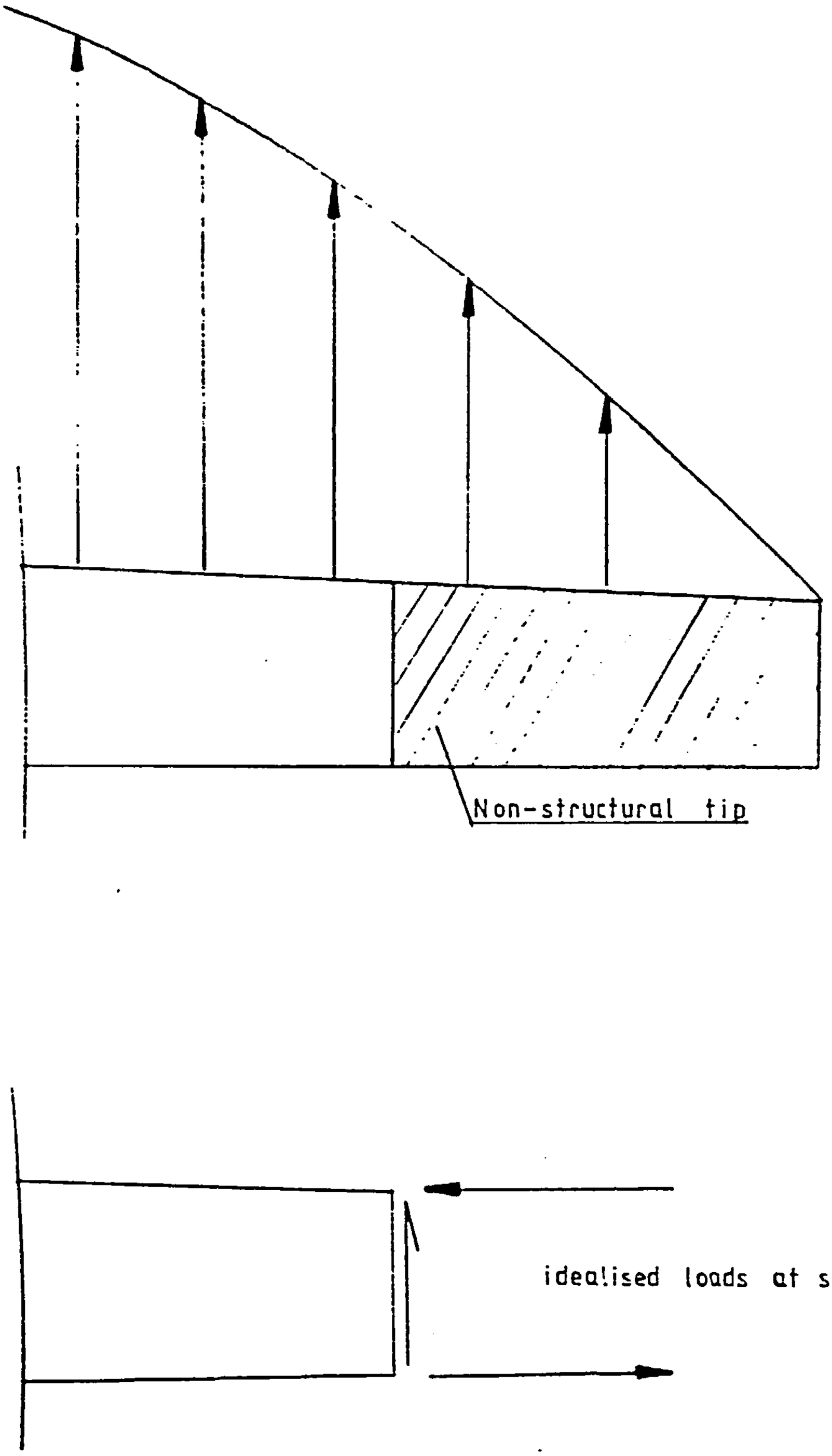


FIG. 13·9 Tip load idealisation

13.2.1.4 Tapered Box with Representative Fixations and 16 by 8 Mesh (Ref fig 13.4)

This model had the fixations shown in fig 13.11, a mesh described by 16 rib and 8 nodes along the center chord. The plan form was representative of the real box but the chord depth taper towards the tip was exagerated.

13.2.1.5 Tapered Box with Representative Fixations and 8 by 5 Mesh (Ref fig 13.5)

This model had a representative planform and fixations as shown in fig 13.11. The chord depth taper was exaggerated as in the previous case and the mesh was described by 8 ribs and five nodes along the center line chord.

The main purpose of this analysis was to investigate the effect of mesh density on the results. Clearly only half the ribs are represented.

13.2.1.6 Tapered Box with Outboard Fixations and 16 by 8 Mesh (Ref fig 13.6)

This model is the same as that described in 13.2.1.4 except for the pick-up fixations which were placed on the rib outboard of the rootrib as shown in fig 13.12.

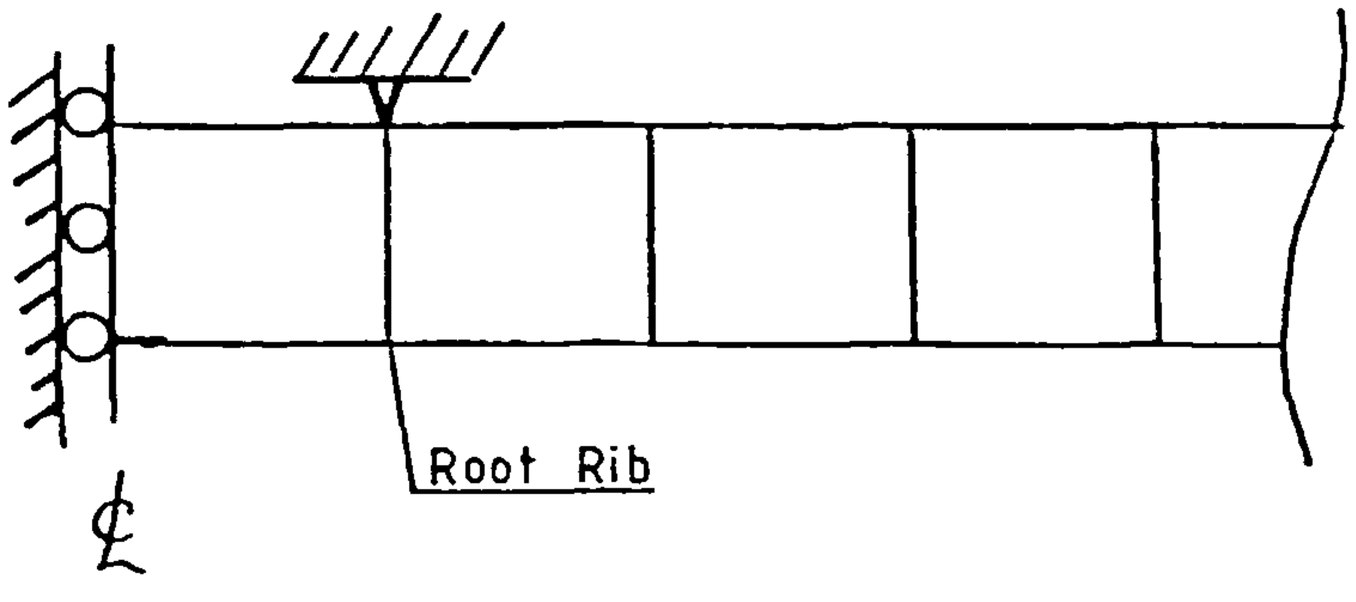


Fig 13.11 Representative Root Fixations

13.2.1.7 10 by 8 Mesh in Aluminium (Ref fig 13.7)

This model has the correct planform and chord thickness represented. This is a model of a design recently drawn up at Cranfield by W Brooks for a post-buckled C.F.R.P wing. In this case though the

loads due to the tip had to be transmitted to the tip of the box as a combination of shear loads and large differential loads to simulate the moment due to the offset of the shear as shown in fig 13.9. These large loads as we will see later lead to local problems.

13.2.1.1 Untapered Box with Center Line Fixations and 16 by 8 Mesh (Ref. fig 13.1)

This box had the correct planform for the A1 box and each rib is correctly positioned, but the chord depth was deliberately kept constant (instead of tapered) to investigate the effect of minor geometrical errors due to over simplification or lack of information on details. This type of error is most likely in the early design stages.

The fixations were also simplified by "building-in" the structure at the centre line as shown in fig 13.10. The measure of mesh density given here as "16 by 8" refers to the number of ribs (16) and the number of nodes along the length of the chord at the root (8).

13.2.1.2 Tapered Box with Centerline Fixations and 16 by 8 Mesh (Ref. fig 13.2)

This box is similar to the one described earlier except the chord depth is correctly modelled.

13.2.1.3 Untapered Box with Representative Fixations and 16 by 8 Mesh (Ref fig 13.3)

This box differs from the first one only in the nodal fixations. Here there is symmetry at the center line but vertical and drag fixations are at the root rib position representing the actual fixations as shown in fig 13.11.

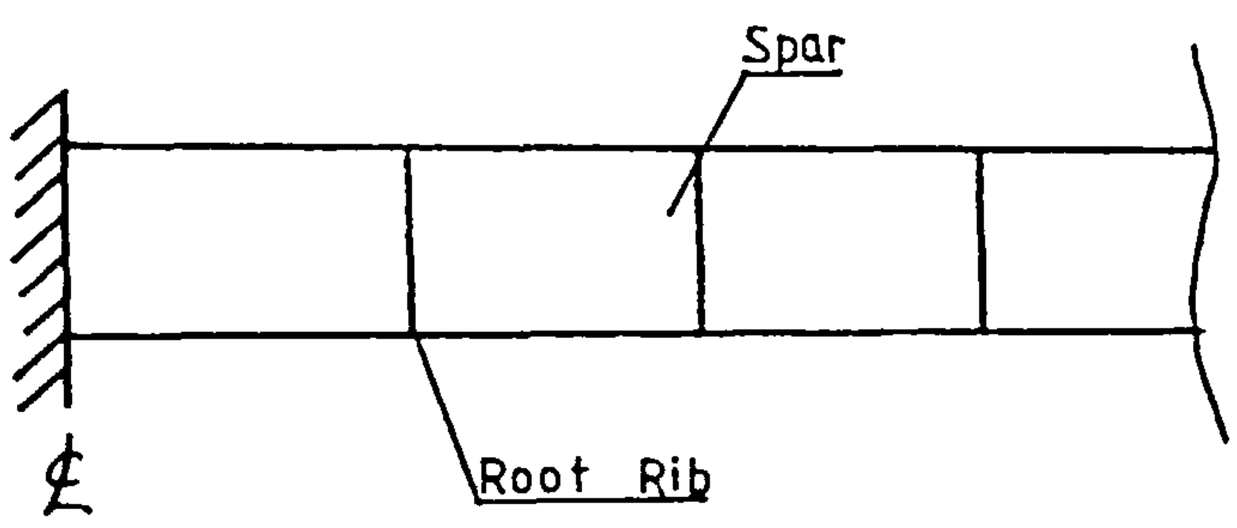


Fig 13.10 Built in Root Fixations

material is aluminium. There are 10 ribs and nine nodes along the center chord.

13.2.1.8 10 by 8 Mesh in C.F.R.P (Ref fig 13.8)

This is the same model of the wing designed by W. Brooks with the C.F.R.P material modelled. A simplification was made in that the skin layups were assumed to have constant orthotropic properties with varying thicknesses.

13.2.1.9 Variation of Stringer Areas

Further analyses were carried out to investigate the effect of design variable constraints (see table 2 and 13.5)

13.2.2 The Design Process and Data

A simple and expedient design process was used for this analysis. Due to the simplicity of the wing (the lack of complicating non-structural components found in say an airliner wing) this procedure gave solutions of sufficient accuracy and a more involved process would have been a waste of time. A flow diagram of the process is given in fig 13.13.

The maximum number of optimising iterations was set at 10 since in practice convergence was usually reached after 5 iterations, more than that indicated an error. The procedure was interrupted every 30 minutes C.P.U. time and continued if necessary. The Stanton Jones aerodynamic analysis was carried out with 50 reference points.

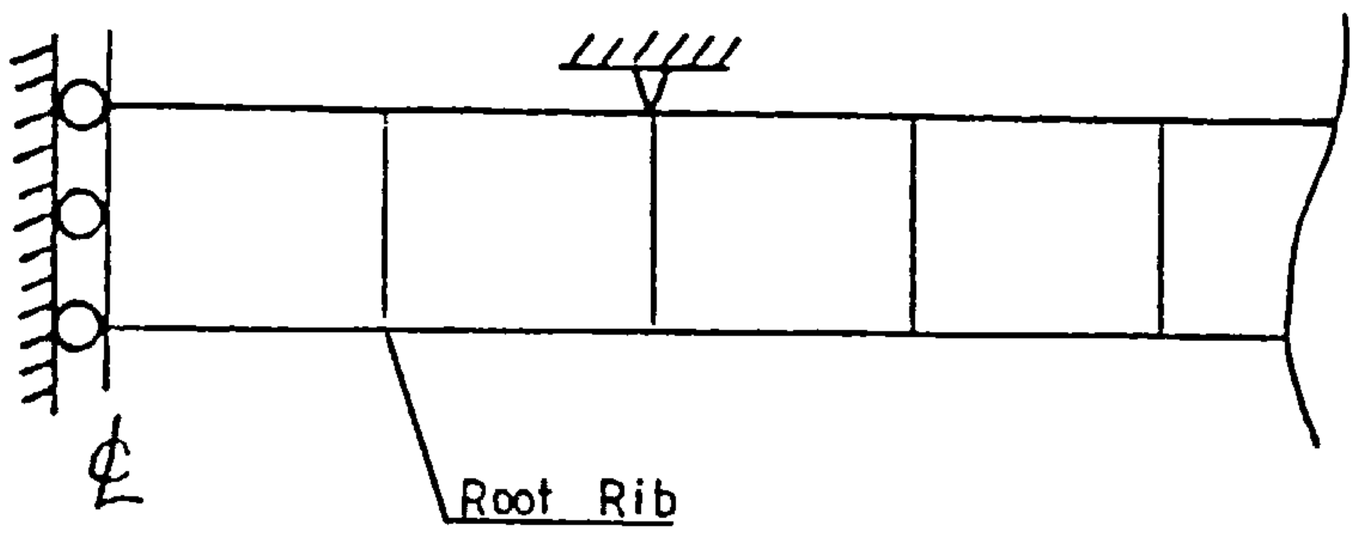


Fig 13.12 Outboard Fixations

The following design prameters and data were used as input.

Material Data;

Youngs Modulus = 70000 N/mm^2
Density = 2.72×10^{-6} kg/mm^3
Strength = 247 N/mm^2

Geometric Data;

Quarter chord sweep	= 10 degrees
Root chord	= 2070 mm
Tip chord	= 900 mm
Design Mach number	= 0.36
All up weight	= 4000 N
Design vertical acceleration	= 9g
Ultimate Factor	= 1.5
Fuselage overlap	= 450 mm

Root Rib Geometry Data/mm;

Top		Bottom	
Y	Z	Y	Z
812	-53	812	105
707	-76	714	106
604	-110	612	108
501	-123	510	108
400	-144	405	109
297	-156	306	111
194	-167	205	114
92	-176	103	117
0	-183	0	117

Y = positive aft
Z = positive down

Analysis Number	Description
1	16x8, untapered,root fixations only
2	16x8, tapered,root fixations only
3	16x8, untapered, pickup rib2
4	16x8,overtapered,pickup on rib2
5	8x5,overtapered,pickup on rib2
6	16x8,overtapered,pickup on rib3
7	10x8, tapered,pickup on rib2
8	10x8,tapered,pickup on rib2, C.F.R.P
9	16x8,overtapered,pickup on rib2 stringer area 50m^2
10	10x8,tapered,pickup on rib2 stringer area 150
11	8x5,overtapered,pickup on rib2 stringer area 150
12	8x5,overtapered,pickup on rib2 stringer area 200
13	(12) + max skin=1.5 min skin = 0.5
14	(12) + max stringer = 200 min stringer = 50
15	(12) + Outer 3 panels (combine top & bottom C,I,O)
16	(12) with different load conversation
17	8x5,tapered,pickup on rib2 stringer 200m^2

Table 13.2 Summary of Analyses

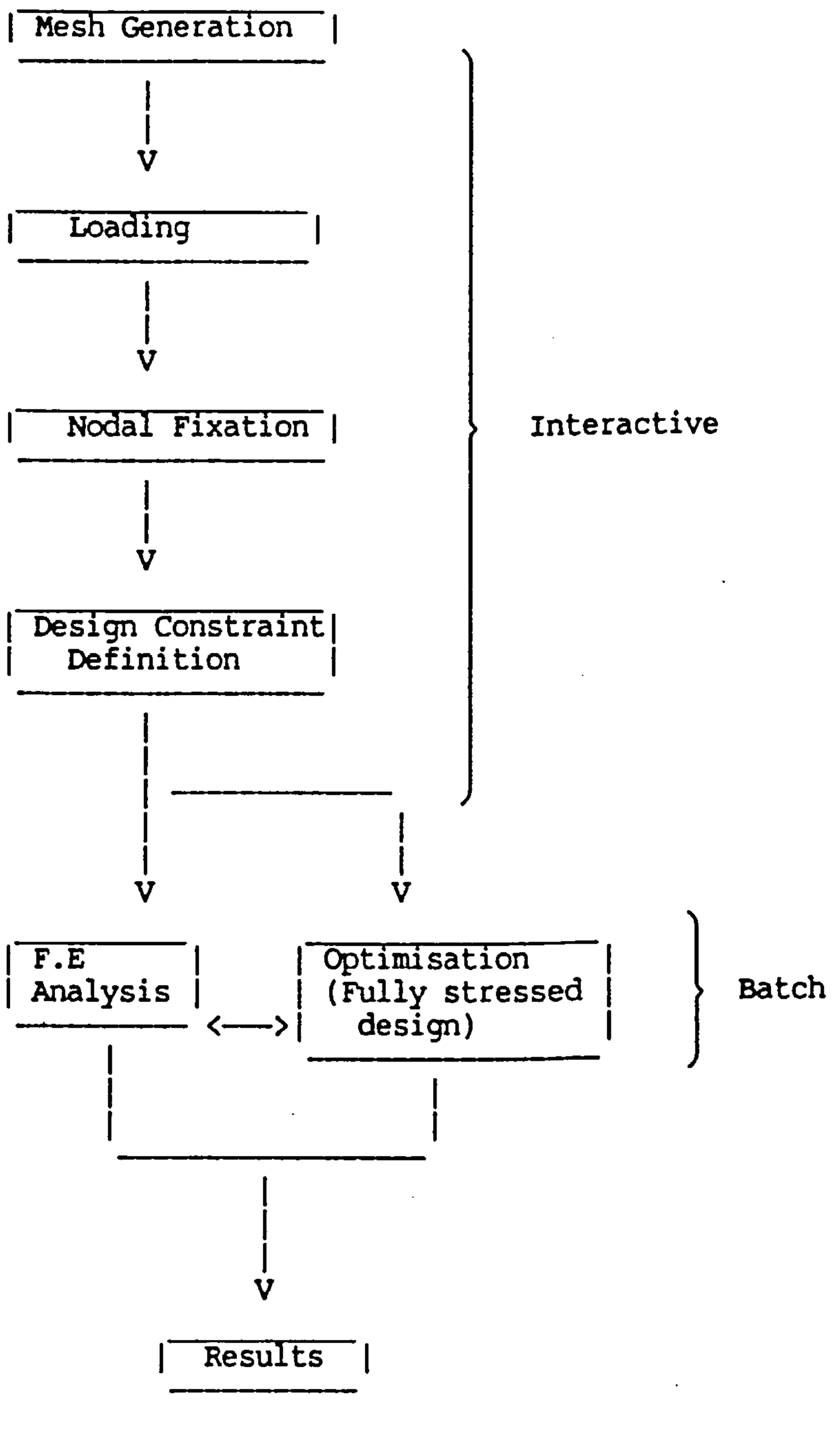


Fig 13.13 Design Process

13.2.3 Speed of the Computation

It is worth noting that the machine used was a VAX 11/750 mini computer which under multi-user condition, prevailing at the time of the test, is "slower" than most modern micros.

Bearing this in mind it took an inexperienced user (in this case a person who was knowledgable about F.E. techniques but to whom the WEIGHTS package was completely strange) one hour of interactive terminal time to input all the required data and make corrections to the inevitable mistakes. This time reduced with familiarity.

The computer then took a further hour of batch (background) processing time to complete the task. All eight analyses were completed in one working day.

This could be greatly speeded up by use of a main frame computer.

Typically, four iterations were needed to converge to final design.

13.2.4 Variable Constraints.

The type of analyses carried out is in fact extremely limited. The reason for this was to reduce the possibilty of eccentric behaviour of the type discussed in earlier chapters. In this case the major restrictions were;

a) During the fully-stressing procedure the stringer/boom areas were held constant. (The fixed areas were varied over a range for each analyses to investigate this restriction).

b) Rather then varying each elements geometric properties individually, properties of groups of elements were varied. Top skins, bottom skins, stringers, ribs and spars were grouped in bays. For example if the stress in an element in the top skin in bay 5 exceeded the allowable stress all the top skin elements in that bay were increased, even though the other elements might have had lower stresses.

c) Element properties were restricted between a range of 0.2mm to 5mm. This effectively (in the context of a light aircraft) is an unconstrained problem. In practice one would expect gauges to be between 24 and 18 gauge. The reason for this were discussed in earlier chapters.

13.3 Results

The results may be broken down into several points, dependant on the purpose of each investigation, these are broadly speaking;

a) General accuracy.

b) The effect of the finite element mesh density.

c) The effect of variable constraints.

d) The effect of loading.

e) The effect of geometric discrepancies.

f) The effect of fixation constraints.

g) Breakdown accuracy.

The discussion that follows will consider each of these in turn.

13.3.1 General Accuracy

A quick look at the overall weight results (table 3) shows they are, in general quite good. The accuracy appears to deteriorate with decrease in mesh density with an accuracy of around 2% for high mesh densities deteriorating to + 32% - 10% at low mesh densities. This deterioration is not monotonic however which points to something more involved than merely mesh density as we will see later.

In general table 13.3 shows that the weight breakdowns for high mesh densities are also good.

A1 Wing Example

Aluminium Wing

ANALYSIS	SKINS	RIBS	REAR SPAR	FRNT SPAR	STRINGERS	TOTAL
A1 actual	13.66	3.81	2.20	3.122	20.028	43
(1)	16.00	2.08	1.03	2.01	21.06	43.09
(2)						43.08
(3)	15.03	2.09	1.03	2.01	21.06	43.02
(4)	17.01	1.72	1.01	2.00	21.06	43.05
(5)	37.00	3.16	2.02	2.06	11.09	56.09
(6)	16.02	1.81	1.01	1.07	21.06	42.05
(7)	16.02	1.82	1.07	1.06	18.01	38.8

ANALYSIS	SKINS	RIBS	REAR SPAR	FRNT SPAR	STRINGERS	TOTAL
ACTUAL	20.91	3.2	1.02	1.05		26.8
(8)	11.07	1.21	1.01	1.00	13.55	27.8

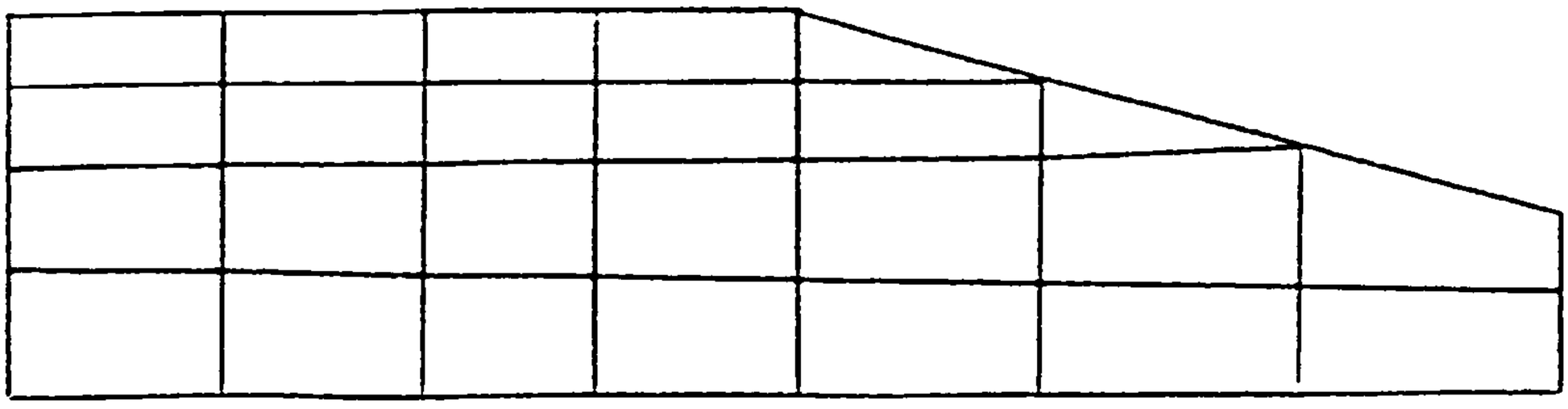
note: stringer mass lumped with skins in VFRP test box.

TABLE 13.3

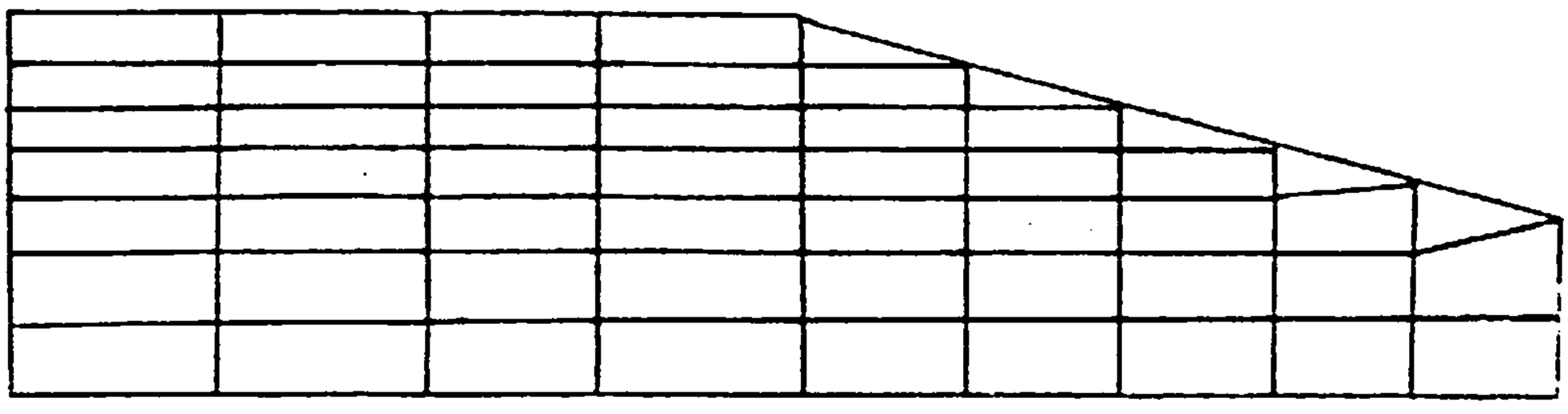
The Effect of the Finite Element Mesh and Geometry

13.3.2 The Effect of Finite Element Mesh Density

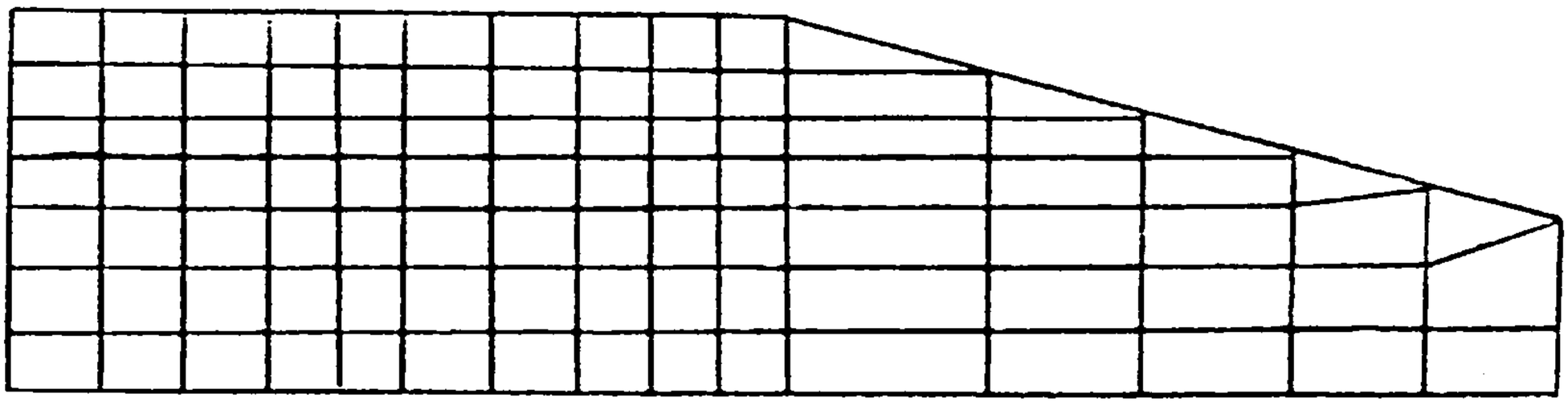
The basic meshes shown in fig 13.14 were used



8 x 5 mesh



10 x 8 mesh



16 x 8 mesh

Fig 13.14 mesh densities investigated

The results in table 13.3 appear to be pretty damning for course finite element grids. Closer inspection shows however that mesh density is not the real culprit for the low accuracies achieved by course meshes.

The main cause of the large variations in weight predictions are due to the skin and stringer weights. (The spar and rib weights are so small as to have little real effect on the final result).

Since the stringer/boom areas are held constant (in the case of table 13.3 they are held at 100mm^2) the fewer stringers there are the less their total weight. In which case the skin areas should increase to compensate for the reduction in moment carrying area. This appears to happen in analysis 5 in table 3 of the 8 x 5 mesh in comparison with the 16 x 8 meshes. But in the case of analysis (7) of the 10 x 8 mesh the skin weights have not increased whilst the stringer weights have decreased. This means that the total moment carrying material has decreased.

Something is amiss. A closer inspection of detailed results provided in the form of itemized weights gives the first clue, when we notice that there are "active" elements in the skin panels in the outer-most bay. This means that these elements have stresses which exceed the allowable value. Inspection of the final design shows that the skin panels at the tip have reached their maximum thickness constraint of 5mm in all the meshes.

This means that density is not, directly the reason for this variation in accuracy. In fact there is evidence to suggest we could get as good results with the 8 x 5 mesh as with the 16 x 8 mesh. What is that?

Well consider the 16 x 8 at 10 x 8 meshes shown in fig 13.4 and notice that the only difference between these two is the mesh density in the untapered part of the wing. In the tapered portion of the wing the meshes are identical.

The 8 x 5 mesh however differs considerably throughout but notice how much bigger the elements are at the tip bay compared with the finer meshes.

So we know that there is some local effect at the tip which causes the skin at the tip to be built up. The effect gets out of hand when the elements within the scope of this local effect are large. The finer grid meshes tend to contain this effect in a smaller region reducing erratic weight prediction.

The expected effect on the total skin area, of reducing the total stringer area does not happen in this case because (as closer scrutinity of the detailed results show) the stringers are understressed. More of this later. Appendix C shows a sample of the full output data.

One problem noticed with the course mesh is the tendancy for the structure to become heavier because the loads are being applied on fewer nodes. This leads to higher local loads and consequently higher local stresses. But since the elements are linked in large groups, large regions of the structure become over designed.

13.3.3 The Effect of Loading

This is the culprit causing that local effect leading to very thick elements at the wing tip. The reason is the rather rough assumptions we made about the way in which the aerodynamic loads are distributed to the nodes in the structure.

The way we do this was discussed in earlier chapters but the full implications were clarified. Consider a load distribution shown in fig 13.15 calculated using the Stanton-Jones method.

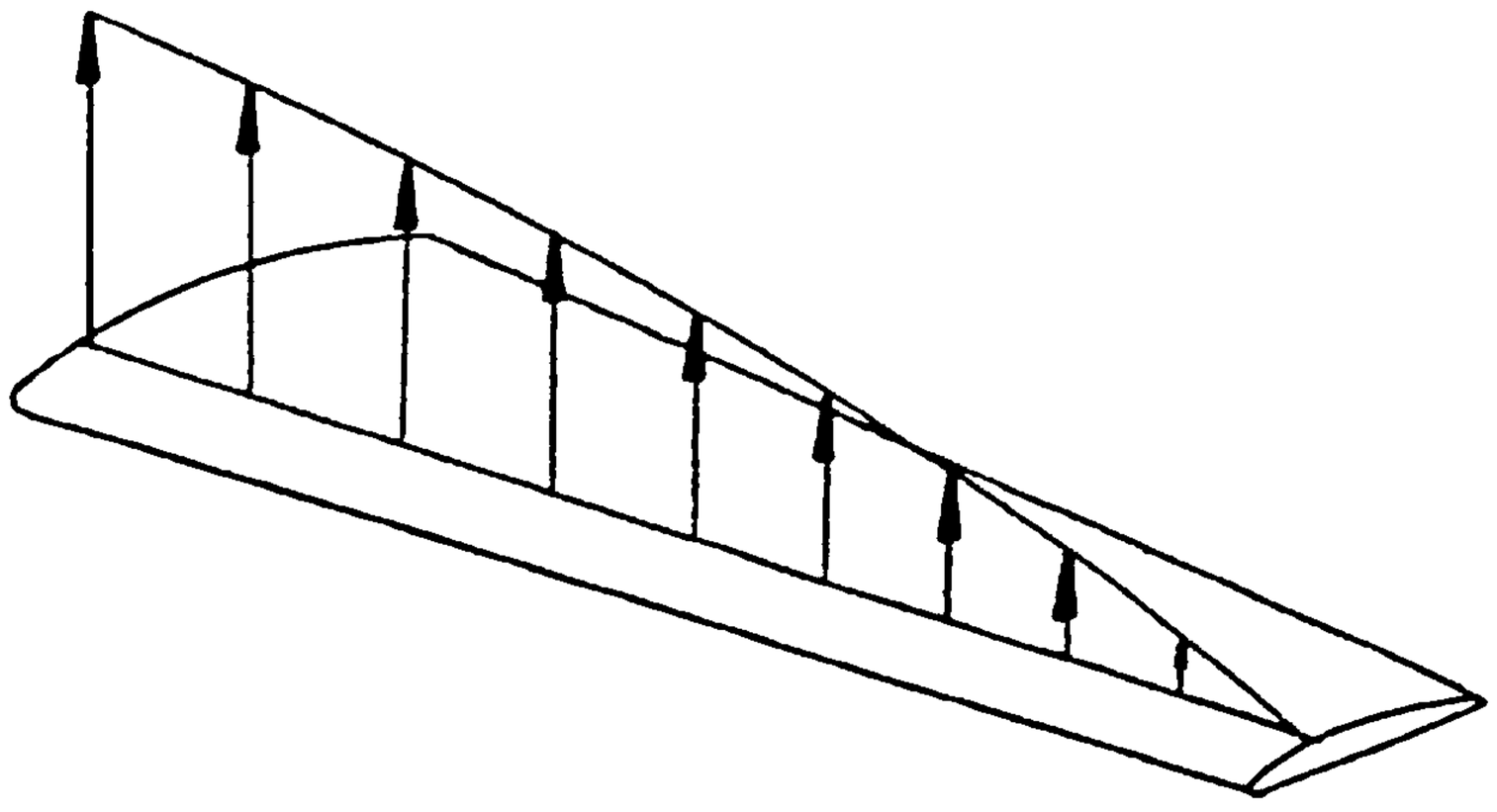


Fig 13.15 Load Distribution

Now consider two wing bays as shown in fig 13.16.

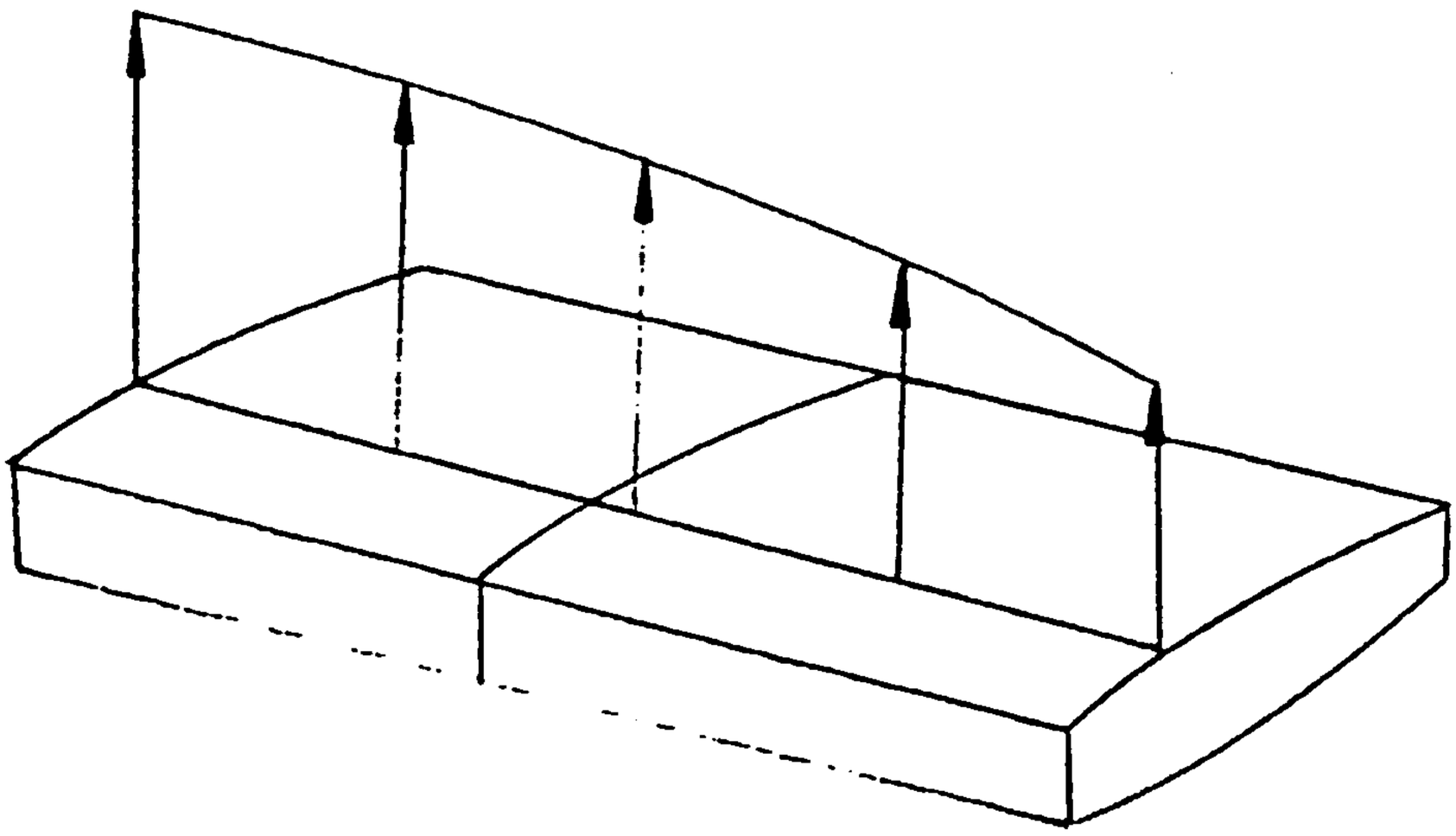


Fig 13.16 Two Rib Bays and Aerodynamic Load

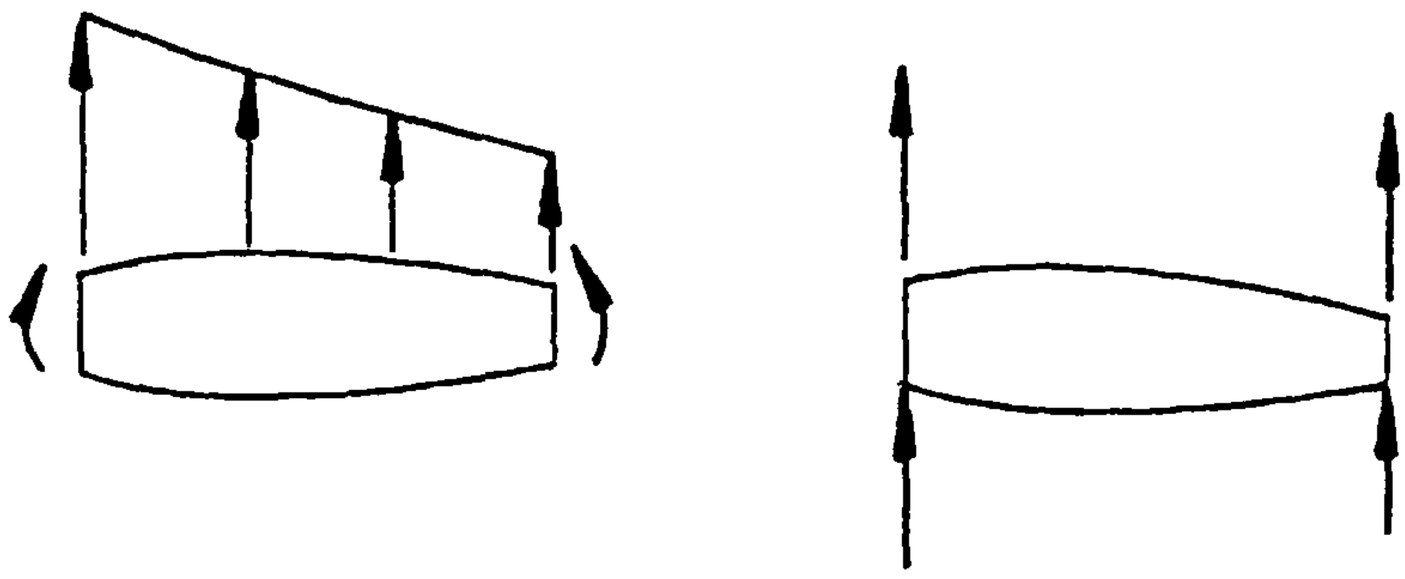


Fig 13.22 Real Situtation Idealised

This leaves the ribs with only the brazier and shear loads due to torsion to resist. This problem can be allieviated by specifying that; concentrated loads should not exceed more than say, 10 times the major shear loads by carefully redistributing the loads. This has been done in analysis 16 shown in table 13.7, here the loads have been moved inboard a little and consequently so have the restoring moments. Compare these results with the "control" experiment analysis 12.

13.3.4 The Effect of Geometric Descrepency

The results to look at are again in table 13.3. In particular analyses 1,2,3,4, and 6. The differences in geometry here were deliberate "mistakes" in the definition of the taper of the wing, with errors of $\pm$ 50%. The effect on the results was minimal however. The differences in results have been attributed to differences in fixation constraints discussed later.

The insensitivity to geometrical error appears to be due to two opposing effects. As the box section increases in area there is a tendancy for the weight to decrease due to the greater effective depth of the box and so less material is needed to withstand the bending moments. But this decrease is arrested when the material thickness reaches the minimum gauge allowed, after which there is a tendancy for the weight to increase due to increase in size of the box.

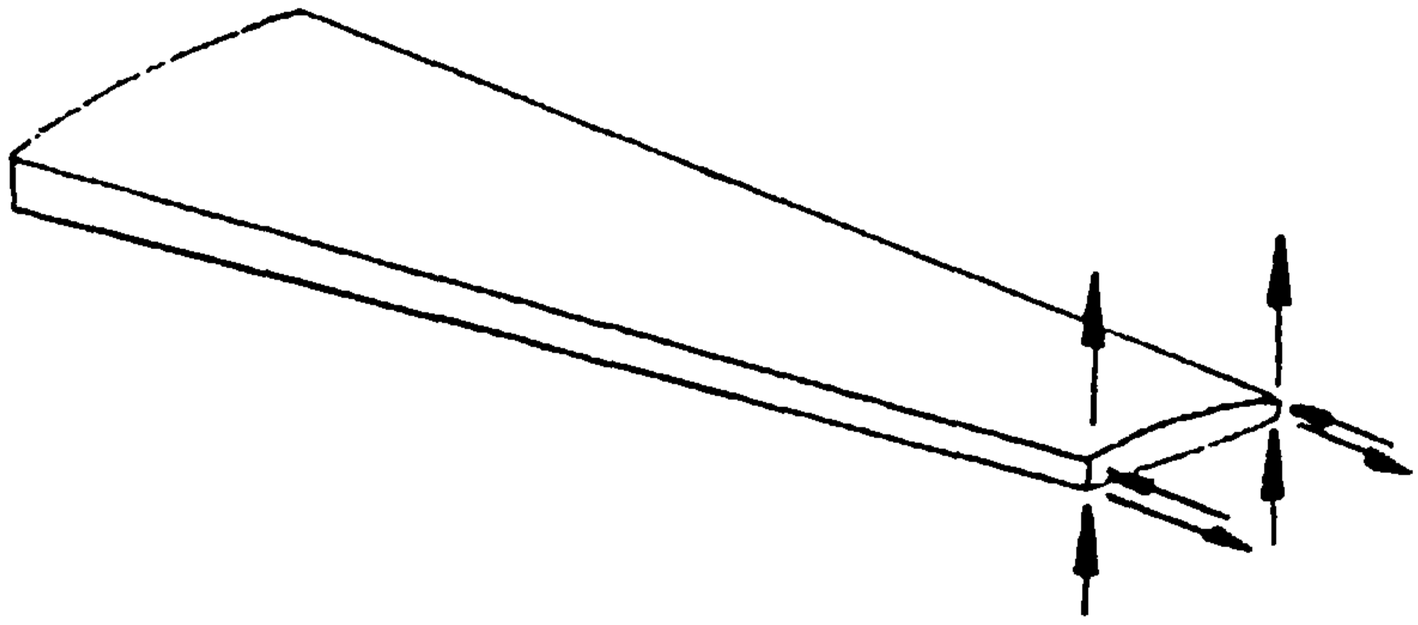


Fig 13.21 Major Equivalent Loads at Wing Tip.

In the case of the tip rib the loading is lopsided resulting in high restoring moments on the tip as shown in fig 13.21.

This leads to highly concentrated loading and consequent high stresses in the tip region, which the optimising procedures tries to alleviate by increasing the thickness of the skins in this region.

The problem here is one of conversion of loading data from the aerodynamic model to the structural one.

How this transformation can be improved is not altogether clear. (see conclusions and recommendations).

Another way around this problem is to place greater restrictions on the range of skin thickness, more on this later.

Another look at table 13.3 shows that the rib weights are consistantly underestimated. This is because, by placing the loads on the front and rear spars we are bypassing the ribs. i.e. one major task the rib carries out is to collect the shears from the skins and transfer them to the spars as shown in fig 13.22.

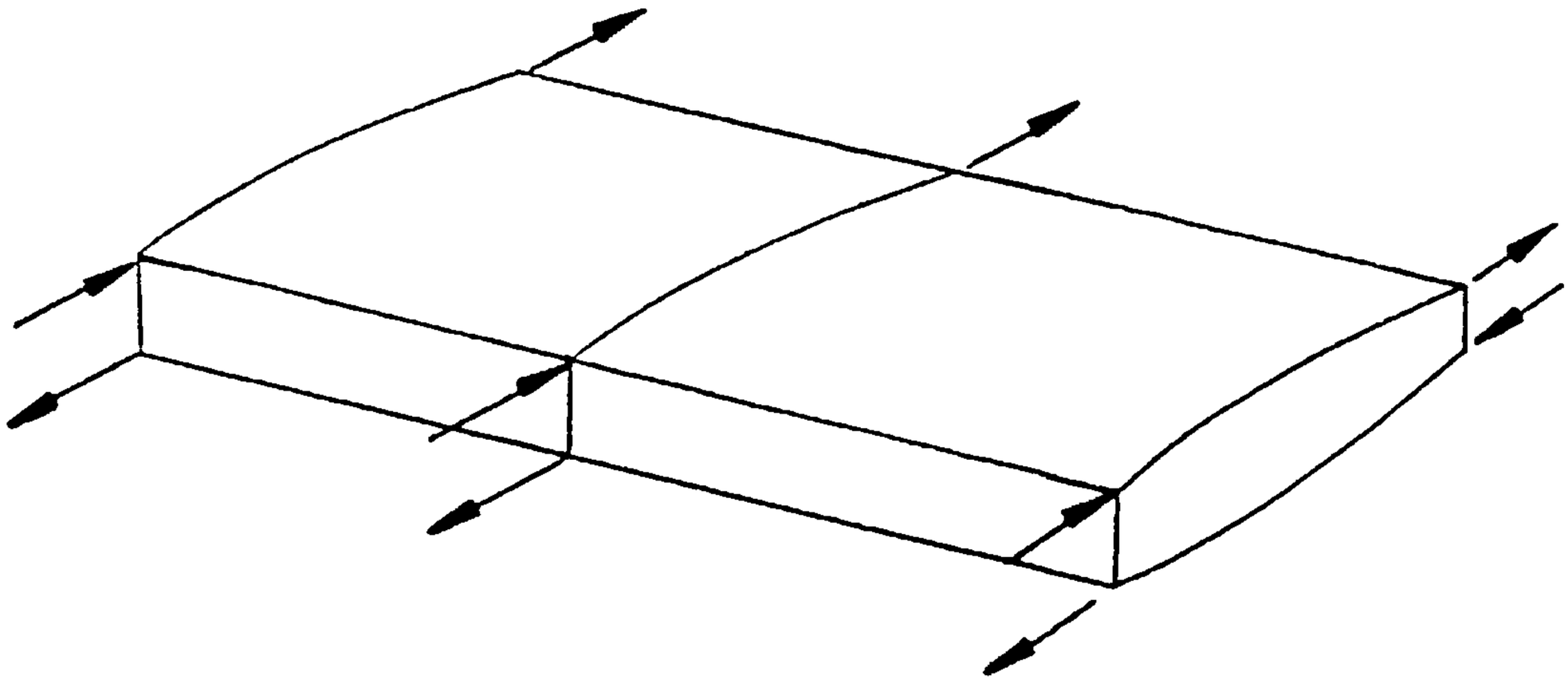


Fig 13.19 Two Rib Bays and Typical Torsional Loads

Fig 13.18 shows the typical balancing moment loads. These are there to ensure that given the major shear loads in fig 13.17 the resultant bending moments are the same as those due to the original load distribution in fig 13.16. Fig 13.19 shows the typical torsion loads.

The situation becomes unbalanced at the wing tip as shown in fig 13.20.

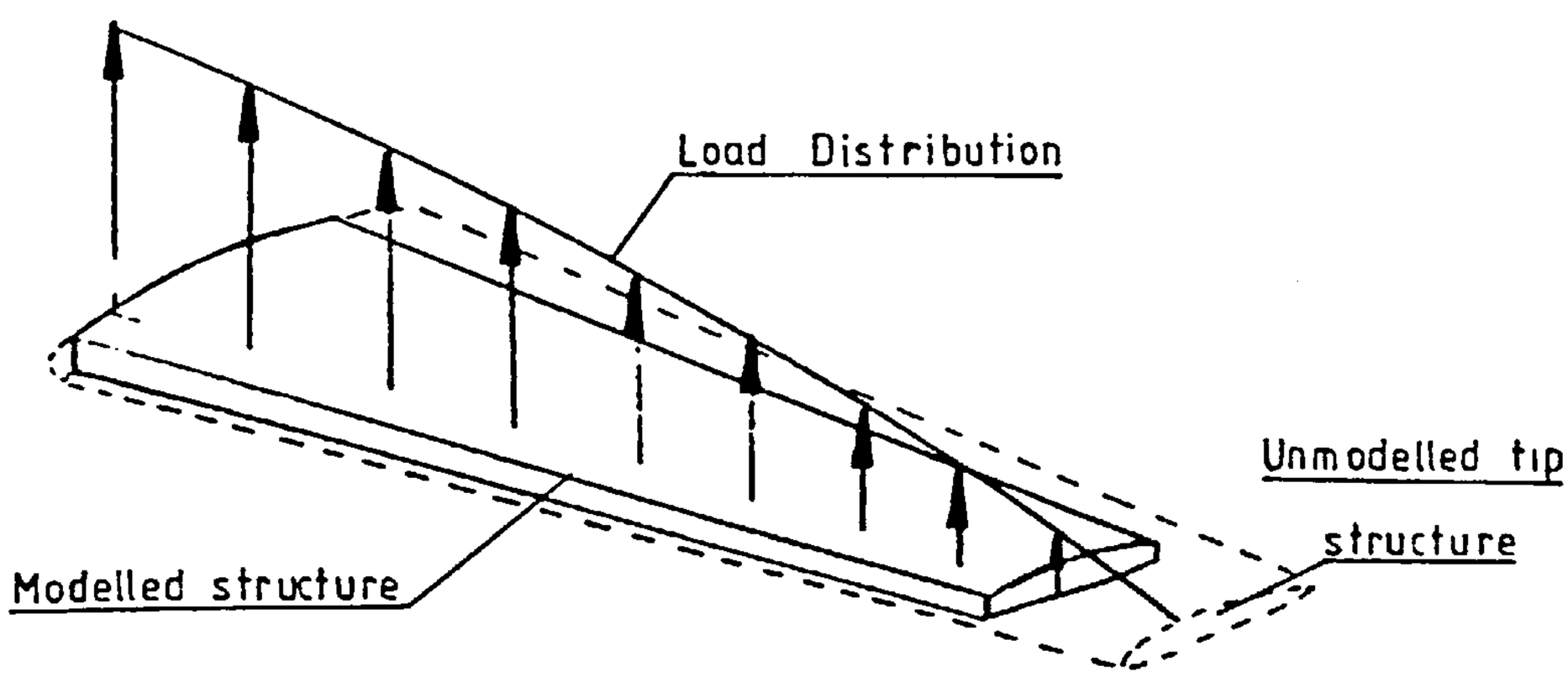


Fig 13.20. Aerodynamic loads at wing tip.

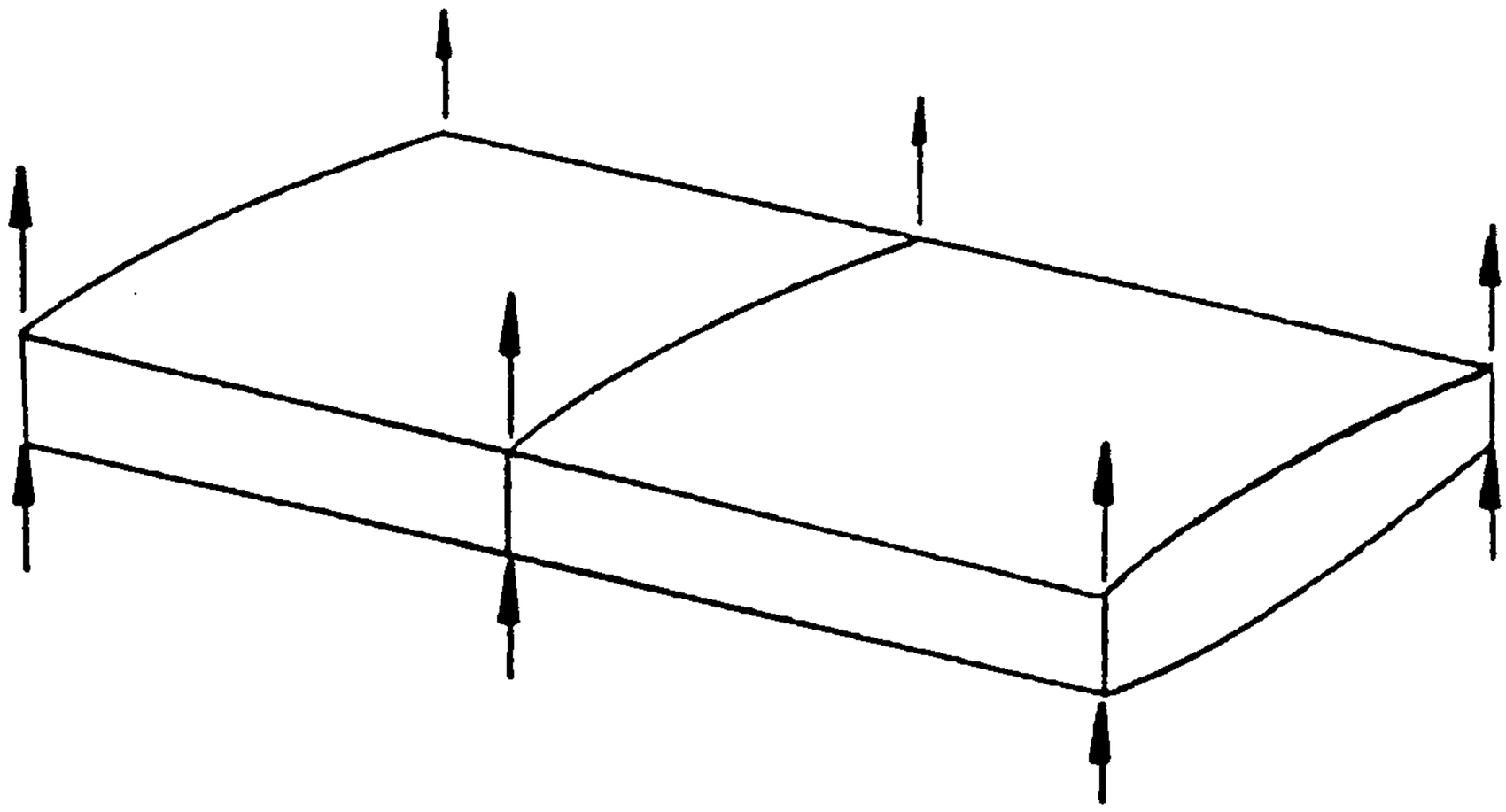


Fig 13.17 Two Rib Bays and Typical Equivalent Major Loads

The aerodynamic load distribution shown in fig 13.16 is converted to loads on nodes on the front and rear spar. Fig 13.17 shows the major equivalent loads acting on a typical rib.

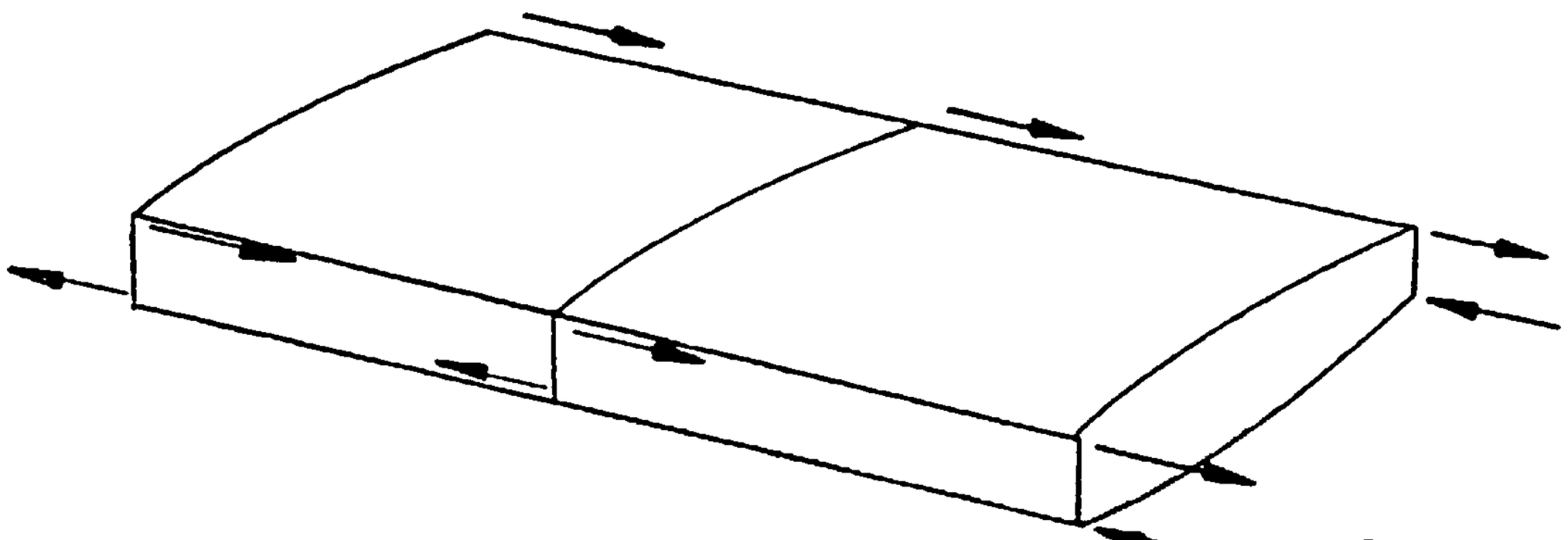


Fig 13.18 Two Rib Bays and Typical Balancing Bending Moments

There is one noticeable effect of geometry hidden in the results. Looking at the raw loading data it was noticed that the concentrated differential loads which represent torsional loads become very large if the geometry is underestimated. For example if the depth of the box is underestimated by a factor of two the loads go up by two. This leads to high local stresses and over design of the entire "linked" region. This has been demonstrated in analysis 17 in table 7 where the 17 correctly represented compared with analysis 12 where the box has an exagerated taper.

13.3.5 The Effect of Fixation Constraints

Once more we refer to table 3 and direct our attention to analyses 1, and 3 or 4 and 6. The main difference in fixations between the analyses are shown in fig 13.23.

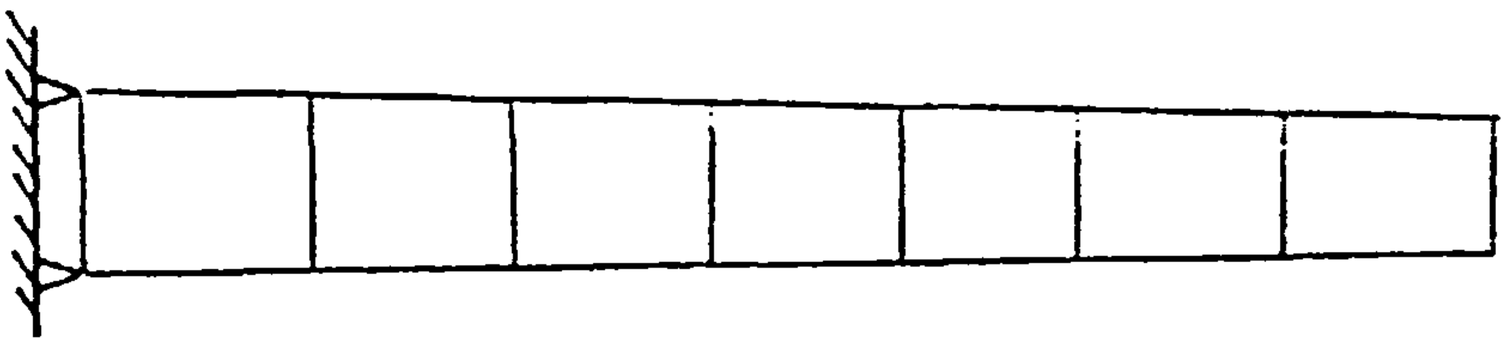


Fig 13.23 (a) Analysis 1 built in at root.

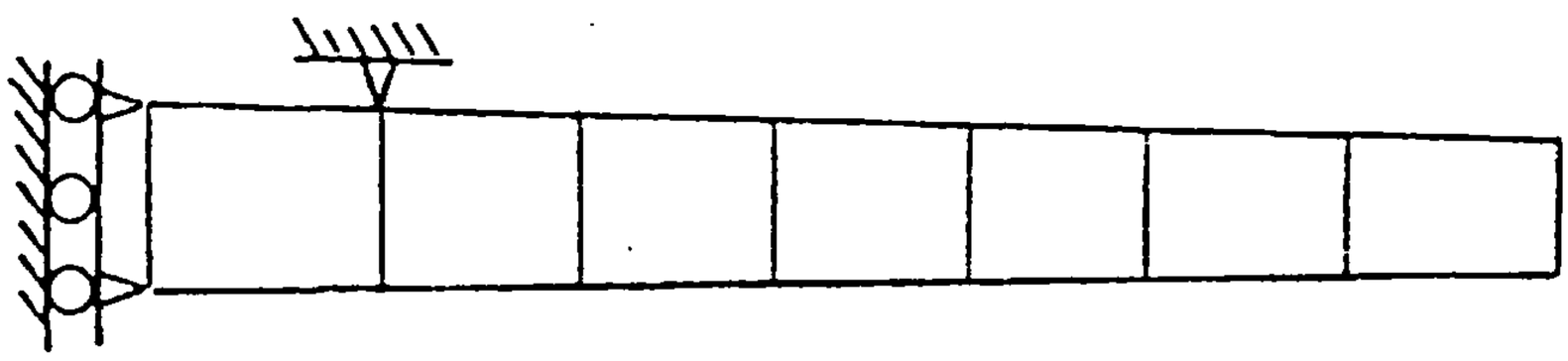


Fig 13.23 (b) Analysis 3 and 4 symmetry at root and pickup at 2nd rib.

More details are also shown in the post analysis drawings of the finite element grids in figs 13.24 to 13.27.

Intuitively one would expect the layout in fig 13.23 (b) to be lighter than the one in fig 13.23 (a) which is built in at the root. A look at the results showing the detailed weight breakdown shows that the layout in fig 13.23 (a) is heavier at the root.

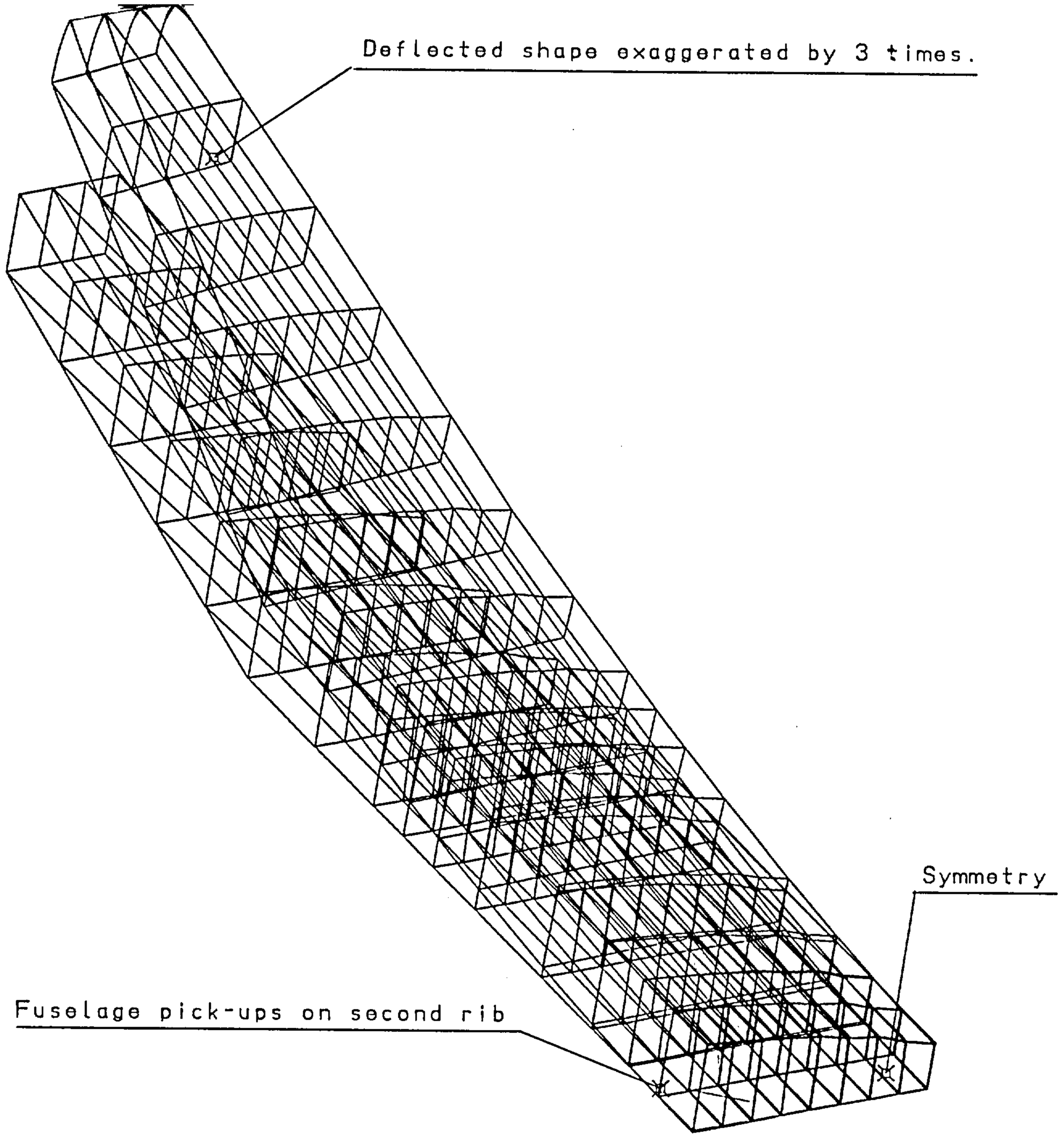
Table 13.4 shows an extract from the results. They show that our intuition is borne out.

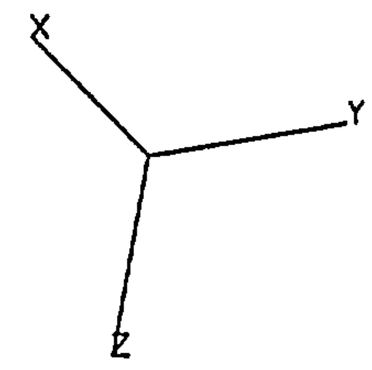
Top Skin Bay	Weight Analysis 1/kg	Weight Analysis 3/kg
1	.6976	.5470
2	.5757	.5217
3	.5458	.5567
4	.6597	.6573
5	.7765	.7719
6	.4438	.4416
7	.2925	.2915
8	.2599	.2594
9	.4632	.4624
10	.4723	.4717
11	.3256	.3257
12	.2452	.2456
13	.1684	.1689
14	.2588	.2555
15	1.8443	1.8402

Table 13.4 Extract from Detailed results Comparing Analysis 1 and 3

Top Skin Bay	Weight Analysis 4/kg	Weight Analysis 6/kg
1	.5496	.4428
2	.5244	.4693
3	.5603	.5234
4	.6515	.6553
5	.7636	.7760
6	.4361	.4381
7	.2872	.2878
8	.2551	.2566
9	.4577	.4559
10	.4308	.4302
11	.3094	.3110
12	.2589	.2614
13	.1956	.2143
14	.3797	.3255
15	2.575	2.413

Table 13.5 Extract from Detailed Results Comparing Analysis 4 and 6.





-117-

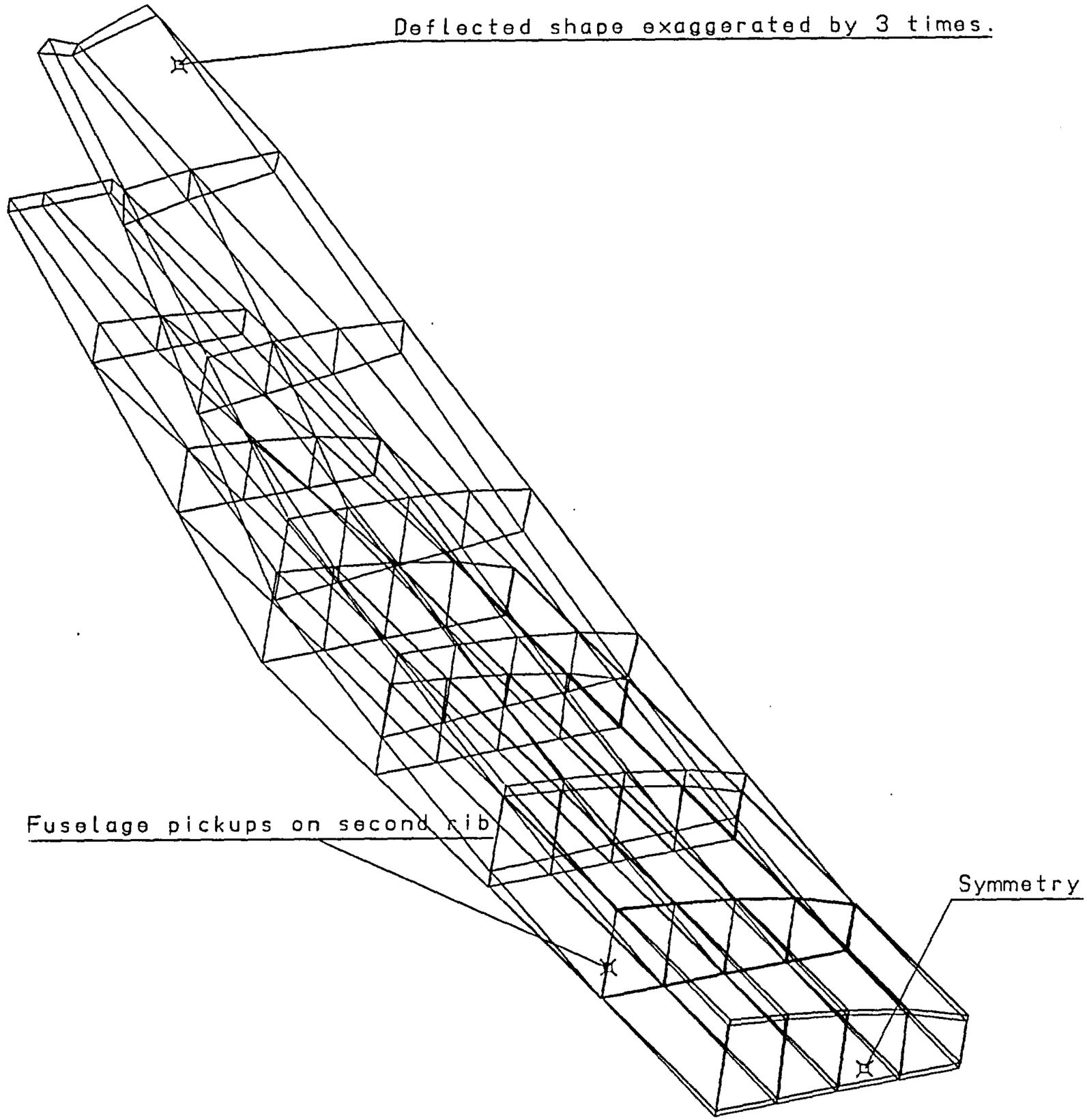
FIG. 13.24

WEIGHTS
A1 wing analysis

3

DATE: 19- 1-86 SCALE 1:15.70

Deflected shape exaggerated by 3 times.



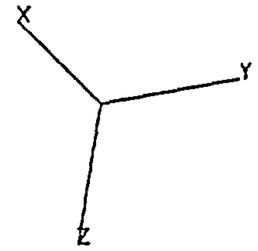
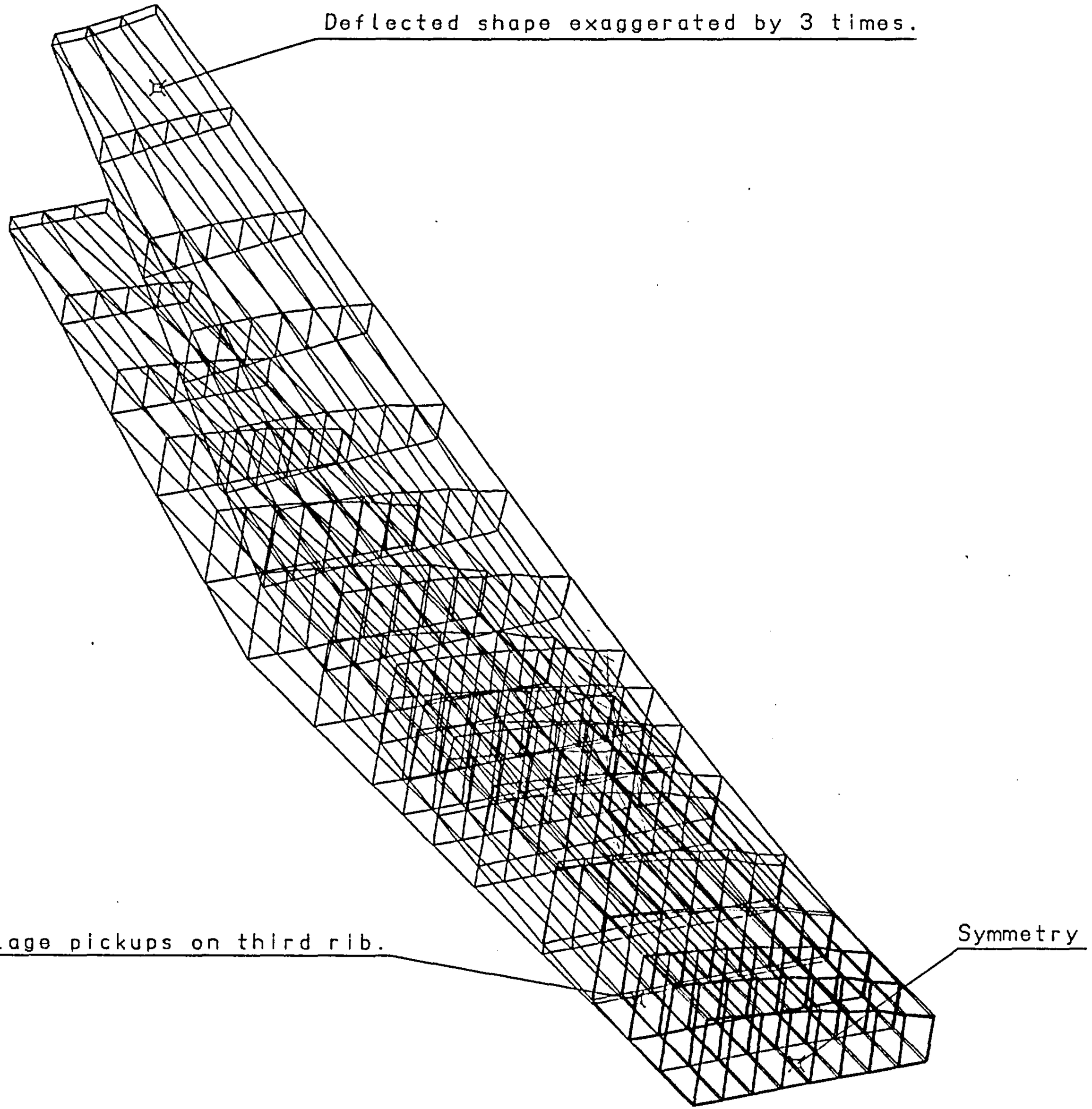


FIG. 13·25

WEIGHTS
A1-wing
analysis 5

DATE: 19- 1-86 SCALE 1:14.71

Deflected shape exaggerated by 3 times.



Fuselage pickups on third rib.

Symmetry

LUSAS

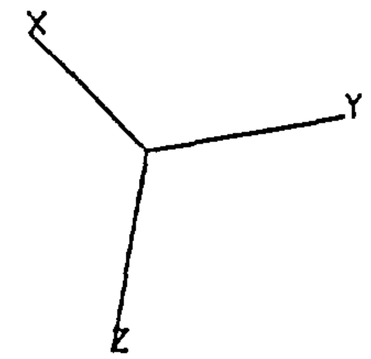
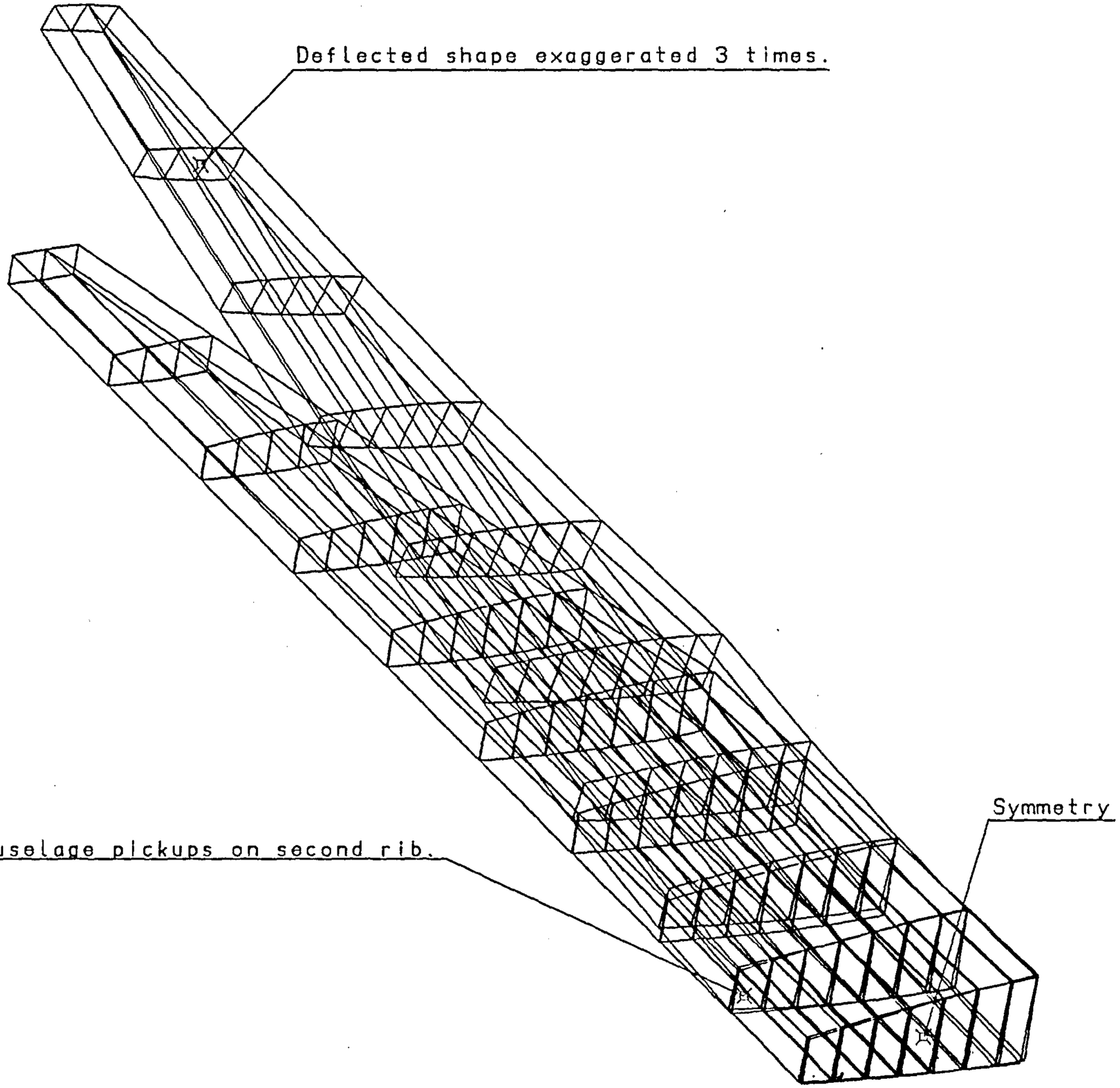


FIG.13·26

WEIGHTS
A1-wing
analysis 6

DATE: 19- 1-86 SCALE 1:15.28



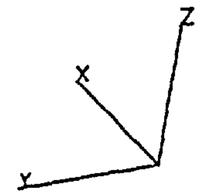


FIG. 13·27

WEIGHTS
A1-wing
analysis 7

DATE: 19- 1-86 SCALE 1:15.38

13.3.6 The Effect of Variable Constraints

Under this heading we have three items to consider

(a) What is the effect of holding the boom areas constant instead of ratioing them at the same time as the skins?

(b) What is the effect at grouping the thicknesses of several elements together to represent panels etc?

(c) What effect has the range of allowed skin thicknesses used have on the results?

We will now consider these in turn.

13.3.6.1 Effect of Holding Stringer Areas Constant

By its nature the skin is usually "worked" harder than the booms as it has more jobs to do. This means that, in general the more material placed in the skins the better. The results in Table 13.5 which compares the results for similar grids bears this out. (Note the local effect in the 8 x 5 grid still swamping the results).

However, the criterion for placement and sizing of stringers is one of stability and not strength. So there is a need for a stringer panel design module here.

A1 Wing Example One Semi - Span Only

Analysis	C/S	SKINS	RIBS	REAR SPAR	F.SPAR	BOOMS & STRINGERS	TOTAL
Actual	13.6		3.8	2.2	3.11	20.03	43
16x8 9	50	19.6	1.75	1.2	2.6	10.8	35.9
16x8 4	100	17.1	1.72	1.1	2.0	21.6	43.5
10x8 7	100	16.2	1.82	1.7	1.6	18.1	38.8
10x8 10	150	14.3	1.83	1.1	1.22	27.2	45.6
8x5 5	100	37.0	3.16	2.2	2.6	11.9	56.9
8x5 11	150	35.8	3.3	2.1	2.3	17.0	61.5
8x5 12	200	35.1	3.3	2.0	2.0	23.9	66.4

Note: C/S refers to stringer cross-sectional area.

Table 13.6 Effect of Variations in Stringer Area Constraints

Analysis	C/S	SKINS	RIBS	REAR SPAR	F. SPAR	BOOMS & STRINGERS	TOTAL
Actual		13.66	3.81	2.20	3.11	20.03	43
5	100	37.0	3.16	2.2	2.6	11.9	56.9
11	150	35.8	3.3	2.1	2.3	17.9	61.5
12	200	35.1	3.3	2.0	2.0	23.9	66.4
13	200	20.8	2.27	1.75	2.0	23.9	50.7
14	50-200	37.0	3.36	2.11	2.69	8.8	54.0
15	200	26.3	3.39	2.36	2.6	23.9	88.0
16	200	18.6	1.32	1.43	1.90	23.9	47.1
17	200	19.7	1.52	1.52	1.65	23.9	48.8

Note: C/S refers to stringer cross-sectional area.

Table 13.7 Miscelaneous Variations on Wing with Course Grid

Table 13.7 shows the results from several variations of the same 8x5 grid. The results of interest here are Analysis 14 compared with 5. In both 5 and 14 the skins were optimised, the difference is that the stringers were fixed at $100mm^2$ in analysis 5 but were allowed to vary between $50mm^2$ and $200mm^2$ in analysis 14. The analyses were well behaved and the difference between the two solutions is small. Analysis 14 took twice as many iterations to achieve convergence as analysis 5.

The conclusion to draw here is that selection of stringer size is important (in inefficient designs) to the result, effecting the result by up to 20% in our test. However, as the design becomes more efficient and as long as the stringers are being "worked hard" the effect on the final overall results is small, though the effect on the component breakdown is great.

There is not a lot one can do about this problem since stringer arrangement and design is a complex matter often dictated by constraints such as manufacturing problems, wing arrangement, fail safety, geometry constraints and even personal preference of the designer.

Many stringer/panel optimisation procedures have been devised. These usually trade off the benifits of having as much material at the extreme fibres of the wing as a whole against maximising the local skin stiffners by increasing the depth of stringers. (This is a trade off because the deeper the stringers the more material one is moving away from the extreme fibre of the wing). Finally the stringers are designed against web and flange stability constraints.

As a guide towards the efficiency of the stringer design a module has been incoorporated in WEIGHTS which given the results of the analysis calculates a typical depth of stringer. Given this information the user may decide to resize the stringers. Say this module tells the user the typical size is very small, say 2mm, this would indicate an inefficient design; whilst a stringer depth which is greater than half the box depth would obviously be impossible.

13.3.6.2 The Effect of Variable Linking

This is a practical aspect of optimisation with the additional benefit of improving the behaviour of the analysis. Quit often several elements may be used to modal regions with similar thicknesses or areas, like a skin panel on the A1 fabricated from one sheet of aluminium. A dozen elements maybe used to idealise the panel but they must all vary in thickness by the same amount during the resizing procedure since they are all formed from the same sheet.

In this case the relevant results are for analysis 12 and 15 where in the later the skin elements were grouped into three panels as on the actual aeroplane. The results are somewhat unexpected.

The expected effect was one of an incorrect weight due to an enlarging of the territory of the erroneous tip effect. This does in fact happen but there is a second effect which swamps this one leading to a lower weight.

The design has "converged" to a "non-optimal" solution. In fact in analysis 15 30% of the elements are still active to a very high degree. Obviously there is a need for a more complex test for convergence. More about this later, but in essence This simple optimiser has broken down.

13.3.6.3 Effect of Constraints Skin Thickness Range

The results of interest here are for analysis 12 and 13 in table 13.7, where in analysis 12 the skins are virtually unrestricted whilst in 13 they are limited to between 0.5mm and 1.5mm thick. The weight drops in the constrained case due to a drop in the skin weight.

This is because the local high load that causes the thickning at the tip has in the unconstrained case less effect. What is happening is that the analysis ignores the need for more material to resist these loads.

This restriction would obviously give a more reliable answer but it is worth running an "unconstrained" analysis as that makes it easier to pickup on the disturbances (such as the tip errors found here) due to the large variations possible, in the "constrained" case the variations allowed are smaller and consequently harder to pick up on.

13.3.7 The Effect of Using C.F.R.P.

The results for the analysis of the CFRP structure is shown in table 13.3 under analysis 8. The results are remarkable good considering the crudeness of the failure criteria used and that only three basic layups were used in the model.

Also one should remember that the model was not constrained to increment thickness by finite layer steps and to make continuous layers. This would account for the slight optimism of the results.

The analysis was quite sensitive to the assumed material strengths but no more so than with a metallic structure. The difference is that material strengths are less definite with FRP material.

For this reason, empirical methods might be the safer bet in the hands of the inexperienced since details such as failure criteria are built-in to them. However as composites become more well known this advantage would dissappear. The problem has to do with dearth of data rather than the techniques itself.

13.3.8 Weight Breakdown Accuracy

Though a quick glance at table 3 would indicate that the accuracy of the weight breakdown is good, this particular wing was an unfair test.

In fact much of the wing was of a "minimum gauge" design. In this cause the actual minimum gauge was 22 gauge or about 0.56mm. Since in most cases the specified minimum allowable thickness was 2mm leading to slightly low weight values for components such as ribs and webs.

And so apart from the possibility that the ribs "light" in the breakdown due to incorrect loading mentioned in Section 4.3, no other conclusion on this matter can be reached till a "heavier" wing is analysed.

13.4 Al Wing Empirical - Analytical Weight Estimate

The results produced by the computer based method has been compared with actual weights in section 13.3. Hence the Lewis and St.John method described in section 2.2 was used to make the same estimate. The working out is presented to supplement section 2.2 as an example.

The analysis is started bearing in mind that the approach was developed for straight, untapered and unswept wing boxes. However, in its favour is the fact that skin-stringer wings were considered.

13.4.1 Weight Estimate Calculation

Some sacrifice in conciseness was made here to avoid the neccesity to refer back to section 2.2 for the required equations though one would have to refer to the notation.

13.4.1.1 Compressive Material Calculations

Assume
$E = 10^7$ p.s.i.
$h = 10$ inches
$h_w = 10$ inches
$C = 32$ inches
$f_{cy} = 35800$ p.s.i.
$f_{su} = 32200$ p.s.i.
ftu $= 35800$ p.s.i.
density $= 0.098$ p.s.i.

These are the same values as assumed for the f.e. analysis approach.

The Stanton - Jones method gives a root bending moment;

$$M = 385000 \text{ lbf.in}$$

The load index;

$$X = \frac{M}{Ch^2}$$

$$= 120$$

Using equation f,

$$f_{ID} = 0.88\, X^{1/2}\, E^{1/2}$$

$$= 30500 \text{ p.s.i.}$$

Substituting in equation g,

$$X_1 = 165.5$$

Using fig 2.7 we obtain a value for allowable stress;

$$f_{AL} = 24700 \text{ p.s.i.}$$

Hence the compressive cross-sectional area at the root is;

$$A_{COMP} = \frac{M}{hf_{AL}} \geq \text{min. gauge}$$

$$= 1.56 \text{ sq.in}$$

→ thickness = 0.035 in

(SWG 21)

13.4.1.2 Tension Material Calculations

$$A_{TEN} = \frac{M}{hf_{tu}}$$

$$= 1.08 \text{ sq in}$$

→ thickness = 0.024 in

(SWG 24)

13.4.1.3 Fatigue Analysis

Aerobatic aircraft are in a class of their own in this respect with some aircraft having life spans of only a few hundred hours. In any case the computer analysis of this wing was conducted ignoring the effects of fatigue. This part of the analysis is therefore skipped to maintain the comparison.

13.4.1.4 Shear Web Analysis

Using equation n

$$X_1 = \left(\frac{f_{su}}{1.766E^{1/3}} \right)^{3/2} = 779$$

And equation m

$$X = \frac{V}{h_w^2} = \frac{12100}{100} = 121$$

$$\Rightarrow f_{IDS} = 1.776E^{1/3} X^{2/3} \leq f_{su}$$

$$= 9360 \text{ p.s.i.}$$

Using fig 2.9;

$$f_{AL} = 1.7 \times 9360 = 16000 \text{ p.s.i.}$$

$$\Rightarrow A_{SHR} = \frac{V}{f_{AL}}$$

$$= 7.6 \times 10^{-3} \text{ sq.in.}$$

$$t = 3.8 \times 10^{-4} \text{ in}$$

which is less than the 24 gauge allowed as the minimum gauge hence

$$A_{SHR} = 0.5 \text{ sq.in.}$$

13.4.2 Results

Weight = density x volume

= 0.098 x 3.14 x 196.8

= 60.6 lbs

Compared with 94.8 lbs in reality

Part of the error would be caused by the straight wing assumptions. Another error would be introduced if the wing root was not representative of the rest of the wing since that is the part of the wing which was analysed. In this case another large error is creeping in due to material present in the real structure in the front spar booms to carry landing loads. This would account for most of the error, in general the thicknesses derived by this method for the skins are representive of reality. This was easily spotted in the computer based analysis due to the itemisation of weights, not so in this analysis.

14. CONCLUSIONS and RECOMMENDATIONS

Given the package as it stands, the overall accuracy in the case of the A-1 wing is good, given some provisos. These are that; the mesh should be of sufficiently high density (at least as high as the 10x8 mesh used here); and the geometry is accurate to about 20% of reality.

The program appears to be susceptible to a combination of factors which means that it needs improvement in the area of load calculation and applications before it can be said to be robust enough for use by a naive user. In general the combination of these effects leads to a breakdown in the simple sizing routine.

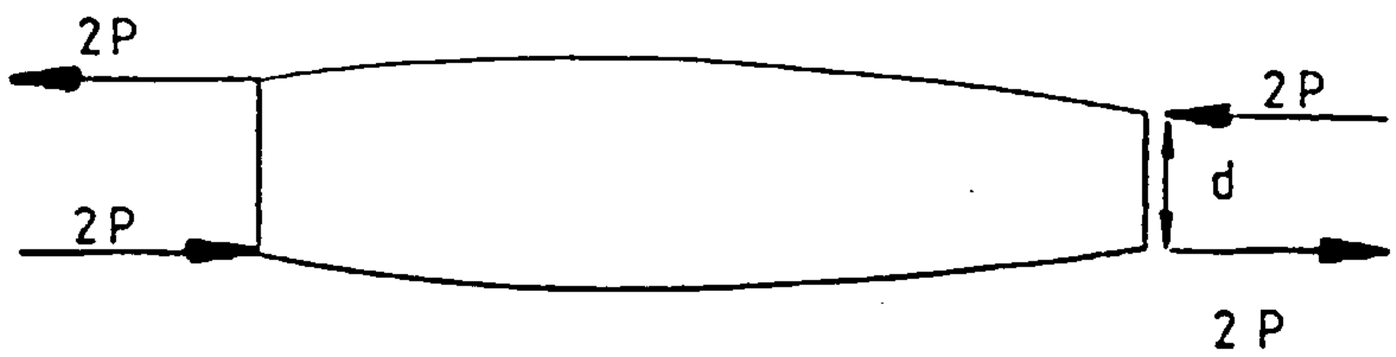
These drawbacks can be alleviated in the short term by providing warning and diagnostic messages advising the user of the problems and possible remedies.

These factors are;

(a) The conversion of aerodynamic loads to nodal loads or the finite element grid is in some cases, clumsy leading to high load concentrations and incorrect sizing.

(b) Incorrect geometry usually has a small effect on results but in some cases when large misrepresentations are made this can effect the load application. For example a torsion box with an error of a factor of two in its depth would cause a similar factor on the differential loads representing torsion as shown in fig 13.28.

The computer based method on the other hand has proved more accurate when properly used, than the advanced analytical - empirical method used in section 13.4.



Torque = 4Pd on
both boxes

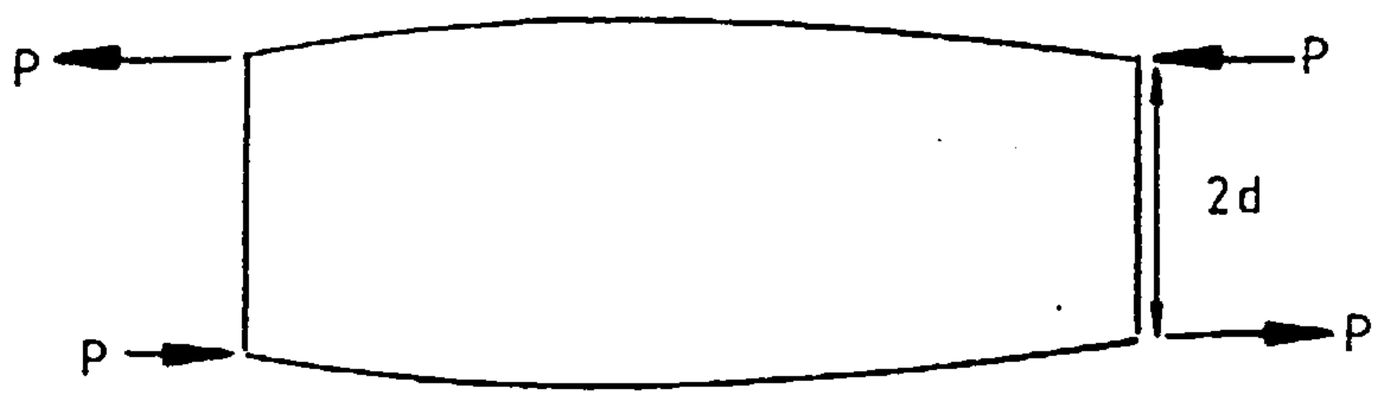


Fig 13.28 Effect of Geometry on Concentrated Loads

(c) Too low a mesh density if incorrectly handled can lead to high concentrated loads as well. However, a low mesh density correctly used can give good results with speed and cost advantages.

(d) Finally the sizing process needs to be monitored, otherwise it can 'converge'to unsatisfactory solutions.

14.1 Loading

All the evidence suggests that these problems will be cured by a more sofisticated and accurate loads package. As for the resizing routine, this seems to behave well in conditions where the loading aberations are absent.

The element of the load package that seems to be at fault is the conversion of the aerodynamic load distribution into a form suitable for the finite element analysis. It is recommended that this should be improved in the standard package.

14.2 Utility

The package was fairly easy to use. However, more time needs to be spent in several areas to make the package more user-friendly and effective. These are;

(a) All input modules should be converted to the interactive editor approach. (update).

(b) The interactive editor approach itself should be made more flexible. (upgrade).

(c) More should be done to enable the packages to identify elements as part of the structure. This had been carried out at the beginning of the project but was neglected at later stages for expediency of testing. This will cut down on some tasks which are done manually at present.

14.3 Validation

Time was allocated to validation "runs" similar to the one conducted on the A1 wing. Initially indications were that the data required for such runs would be supplied by "industrial" sources. Unfortunately, due to the secrecy associated with such data, and the cost of providing it "industrial" sources dried up.

It is recommended that funding be supplied to buy the data required to test this package further, particularly with more complex wing, which have more non-structural components.

15. References

1. C.R.Glatt,
WAATS - A computer program for weights analysis of advanced transportation systems,
NASA CR-2420,
Sept. 1974.

2. M.Hashimoto et al,
Application of the finite element technique to aerodynamic problems of aircraft,
Computers and Structures Vol. 19, No. 1-2, 1984.

3. W.B.Herbst et al,
Application of computer aided design programs for the technical management of complex fighter development projects.
A.I.A.A. paper No. 70-364,
March 1970.

4. J.G.Hutton et al,
Application of finite element analysis to derivation of structural weight,
S.A.W.E. Paper No. 1271,
May 1979.

5. N.S.Khot et al,
Optimization of fiber reinforced composite structures,
Int. J. Solids Structures Vol 9,
1973.

6. F.K.Ladner et al,
A summary of the design synthesis process,
S.A.W.E Paper No. 907,
May 1972.

7. W.Lansing et al,
Application of fully stresses design procedures to wing and empennage structures.
Journal of Aircraft Vol. 8,
Sept. 1071.

8. L.E.Lewis et al,
Allowable stress estimation methods for preliminary design weight prediction,
S.A.W.E. Paper No. 1044,
May 1975.

9. J.W.Nisbet et al,
A new role for structures technology in aircraft configuration development,
A.S.M.E.
December 1976.

10. C.R.Ritter,
Rib Weight estimation by structural analysis,
S.A.W.E. Technical Paper No. 259,
May 1960.

11. R.S.St.John,
The derivation and application of analytical - statistical weight prediction techniques,
S.A.W.E. Paper No. 810,
may 1969.

12. M.J.Turner,
Optimization of structures to satisfy flutter requirements,
A.I.A.A. Journal Vol. 7. No.5,
May 1969.

Appendix A

WEIGHTS User Manual

Version 86.05

CONTENTS

1. INTRODUCTION

WEIGHTS is a modular computer package which may be used for estimating or predicting weights of aircraft wings. The basic package is spartan and has limited capabilities. However, the package was designed to allow for the user to include user modules to extend the capabilities of the package.

Originally it was proposed that the package be designed with portability in mind, so that implementation on many computer types may be carried out. This decision was changed during the development stage as it imposed many limitations. Much use has been made of the VAX/VMS operating system capabilities. Although other implementations are being considered (e.g. UNIX based HP9000) this manual refers to a VAX/VMS implementation. Thus a basic knowledge of VAX/VMS is useful but not essential for the user of this package. (Ref. VAX USERS GUIDE).

The package has "on-line" documentation which facilitates use of the system during a specific implementation. The usefulness of hardcopy documentation cannot be completely replaced by on-line documentation (for example, during those quiet times when the computer is off-line or in mid run) hence the reason for this document. This document is not concise since it forms the basis of the on-line documentation which has to have information on each topic which can stand alone.

This manual covers several aspects of the use of the package and breaks down into the following topics.

(a) Invoking WEIGHTS

(b) Invoking WEIGHTS modules

(c) Operation in VAX/VMS environment

(d) Validity of module ordering

(e) Design procedures

(f) Including your own module

(g) User quotas and installation

In the following text, VDU refers to a Visual Display Unit, and VAX/VMS is the computer operating system.

2. INVOKING WEIGHTS

To start execution of the package a startup module must be executed which sets up work directories and global symbols. Provided the package has been correctly installed the user need only type WEIGHTS after logging on. If this command symbol does not produce the required effect contact the person responsible for software installation it is possible that access to the package is restricted.

The VDU screen should then clear and respond with the prompt,

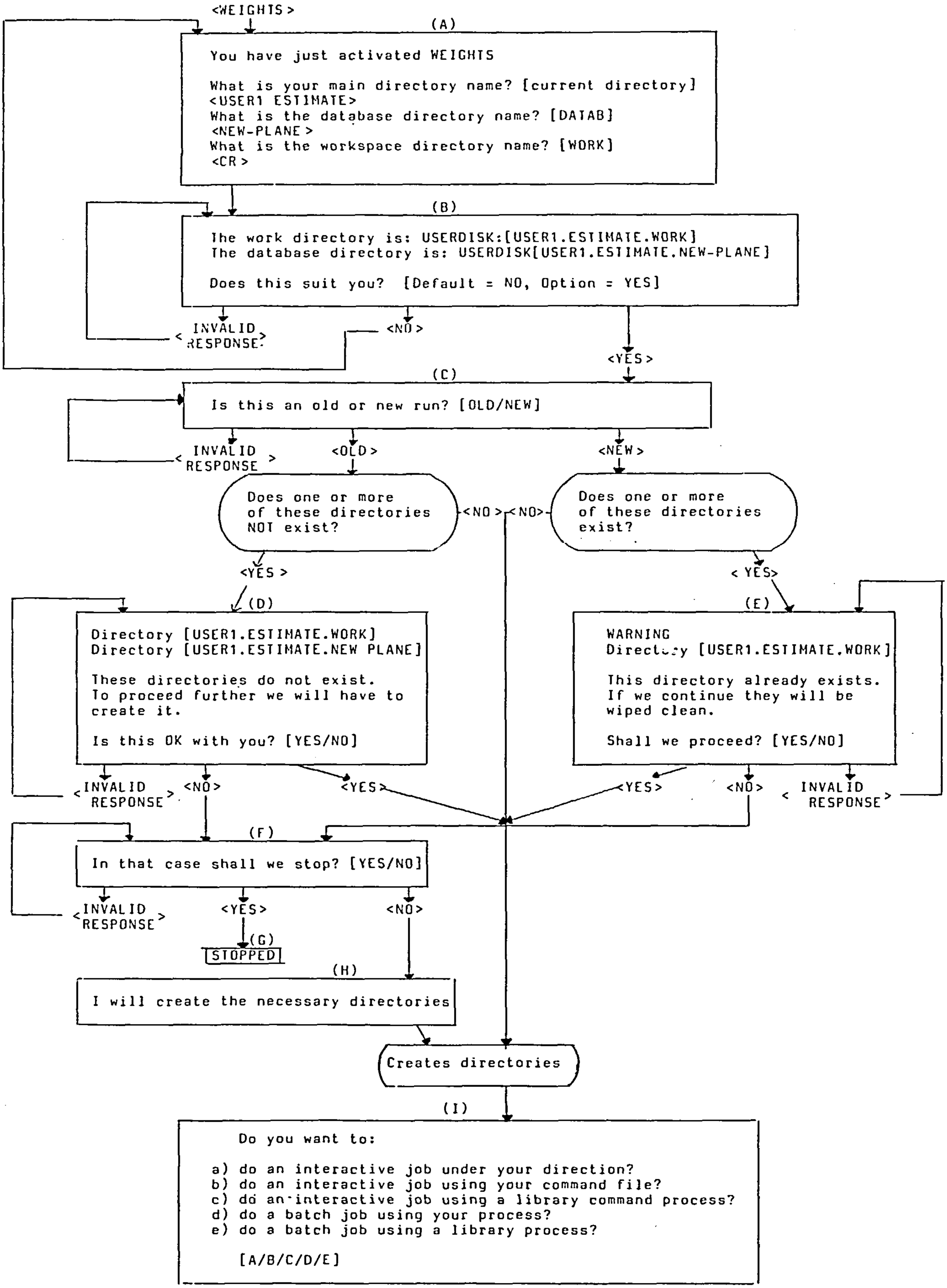
"You have just activated WEIGHTS".

After this the procedure shown in the following flow chart is carried out.

Key to the flow chart

<word>	indicates a user entry.
[one word]	indicates the default answer assumed if no answer is given.
[several options]	indicates the valid choice of answers, sometimes there will be a default answer.
┌──────┐ │ text │ └──────┘	text produced on the V.D.U before being cleared to make room for more text.
text	the process carried out by the computer.
——→	path taken given a test.
(character)	prompt identifier used for reference in the description later.

WEIGHTS ACTIVATION FLOWCHART



2.1 Valid responses to invoking routine

If you are familiar with VAX/VMS the flow chart should be self explanatory, this section is to provide more details for those who are not familiar with VMS.

2.1.1 Box (A)

As mentioned earlier this routine prepares areas on your machine's disk for use. Referring back to the flow chart, the information needed for this preparation is provided in box (A). It is possible to use the package without knowledge of the VMS directory structure but your ability to use the package effectively would be enhanced by that knowledge.

In response to the first question;

What is your main directory name? [current directory]

You may respond with any top level directory or subdirectory that you have access to or would like to create. e.g.

```
JOE             ->    [JOE]

JOE.BLOGGS      ->    [JOE.BLOGGS]

.BLOGGS         ->    [.BLOGGS]
```

If you give no answer it will be assumed that the directory you are currently in, will be used.

In response to the other two questions in Box (A) of the flow chart the following responses are valid;

```
SUB_DIR_NAME

SUB_DIR_NAME.SUB_SUB   etc.
```

The routine takes the information given and concatenates the strings to form valid subdirectory names. In the flow chart the resulting directories were concatenations of USER1.ESTIMATE and NEW_PLANE both user provided; and USER1.ESTIMATE and WORK the latter being the default.

The rationale behind this is to make house keeping easier. A user may have to analyse several different aircraft or do several analyses on one type. In order to make it easier to keep track of each analysis the following strategy may be adopted by a user running an analysis on a commercial airliner and several analyses on a fighter.

```
[USER.WEIGHTS.AIRBUS.DATAB]       )
[USER.WEIGHTS.FIGHTER.WING_1]     } Database
[USER.WEIGHTS.FIGHTER.WING_2]     ) directories

[USER.WEIGHTS.AIRBUS.WORK]        )
[USER.WEIGHTS.FIGHTER.WORK]       } Scratch
                                  ) space directories
```

In this example the user has chosen to do all weights related work in the subdirectory [USER.WEIGHTS...]. All work relating to the airliner is found in [USER.WEIGHTS.AIRBUS...] and all the work relating to the fighters is found in [USER.WEIGHTS.FIGHTER...]. Further subdirectory branches WING_1 and WING_2 would contain different analyses of the same aircraft.

2.1.2 Box (B) to (H)

These portions of the flow chart are self explanatory and need no further explanation except to say that they are fail-safe mechanisms to prevent accidental deletion of existing databases.

2.1.3 Box (I)

Only one option in Box (I) is currently available. That is option A. The reason is that not enough is known about the behaviour of the package to make the other options practical. They exist, though, and can be implemented at any time it is thought necessary.

3. INVOKING WEIGHTS MODULES

Having prepared your work space and defined the WEIGHTS command the following text is placed on the VDU.

PLEASE TYPE IN THE WEIGHTS COMMAND YOU WANT ME TO DO.

If you make an invalid response to this prompt it will reply,

THE PROGRAM NAME YOU HAVE JUST GIVEN ME IS NOT IN MY LIST. COULD YOU PLEASE TRY AGAIN?

If a valid response is made you get the response

PROGRAM NAME SET.

Descriptions of valid WEIGHTS commands and descriptions of what they do follow.

3.1 AIRLOAD1 and AIRLOAD1 INPUT

This module calculates the lift and torque distribution on a tapered swept back wing at subsonic speeds (compressible flow allowed) using the Stanton Jones Method.

The data required for running this module may be compiled using the AIRLOAD1_INPUT module which is self explanatory and prompts the user for the input required.

The input filename must be AIRLOAD1.IN and the output filename will be AIRLOAD1.OUT. The input may be free formatted and must contain the following in order;

GAP	Interval between reference points.
PTS	Number of reference points.
QSWEP	Quarter chord sweep back angle in degrees.
RCORD	Root chord length.
TCORD	Tip chord length.
MD	Dive Mach number.
WGHT	Design weight.
ACCN	Design case "g" load.
ULTFAC	Ultimate factor
ETAII	Fuselage overlap length (center to wing root).
DUMP	Variable dump flag. 1 for yes 0 for no.

This last variable was included for development work and is not needed by the user, it is set to 0 by AIRLOAD1_INPUT.

3.2 BUCKLE

This is a simple demonstration module. The input is interactive as shown in fig 1.

```
$ BUCKLE
enter material failure stress
247
enter rib bay span
450
enter material modulus
70000
enter chord length
2070
enter skin thickness
.6
enter total stringer area
100
 the stringer depth is about    31.16009255016619
```

Fig 1. Demonstration of Buckle Utility

Given the requested parameters a notional stringer depth is calculated. The designer can decide whether or not this is suitable on practical grounds.

Future versions would have an automatic version of this module with the user either acting in a supervisory capacity or not being part of this process at all.

3.3 CLEAR

Clears the VDU screen.

3.4 DESVAR INPUT

This module prompts the user for design variable link data. It is self explanatory and requires an input for each design variable;

(a) Design variable number.

(b) Number of linked elements in that group.

(c) Element numbers in that group.

This module is one of the least helpful and it is currently simpler to use the standard VAX editors to create the data. Using free format and the order shown above. e.g.

1 2	variable 1 has 2 elements
5 17	they are elements 5 and 17
2 5	variable 2 has 5 elements
16 2 3 1 4	they are 16 2 3 1 and 4

A more sophisticated input module is under development and will eventually replace all the input modules.

An alternative non-interactive and more concise module which carries out the same task is RANGER.

3.5 ELPROPS2

This module prompts the user for element properties, gauge limits and material strengths which it then uses to generate element property files. The routine was devised before the advent of screen management routines which are used in the newer interactive routines. However this does make the program more portable.

Before this module can be executed, a mesh must have been generated either manually or by using WMESH1, SUB_WMESH1 or WMESH2 modules. This is because, apart from the input received from the user, standard weight database files containing mesh and run statistics data are used, and standard database files are produced. This is transparent to the user and is only given as an explanation why the mesh must be created first. For more information on these topics refer to the chapter on valid module ordering and a seperate document on the WEIGHTS DBMS.

The user is asked to answer a series of prompts some of which are requests for starting values for certain physical parameters which the program will change in converging to a design, others represent constraint or fixed values. Hence:

What thickness do you want for the top skin panels?

What thickness do you want for the bottom skin panels?

What thickness do you want for the spar webs?

What area do you want for the stringers?

What thickness do you want for the ribs?

What is the Youngs modulus?

What is the density of the material?

What is the Poisons ratio of the material?

What is the minimum skin thickness?

What is the maximum skin thickness?

What is the allowable skin stress?

What is the minimum stringer area?

What is the maximum stringer area?

What is the allowable stringer stress?

Having provided this data you are now given a chance to correct any mistakes. The following text (fig 2.) appears on the screen listing the values you have typed in.

```
     WHAT IS THE MAXIMUM STRINGER AREA?
50
     WHAT IS THE ALLOWABLE STRINGER STRESS?
247
     THESE ARE THE VALUES YOU HAVE GIVEN ME.

 1. TOP SKIN THICKNESS          ---- 2.000000000000E-01
 2. BOTTOM SKIN THICKNESS       ---- 2.000000000000E-01
 3. SPAR THICKNESS              ---- 2.000000000000E-01
 4. STRINGER AREA               ---- 5.000000000000E+01
 5. RIB THICKNESS               ---- 2.000000000000E-01
 7. YOUNGS MODULUS              ---- 7.000000000000E+04
 8. MATERIAL DENSITY            ---- 2.720000000000E-06
 9. POISONS RATIO               ---- 3.000000000000E-01
10. SKIN MINIMUM THICKNESS      ---- 5.000000000000E-02
11. SKIN MAXIMUM THICKNESS      ---- 5.000000000000E+00
12. SKIN ALLOWABLE STRESS       ---- 2.470000000000E+02
13. STRINGER MINIMUM AREA       ---- 5.000000000000E+01
14. STRINGER MAXIMUM AREA       ---- 5.000000000000E+01
15. STRINGER ALLOWABLE STRESS   ---- 2.470000000000E+02
     IF YOU WANT TO CHANGE A VALUE TYPE THE LIST NUMBER
     OF THE VALUE AND HIT RETURN , OTHERWISE TYPE 0.
0
```

Fig 2. Sample VDU Output for ELPROPS2

To correct a mistake follow the instructions and type in the list order number of the value you wish to correct. e.g. type 9 if you want to correct the Poisons ratio. You will then be prompted for the new value. After that the corrected value appears in the values table (fig.2) and you are given more chances to correct other mistakes.

If everything is correct type 0 and the routine carries out its task.

3.6 ELPROPS3

Whereas ELPROPS2 replaced ELPROPS1, the module ELPROPS3 has extended capabilities but is not intended as a replacement to ELPROPS2. ELPROPS3 prompts the user for data required to make up element property data and design limits data. It operates in exactly the same way as ELPROPS2. ELPROPS3 has an orthotropic materials capability. For further details see ELPROPS2, module ordering and WEIGHTS DBMS Manual.

The data input required of the user is more extensive than in ELPROPS2 and is divided into six main sections which are;

Top skin data,
Bottom skin data,
Rib skin data,
Front spar web data,
Rear spar web data,
Stringer data.

The skin and web sections all follow the same format and the prompts for the top skin are given here as an example.

What is the modulus (Ex) value for the top skin panels?

What is the modulus (Ey) value for the top skin panels?

What is the modulus (Gxy) value for the top skin panels?

What is the material density in the top skin?

What is the fibre orientation on the top skin?

What is the material Poisons ratio of the top skin?

What is the top skin minimum skin thickness?

What is the top skin maximum skin thickness?

What is the allowable top skin stress?

In the case of the stringer section, there are slight differences in the data required and the following prompts are given;

What is the modulus (Ex) value for the stringers?

What is the material density in the stringer?

What is the material Poisons ratio of the stringer?

What is the stringer minimum C/S area?

What is the stringer maximum C/S area?

What is the allowable stringer stress?

After answering the prompts in each section the user is given a chance to correct mistakes as in ELPROPS2 before moving on to the next section. A sample table is shown in fig 3.

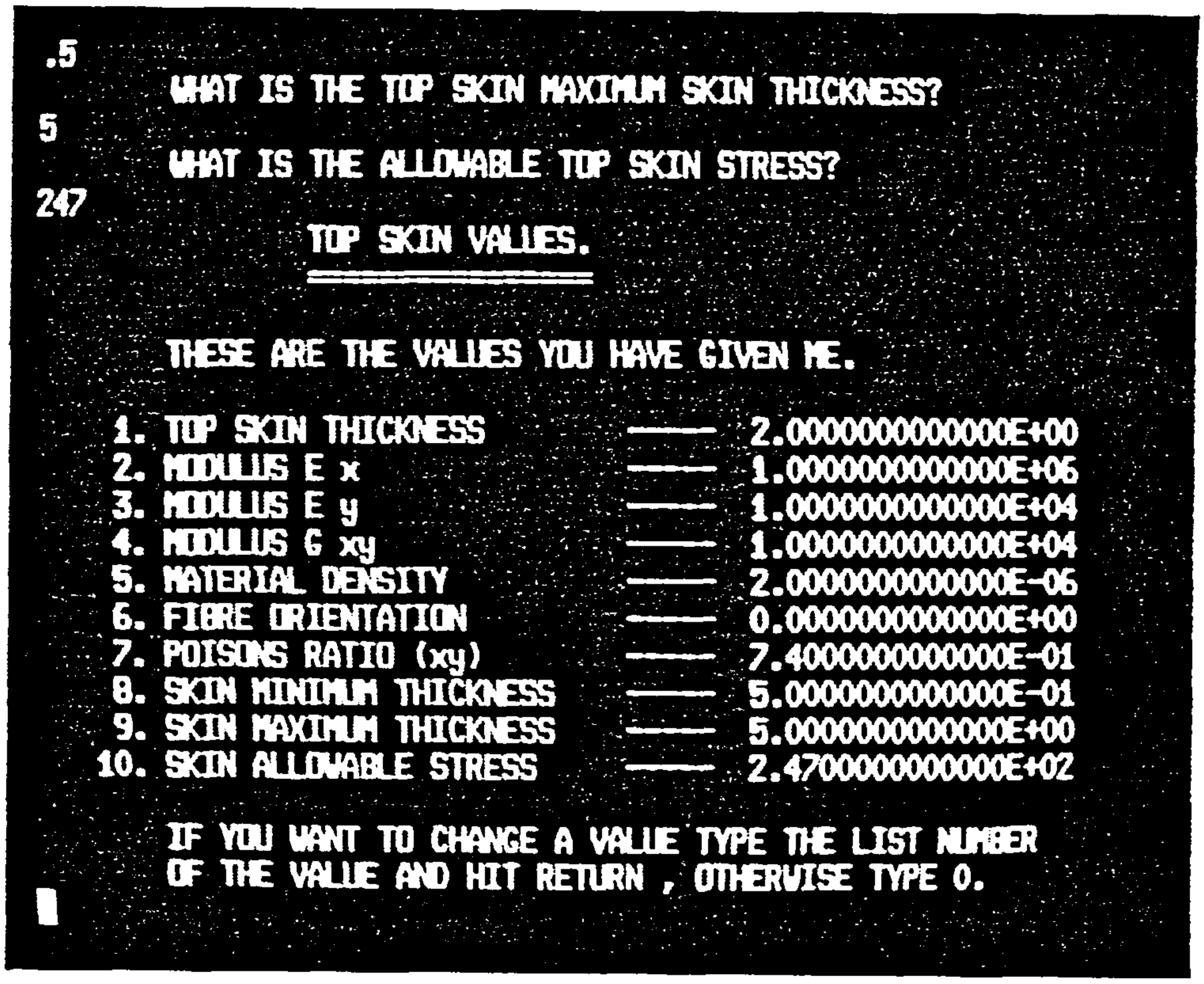


Fig 3. Sample VDU Output for ELPROPS3

For example if you wish to make a correction to the Ex value type in 2 and you will be prompted for a new Ex value. This table will reappear with the new value and you can make other corrections before typing 0 to move on to the next section for finishing the task.

Note: Unlike the newer routines there is no recovery or intermediate save action available. It is imperative therefore that you have all the data required and enter it correctly. The only solution to uncorrected mistakes would be to start again.

3.7 FE ANALYSIS

This module is currently redundant and is only for "knowledgable" users. It invokes FINEL version 3 which is a finite element package. This package has been upgraded and changed since this module was devised and FINEL is no longer supported at the College of Aeronautics.

3.8 FULLY STRESSING and FS INIT

This module comes in several versions which are site dependant and you should refer to your systems manager's installation notes. The following holds true for the College of Aeronautics system.

This module submits a fully stressing optimisation routine to a batch queue for analysis. Some versions of this analysis are capable of considering orthotropic membrane materials, some can consider plate bending and others consider only simple membrane, stiffner problems. Table 1 lists the module names, their capabilities and related batch queues.

MODULE NAME	QUEUE	APPLICATION
FULLY_STRESSING	INTERACTIVE	Plate bending/Stringer
FULLY_STRESSING_BATCH	SYS$BATCH	Plate bending/Stringer
FULLY_STRESSING_3DOF	INTERACTIVE	Membrane/Stringer
FULLY_STRESSING_3DOF_SLOW	SLOW	Membrane/Stringer
FULLY_STRESSING_3DOF_BATCH	SYS$BATCH	Membrane/Stringer
FULLY_STRESSING_3DOF_ORTHO	INTERACTIVE	Membrane/Stringer/Orthotropic
FULLY_STRESSING_3DOF_ORTHO_SLOW	SLOW	Membrane/Stringer/Orthotropic
FULLY_STRESSING_3DOF_ORTHO_BATCH	SYS$BATCH	Membrane/Stringer/Orthotropic

Table 1. COA Batch Queues

These modules use the LUSAS finite element package to carry out the stress analysis. The user may limit the number of resizing iterations carried out by entering a maximum iteration value as a parameter; e.g.

```
FULLY_STRESSING_3DOF_SLOW 15
```

In this example a membrane/stringer analysis is submitted to the SLOW batch queue with the maximum number of iterations of resizing set at 15. If this parameter is excluded the default maximum is 2.

The design process stops either when "convergence" of the weight occurs or when the maximum specified redesigns have been carried out. There is a possibility that the analysis could stop because of system related parameters, such as exceeding your disk quota or CPU limit. In this case contact your system manager.

The process may be carried on where it stopped by typing in the command FS_INIT followed by the required FULLY_STRESSING command.

3.9 HELP and HINT

These are the "online" documentation modules. HINT contains the same information as the pages in this manual whilst HELP is the VAX/VMS online documentation.

For example if you type HINT you will be given a list of topics on which documentation is available as shown in fig 4.

```
HELP

  This is the WEIGHTS help file.

 Additional information available:

 AIRLOAD1   AIRLOAD1_INPUT        ASK_FOR_PROCESS_NAME  ASK_FOR_PROCEDURE_NAME
 DESVAR_INPUT          ELPROPS    ELPROPS2   FE_ANALYSIS
 FULLY_STRESSING       FULLY_STRESSING_BATCH GET_PROCEDURE_NAME   HELP
 MENUS      MENU_A     MENU_B     MENU_C     MERGER1    NODELOAD_INPUT
 NODFIX_INPUT          NOT_YET_AVAILABLE     OPTIMISE   PLOT
 PLOT_INPUT POST_STARS RUN_BEAMING8          RUN_WGHT_STARS        SET_FILE
 SET_RUN_MODE          STARLINK   STOP       SUB_WMESH1 WGFAC1     WMESH1
 WMESH2     WMESH2_INPUT

Topic? _
```

Fig 4. Example of Basic HINT Facility

To obtain information on a specific topic you may either type HINT <CR> and then type in part of the topic you require or simply type the whole thing in one go, for example the command HINT PLOT_INP gets the response shown in fig 5.

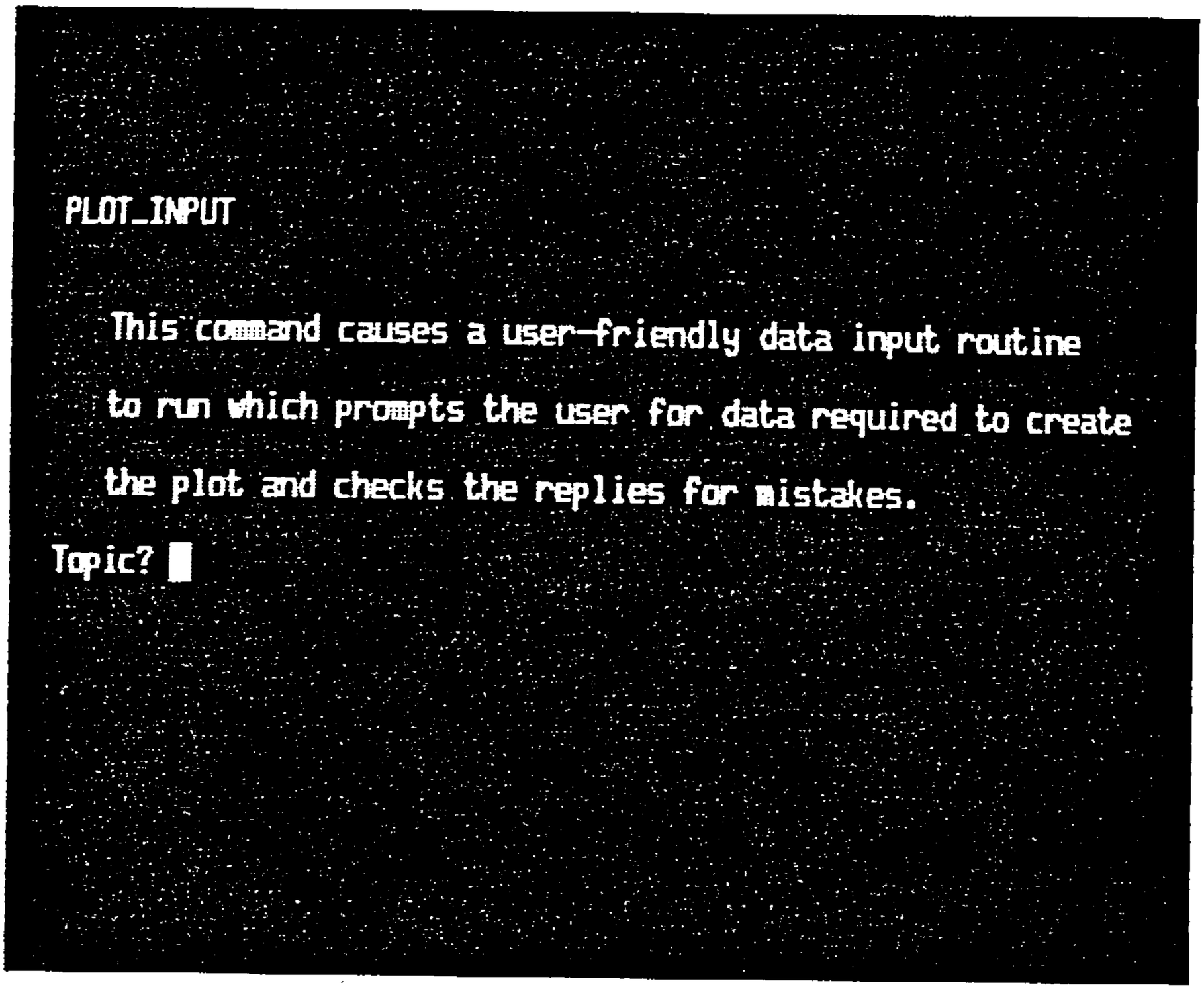


Fig 5. Example of Specific HINT Facility

If you are not very specific in the type of information you require all the information matching your specifications will be provided. For example if you typed HINT PL you would receive information on the PLOT module as well as the PLOT_INPUT module.

3.10 ITEMIZE

This command causes a scan of the database during which weights of various compounds are extracted and stored in an output file ready for printing or viewing on the VDU.

This procedure requires two input files, the first one is produced by another module called SCAN (refer to SCAN and module ordering). The other file must be called ITEMIZER.IN. This contains information on how the breakdown should be carried out. In previous releases and in future releases of WEIGHTS this file will be generated by other modules with user intervention being needed only in specialised cases. In the current development package this file must be made up by the user.

The datafile ITEMIZER.IN must contain the following for each component.

Number of element groups component name.

First element last element } for each group

For example:

```
9 Bottom Skin Sub Total
64 70
80 86
96 102
112 118
128 134
143 148
156 160
167 170
176 178
1 WING WEIGHT TOTAL.
1 303
```

In this example there are two component items. The first is the total bottom skin weight which is composed of 9 groups of elements these groups being 64 to 70 inclusive, 80 to 86 inclusive, 96 to 102 inclusive, etc. The second item is the total wing weight which is composed of one group of elements, this being elements 1 to 303 inclusive (i.e. all the elements). The resultant output is shown next.

```
Bottom Skin Sub Total
Number of elements          =  53
Number of active elements   =   3
Component weight            = 7.02446E+00
WING WEIGHT TOTAL.
Number of elements          = 303
Number of active elements   =  21
Component weight            = 4.55844E+01
```

3.11 MENU

These modules have been superseded by the HINT module. They give lists of available menus and modules.

```
MENUS
MENU_A
MENU_B
MENU_C
```

3.12 MERGER1

This module combines the output of several applications of the module SUB_WMESH1. The user is given a list of available meshes generated so far and is asked which ones should be concatenated. The module combines several sub-structural meshes into one, eliminating any redundancies.

When the command MERGER1 is entered the following appears after a short interval;

```
The number of sub meshes stored = 10

How many do you want merged?
```

This means that you have created 10 sub-structures so far using SUB_WMESH1. Supposing you wish to merge three of those you would respond now by typing 3 and the follow on from this will be;

```
3
Which ones do you want to merge?
Enter code number of sub mesh 1
3
Enter code number of sub mesh 2
4
Enter code number of sub mesh 3
7
```

In this example the 3rd, 4th and 7th sub-structures would be concatenated.

3.13 NODELOAD INPUT

This is an interactive input module (instructions and error messages are given) which needs to be run before executing module RUN_BEAMING8. It prompts for data describing the nodes to which the aerodynamic loads should be applied.

The recommended pattern of nodes is shown in fig 6.

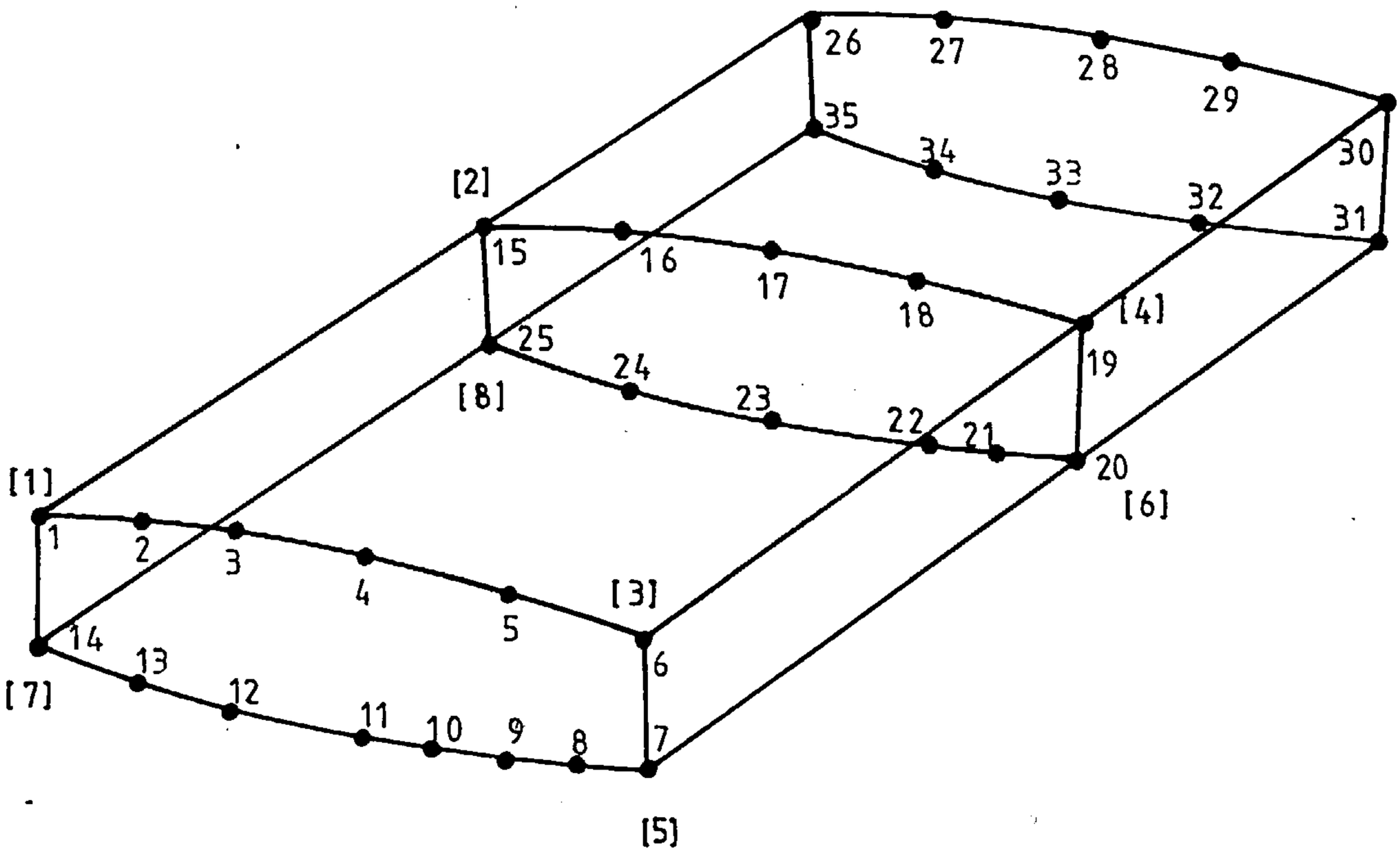


Fig 6. Recommended Loaded Node Order.

The reply to obtain this result is shown in fig 7.

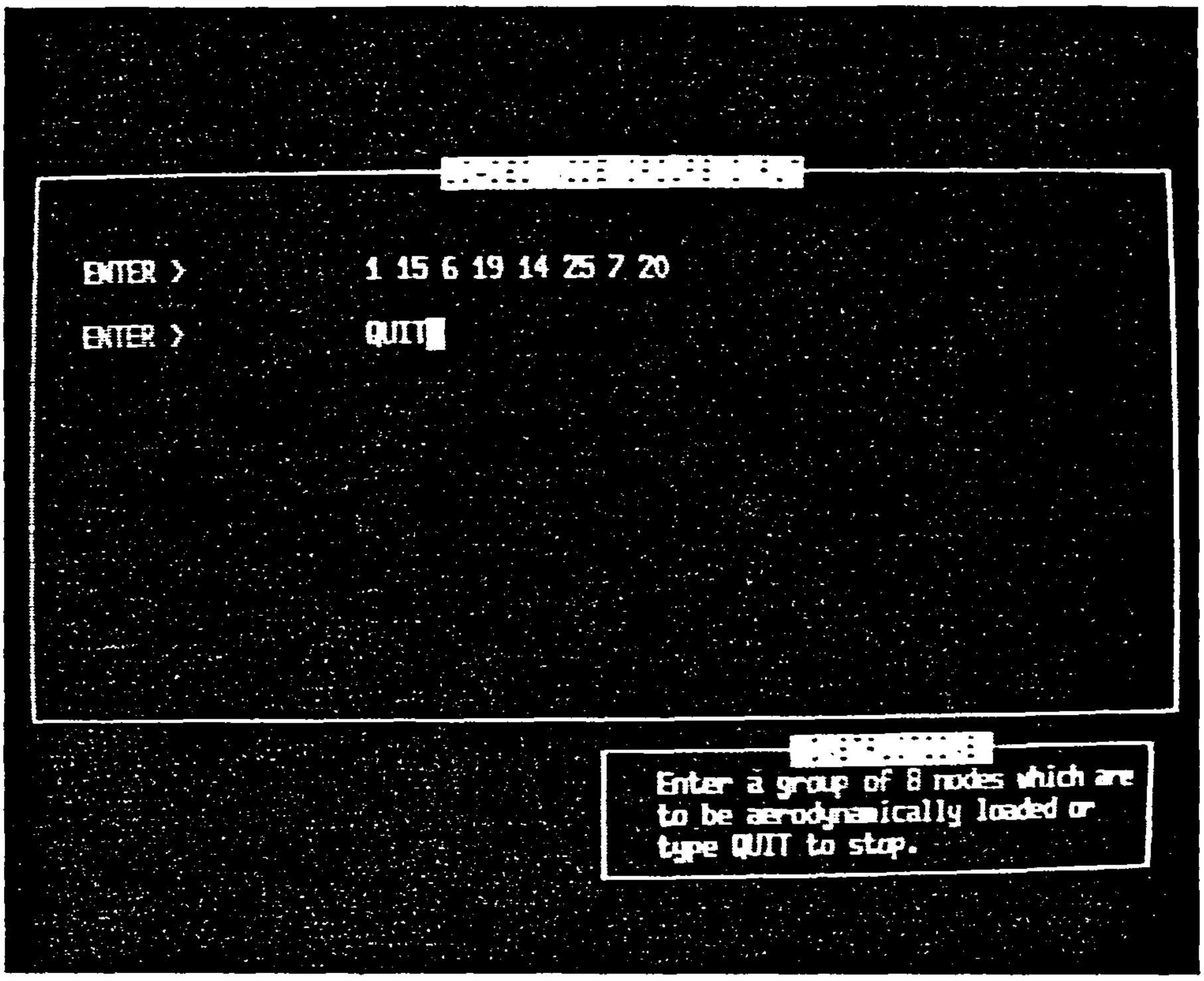


Fig 7. Example of NODELOAD INPUT Display

Note: You must enter the node groups in the same order as the aerodynamic loads you wish to associate them to e.g. if in the example given the group shown in fig 7. were the first values entered, they would apply to the aerodynamic loads at the reference station nearest the centre line.

This process will be automated and hence eliminated in the production program. Currently there is a capability to save data at anytime but none to restore old data.

Should the user wish to use the VMS editor the following format should be used in a datafile called NODELOAD.DAT. This is the output file from NODELOAD_INPUT.

Number of reference stations
Group of 8 reference stations repeated

For example:

```
 50
  1   17   8   24   9   25   16   32
  1   17   8   24   9   25   16   32
  1   17   8   24   9   25   16   32
  1   17   8   24   9   25   16   32
  1   17   8   24   9   25   16   32
  1   17   8   24   9   25   16   32
 17   33  24   40  25   41   32   48
 17   33  24   40  25   41   32   48
 17   33  24   40  25   41   32   48
 17   33  24   40  25   41   32   48
 17   33  24   40  25   41   32   48
 33   49  40   56  41   57   48   64
 33   49  40   56  41   57   48   64
 33   49  40   56  41   57   48   64
 33   49  40   56  41   57   48   64
 33   49  40   56  41   57   48   64
 49   65  56   72  57   73   63   80
 49   65  56   72  57   73   63   80
 49   65  56   72  57   73   63   80
 49   65  56   72  57   73   63   80
 49   65  56   72  57   73   63   80
 49   65  56   72  57   73   63   80
 49   65  56   72  57   73   63   80
 65   81  72   87  73   88   80   94
 65   81  72   87  73   88   80   94
 65   81  72   87  73   88   80   94
 65   81  72   87  73   88   80   94
 65   81  72   87  73   88   80   94
 65   81  72   87  73   88   80   94
 65   81  72   87  73   88   80   94
 81   95  87  100  88  101   94  106
 81   95  87  100  88  101   94  106
 81   95  87  100  88  101   94  106
 81   95  87  100  88  101   94  106
 81   95  87  100  88  101   94  106
 81   95  87  100  88  101   94  106
 95  107 100  111 101  112  106  116
 95  107 100  111 101  112  106  116
 95  107 100  111 101  112  106  116
 95  107 100  111 101  112  106  116
 95  107 100  111 101  112  106  116
 95  107 100  111 101  112  106  116
 95  107 100  111 101  112  106  116
107  117 111  120 112  121  116  124
107  117 111  120 112  121  116  124
107  117 111  120 112  121  116  124
107  117 111  120 112  121  116  124
107  117 111  120 112  121  116  124
107  117 111  120 112  121  116  124
117  125 120  127 121  128  124  130
117  125 120  127 121  128  124  130
117  125 120  127 121  128  124  130
117  125 120  127 121  128  124  130
117  125 120  127 121  128  124  130
117  125 120  127 121  128  124  130
117  125 120  127 121  128  124  130
```

Here, there are fifty aerodynamic loading stations and the loads are distributed to nodes on the front and rear spar of each rib. A typical group is shown in fig 8.

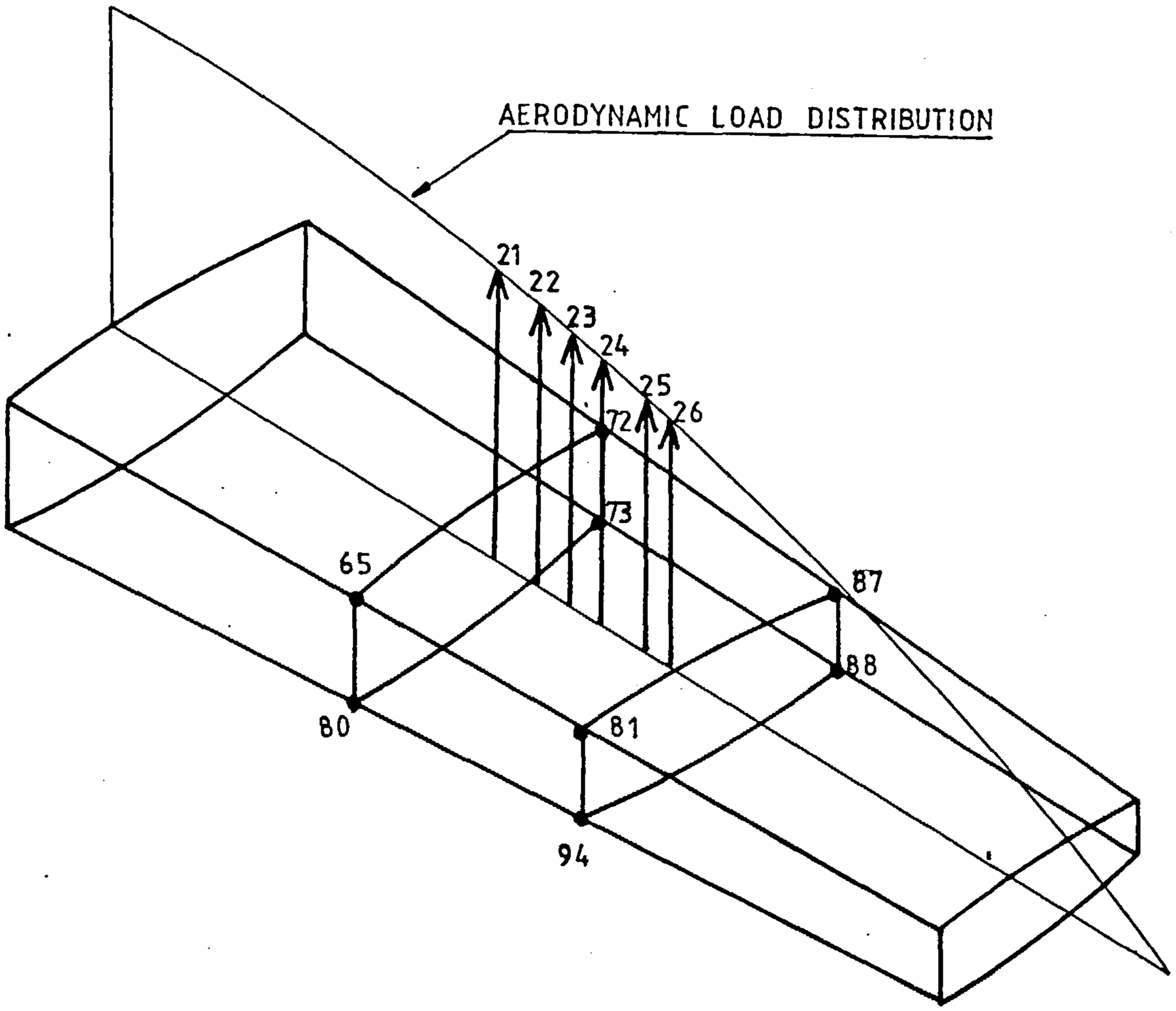


Fig 8. Aerodynamic loads at stations 21 to 26 transferred to node group 65,81,72,87,73,88,80,94

Note that the initial load distribution may be provided by the Stanton Jones approximation module (see AIRLOAD1), may be fed in by hand or provided by any other suitable method.

The redistribution process is carried out by the RUN_BEAMING8 module, refer to appropriate chapter.

3.14 NODFIX INPUT

This is an interactive input module which needs to be run before executing module FULLY STRESSING. It prompts the user for data concerning nodal restraints on the finite element model.

The proposed default is to have a built-in wing at the centre line of the wing if the user does not supply any information on this topic.

Instructions and error messages are interactive. There is a capability to save what you have done at anytime but there is currently no error recovery.

The data is entered in free-format and checked for validity before being accepted. You must provide the following data for each fixed node.

node number direction fixed, direction fixed...

For example:

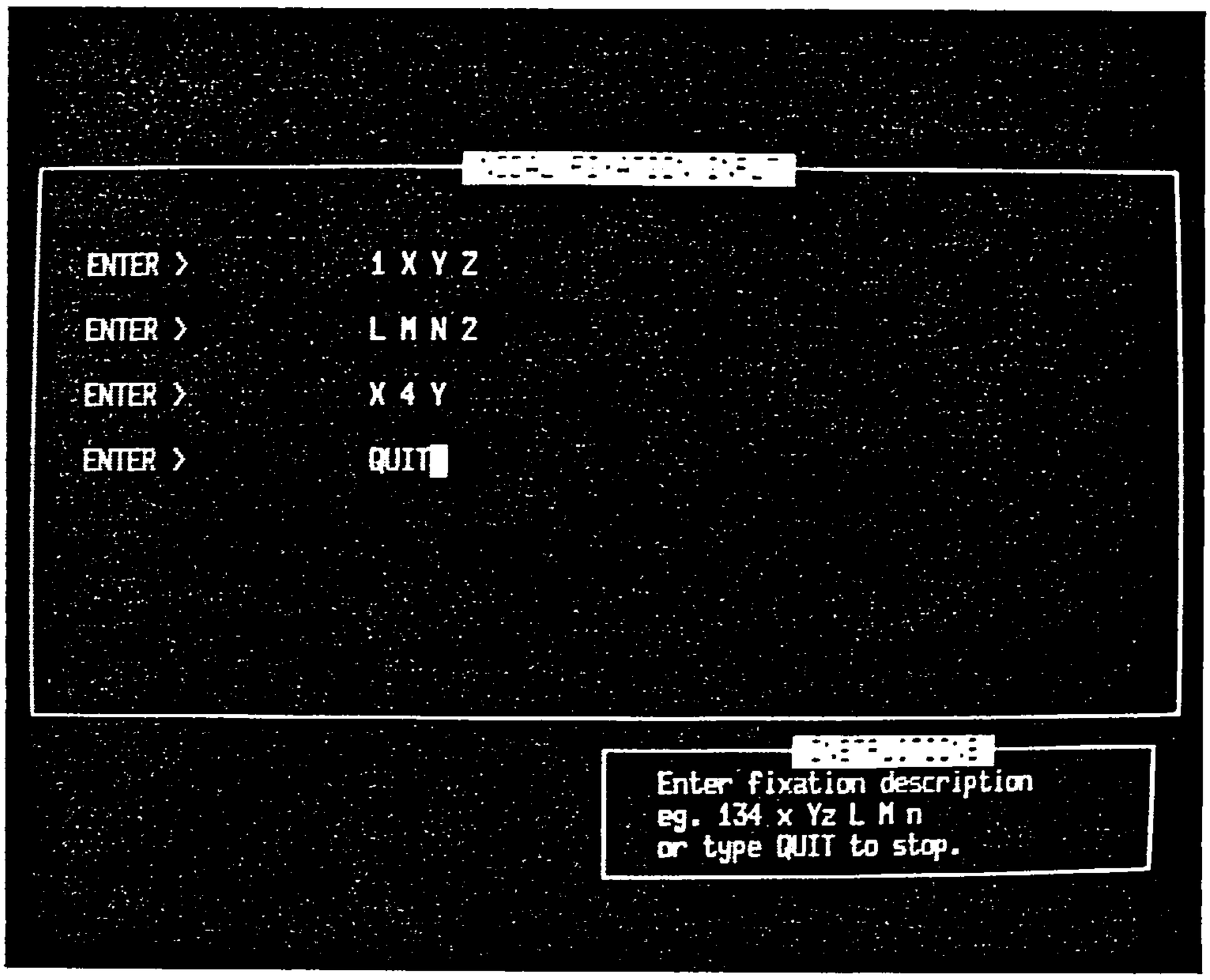


Fig 9. Example of NODFIX INPUT Display

In this example node 1 is fixed in the X Y and Z directions, node Z is fixed in the L M and N rotations and node 4 is fixed in the X and Y directions.

The output file for this program is called NODFIX.DAT and may be generated by the user by alternative VMS editors. In this case the file must contain the following in the following format.

```
line of text
line of text
node number direction
   .      .       .
   .      .       .
   .      .       .
```

For example:

```
          WEIGHTS ANALYSIS NODAL FIXATION DATA.
Node number followed by the direction of fixation.
            1 X
            2 X
            3 X
            4 X
            5 X
            6 X
            7 X
            8 X
            9 X
           10 X
           11 X
           12 X
           13 X
           14 X
           15 X
           16 X
           17 Y
           17 Z
           21 Z
```

Note that if a node such as node 17 in this example has more than one fixation they must appear seperately.

3.15 OPTIMISE

Very simply this command invokes STARS a structural analysis and redesign package which then carries out a predetermined type of optimisation on the structure.

This module will be site dependant as it was devised using the development version of STARS which is no longer supported at the College of Aeronautics.

This module must be used in conjunction with other modules - see module ordering.

3.16 PLOT and PLOT INPUT

PLOT_INPUT is an interactive module which prompts the user for plot orientation and copy information. PLOT actually executes the plot.

When you use the command PLOT_INPUT the menu in fig 10. unfolds as you choose your option.

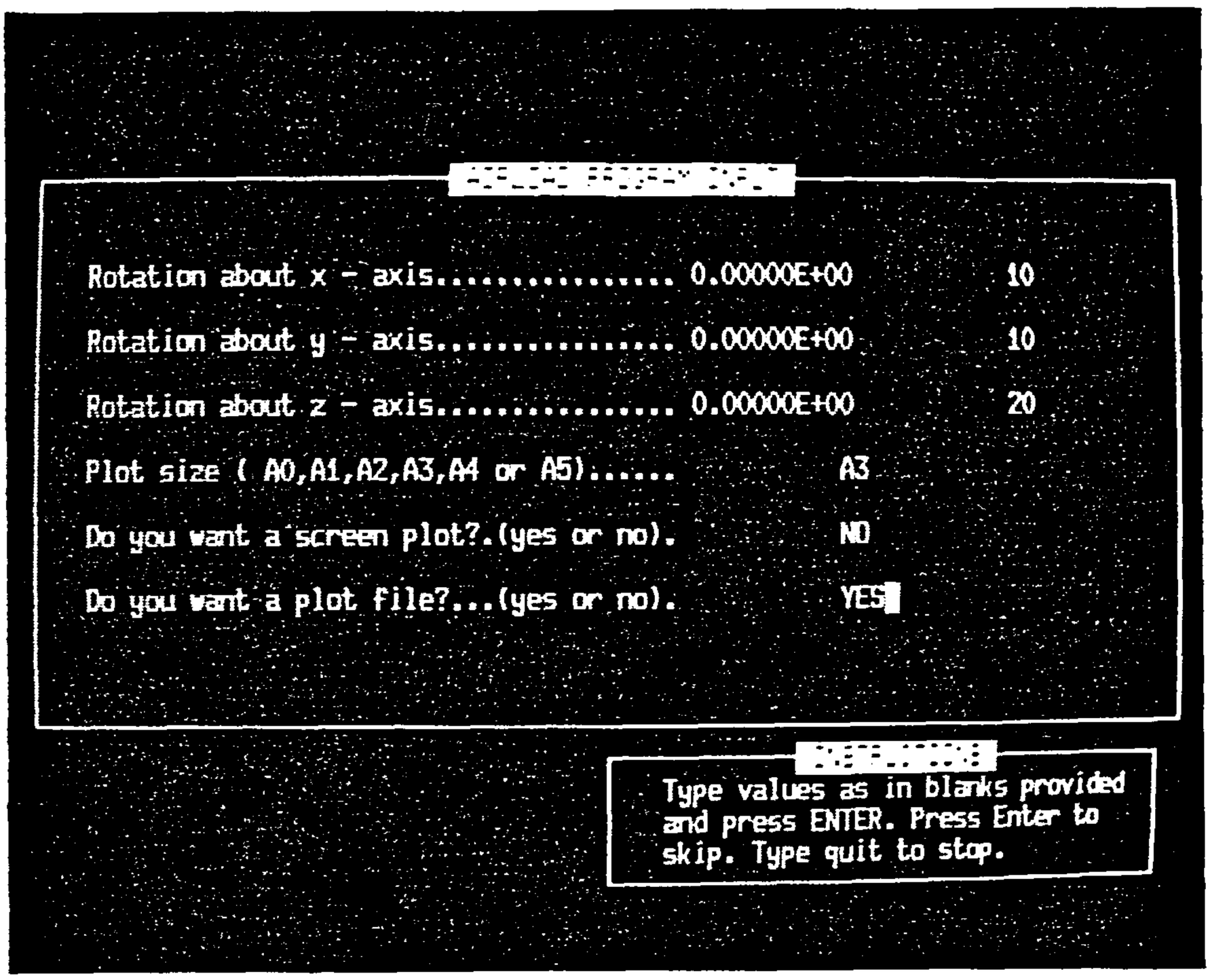


Fig 10. Example of PLOT INPUT Menu

In the example given in fig 10. the view is rotated about the axes to give the drawing shown in fig 11. When the PLOT command is used.

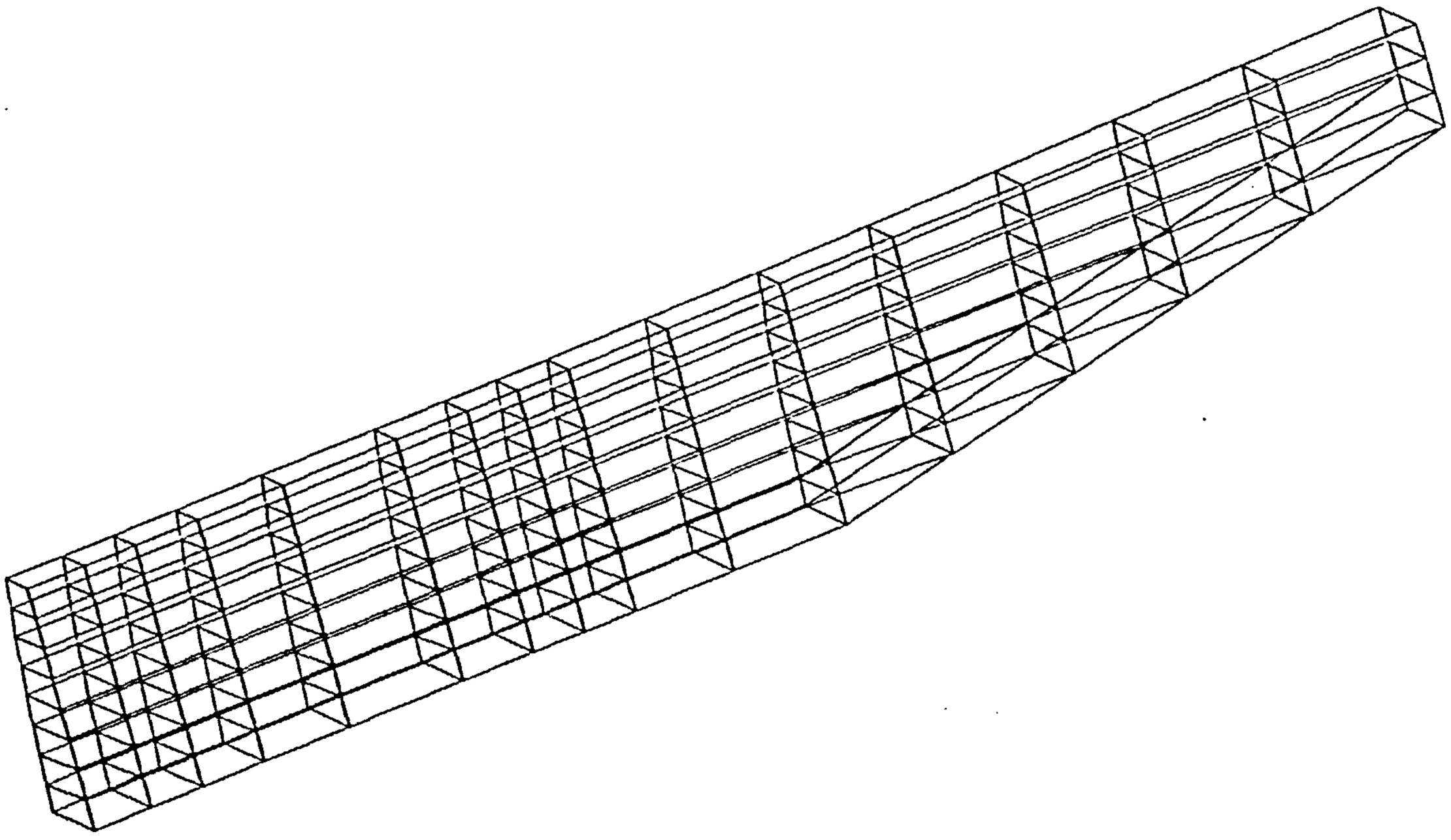


Fig 11. Example of PLOT Display

When you request a hard copy the PLOT command generates a plot file. PLOT uses the GINO graphics package at the College of Aeronautics. Having created the plot file another command called DISPLAY is used to obtain the actual hand copy from the plotter. On your site this may vary because your site may not support GINO or may have a different command to DISPLAY which does the same job.

To obtain a hardcopy at the C.O.A. you type in the following commands

```
DISPLAY
$_To device: BENSON
$_File:      PLOT_FILE

$_Is the plotter ready? (Y/N) Y
```

3.17 POST STARS

As with other modules relating to the structural optimisation package STARS this module is of limited use since STARS is not supported at the College of Aeronautics.

These modules were written mostly as demonstrators to show how a large sophisticated package could be included. The POST_STARS module conducted some data processing at STARS output which could then be analysed to obtain weight estimates.

3.18 RANGER

This module can be used instead of DESVAR_INPUT to compile design variable link data (see DESVAR_INPUT).

The input to this module is in the form of a file called RANGER.IN which must contain the following;

Design variable element range.

. .
. .
. .

For example:

```
19 88 94
20 95 0
21 96 102
22 103 0
23 104 110
24 111 0
25 112 118
```

In this example there are 7 design variables, 19 to 25. Design variable 20 contains the element 95 and design variable 25 contains the elements 112 to 118 inclusive.

3.19 RUN BEAMING8

This routine has many associated modules which create its input data, refer to the section on module ordering for more information. RUN BEAMING8 converts the aerodynamic loads calculated by AIRLOAD1 into nodal loads on the finite element grid.

Merely type in the command. For details of input refer to the WEIGHTS programmer's manual for full details. The following datafiles are used as input;

BEAMING.IN1 (aerodynamic loads)

BEAMING.IN2 (node geometry)

NODELOAD.DAT (node groups)

By using the interactive module NODELOAD_INPUT and AIRLOAD1 you need not be concerned about these files.

3.20 RUN WEIGHT STARS

This module invokes the structural optimisation package STARS. This module was devised as a proof of concept program and is no longer supported in the College of Aeronautics because support for STARS has been withdrawn. The STARS package on the C.O.A system is a development version and was very limited. Newer versions of STARS may require changes to be made to all STARS related modules.

3.21 SCAN

This module scans the weights database and extracts useful post-analysis data for modules such as ITEMIZE. The reason for this is to reduce the time spent by post_analysis modules sifting through data that is mostly only needed prior to analysis.

Merely type SCAN.

3.22 STARLINK

This module extracts data from the WEIGHTS database and creates data in a format suitable for the structural optimisation package STARS.

Refer to the section on module ordering.

3.23 STOP

This command causes the WEIGHTS invocation module to terminate. After the module stops you still have full use of all the WEIGHTS commands up to the time you log off.

3.24 SUMASS

This is a simple module which scans the post-processing sub set of the WEIGHTS database and retrieves the total predicted wing weight and displays it on the screen.

3.25 WMESH1
3.26 SUB WMESH1

The WMESH modules are outwardly alike and need the same input. WMESH1 allows you to generate a simple wing mesh whereas SUB MESH1 does the same thing but does some extra "house keeping" which allows it's output to be combined by module MERGER1 later.

The module is fully interactive but was one of the first modules devised and is very crude with respect to error trapping . A sample run follows and fig 12. shows the mesh generated by that sample run.

Sample Program Execution

Reverse characters indicate an input from the user at a VDU.

```
RUN WMESH

HOW MANY RIBS?
3
HOW MANY NODES ON RIB NO.      1
9
HOW MANY NODES ON RIB NO.      2
6
WHERE DOES THE          1TH STEP OCCUR?
2
HOW MANY NODES ON RIB NO.      3
6
STARTING FROM THE TOP OF THE ROOT RIB

FROM THE L.E. TO THE T.E. AND THEN
REPEATING THIS FOR THE BOTTOM; GIVE
THE COORDINATES OF THE NODES ON THAT
RIB.

REMEMBER THAT:
               X = OUTBOARD.
               Y = AFT.
               Z = DOWN.

GIVE THE COORDINATES OF NODE NO.       1
0 0 0
GIVE THE COORDINATES OF NODE NO.       2
0 1 .1
GIVE THE COORDINATES OF NODE NO.       3
0 2 .1
GIVE THE COORDINATES OF NODE NO.       4
0 3 0
GIVE THE COORDINATES OF NODE NO.       5
0 0 -1
```

```
GIVE THE COORDINATES OF NODE NO.      6
0 1 -1.1
GIVE THE COORDINATES OF NODE NO.      7
0 2 -1.1
GIVE THE COORDINATES OF NODE NO.      8
0 3 -1
DO YOU WANT TO MAKE ANY CORRECTIONS?
NO
WHAT IS THE TAPER RATIO?
1
WHAT IS THE SWEEPBACK ANGLE
0
WHAT IS THE          1TH RIB PITCH?
5
WHAT IS THE          2TH RIB PITCH?
5
```

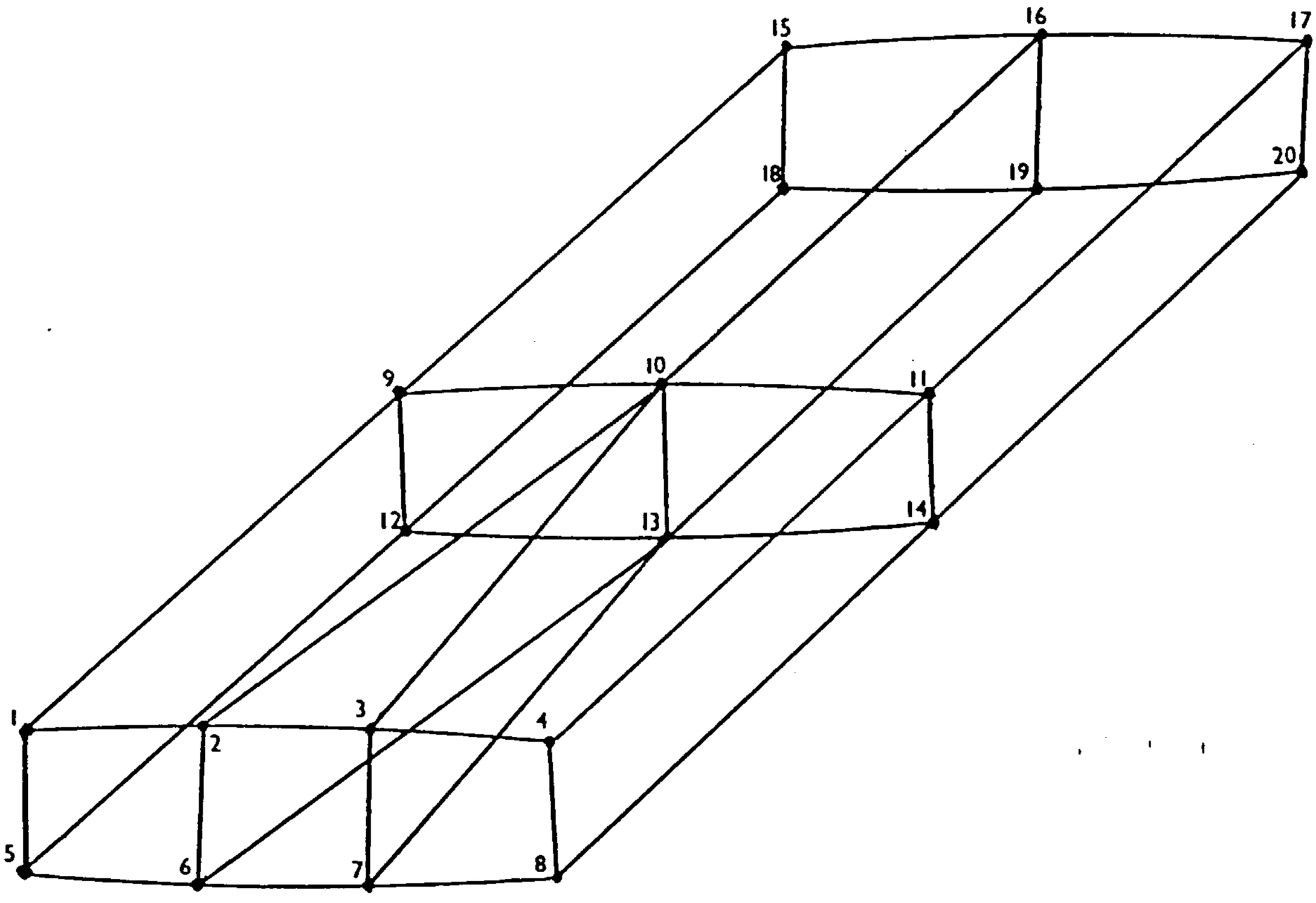


Fig 12. SAMPLE MESH

3.27 WMESH2

This module generates a wing mesh. The module WMESH2_INPUT prompts you interactively for the input data, and this is the most painless way to compile the input data to module WMESH2.

In case you want to use the standard editor a brief description of the input data needed is given here but refer to section 3.28 for more detailed descriptions of the meanings.

The datafile name is GEOM2.IN and it must contain;

Number of Ribs

Number of Defined Ribs

Sweep flag (1 for L.E. sweep 0 for T.E. sweep)

* Rib pitch angles

* Wash out angles

* Dihedral angles

* Sweep angles

*

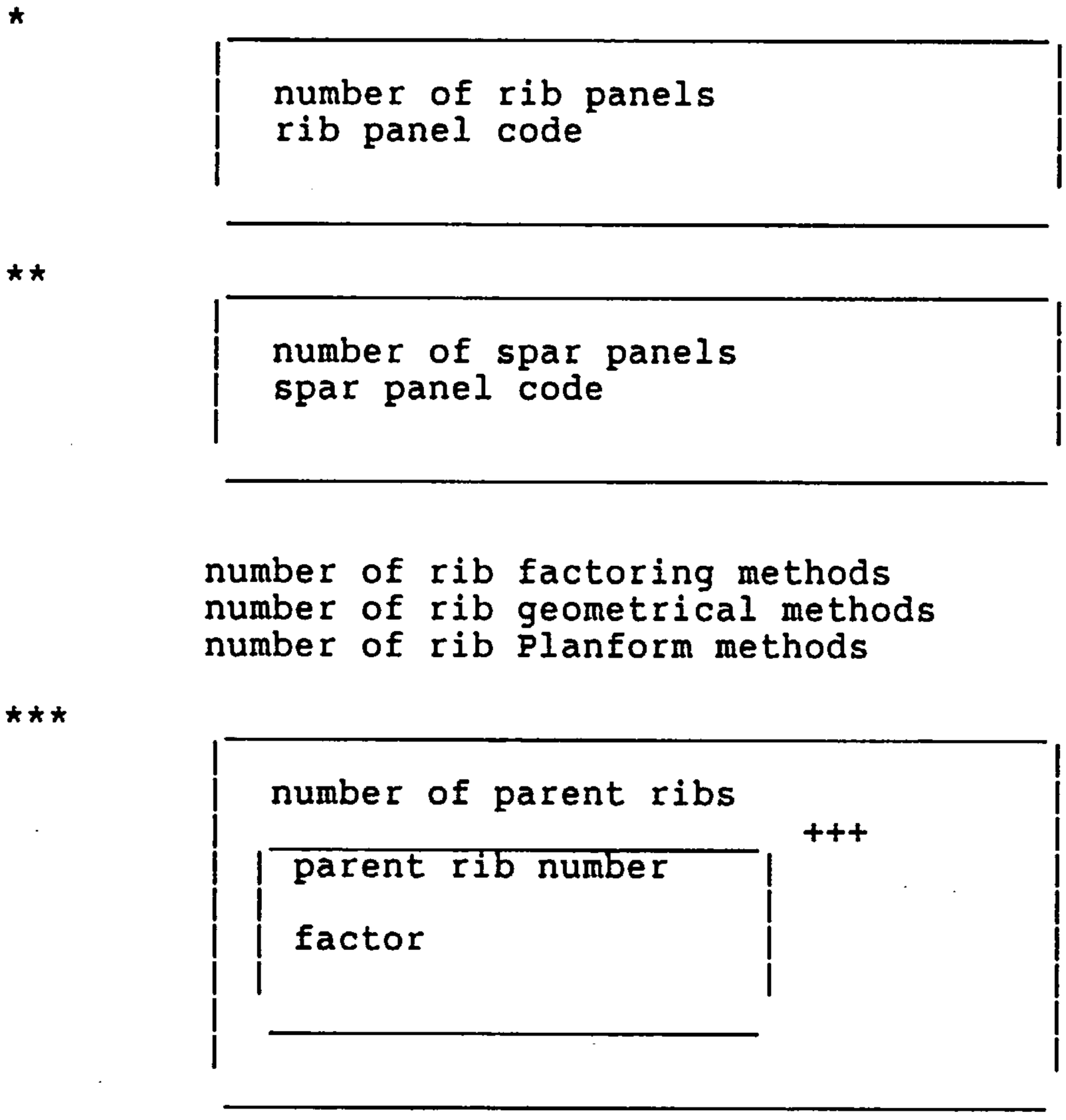
```
+-------------------------------------------+
|                                           |
| number of nodes,                          |
| number of top nodes,                      |
| number of bottom nodes,                   |
| rib configuration                         |
|                                           |
+-------------------------------------------+
```

*

```
+-------------------------------------------+
|                                           |
| rib number,                               |
| X, Y, Z coordinates for each node         |
|                                           |
+-------------------------------------------+
```

*

```
+-------------------------------------------+
|                                           |
| number of stringer starts                 |
| number of stringer runoffs                |
| positions of starts +                     |
| positions of runoffs ++                   |
|                                           |
+-------------------------------------------+
```



+ for each start
++ for each runoff
+++ for each parent rib.
* repeat for each rib.
** for each rib bay.
*** for each factoring method.

With the data compiled merely type in the command to generate the mesh.

The wing mesh data is compiled from information you are likely to know at an early stage in the design, such as wing section geometry, skeletal structural geometry, plan form etc.

3.28 WMESH2 INPUT

This is another proof of concept module for the interactive compilation of wing mesh data. A better module is currently under development which will eventually replace all the input modules.

Some options are optional, you need not use them and default values will then be used. Many must eventually be answered before the mesh generator will generate a complete wing mesh. This means that you must answer all the compulsory options but you need not do so in one session, you may save your session at this stage and return to it later. More of this later.

When the command SUB_WMESH1 is used the VDU will clear and the following menu will appear.

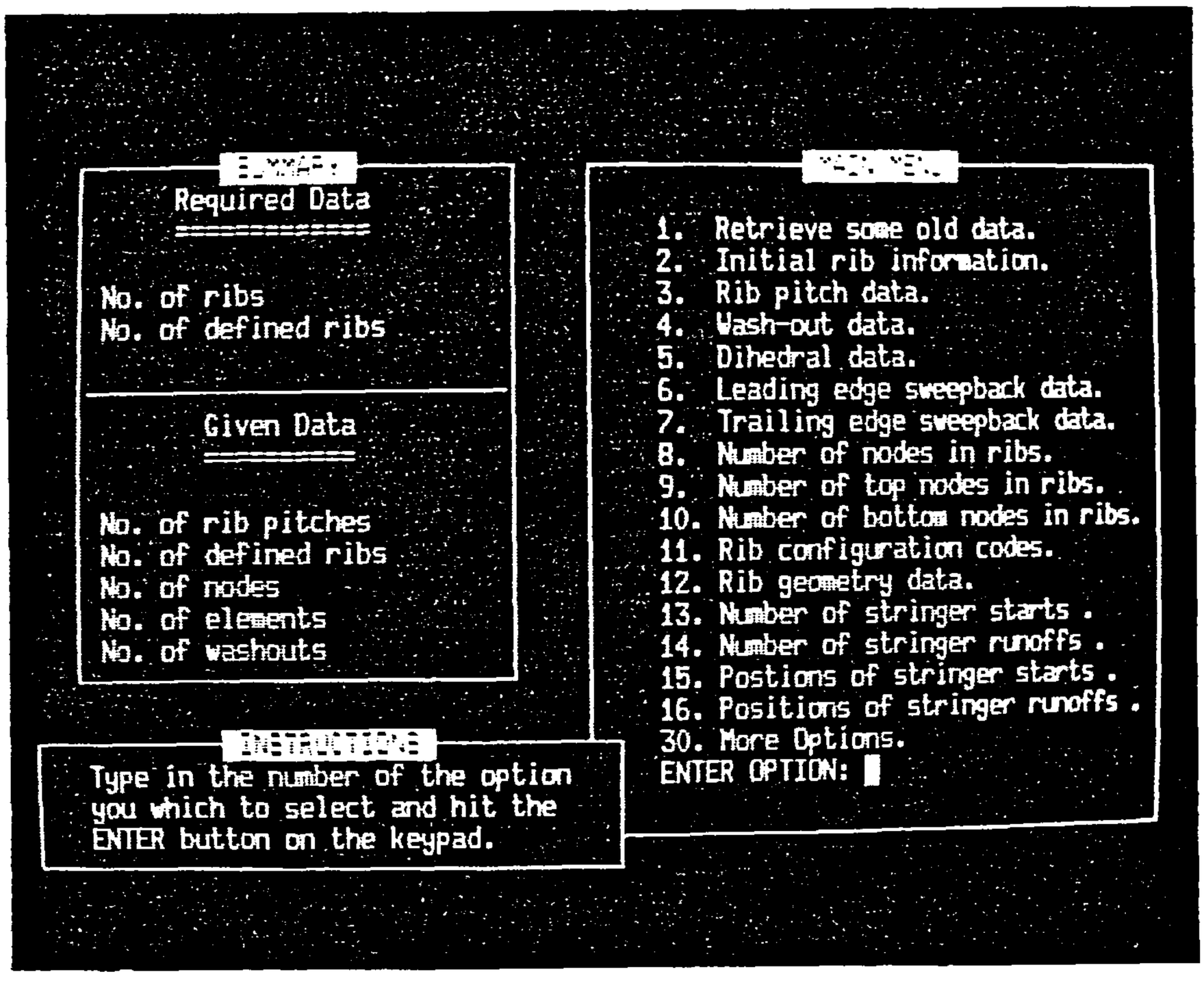


Fig 13. Example of WMESH2 INPUT Main Menu Display

3.29.1 MAIN MENU BOX

The box labeled "Main menu" contains 17 options including an option to obtain the next set of options. If option 30 is selected by typing the number in you will get the rest of the options like this:

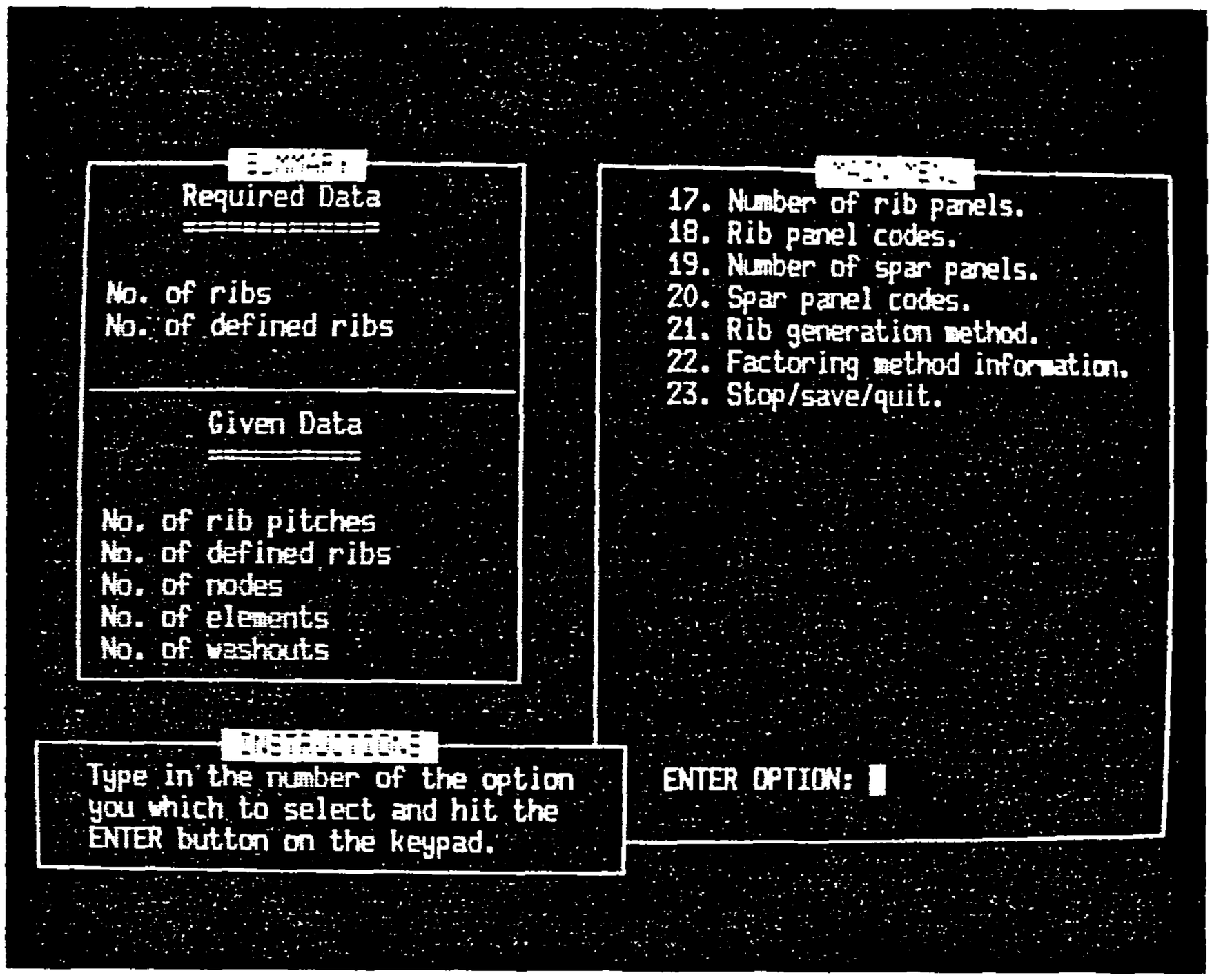


Fig 14. Main Menu Display Part Two

3.29.2 SUMMARY BOX

Ignore the summary box till you become more familiar with the module. It gives an indication of the amount of information you have entered so far. This feature is currently very crude and will be improved in the module currently under development.

3.29.3 INSTRUCTION BOX

There will always be an instruction box in some part of the screen to make the module self explanatory.

3.29.4 ERROR BOX

Most of the data you enter is checked by the module. If a mistake is spotted an error box with an appropriate message indicating what the error was appears as shown in fig 15.

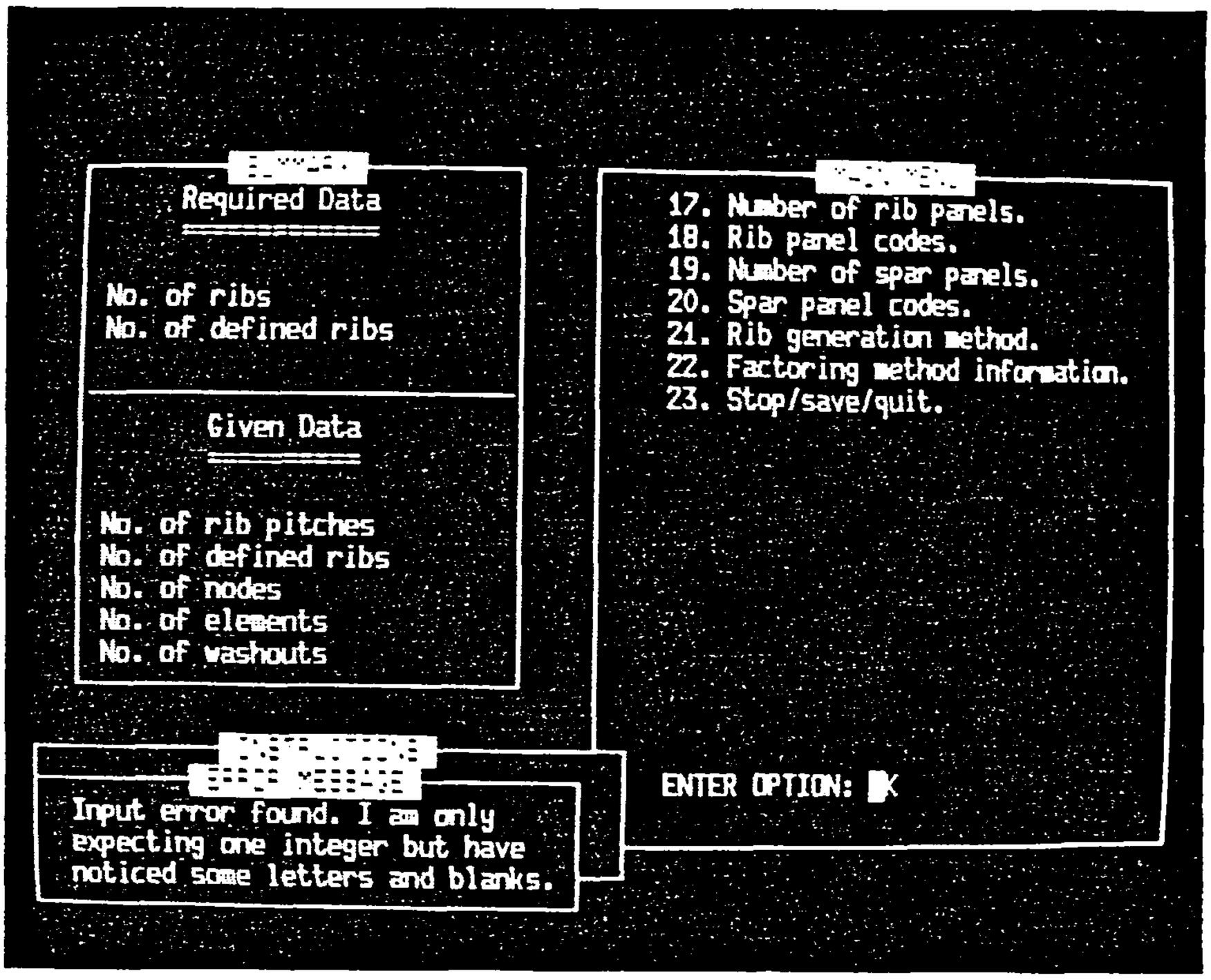


Fig 15. Example of Error Message Display

3.29.5 SAVING DATA and ERROR RECOVERY

You may select option 23 from the main menu at any stage of your data input, in fact it is a good idea to do so and this is why, should the computer break down or you make a catastrophic mistake you can start from the moment you chose option 23. When you choose option 23, this is what appears on the screen.

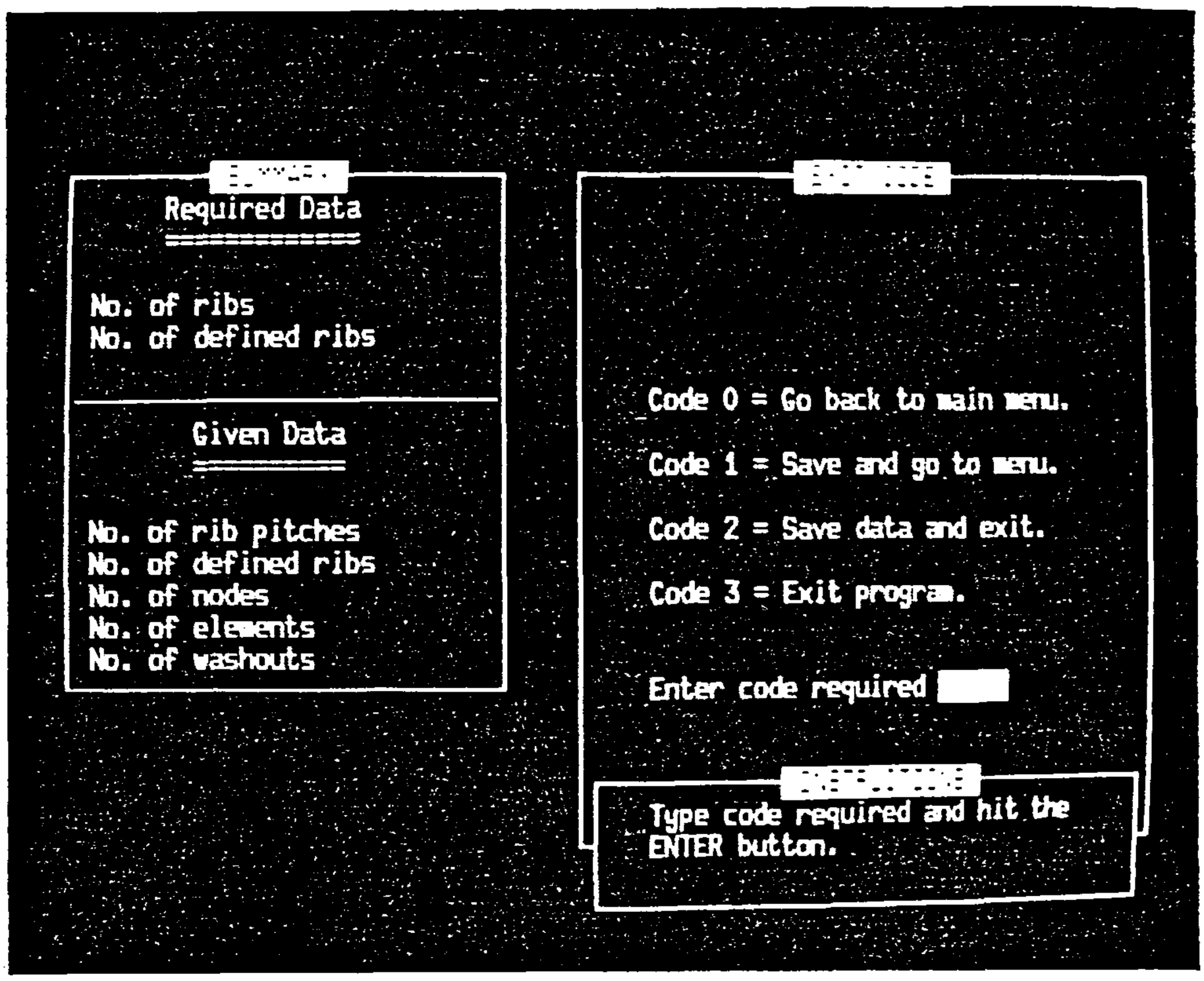


Fig 16. Example of Option 23 Display

The choices here are fairly obvious. Code 0 allows you to slip back to the main menu, this option was provided mainly to cover the case of accidentally choosing main menu option 23.

Both options 1 and 2 save all the data you have entered so far onto the WEIGHTS database updating any old data at the same time. The difference between them is that one causes an exit from the program whilst the other causes a return to the main menu.

Finally option 3 causes an exit from the module without modifying the database. This is to cover the situation where you have gone drastically wrong. The usual interrupts such as Control-C or Control-Y work too.

3.29.6 OPTION 1 Retrieving Old Data

This is option 1 of the main menu and is used to instruct the program to read the WEIGHTS database and collect any data you may have entered and saved previously.

Note: The modules in this development version of WEIGHTS use sequential files but in future the random access database portion of the WEIGHTS database will be used. For the moment though if you are familiar with VMS you may do some manipulation of the sequential files. The file used by this module is GEOM2.IN

When this option is used the screen goes blank till the data has been retrieved and the main menu reappears. After its completion, any menus with default values usually set to zero will contain the retrieved data as defaults.

3.29.7 OPTION 2 Initial Rib Information

When this main menu option is chosen the following appears;

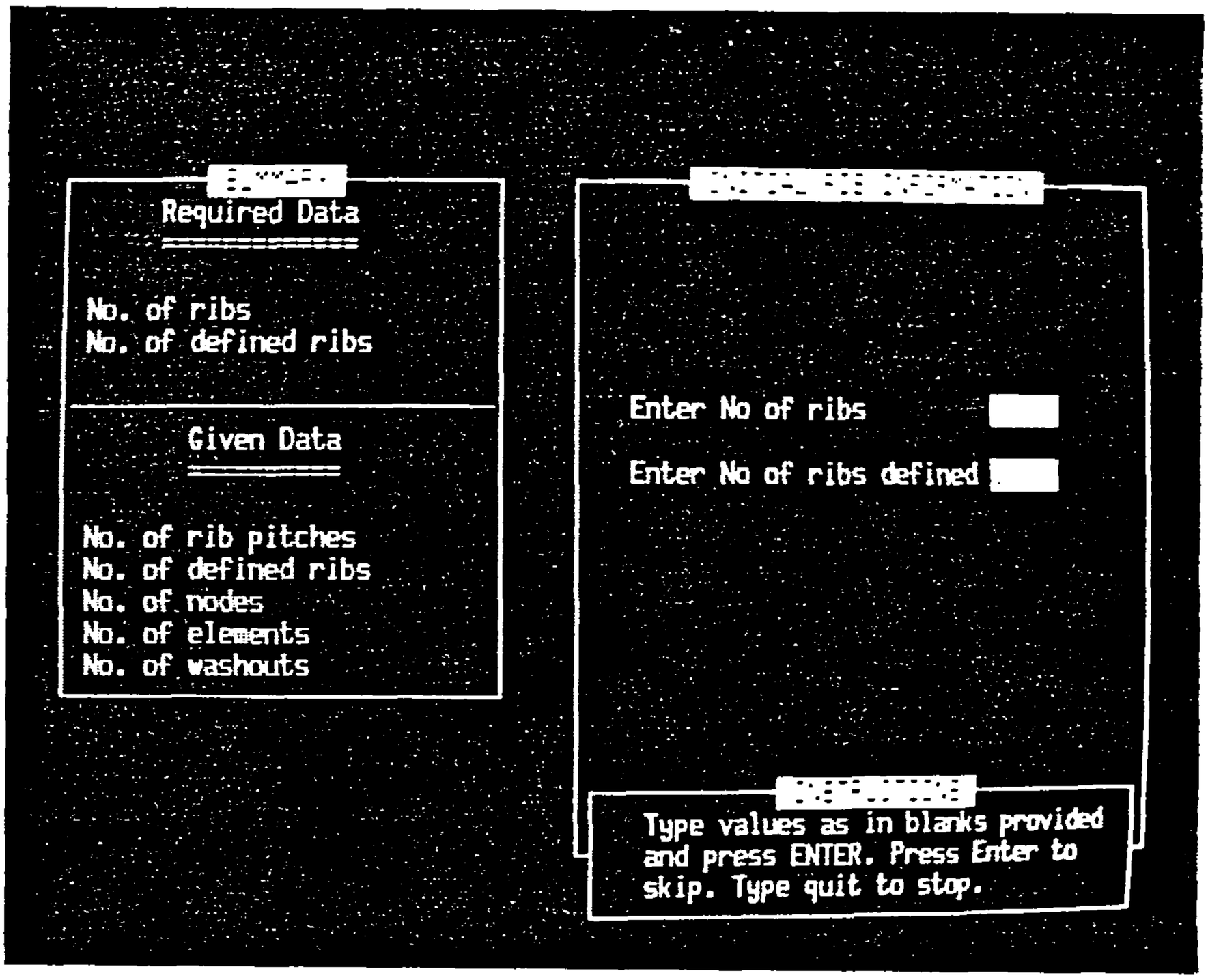


Fig 17. Example of OPTION 2 Display

The total number of ribs you want is fairly obvious but "defined" ribs require some explanation. This is a facility which allows the program to generate rib section geometry from limited information by exploiting factoring information in option 22 to generate the other ribs.

Say you give it the geometry for only one rib, and let that rib be the root rib. The other ribs are generated by factoring the geometry of that root rib to suit. That root rib you gave the program is called a 'defined' rib. You could have chosen to define more than one rib and make other ribs the factored combinations of one or more of those defined ribs.

3.29.8 OPTION 3 Rib Pitch Data

When you choose option 3 the following appears on the screen.

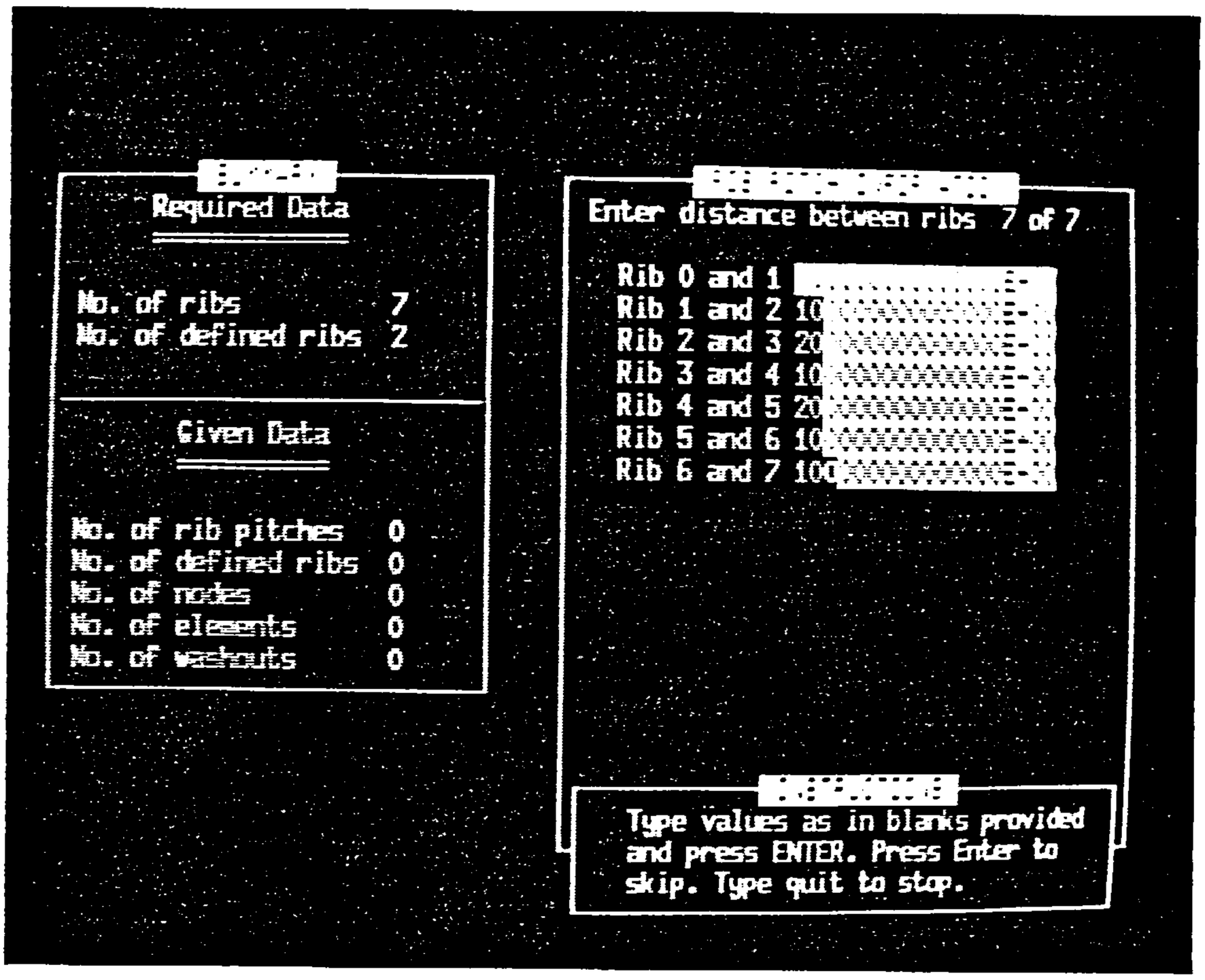


Fig 18. Example of Option 3 Display

In this case rib pitch is defined as the distance between the top corner, leading edges of two ribs (see fig 20. position A) and start with the centre rib as shown in fig 21.
e.g.

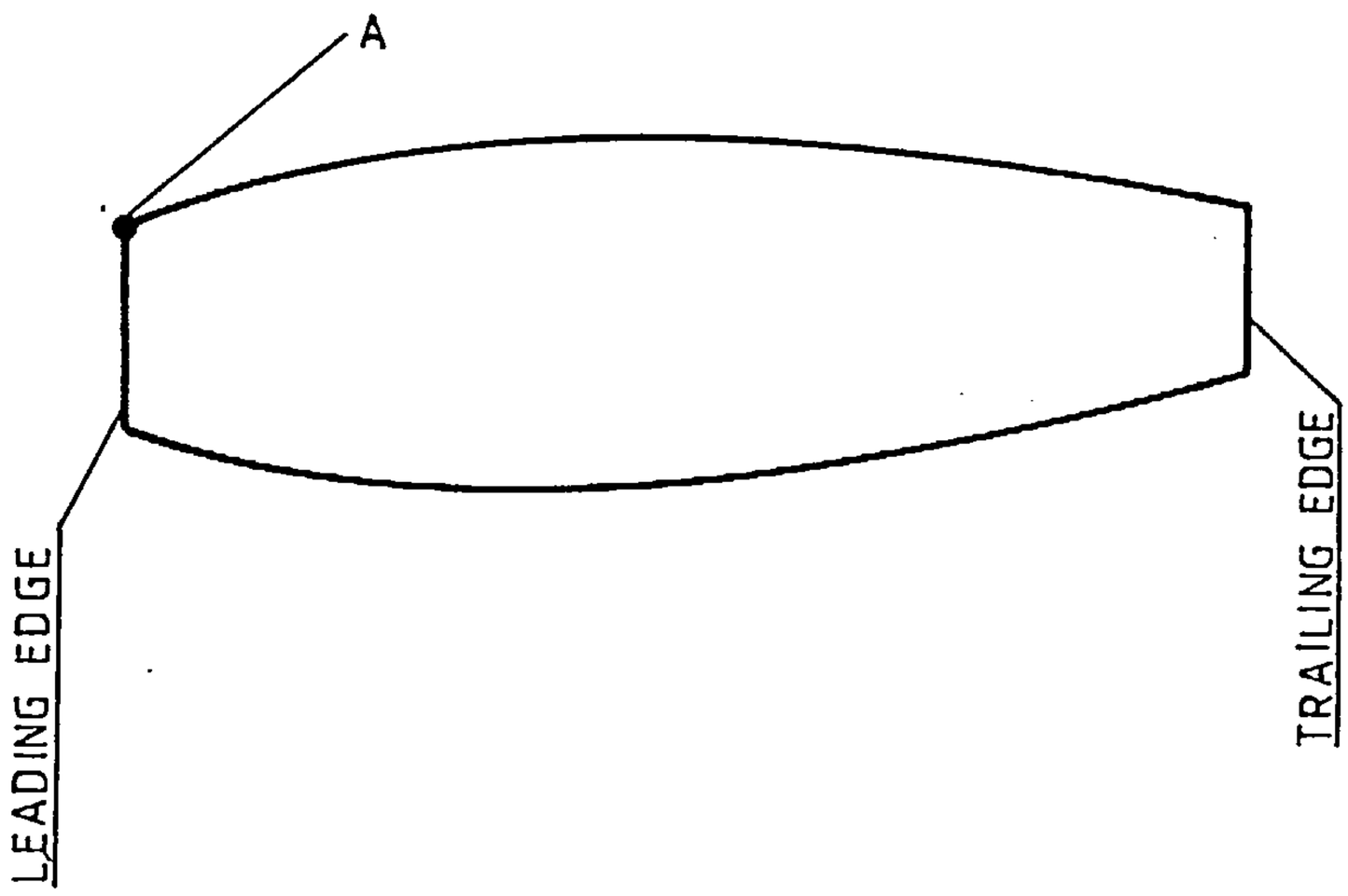


Fig 20. Rib Reference Position A

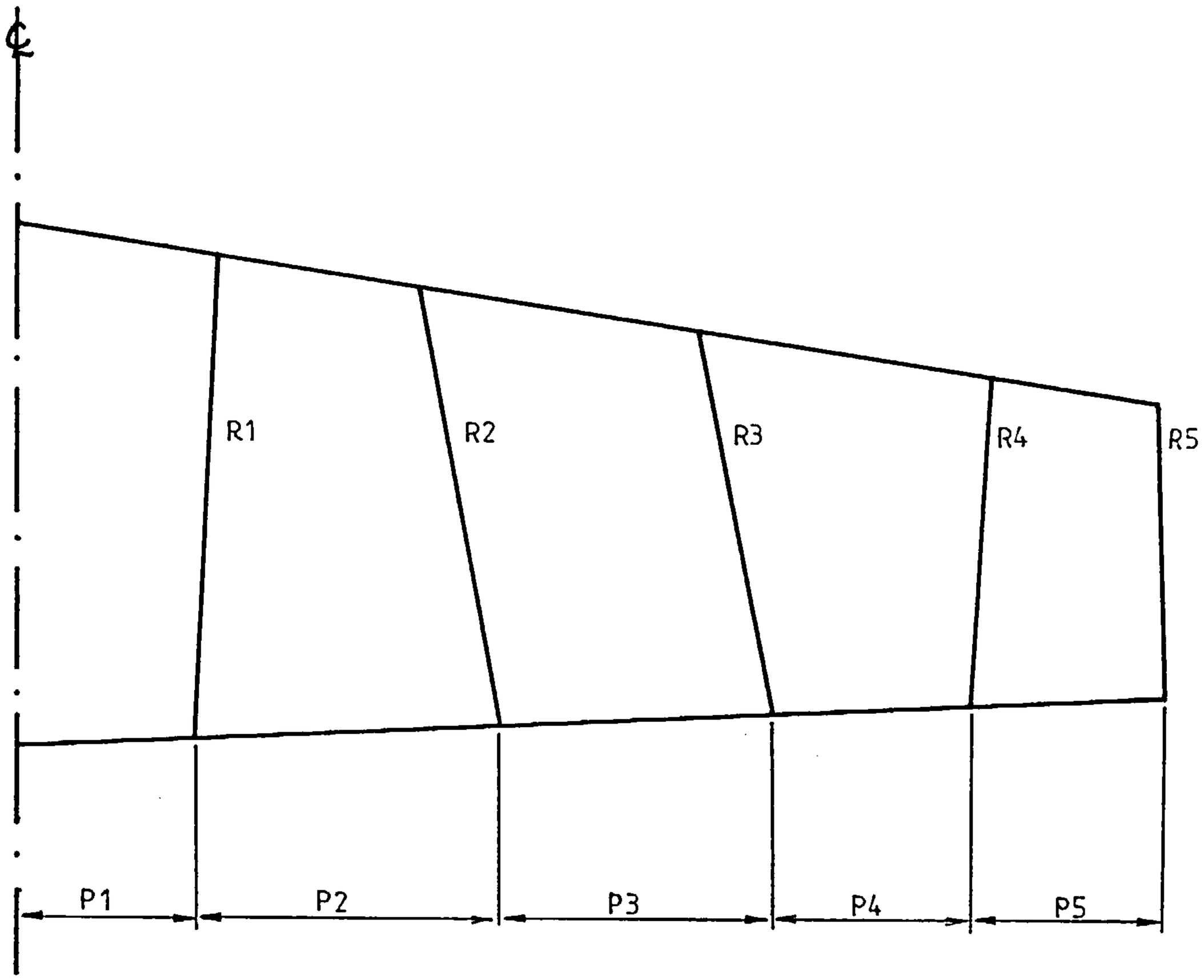


Fig 21. Rib Pitches

In the case of a carry through (continuous) wing, rib pitch 1 will be zero but would have some other value with some sub structure attachments for compicated frame structures, as shown in figure 22.

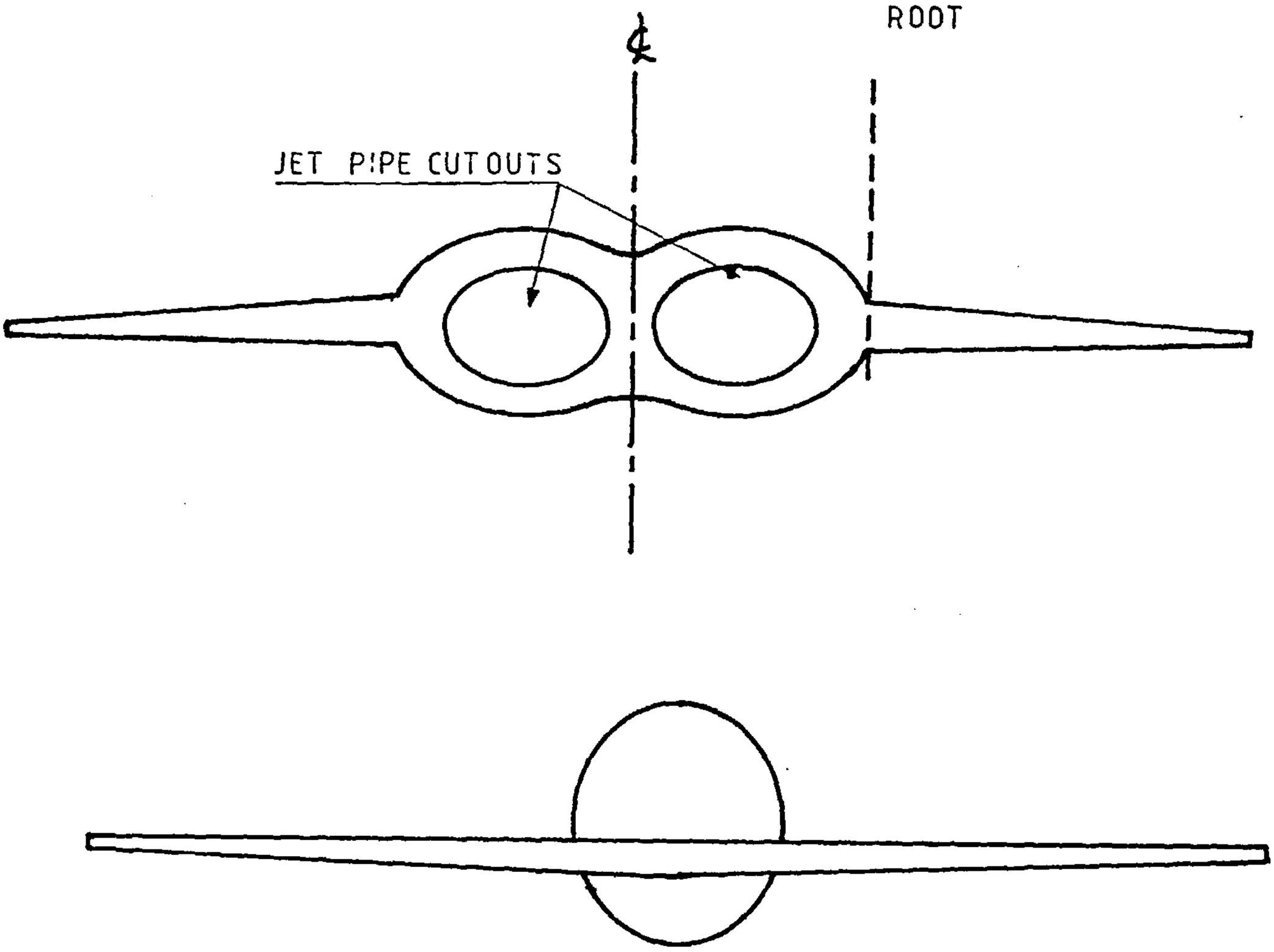


Fig 22. Carry through Wing and Broken Wing

3.29.9 OPTION 4 Wash-out Data (optional default = 0)

When you choose this option the following appears on the VDU. If you bypass this option all washout values are set to 0 zero.

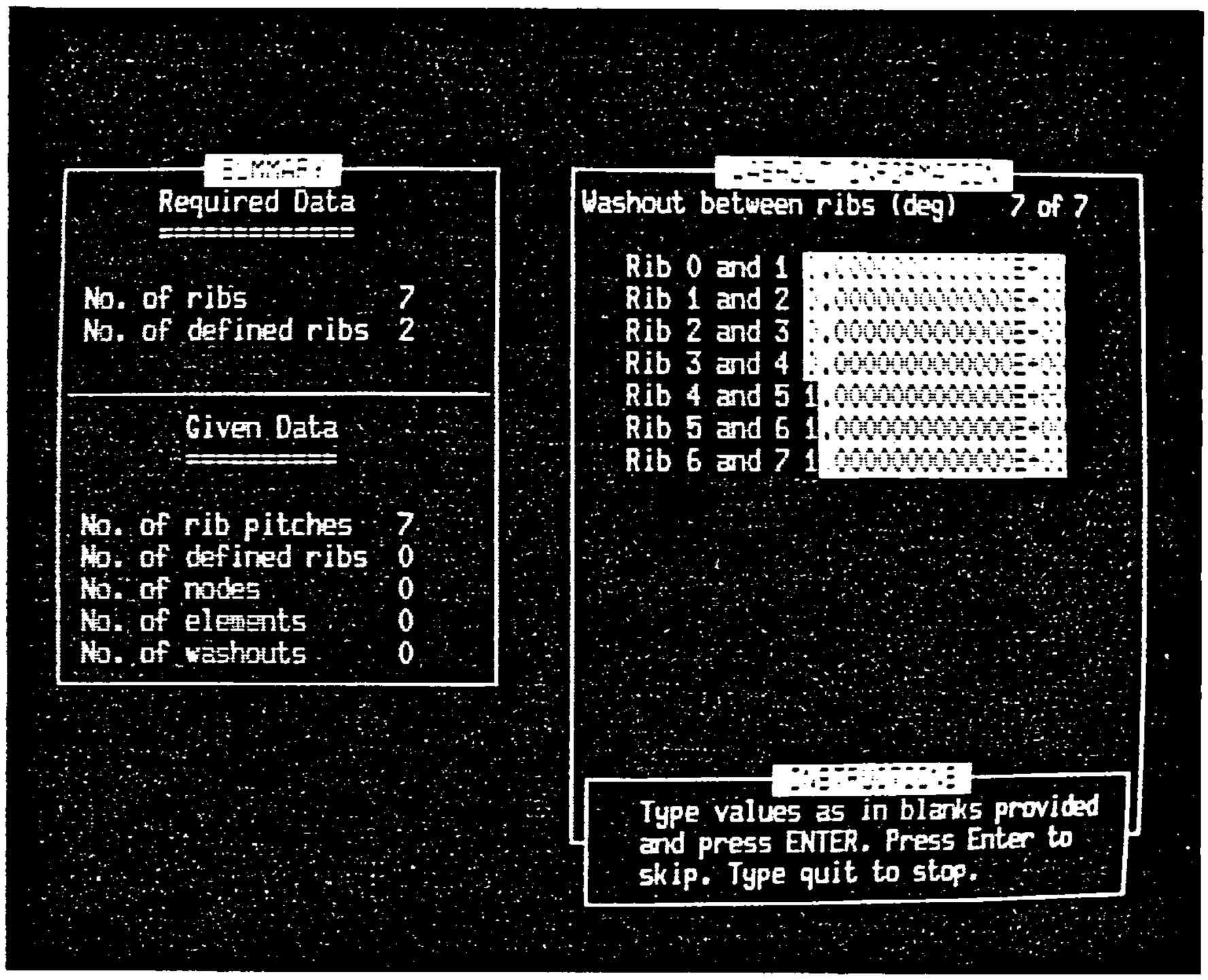


Fig 23. Example of Option 4 Display

This option works in the same sense as the rib pitch data. i.e. the top, leading edge corner of the ribs are used as a datum as shown in fig 24.

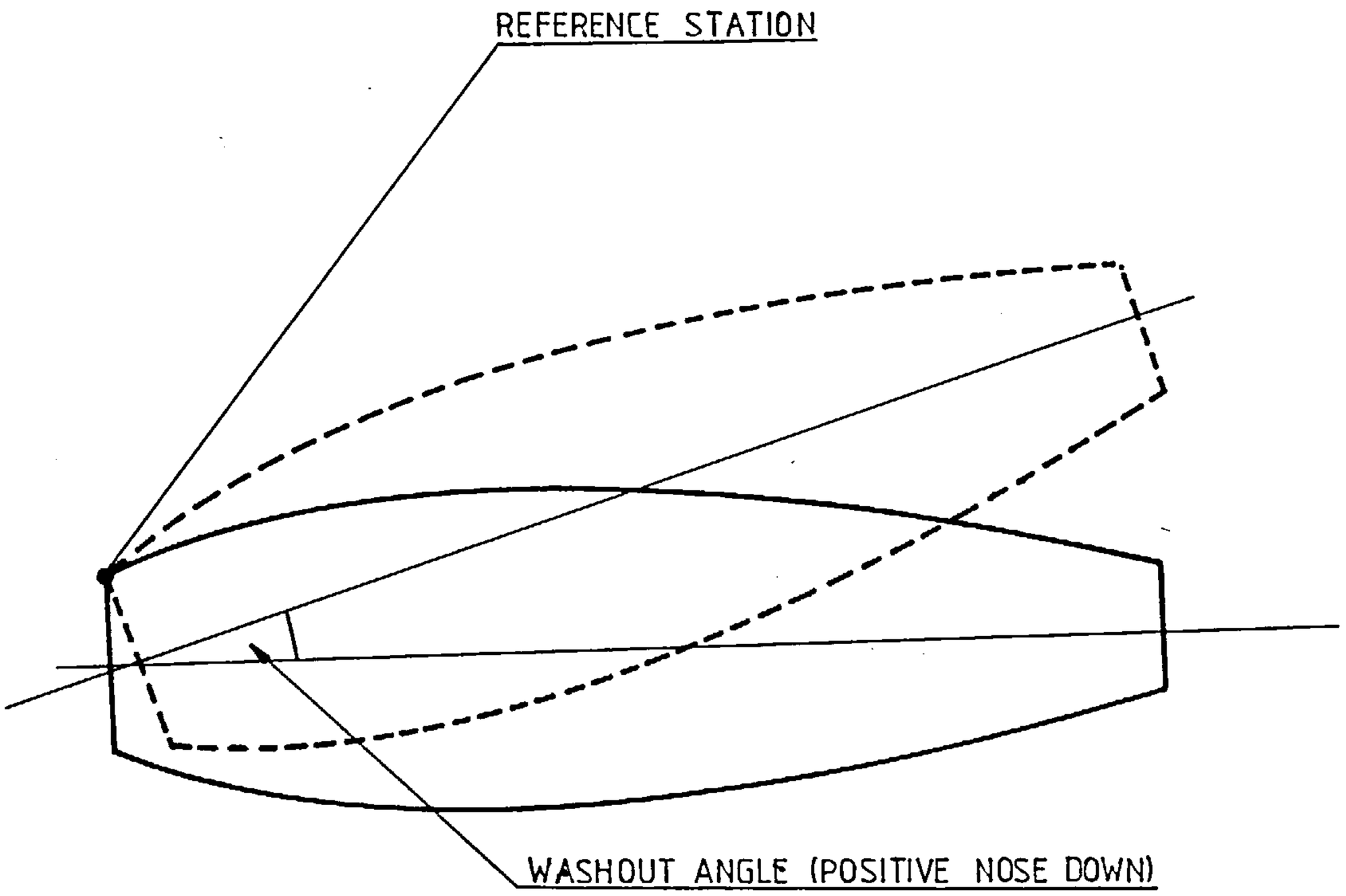


Fig 24. Wash-out Angle Reference Station

The wash-out increment angles are required in degrees and accumulate at each rib. For example if you enter values 0 a rib1, 0.1 at rib2 and 0.2 at rib3 then the total wash-out at rib3 will be 0.3 degrees.

3.29.10 OPTION 5 Dihedral Data (optional default = 0)

On selecting this option the following appears on the VDU. If you bypass this option all dihedral angles are set to zero.

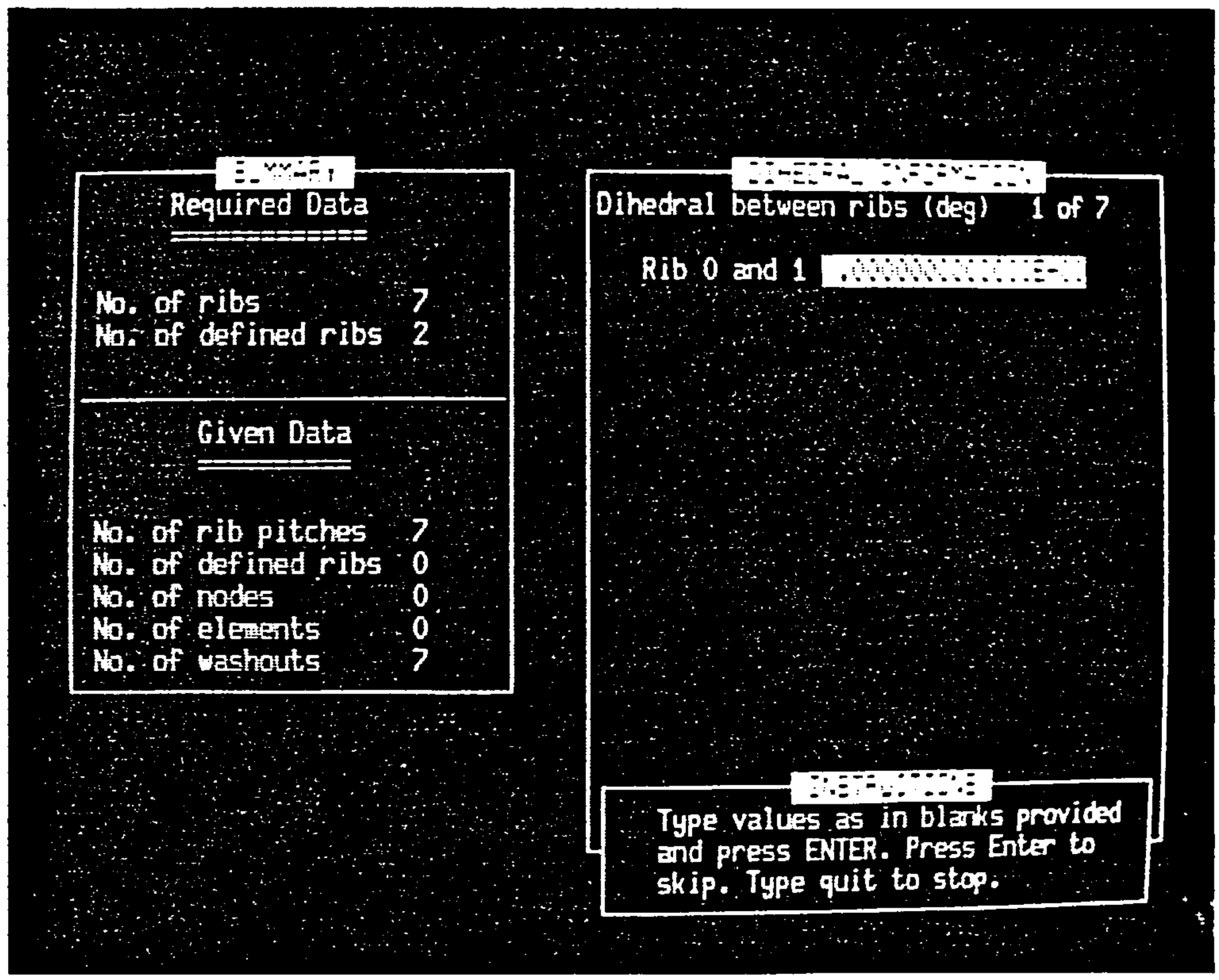


Fig 25. Example of Option 5 Display

The leading edge, top corner of ribs is again used as the datum. The values you give are the incremental values of dihedral at each rib, which means you can enter odd shaped wings as shown below.

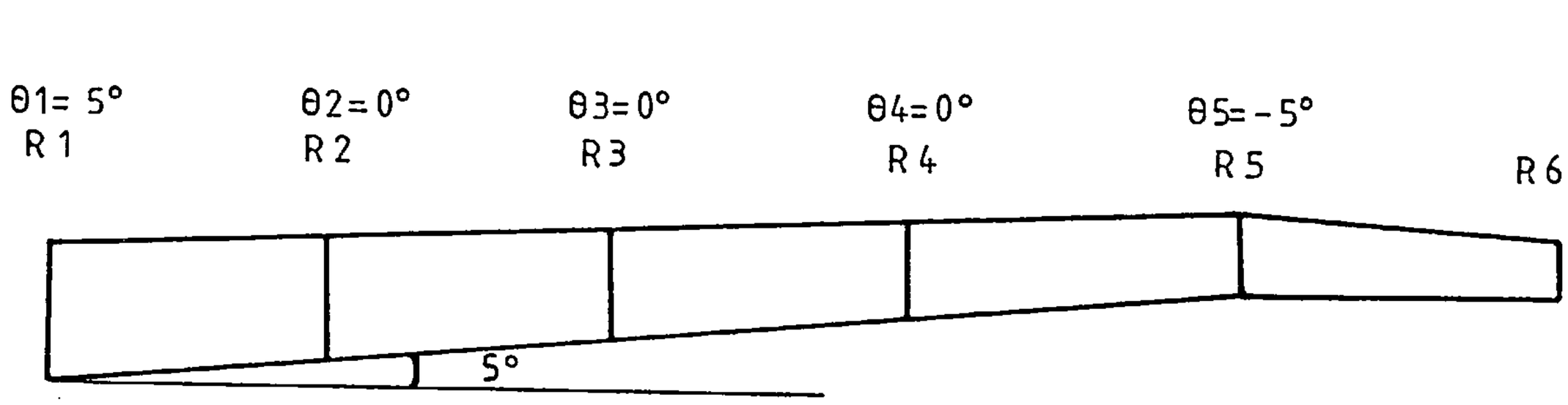


Fig 26. Example of Dihedral Data

3.29.11 OPTION 6 Leading Edge Sweep Back (optional default = 0)

The leading edge, top corner of ribs are the datum here. The values required are absolute, and when you choose this option the following appears.

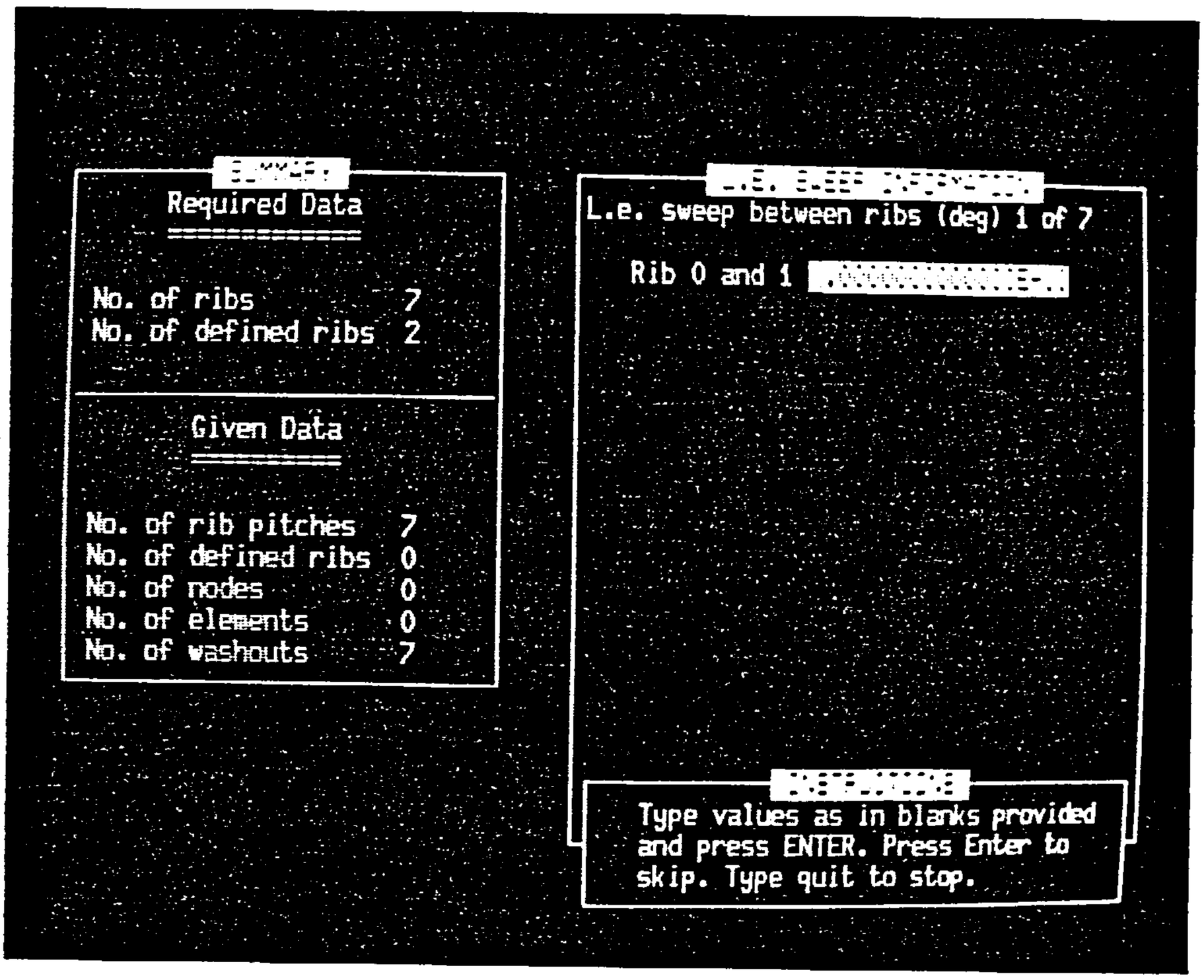


Fig 27. Example of Option 6 Display

You may obtain various wing shapes by varying the numerical values as shown below.

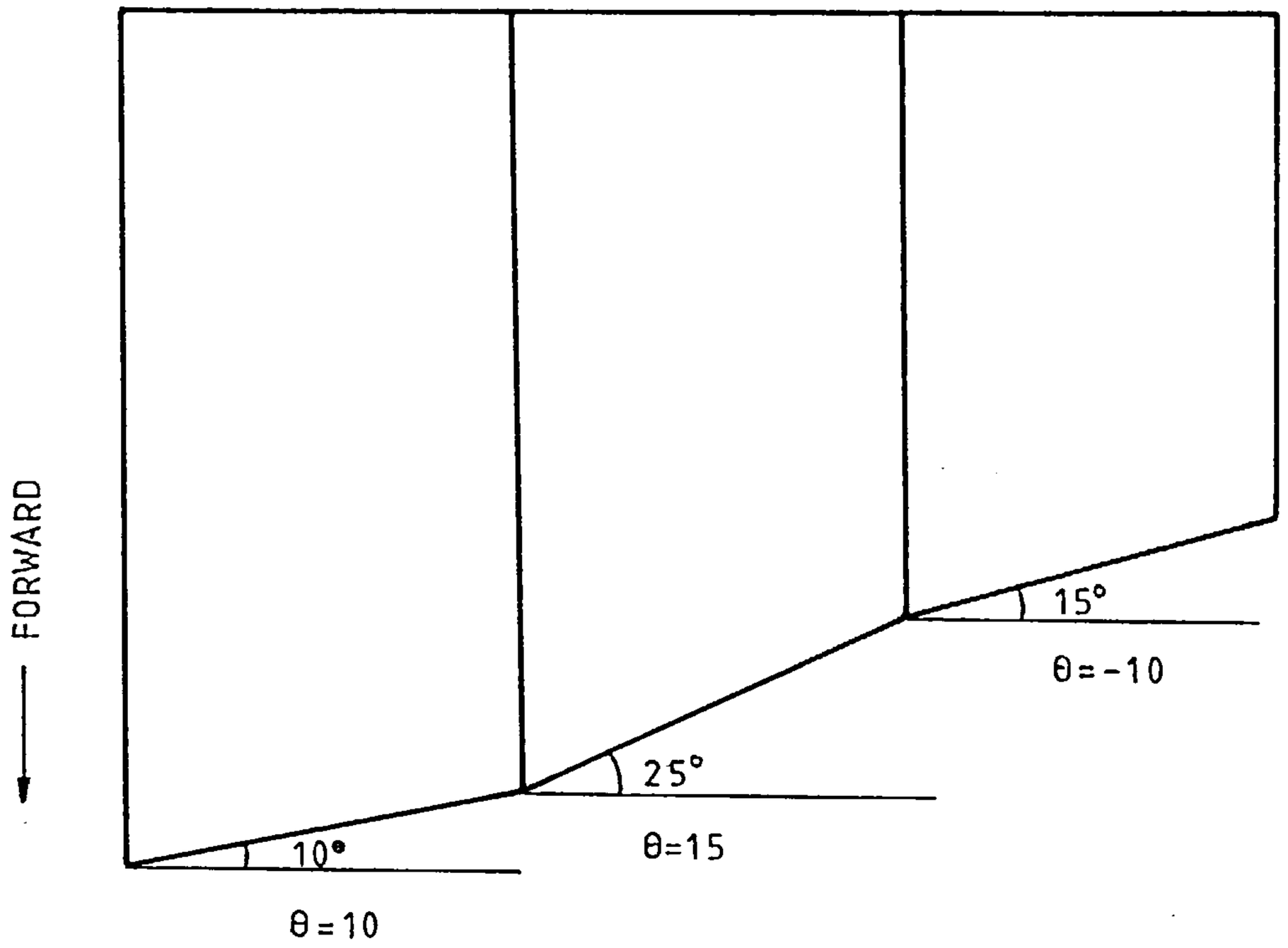


Fig 28. Example of Wing Shape

If you bypass this option all sweep angles are set to zero.

3.29.12 OPTION 7 Trailing Edge Sweep Back (optional default = 0)

This has the same effect as option 6 except that the datum used here is the trailing edge, bottom corner of ribs. The same values input in the example shown in 3.27.11 results in the wing shown below.

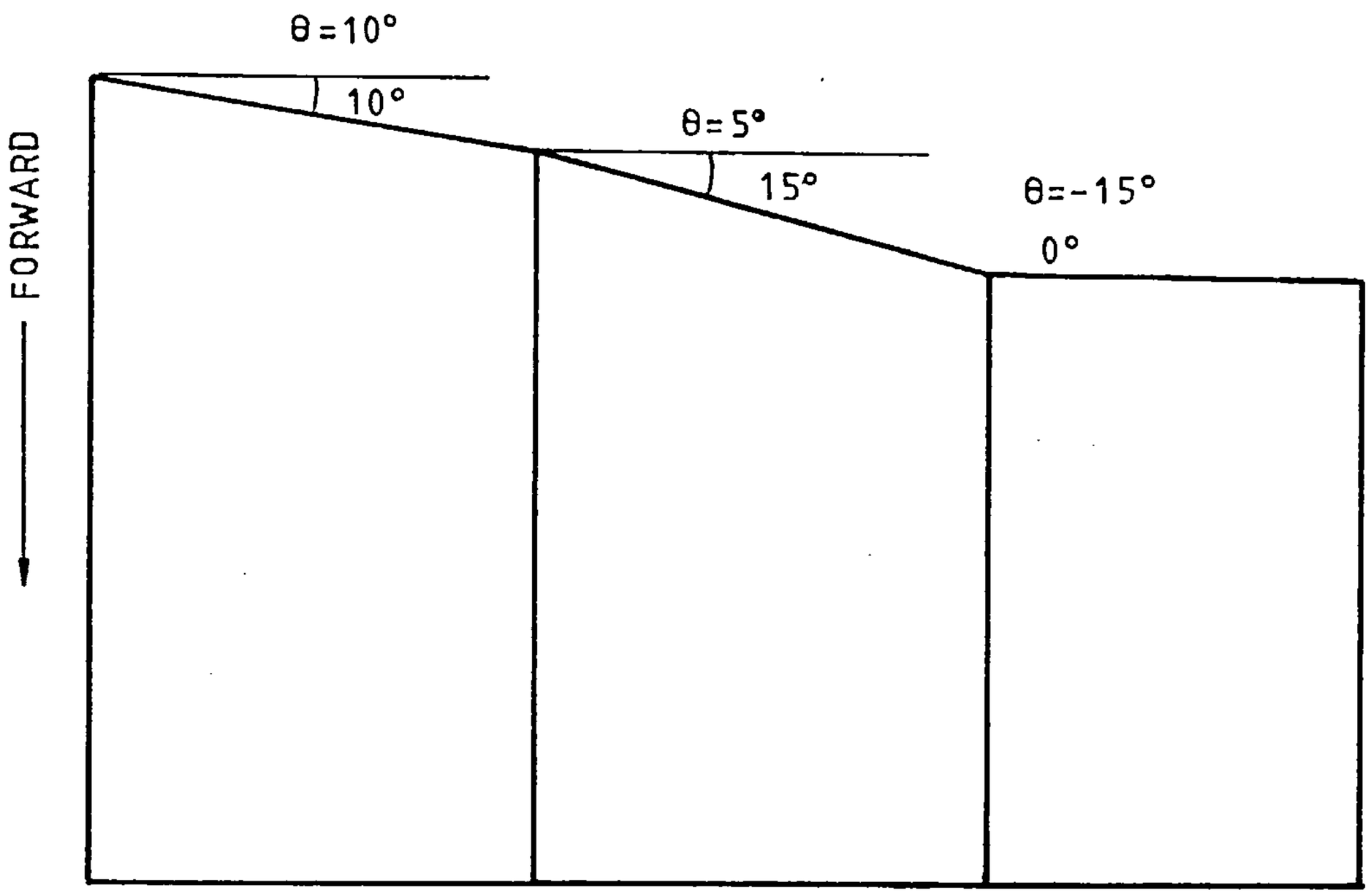


Fig 29. Example of Option 7 Wing Shape

Bypassing this option leads to zero values.

Note: May be positive for sweep back or negative for forward sweep.

3.29.13 OPTION 8 Number of Nodes in Ribs

The following appears on your screen when this option is chosen.

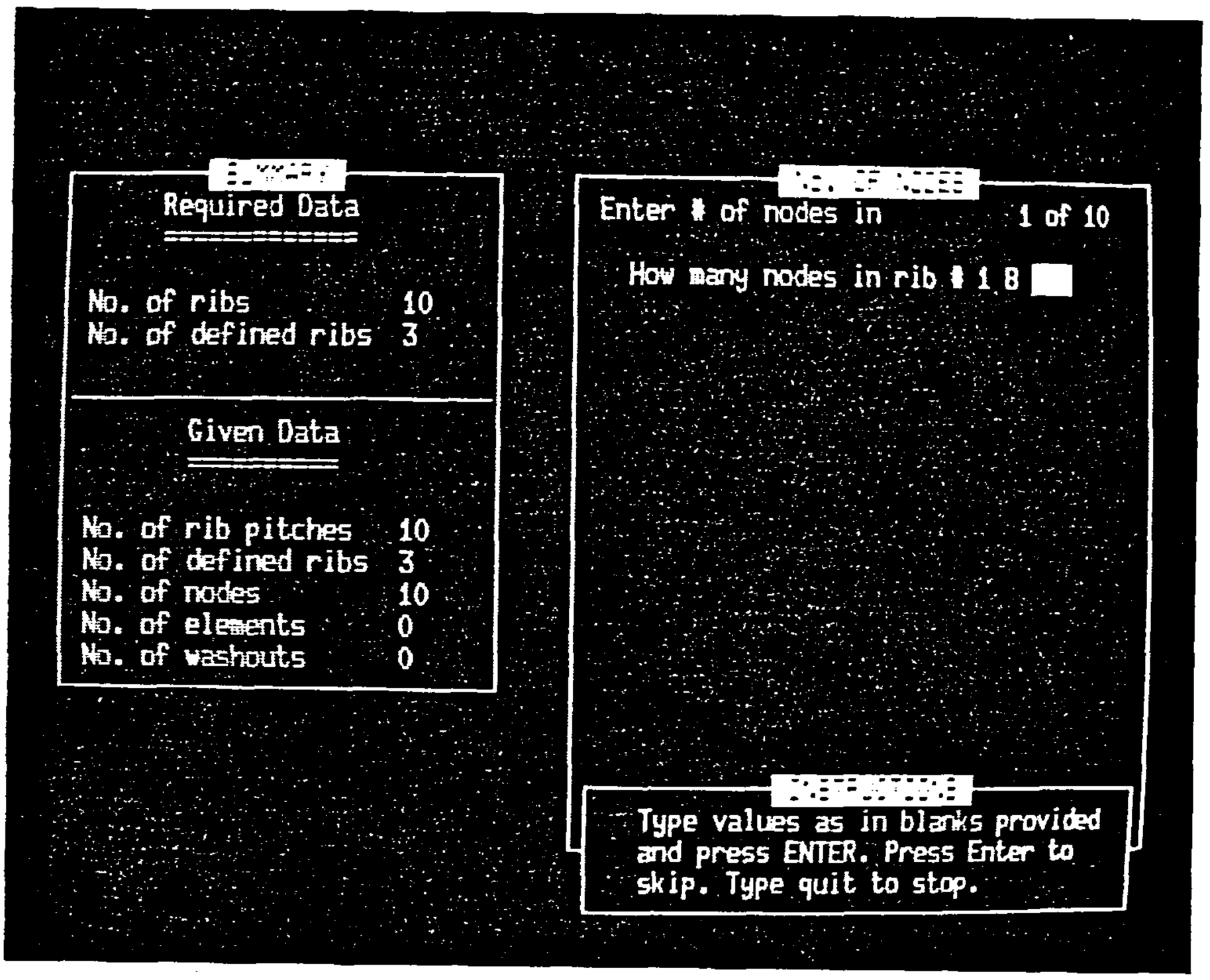


Fig 30. Example of Option 8 Display

What you are required to enter here are the number of finite element nodes which each rib is to contain. For example the rib shown below has 8 nodes.

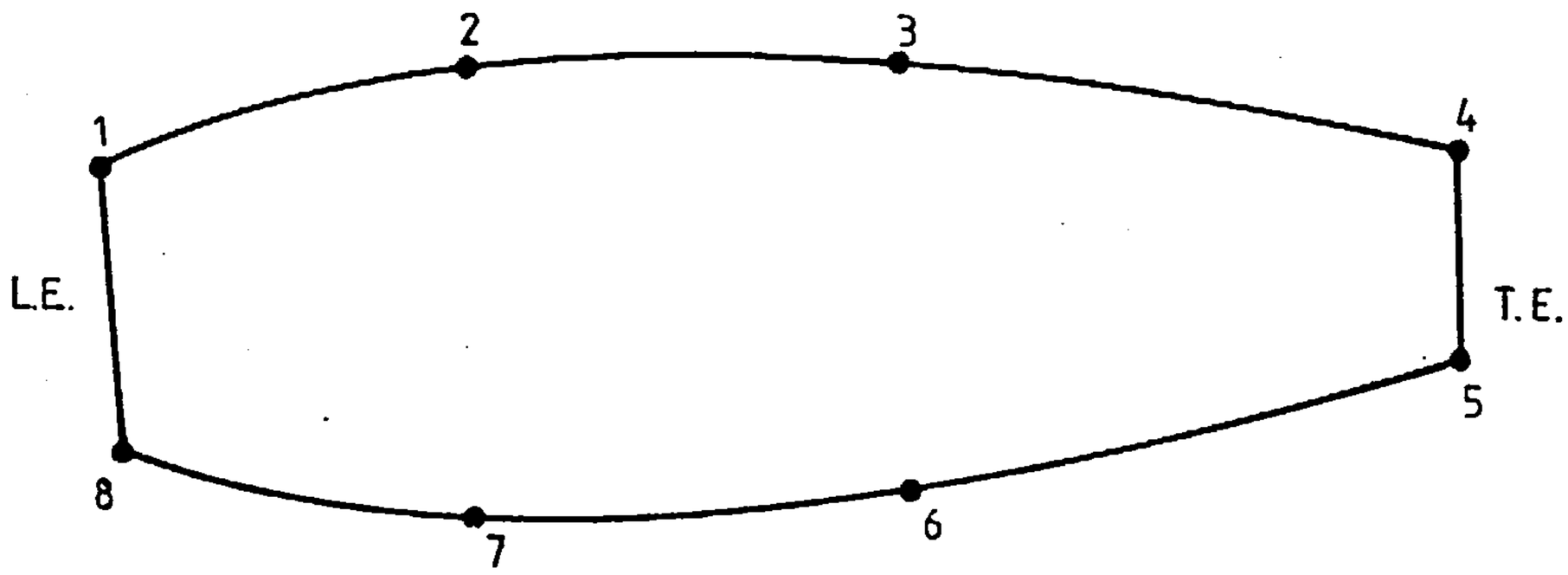


Fig 31. Example of Node Arrangement on a Rib

Note the node order, this becomes important later.

3.29.14 OPTION 9 Number of Top Nodes
3.29.15 OPTION 10 Number of Bottom Nodes

When option 9 is chosen the VDU responds with

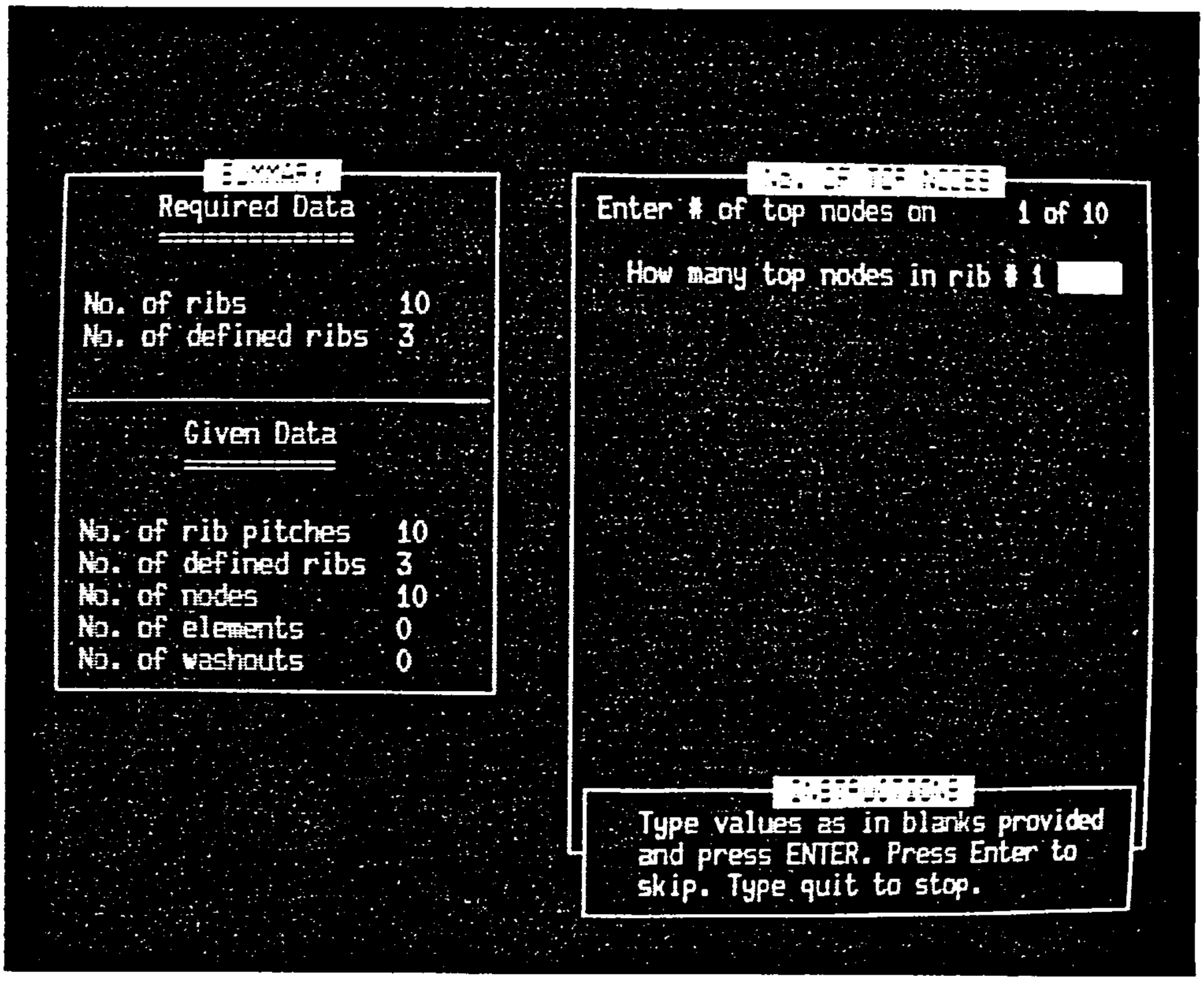


Fig 32. Example of Option 9 Display

And in the case of option 10 we get

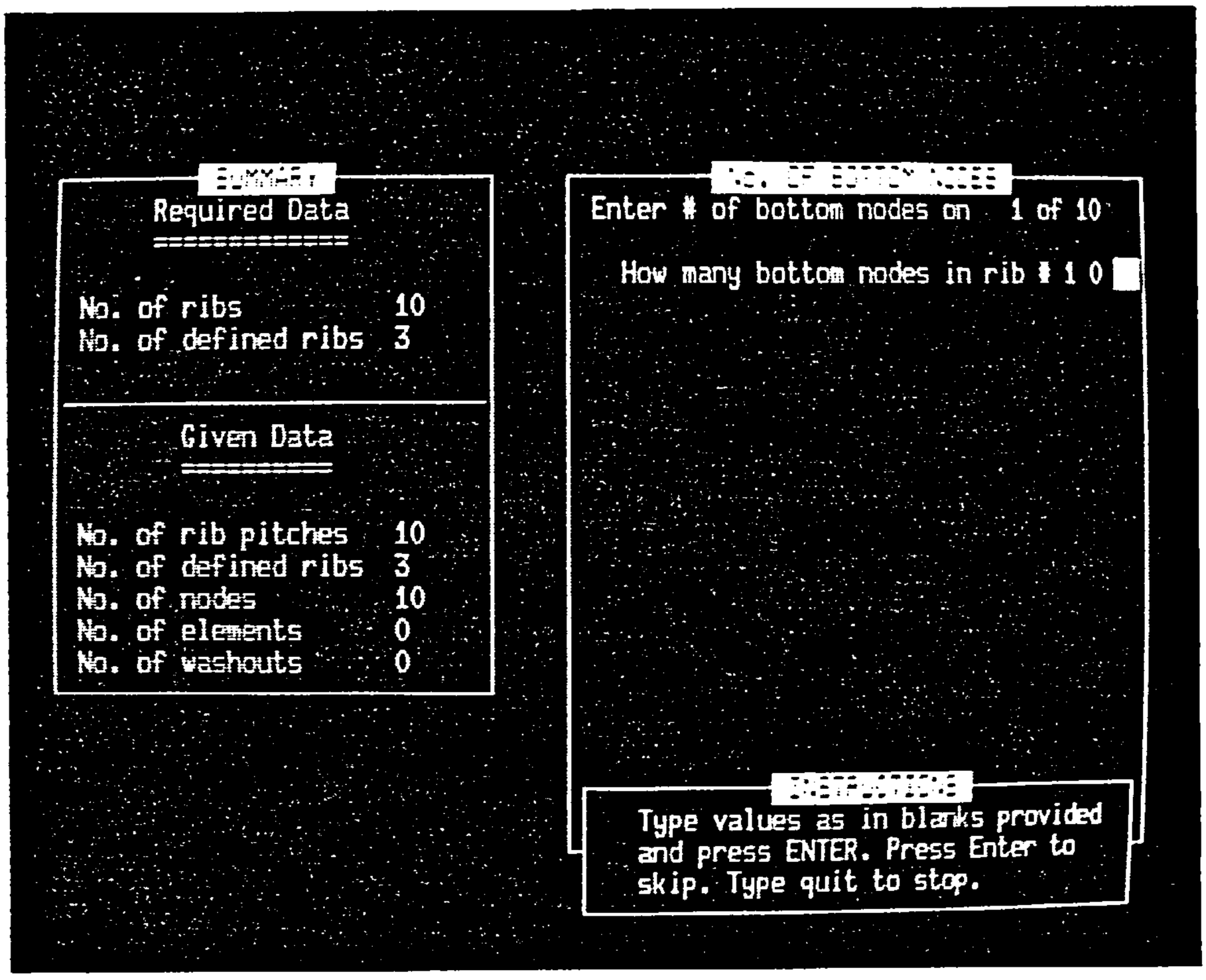


Fig 33. Example of Option 10 Display

The reason we need to give these values is that the generator has no idea about the configuration of the structure. It is quite possible to have the situation in fig 34.

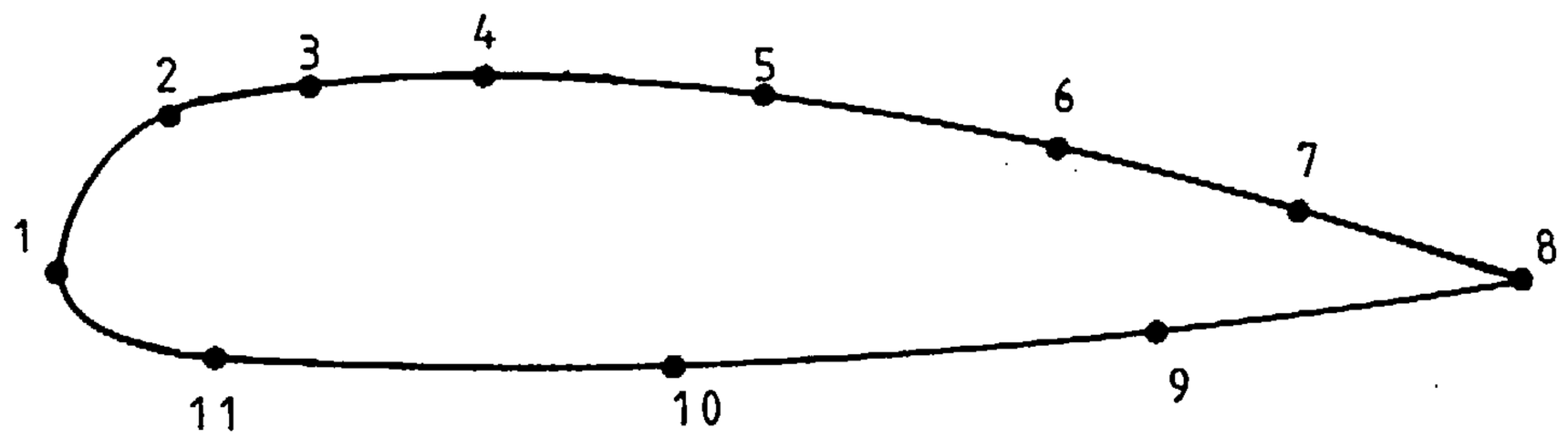


Fig 34. Uneven Node Distribution on a Rib

Here the top and bottom surfaces have different numbers of nodes; Note that in the case shown in fig 34. nodes 1 to 8 inclusive are top nodes and nodes 8 to 11 and node 1 are bottom nodes. i.e. nodes 1 and 8 belong to both surfaces.

In the case of a D shape box as shown in fig 35. nodes 1 to 3 and on the top surfaces and 4, 5 and 1 are on the bottom.

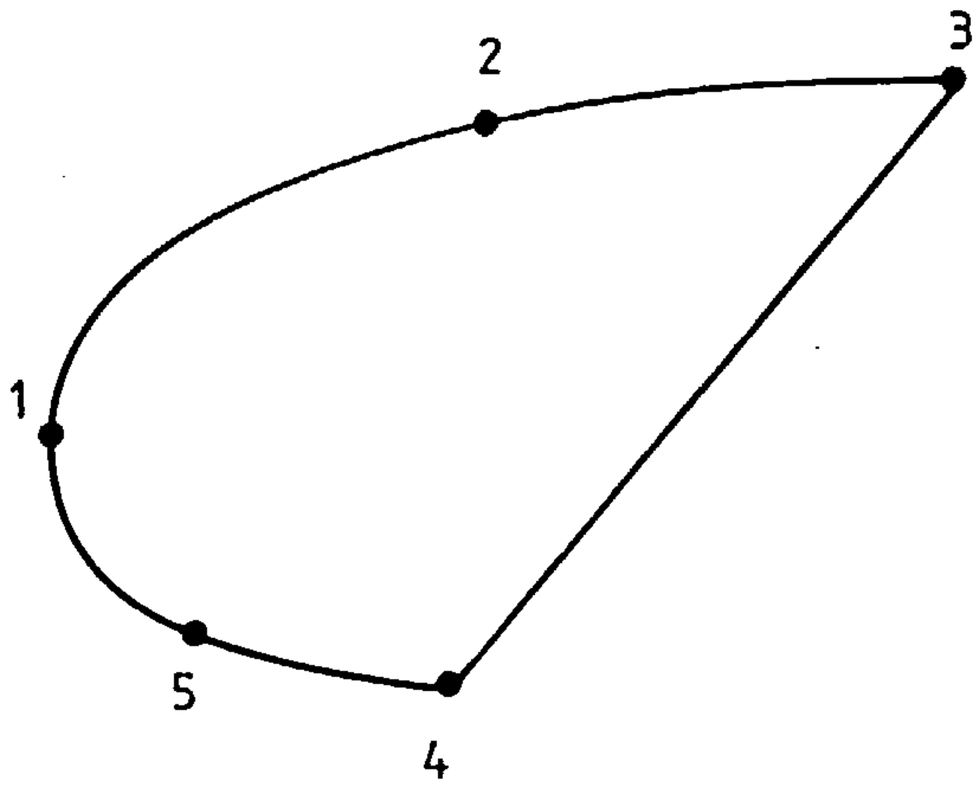


Fig 35. Example of a D-nose Rib

In the case of a straight forward box section there is less room for ambiguity as shown in fig 36.

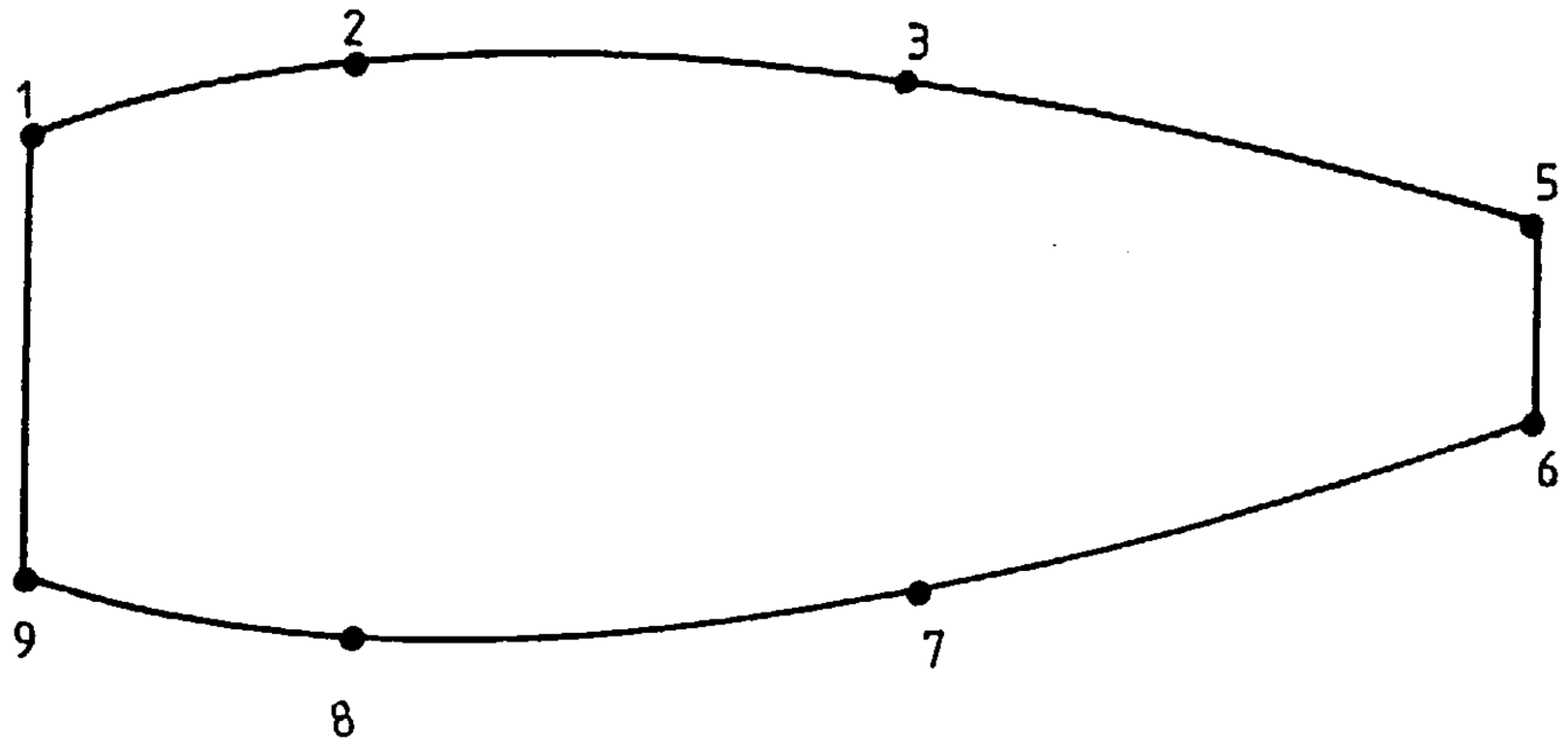


Fig 36. Example of a Box Rib

3.29.16 OPTION 11 Rib Configuration Codes

There are currently three rib configurations available and each rib must be given one. Having selected this option the VDU responds with

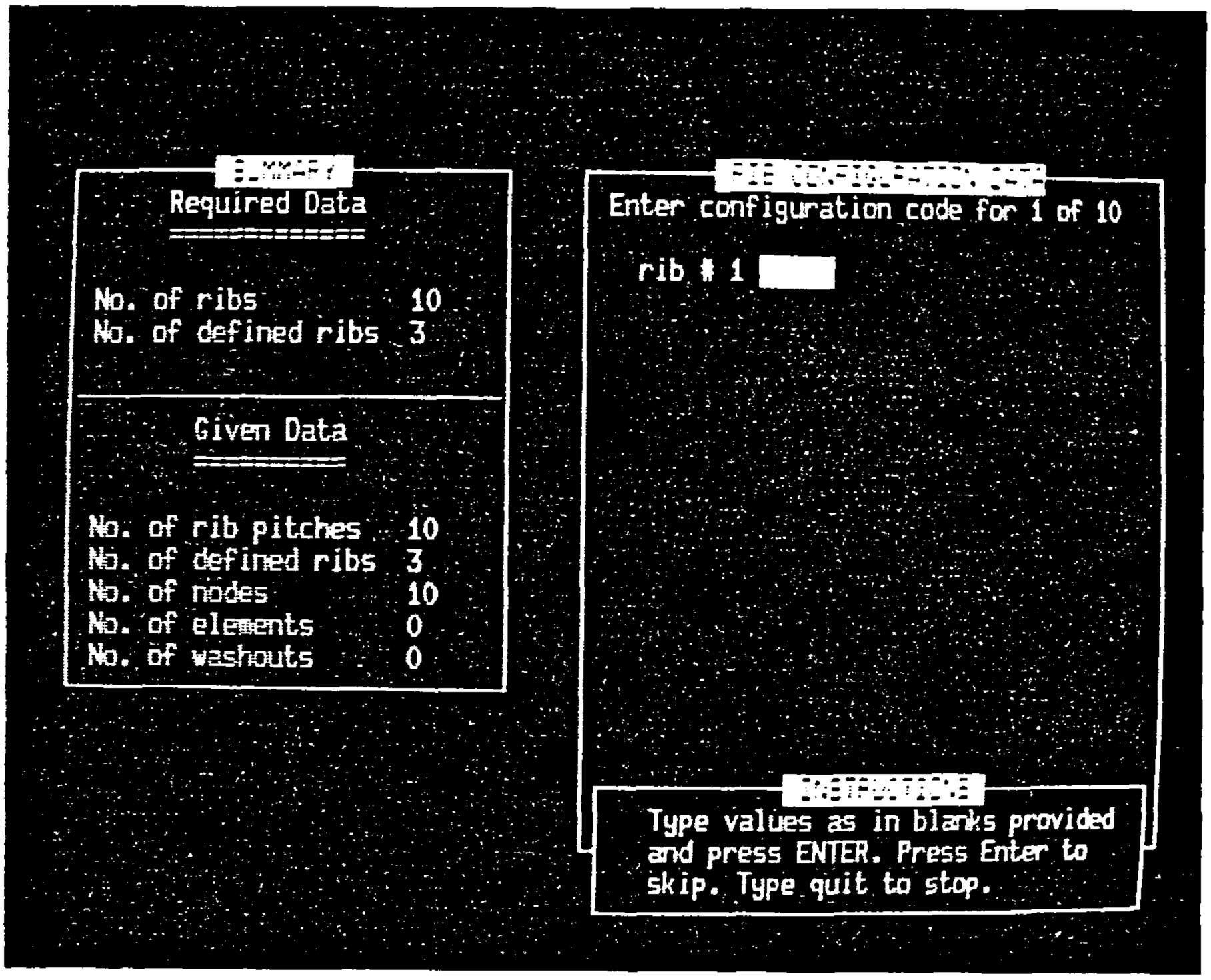


Fig 37. Example of Option 11 Display

You must reply with one of the following configuration codes;

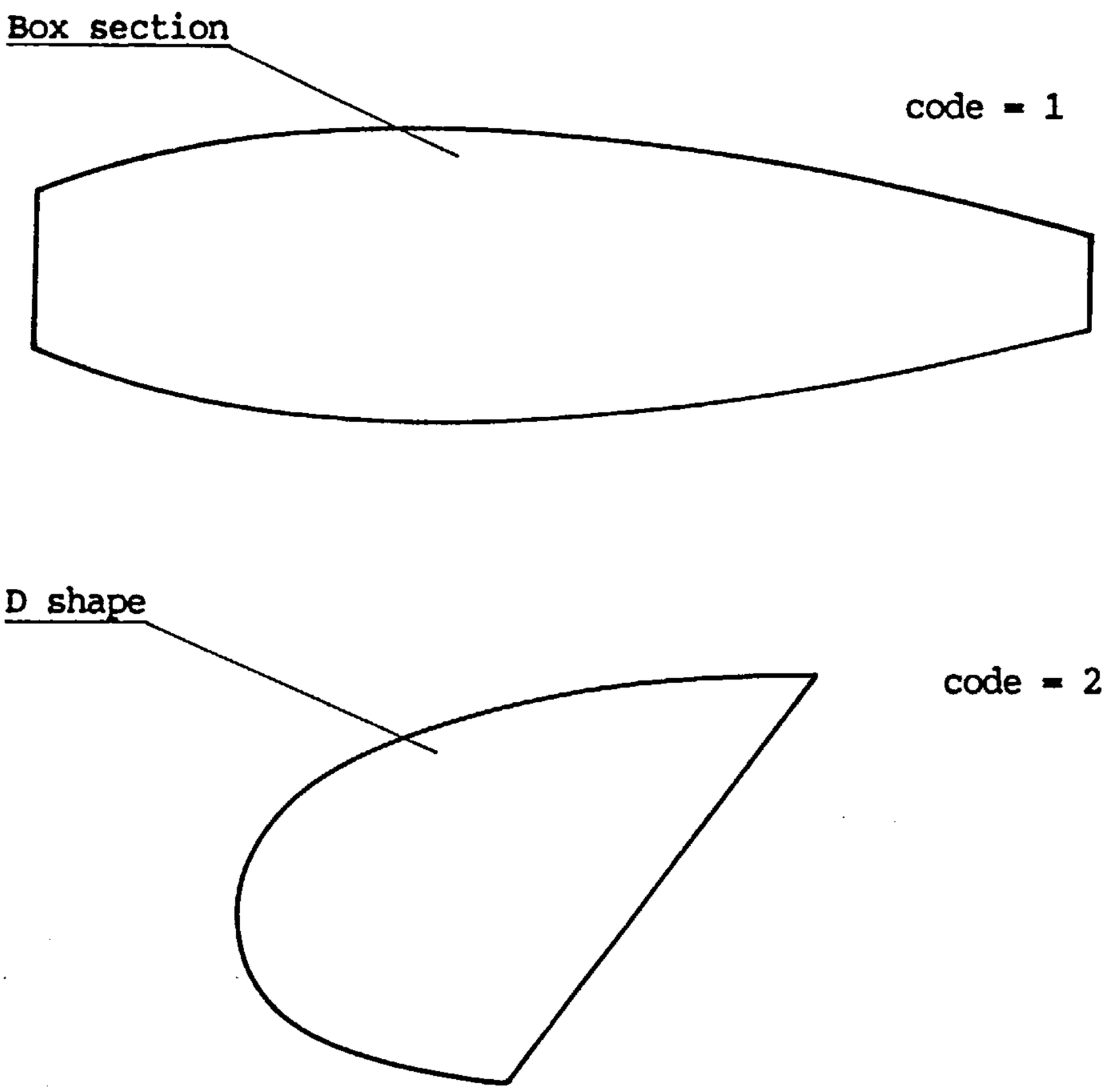


Fig 38. Rib Configuration Codes

3.29.17 OPTION 12 Rib Geometry Data

This is the option where you give geometric details of the "defined" ribs discussed in 3.27.7. When you select option 12 the VDU screen responds with;

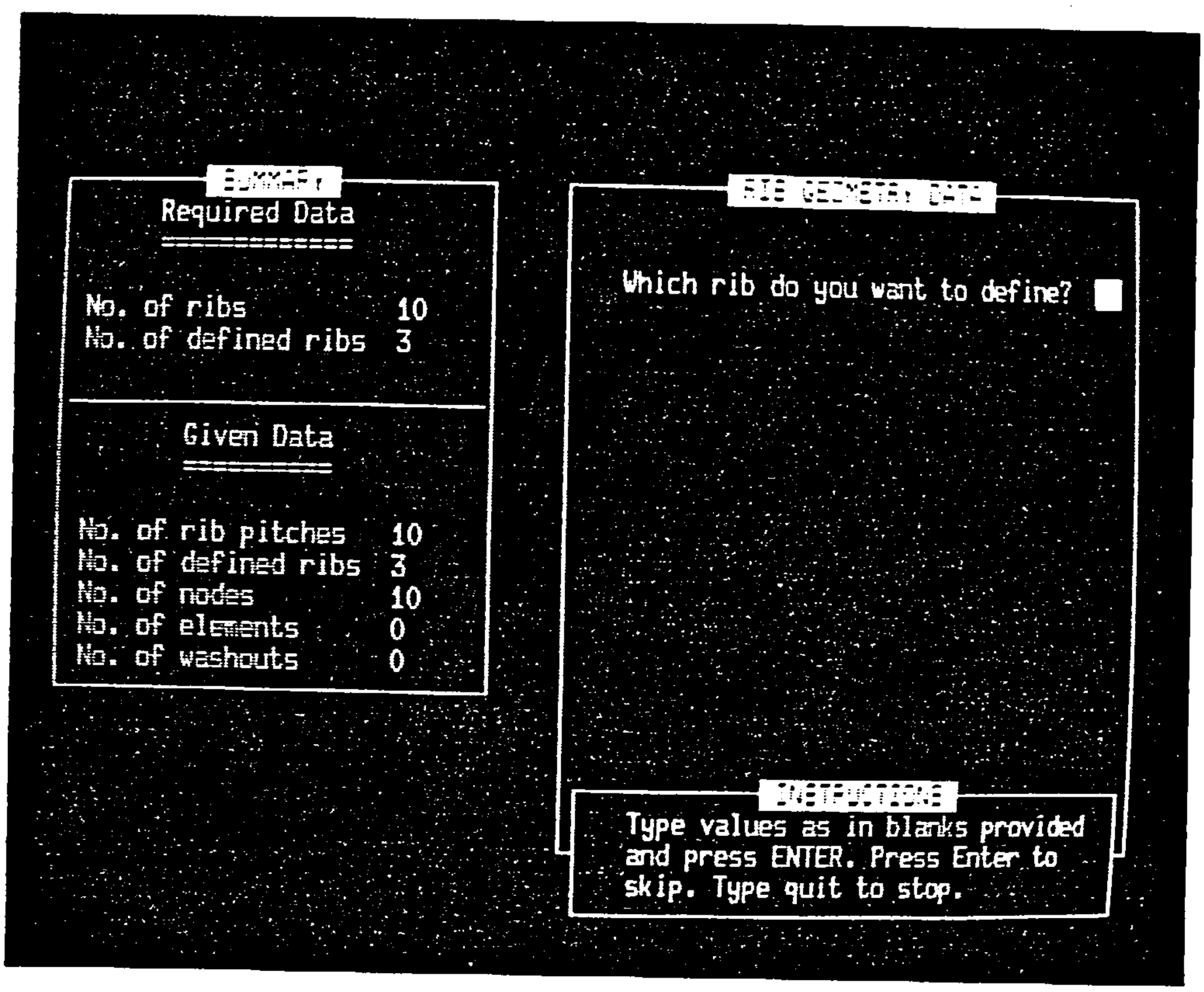


Fig 39. Example of Option 12 Initial Display

In response to this query you must tell it which rib you wish to define or redefine. Say you ask to define rib 1 the VDU then moves on to the next stage and displays this;

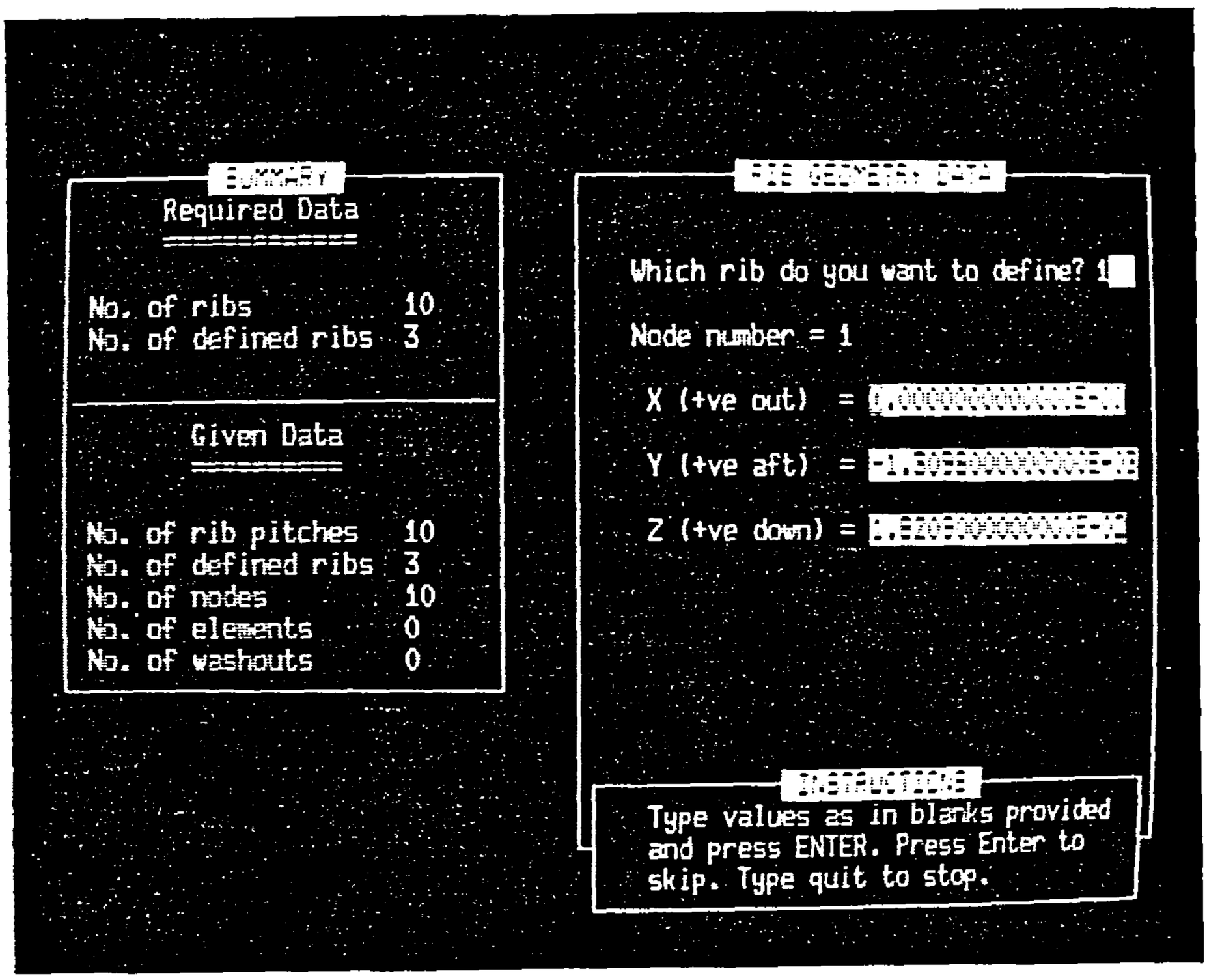


Fig 40. Example of Option 12 Second Display

The module knows how many nodes there are on the rib you have requested to define but still does not know where they are. So in the top right corner of the display it tells you which node number it is expecting you to define. It is important that you follow the conventions requested on the screen. These conventions are shown pictorially in fig 41.

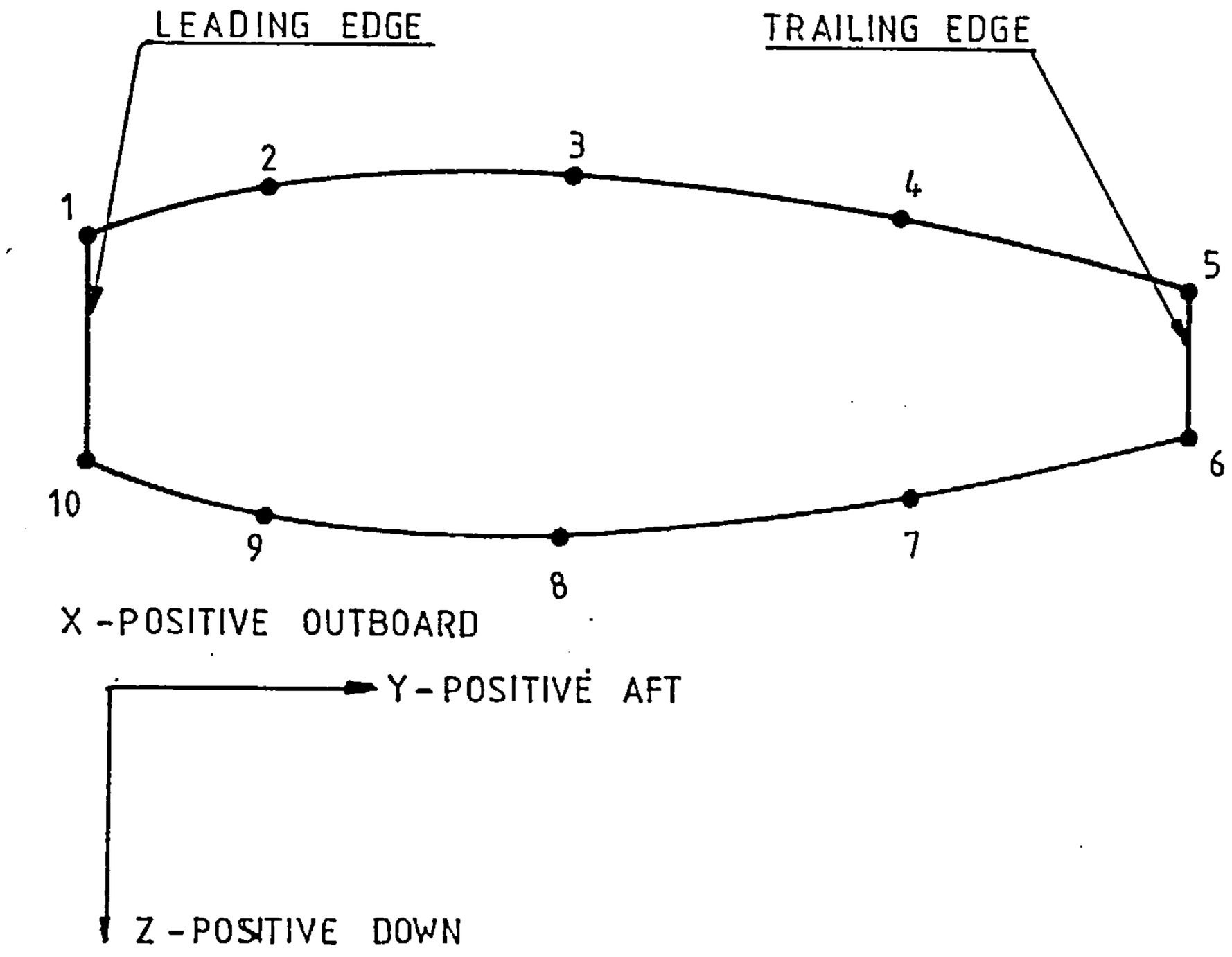


Fig 41. Geometry Conventions

Use the top leading edge corner node as your datum (this does not matter critically but it saves confusion later).

Note that the nodes are ordered from the top leading edge corner and work their way around the rib back to the last node on the bottom leading edge corner.

3.29.18 OPTION 13 Number of Stringer Starts (optional default = 0)

This is another structural definition option. Stringers can become sparser or denser towards the wing tip. In this section you are prompted as shown below for the number of new stringers that start in a bay.

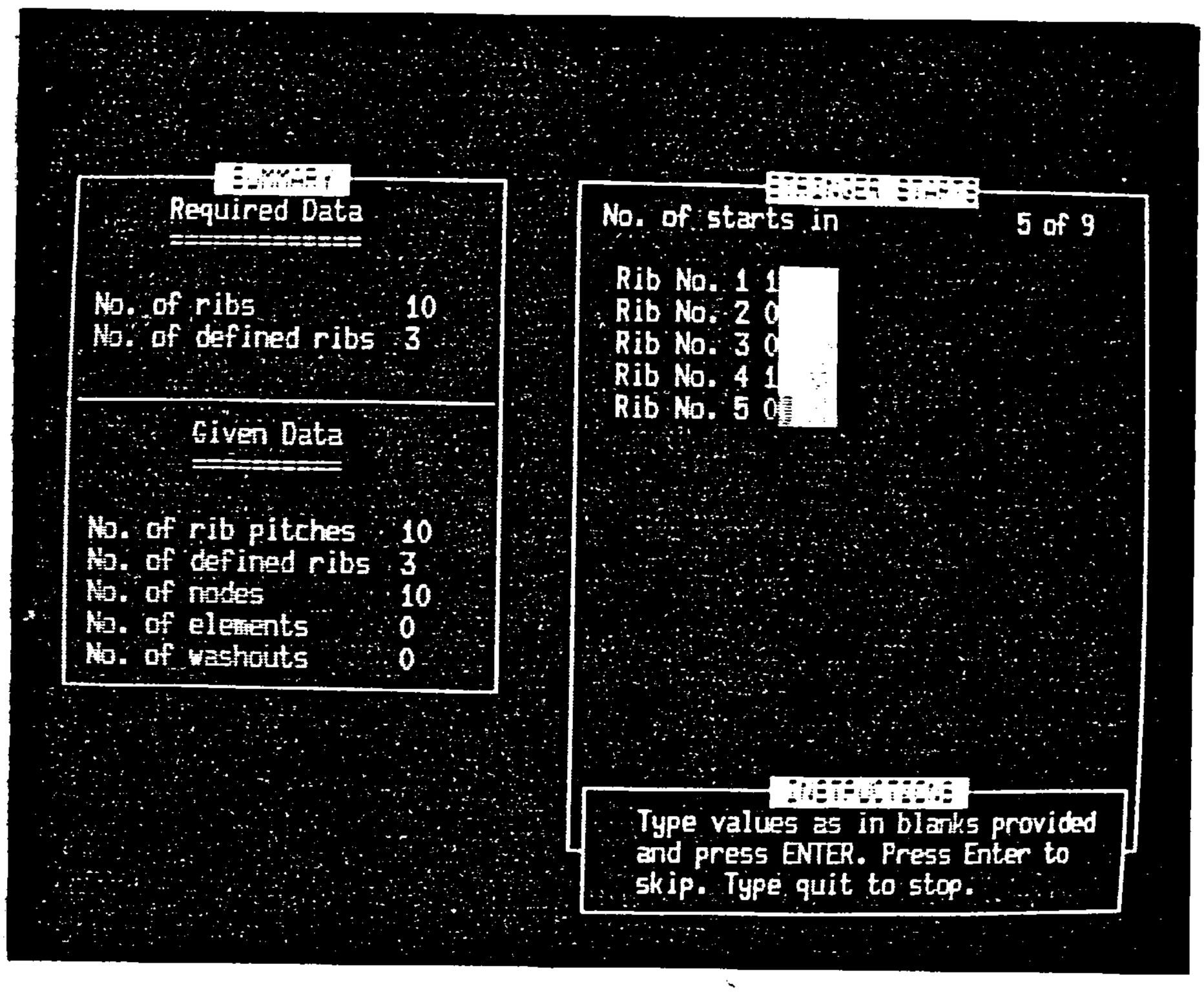


Fig 42. Example of Option 13 Display

The replies given above would lead to the following type of layout.

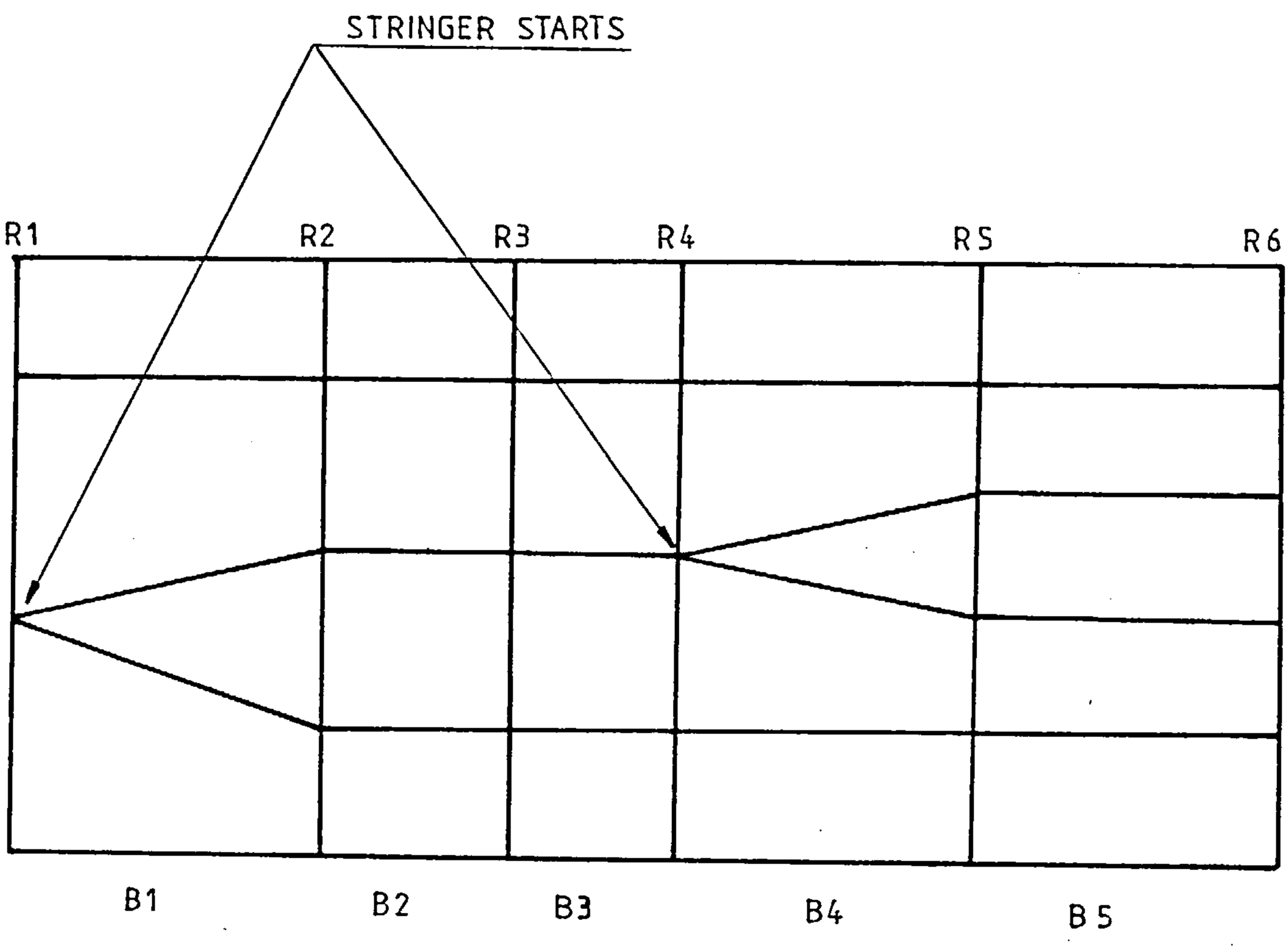


Fig 43. Example of Generated Wing Configuration

Note: A starting position in say bay 1 is on rib 1.

You define where these branches occur on a rib in option 15. The reason for this branching technique is that though in real structure stringers may start anywhere, in the finite element idealisations there can be problems trying to model such things for example;

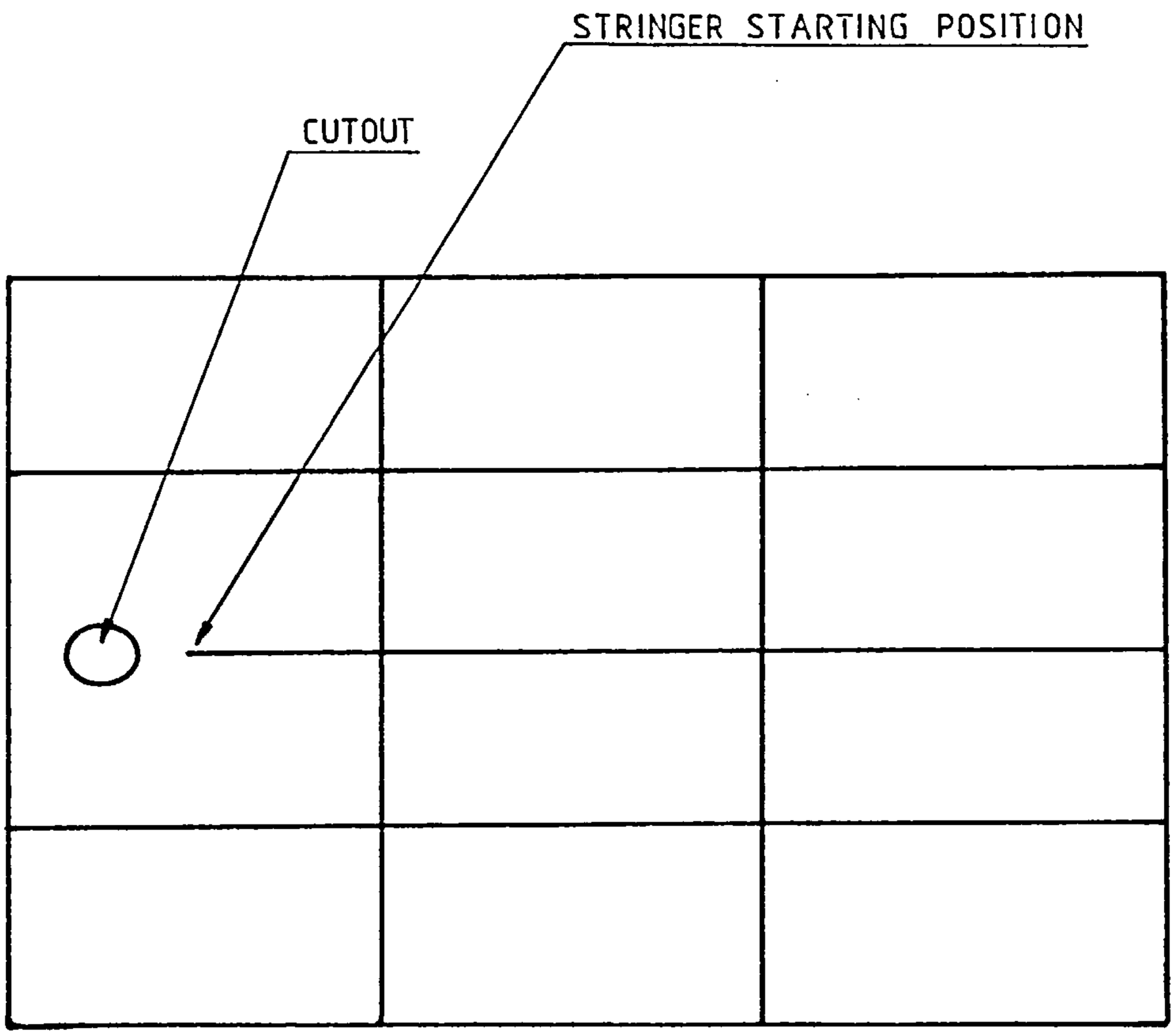


Fig 44. Real Structure

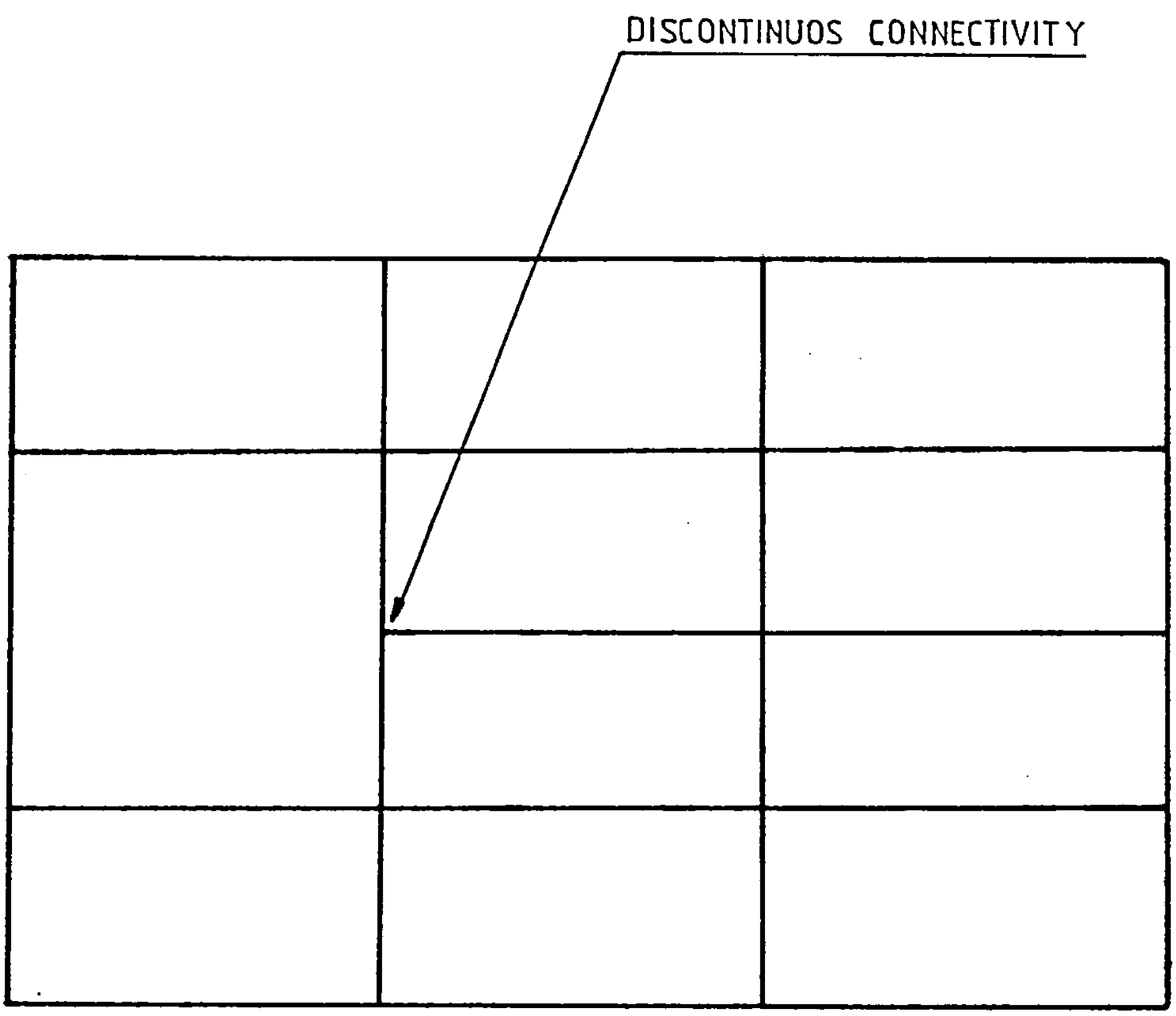


Fig 45. Finite Element Model

If you do not use this option the structure is assumed to have no stringer starts.

3.29.19 OPTION 14 Number of Stringer Runoffs (optional default = 0)

When you select this option the VDU responds thus

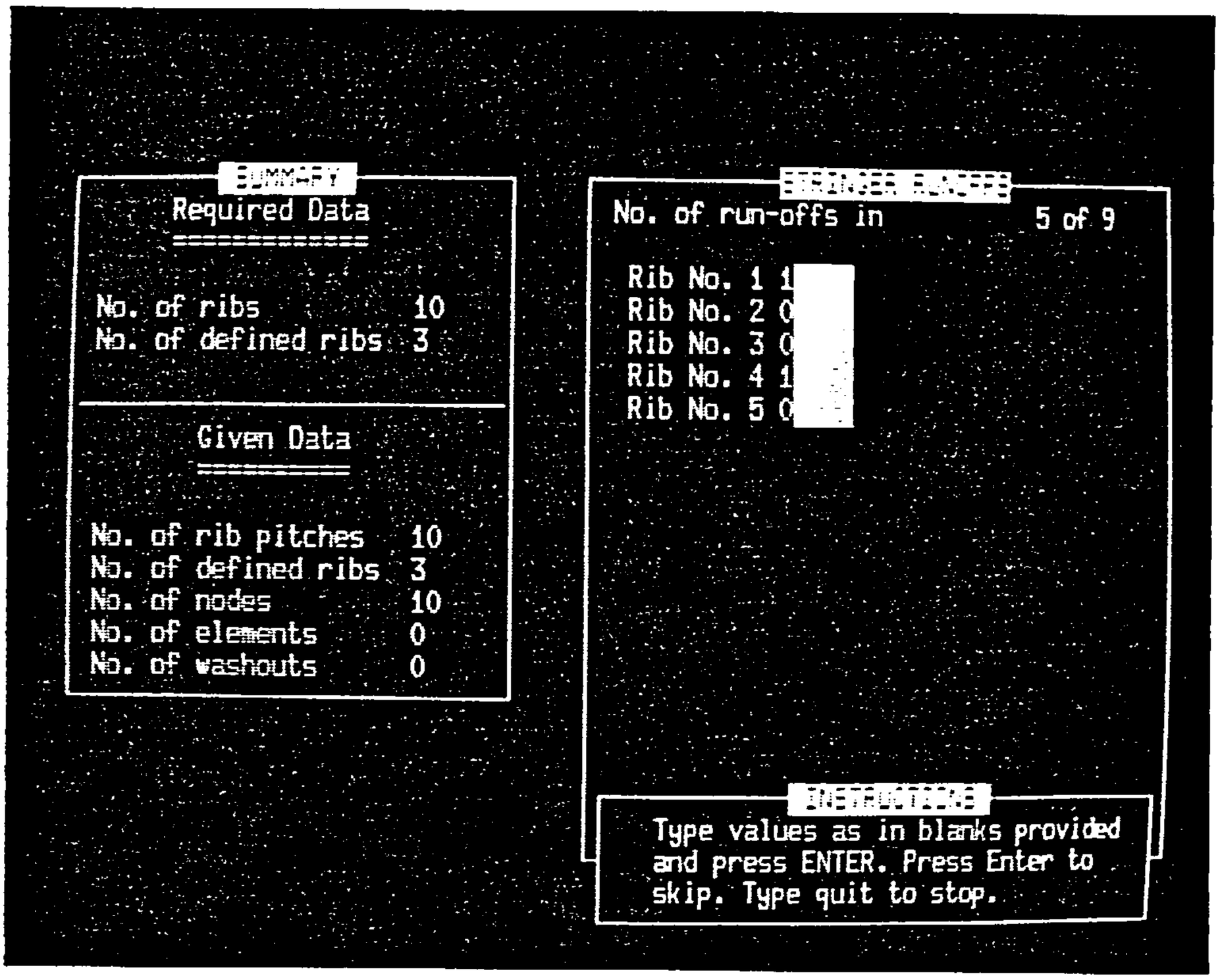


Fig 46. Example of Option 14 Display

The effect is similar to that of option 13 except you are now referring to stringer runoffs as shown below.

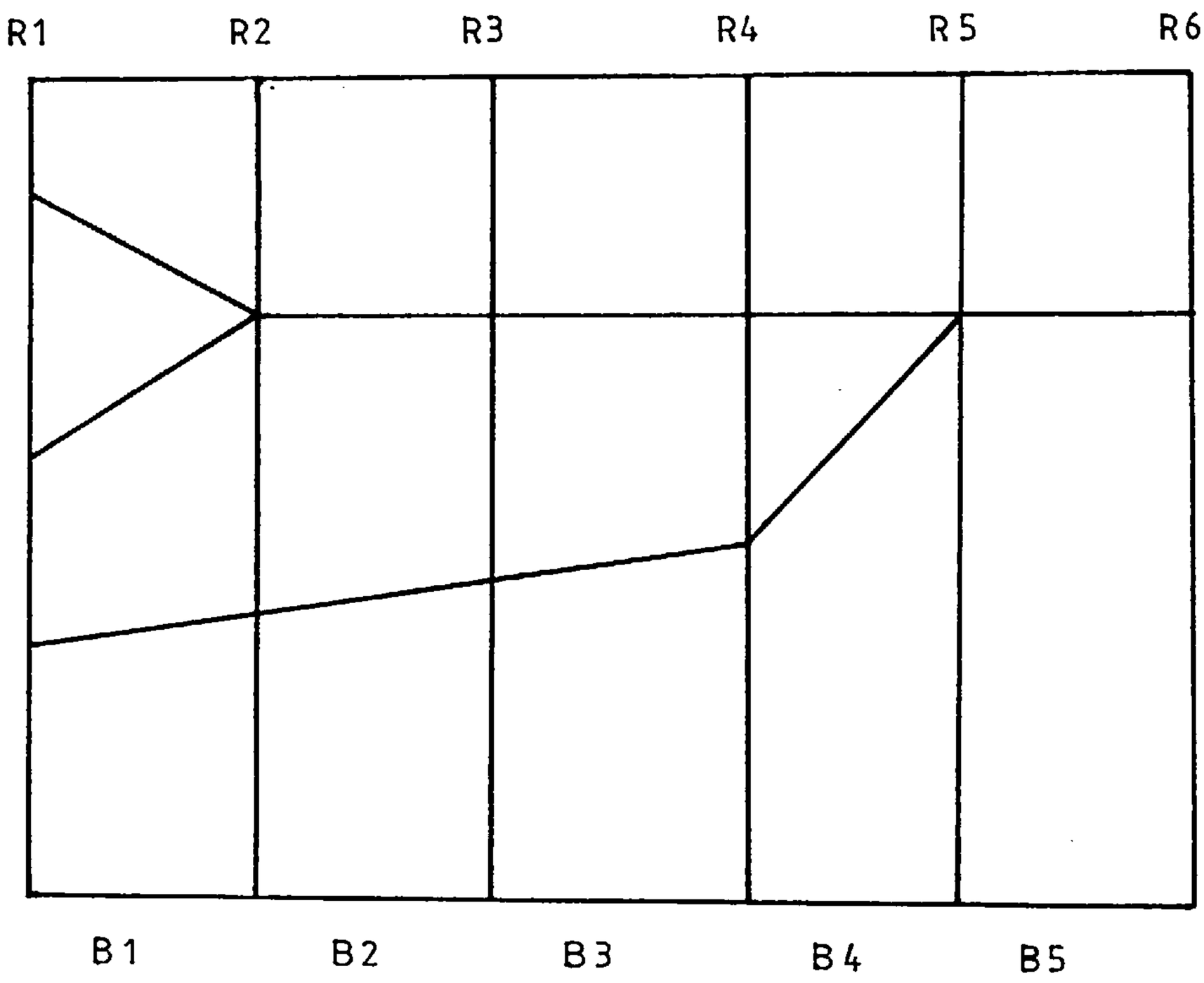


Fig 47. Example of Stringer Runoffs

Note: A runoff position in say bay 1 is on rib 2.

The definition of the positioning of these runoffs on a rib in option 16. Refer to 3.27.18 for modelling rationale.

If you do not use this option the wing is assumed to have no stringer run outs.

3.29.20 OPTION 15 Positions of Stringer Starts (optional)
3.29.21 OPTION 16 Positions of Stringer Runoffs (optional)

Note: If you have bypassed option 13 then option 15 is not needed and the same relationship holds between option 14 and 16.

For each rib which has a stringer change you will receive the following menu.

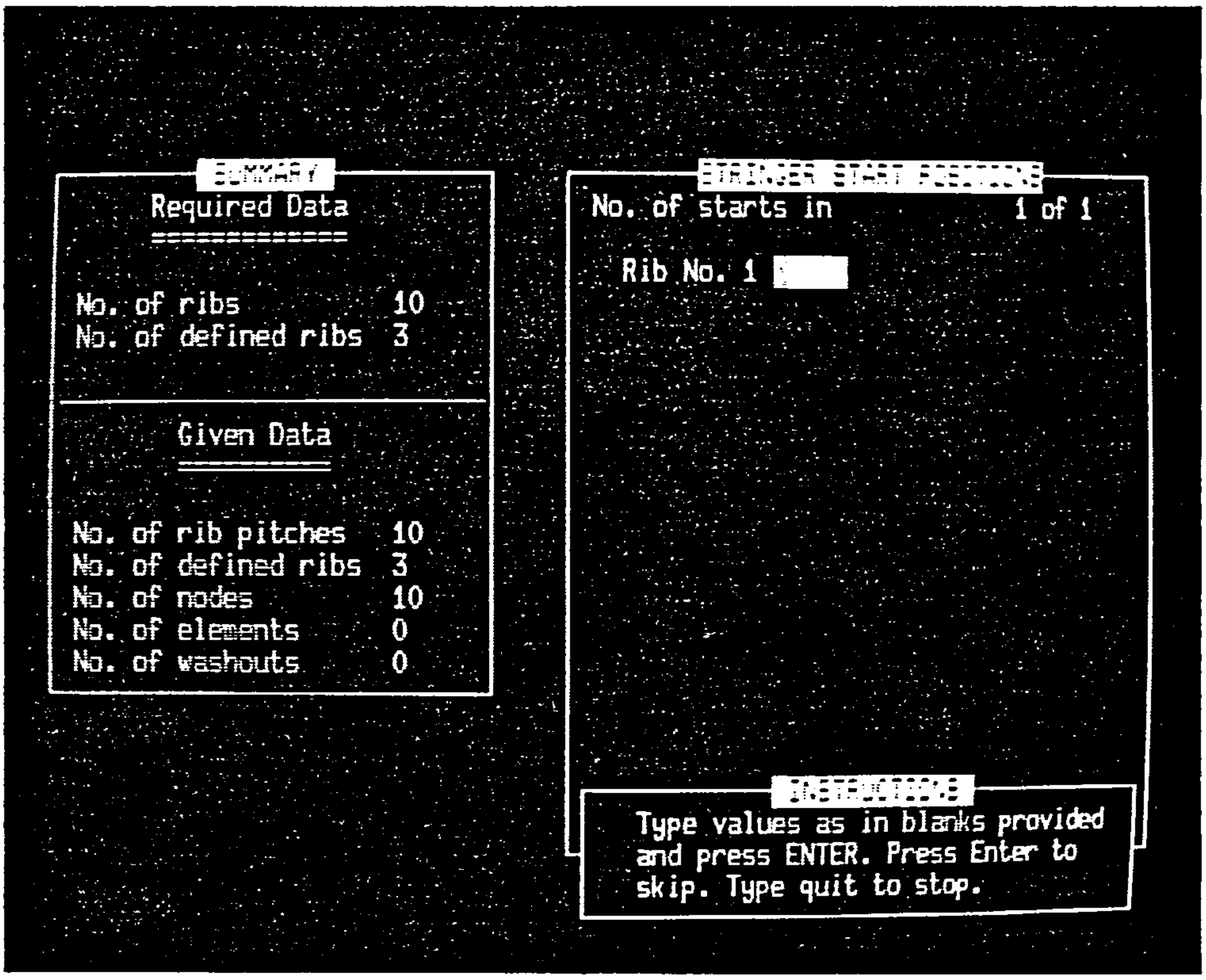


Fig 48. Example of Option 15 Display

Currently you have to keep track of the rib bay being defined, this will change in future versions.

The top right hand corner of the menu shows you the current stringer change being defined. You have the choices shown in fig 49. and fig 50. when defining a change.

Note: The numbers in these figures are not node numbers.

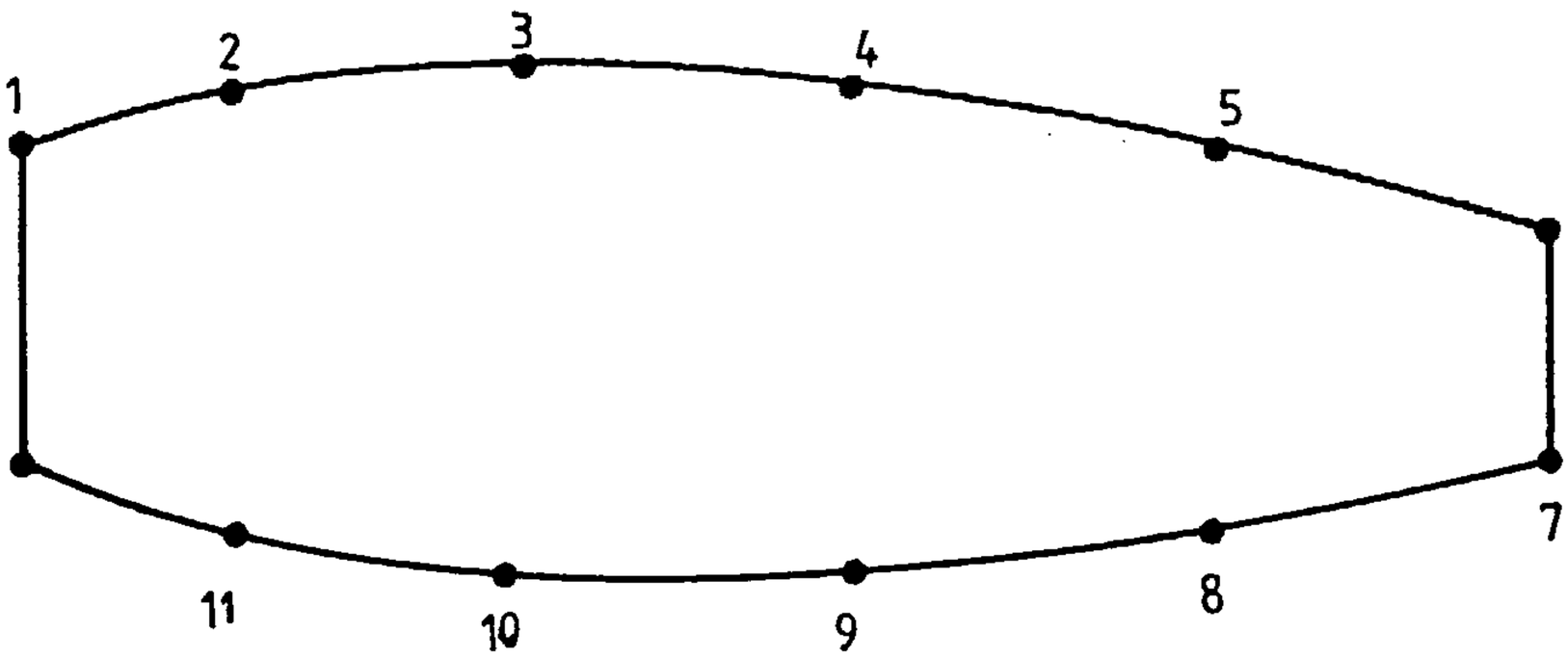


Fig 49. Stringer Run-off Positions Available on a 12 Noded Rib

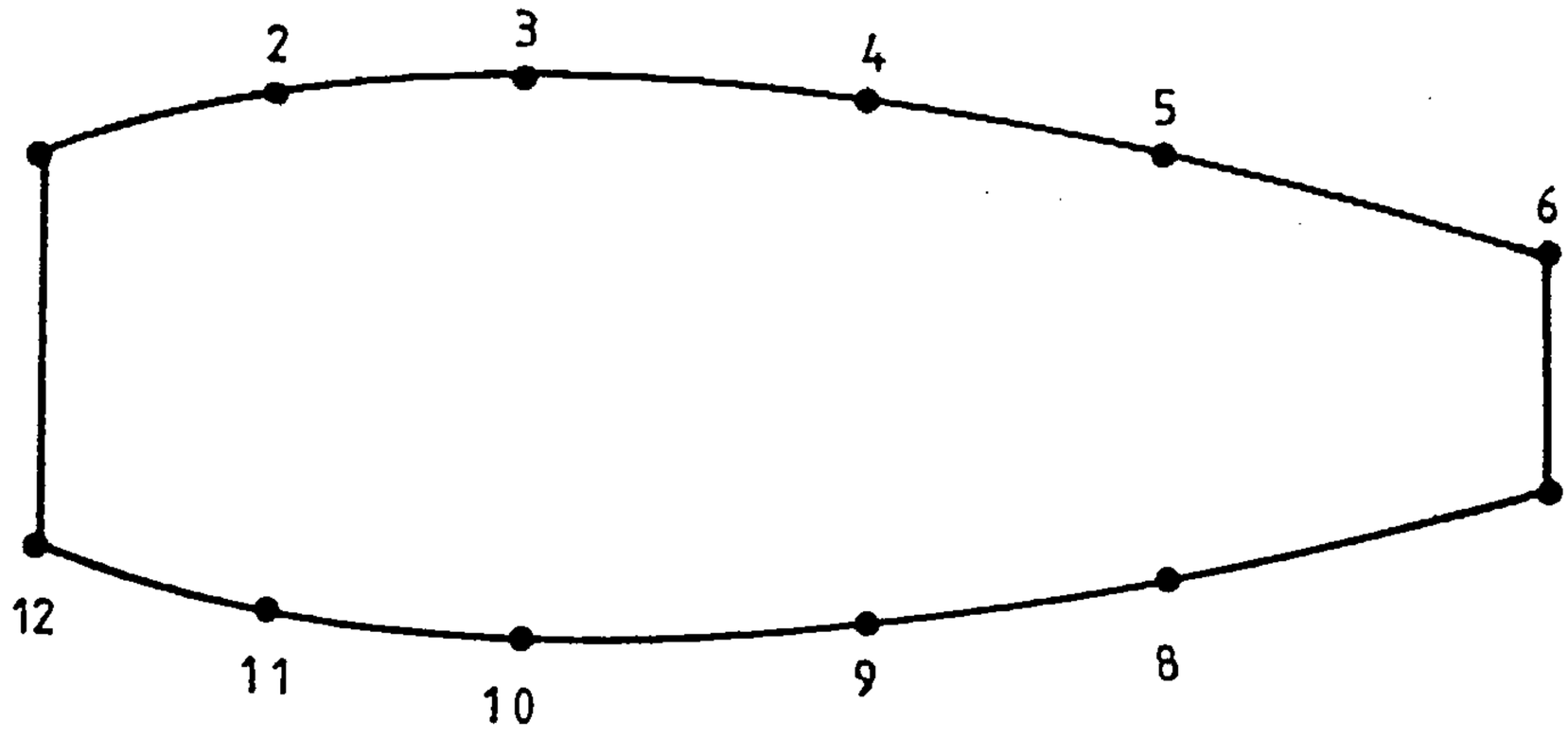


Fig 50. Stringer Start Positions Available on a 12 Noded Rib

For example stringer runoff positions 1 and 4 will cause the following grid to be generated.

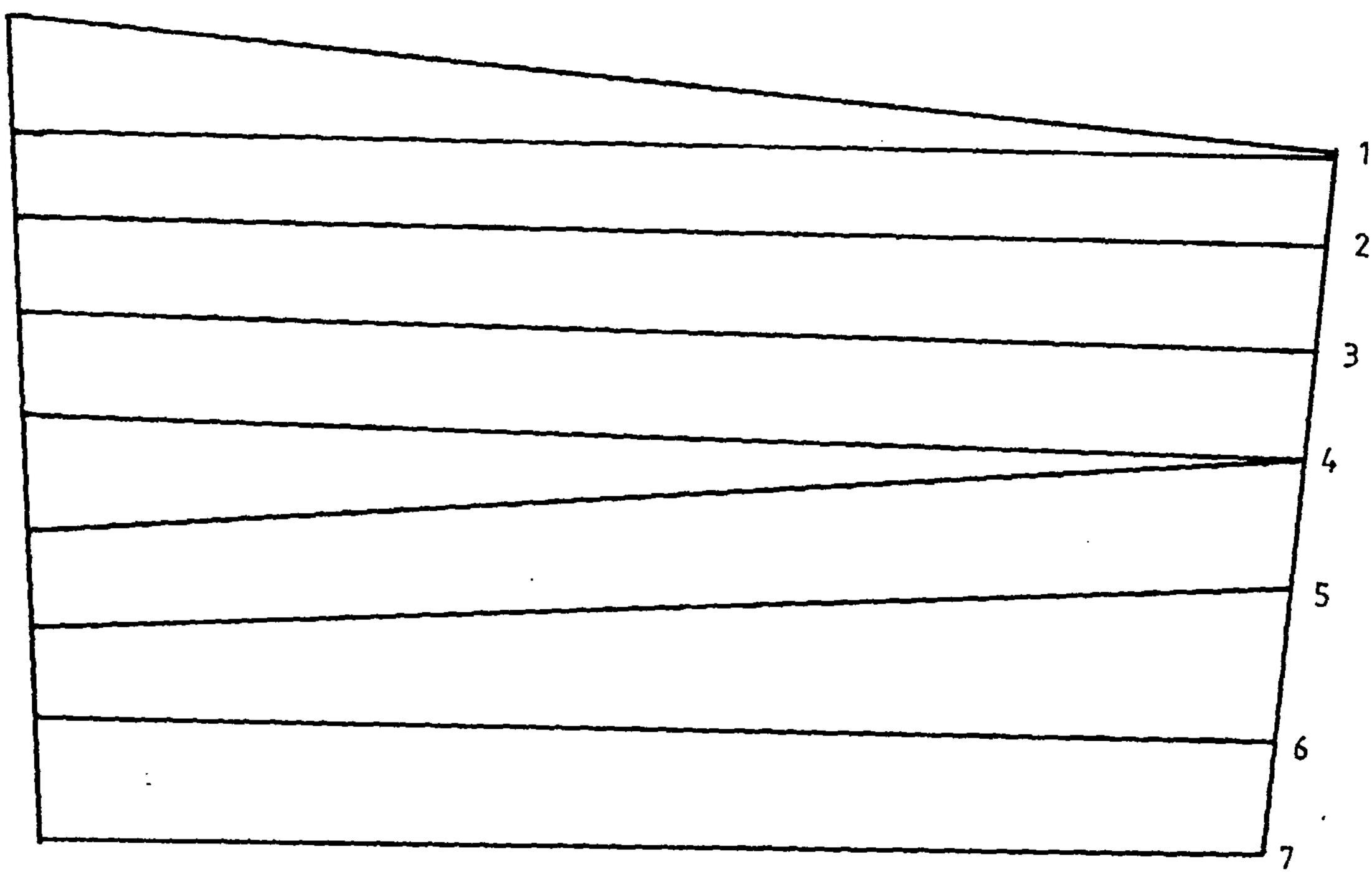


Fig 51. Sample Grid

3.29.22 OPTION 17 Number of Rib Panels

This is another configuration definition option. The following appears on the VDU during this option.

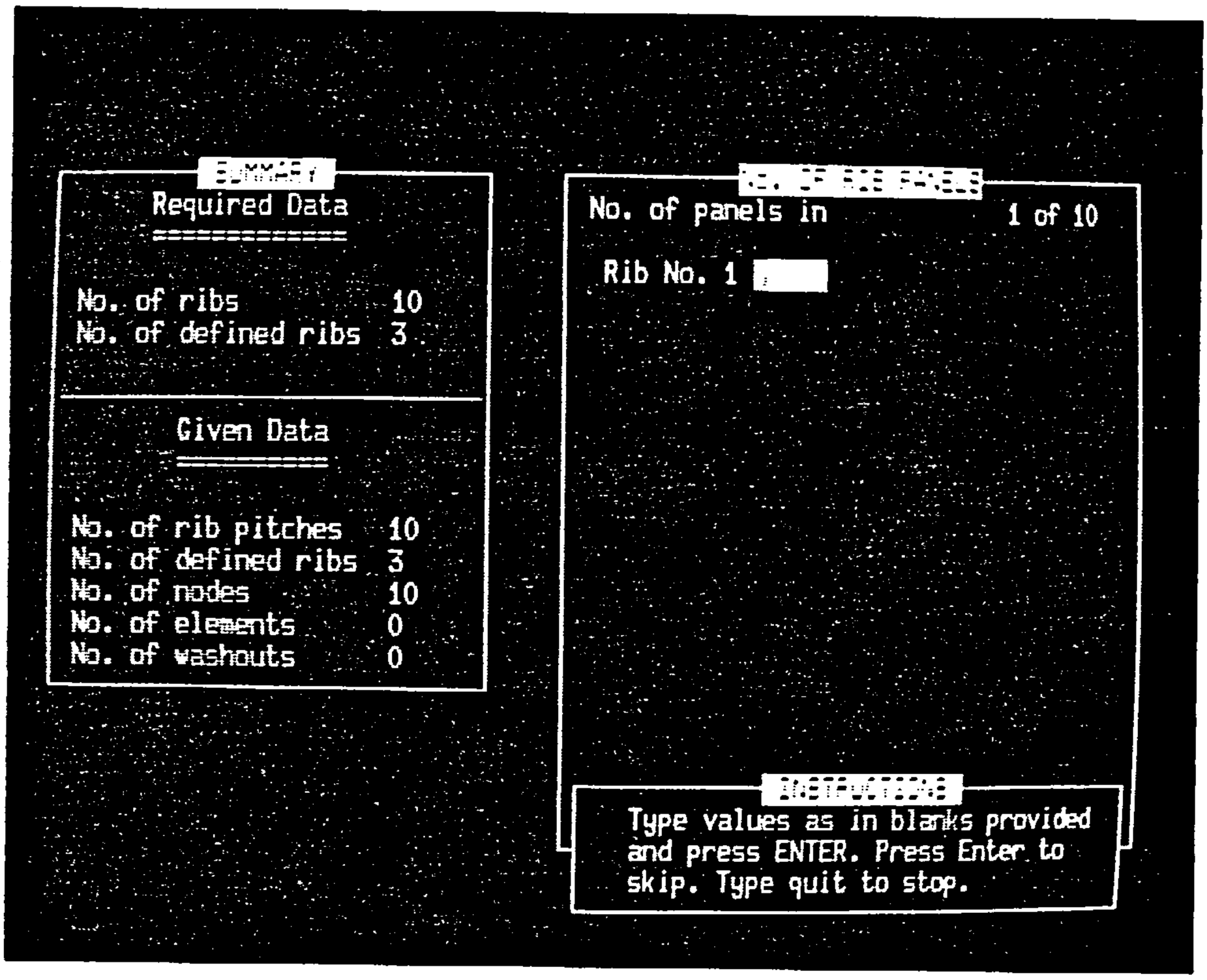


Fig 52. Example of Option 17 Display

In the example shown in fig 53. the rib has 6 panels

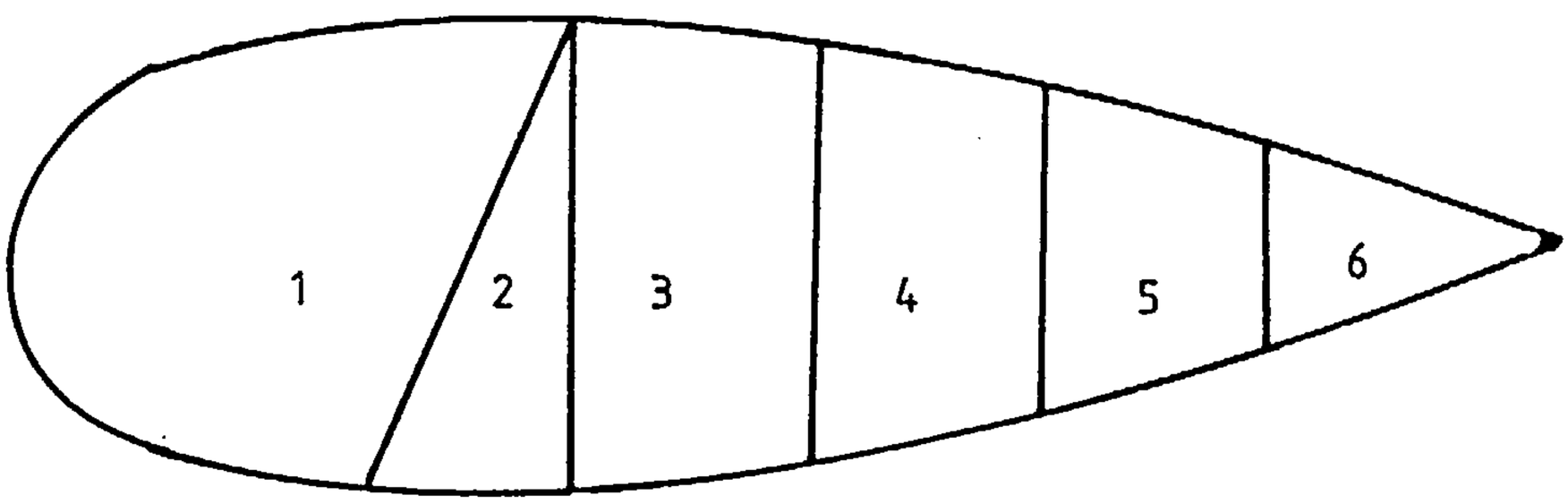


Fig 53. Rib panels

3.29.23 OPTION 18 Rib Panels Codes

The following appears on the VDU for each rib

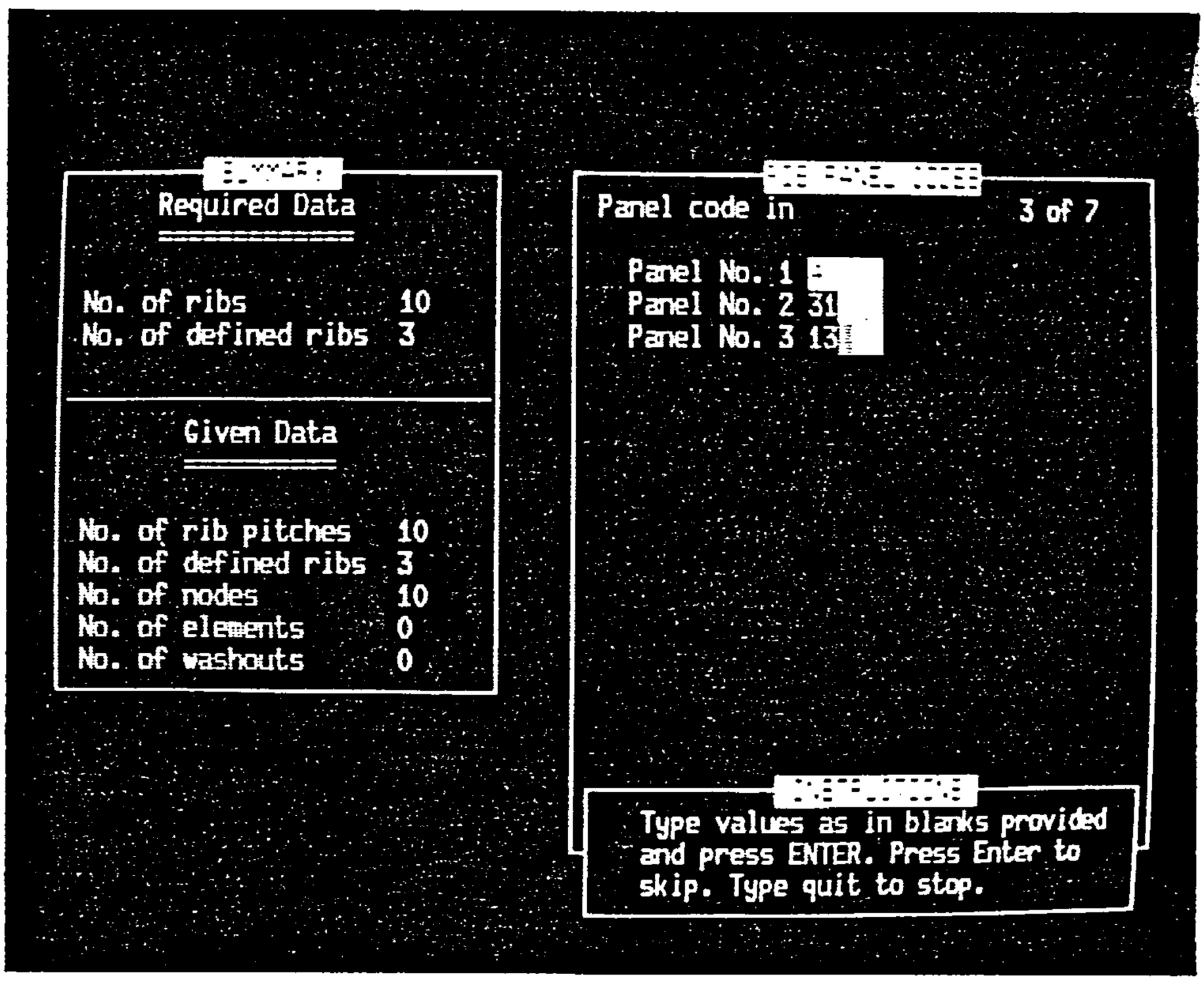


Fig 54. Example of Option 18 Display

There are 3 panel types available as shown below

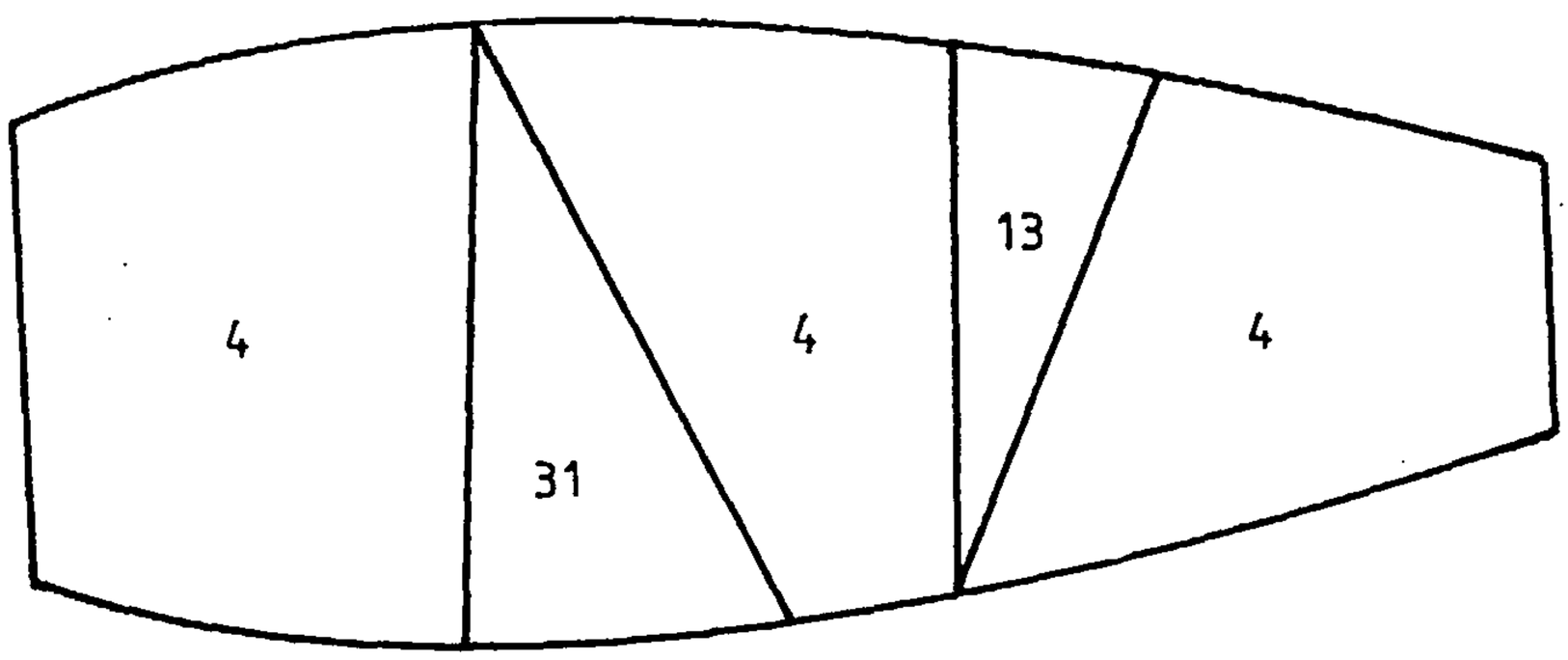


Fig 55. Rib Panel Codes

3.29.24 OPTION 19 Number of Spar Panels (optional default = 0)

This is optional in some cases and is used to define how many extra spars there are in a wing bay. Invoking this option causes the VDU to respond with;

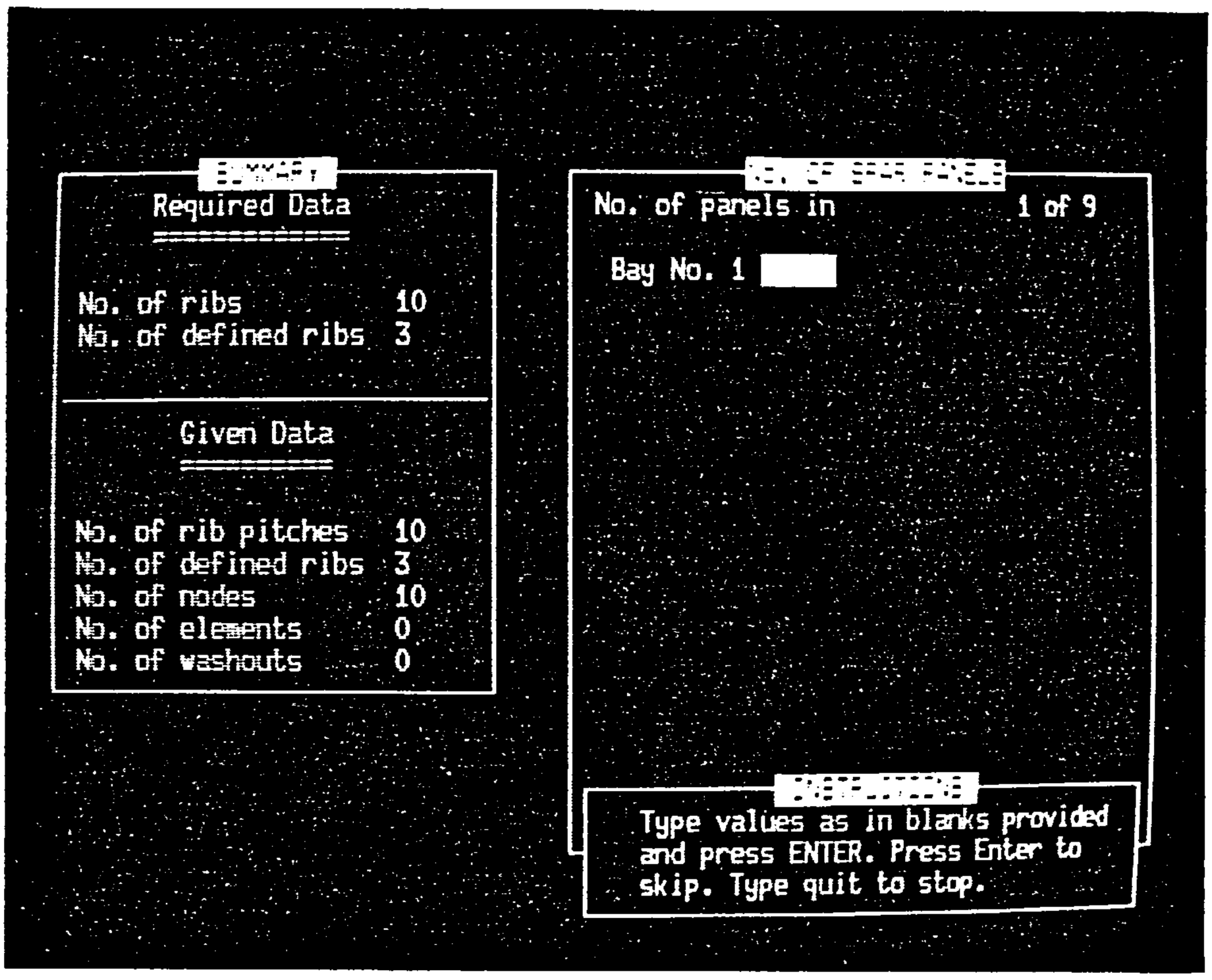
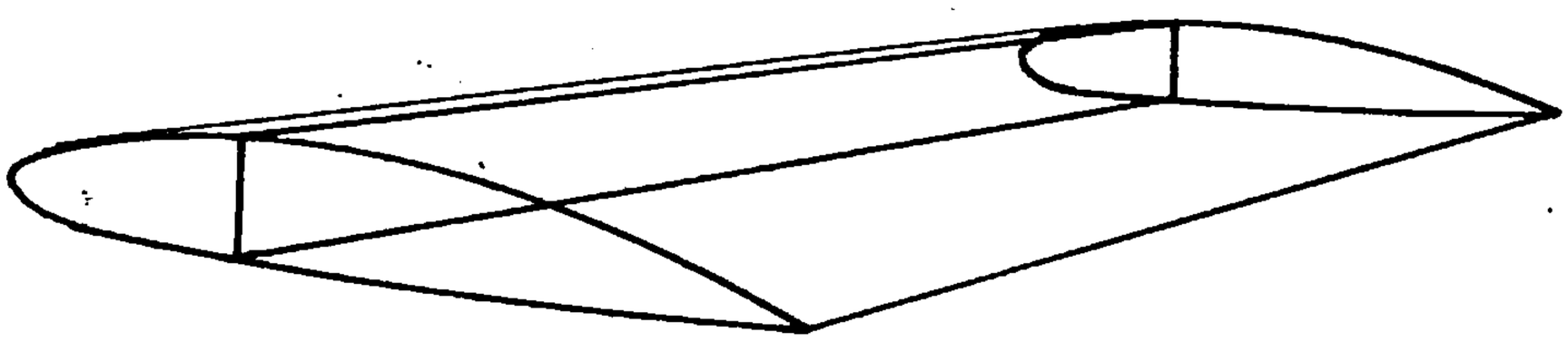
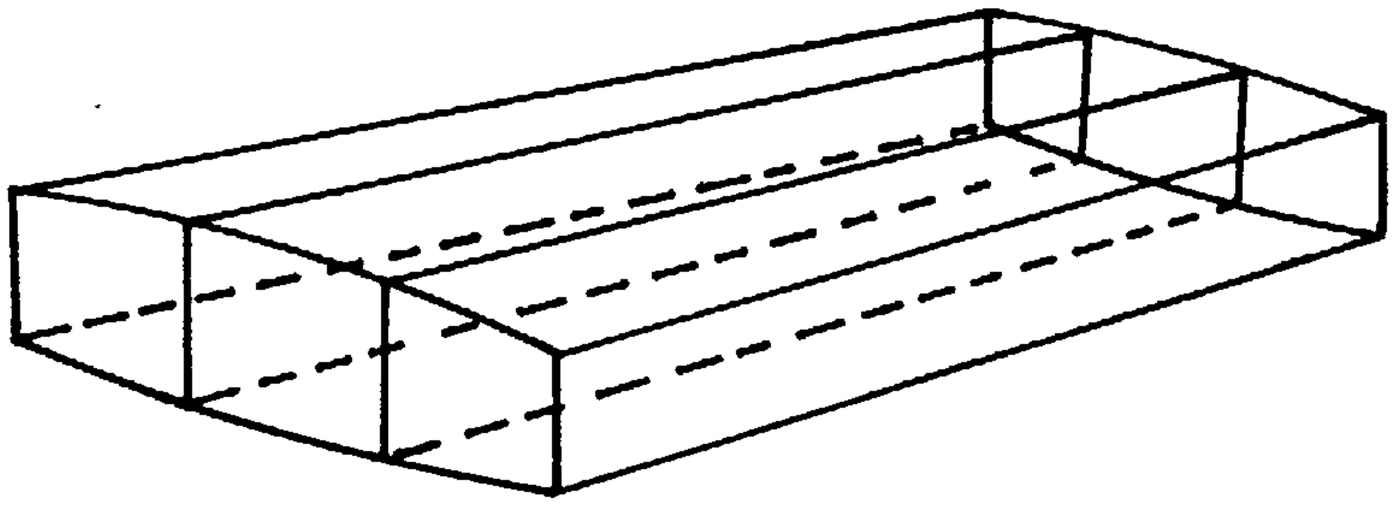


Fig 56. Example of Option 19 Display

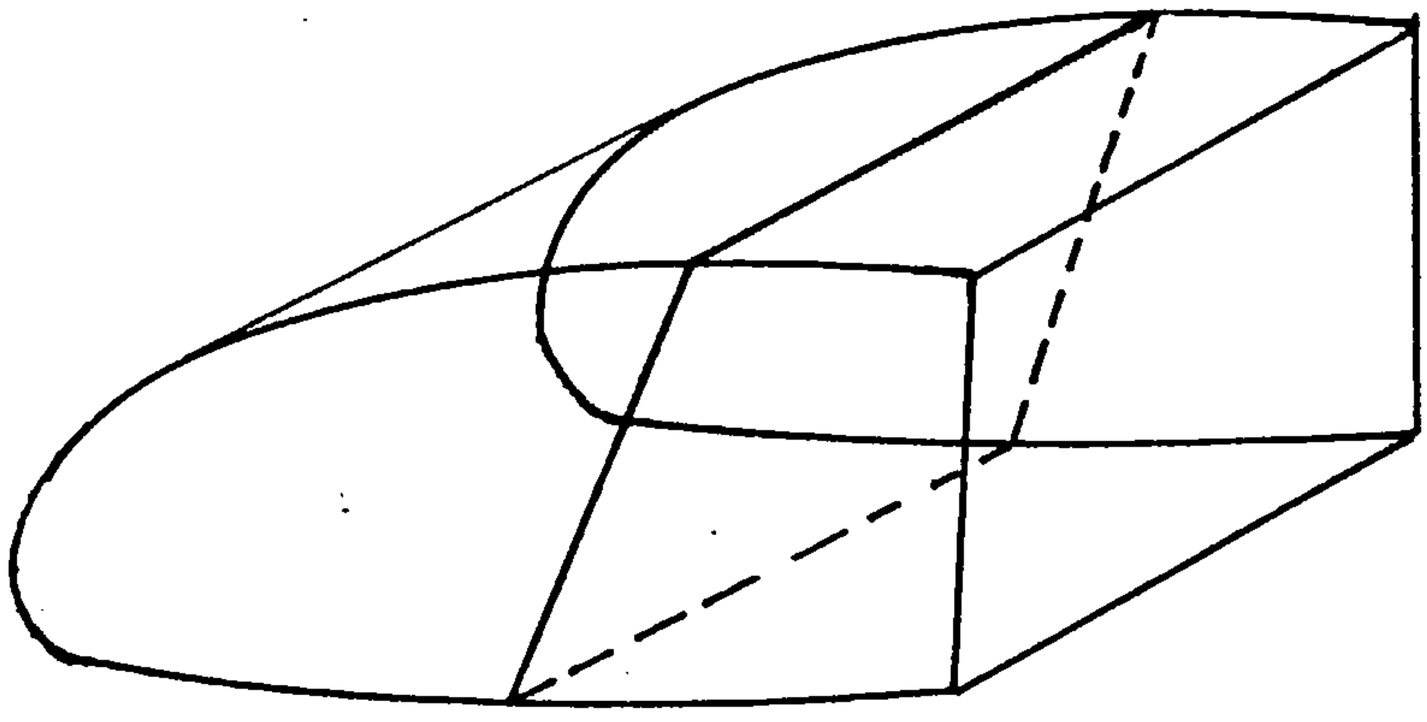
What you are asked here are how many extra spars there are for example;



one extra spar



two extra spars.
(front and rear are defaults in a box-section)



one extra spar
(the rear spar is a default in a D section)

Fig 57. Sample Spar Positions

3.29.25 OPTION 20 Spar Panel Code (optional)

The necessity for using this option depends on whether option 19 has been used. When invoked the VDU displays this;

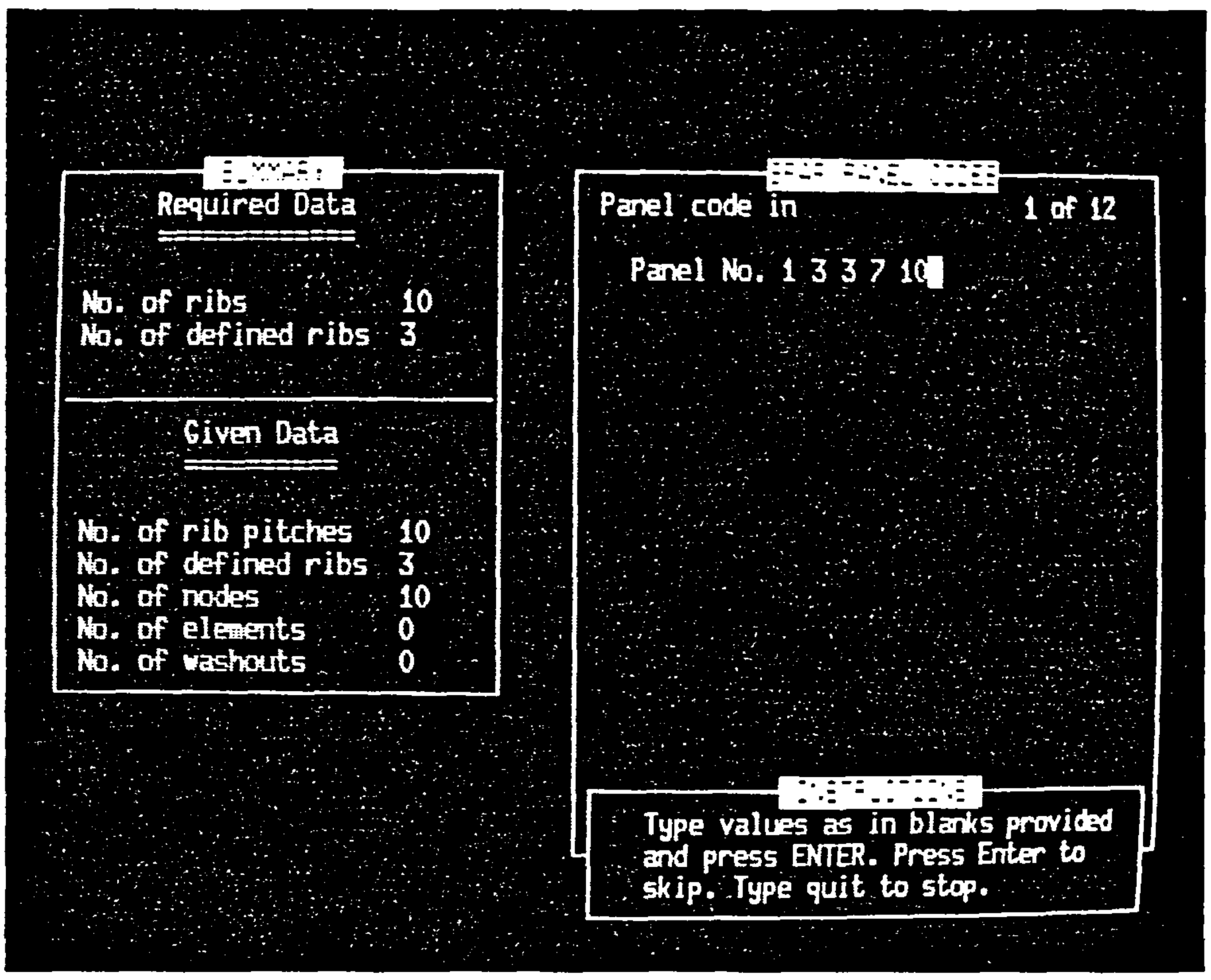


Fig 58. Example of Options 20 Display

The codes you are asked for refer to the extra panels you declared in option 19 and their positioning. The code system is based on node positions in ribs, for example

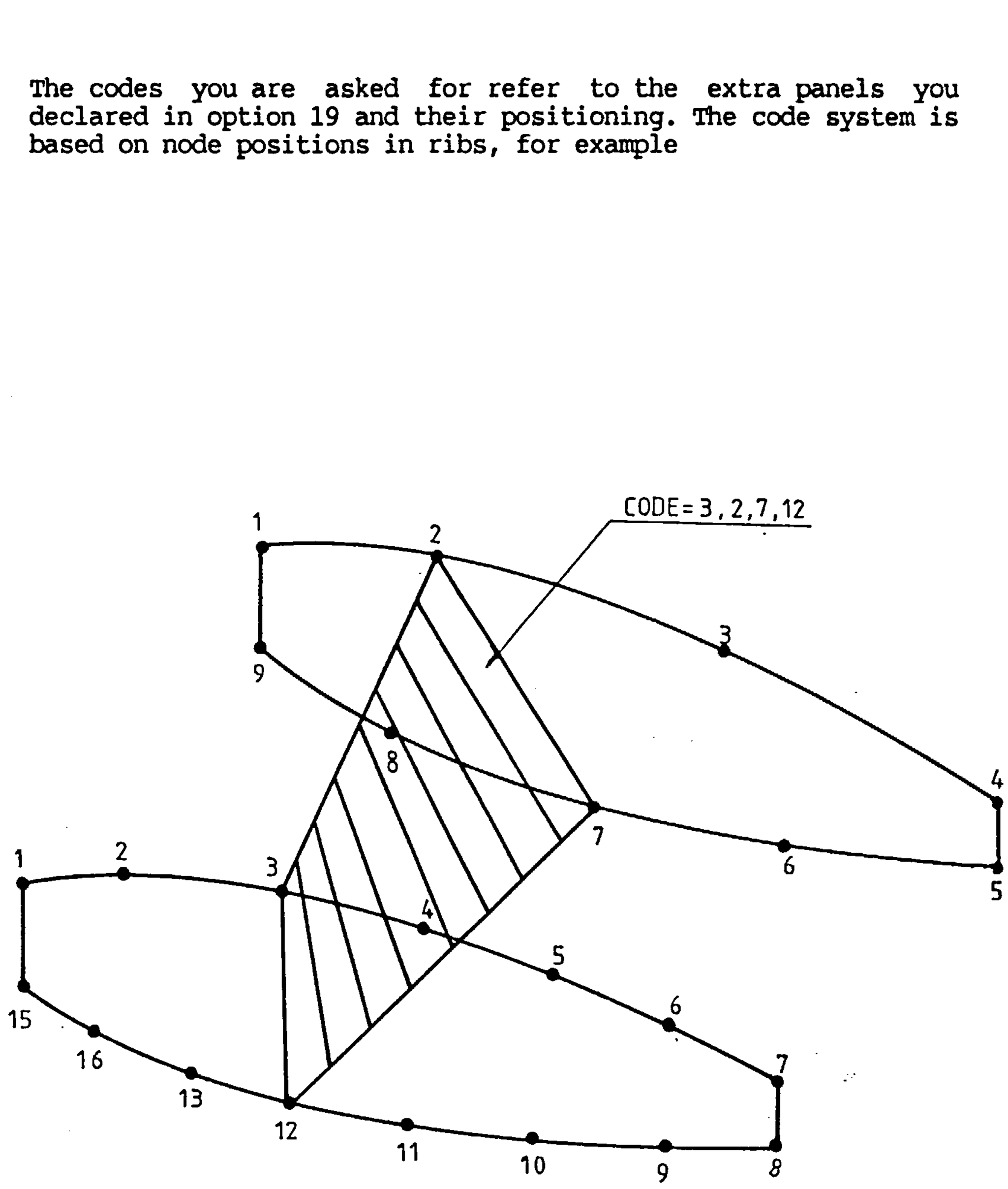


Fig 59. Spar Codes

3.29.26 OPTION 21 Rib Generation Method

This module gives the following display when invoked.

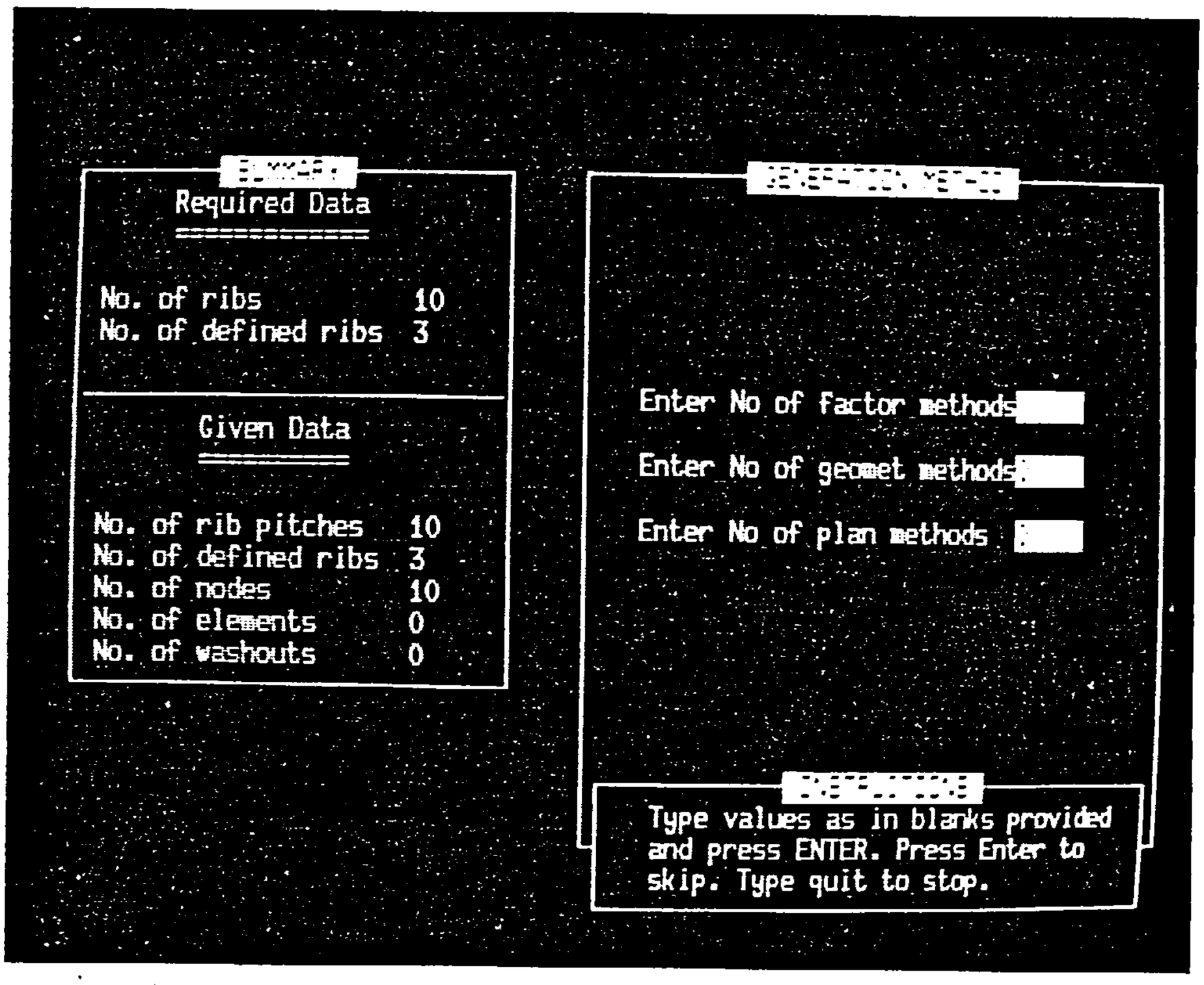


Fig 60. Example of Option 21 Display

Note: When you use this module it is currently necessary to use option 22 before ending the session or an error occurs. This problem will be solved in future versions.

Only one of the options listed on the screen is currently available and that is the factoring method. You are now required to enter the number of ribs you have not defined. For example if the wing has a total of 10 ribs and you have defined 2 the number of factoring method procedures carried out will be at least 8. It may be more than 8 because you may wish to make some ribs complex combinations of several other generated ribs. See option 22.

3.29.27 OPTION 22 Factoring Method Information

You need to carry out this procedure at least once for each undefined rib. When invoked the VDU shows

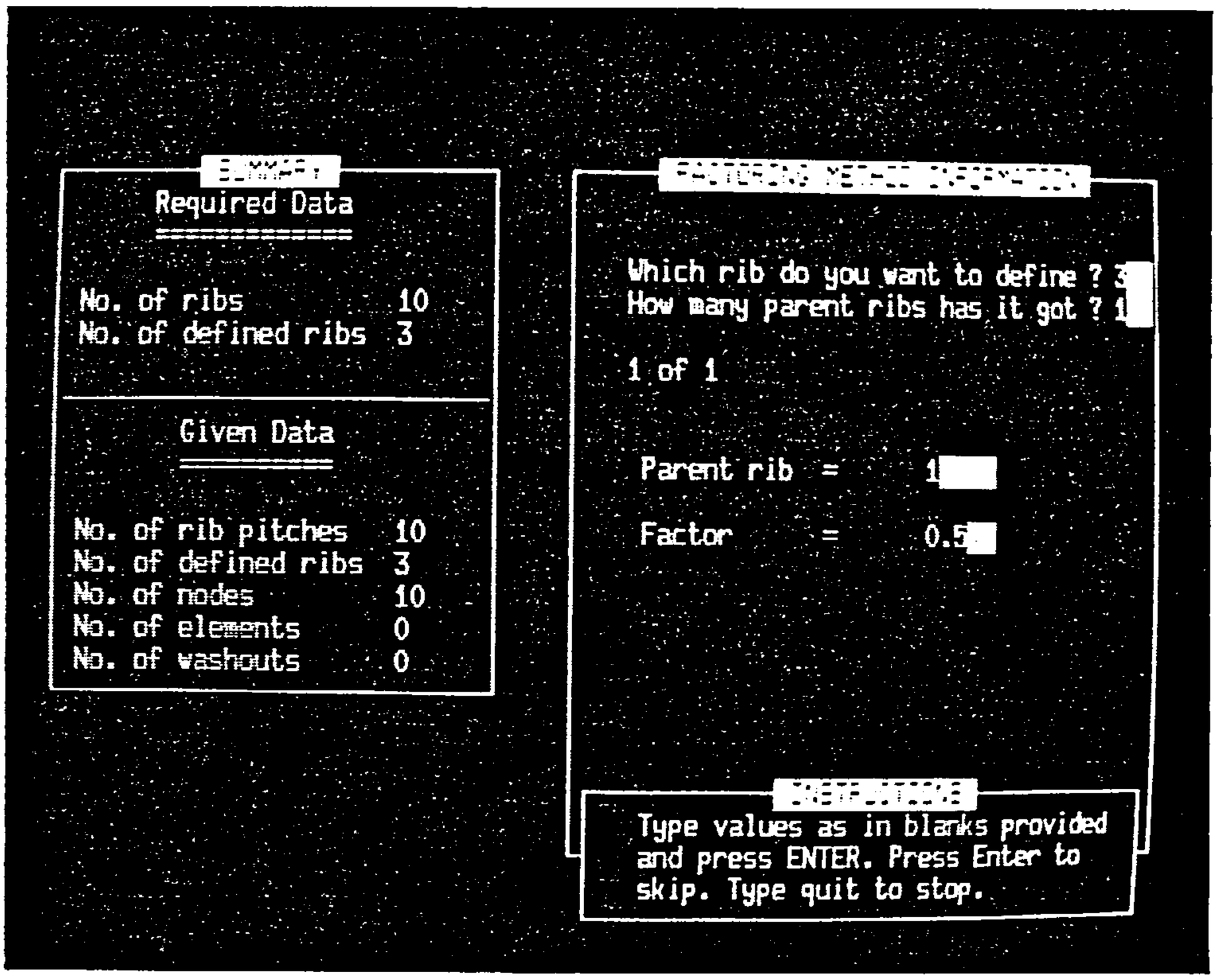


Fig 61. Example of Option 22 Display

You may factor one or more parent ribs to constitute a new rib. So if you make the response shown on the fig 61. You will get a new rib (rib number 3) as shown in fig 62.

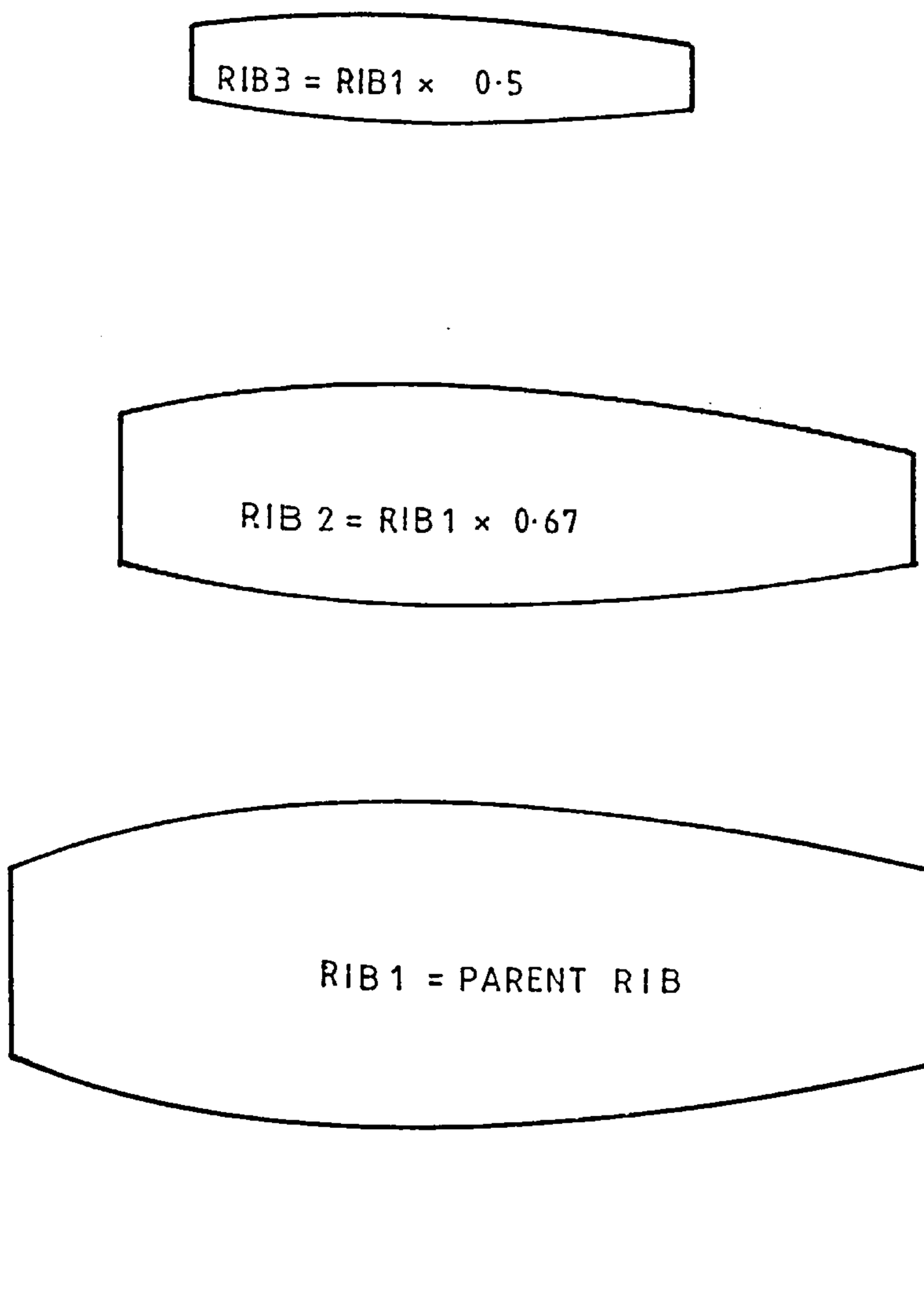


Fig 62. One Parent Rib

On the otherhand you may wish to make a rib a hybrid of two others as shown in fig 63. In this case the factors applied to both parent rib was 0.5.

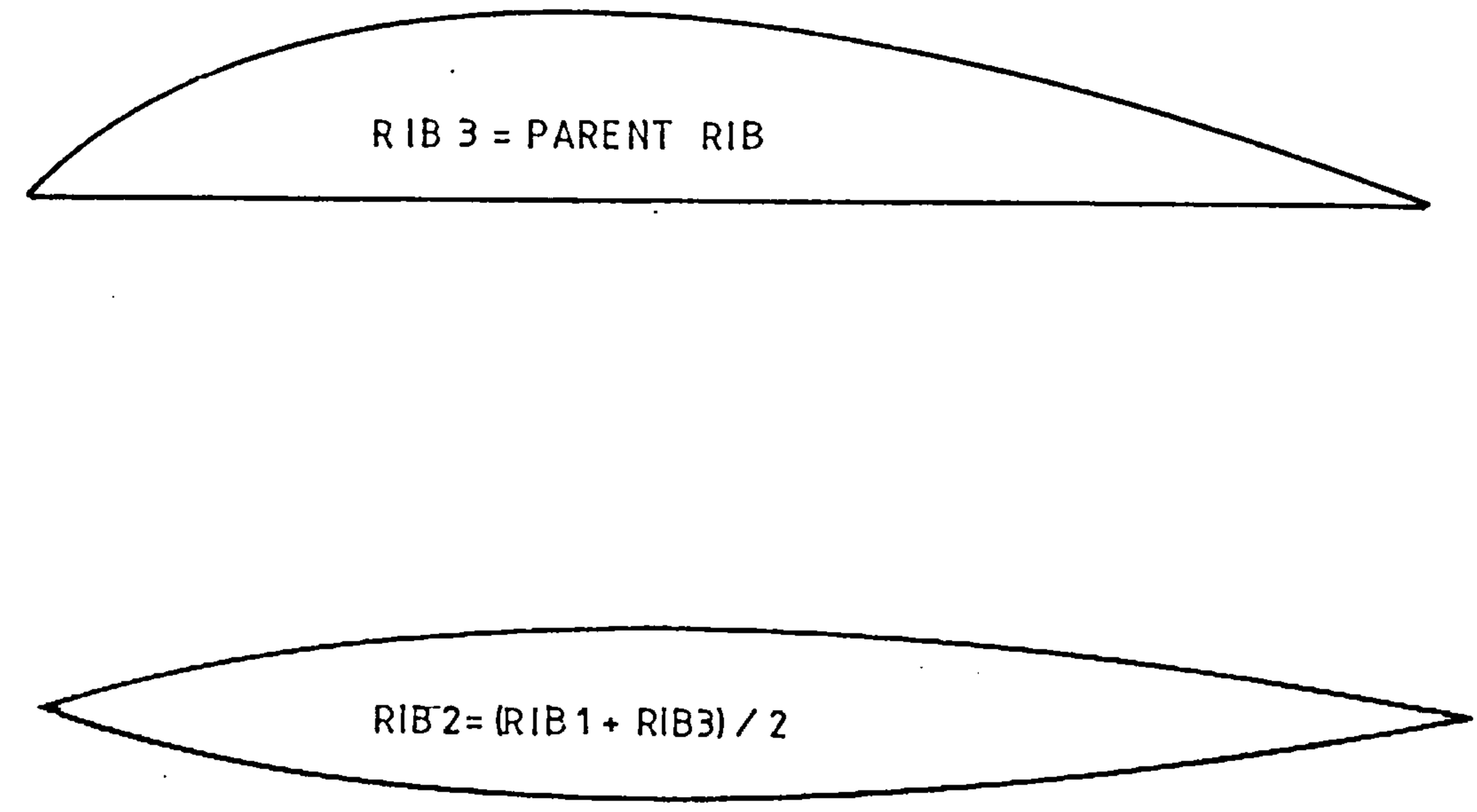


Fig 63. More Than One Parent Rib.

4. Operating WEIGHTS in VAX/VMS

By typing STOP during the WEIGHTS initialization procedure you return to VAX/VMS the VAX operating system. You now have access to all the VAX commands as well as the WEIGHTS commands. This means that you can now monitor the program at your WEIGHTS and other jobs as you work.

It is also possible to start more than one job running at a time. On a small machine such as the College of Aeronautics VAX 750 this is probably not much of an advantage, but can lead to significantly faster execution times in multiprocessor or clustered VAX systems.

5. Module Ordering and Compatibility

The following list shows the order in which module must be executed. Where modules may be executed in any order they are grouped together.

HINT MENUS MENU_C MENU_B MENULC STOP	at anytime

WMESH2_INPUT	before WMESH2
AIRLOAD1_INPUT	before AIRLOAD1
NODFIX_INPUT NODELOAD_INPUT DESVAR_INPUT RANGER	before RUN_BRAMING8

SUB_WMESH1 before MERGER1
WMESH1
PLOT_INPUT
PLOT

STARLINK before OPTIMISE before POSTSTARS

FULLY_STRESSING
FEANALY
SCAN

SUMMASS ITEMIZE

The following groups modules are compatible with each other.

HINT & MENUS

- with any module

WMESH2_INPUT WMESH2 AIRLOAD1_INPUT AIRLOAD1 NODFIX_INPUT NODELOAD_INPUT DESVAR_INPUT RANGER RUN_BEAMING8 PLOT_INPUT PLOT FULLY_STRESSING SCAN SUMMASS ITEMIZE

SUB_WMESH1 STARLINK OPTIMISE POSTSTARS AIRLOAD1 RUN_BEAMING8 NODFIX_INPUT NODELOAD_INPUT DESVAR_INPUT RANGER FEANALY

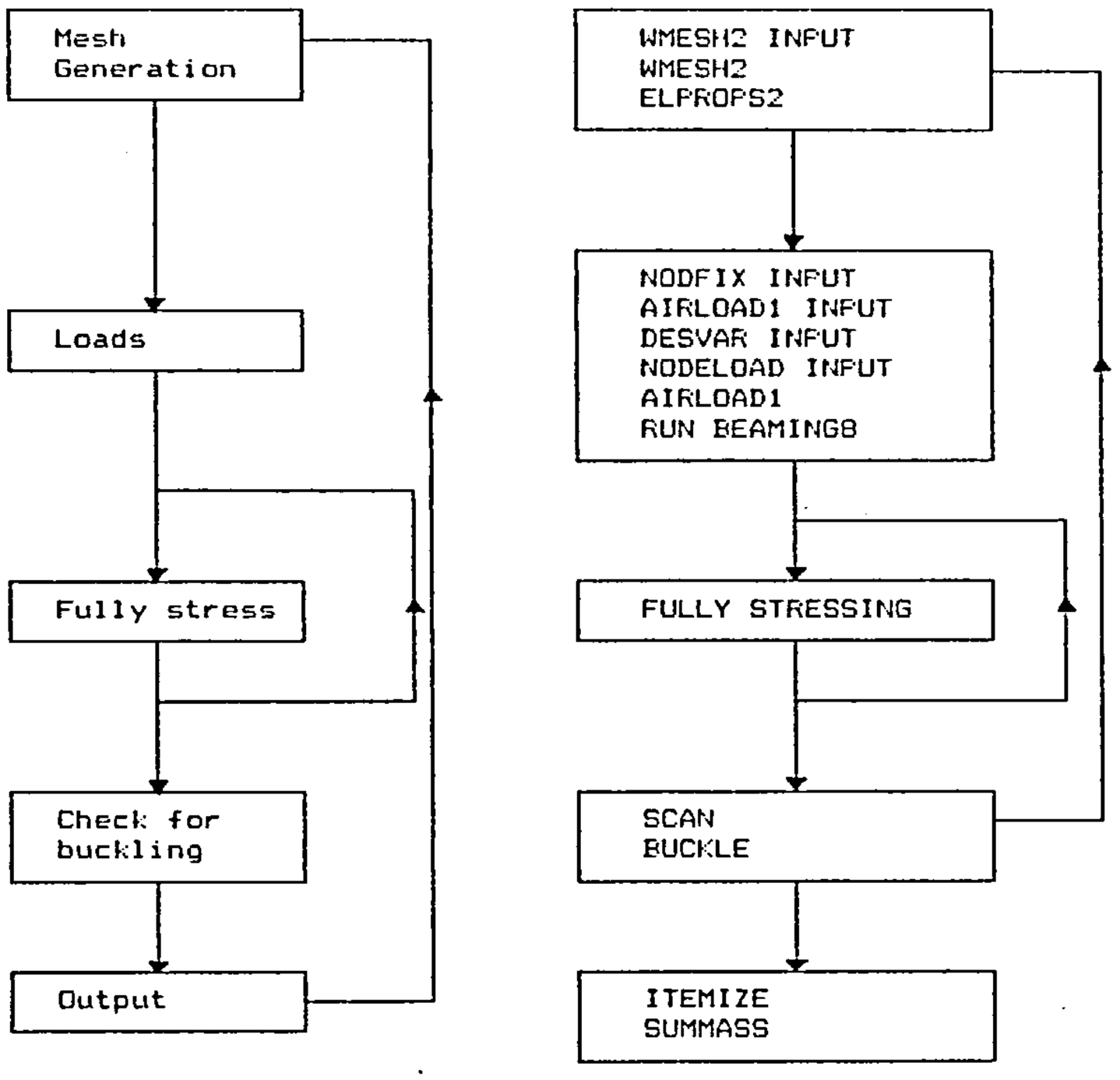
WMESH1 STARLINK OPTIMISE POSTSTARS AIRLOAD1_INPUT AIRLOAD1 RUN_BEAMING8 NODFIX_INPUT NODELOAD_INPUT DESVAR_INPUT RANGER FEANALY

Any other combinations are likely not to work.

6. DESIGN PROCEDURES

Basically when you use WEIGHTS to work out a weight prediction you are going through the entire design procedure for this wing in a fraction of the usual time. The sensitivity of the solution would depend on the complexity of the design and the importance of non structural components. (refer to Analytical Wing Weight Prediction/Estimation using computer based design techniques)

The following procedure was found to be very effective in predicting structural weights of the Cranfield A1 aerobatic plane wing and the VC10 wing.



design process weights commands

It is up to you to decide what the process should be for your wing and execute the modules in WEIGHTS in the correct fashion to achieve that process. Future versions of WEIGHTS will have standard processes but this is subject to testing and validation yet to be completed.

It is possible for you to set up a fixed process for WEIGHTS to carry out automatically. Refer to the WEIGHTS programmers manual on how this is done. Do not confuse this with the process of adding your own modules to WEIGHTS, that is described in the next chapter.

7. INCLUDING YOUR OWN MODULE

Nothing can be simpler. You may develop your program independently but it is worth remembering that there are many data base handling routines available which are described fully in the DBMS manual. Using these for your input and output will simplify "plugging" your module into the WEIGHTS package. If you decide not to comply with the WEIGHTS data base formats (this may be necessary in the case of a program that already exists such as a finite element package) it will be necessary to write some translation programs to allow data to flow between the WEIGHTS database and your module.

Having developed your module and devised any necessary translators, the module may be included in weights. Place the command and image files of your module into the WEIGHTS executive directory (see installation chapter). Then modify the WEIGHTS command file to define a suitable global symbol which will cause the execution of your command procedure. For example if your command procedure filename was YOUROWN.COM then insert a line in WEIGHTS.COM saying;

```
$MYOWN:== @'SYS_DIR'YOUROWN
```

Also insert a line in the file named PROGNAMES.DAT to tell WEIGHTS that your program has been included, in the example above you would have to insert the line

```
MYOWN
```

Your module is now part of WEIGHTS and can be called like any other weights command by typing "MYOWN".

If you do not understand this explanation it would be worth studying the VAX/VMS manuals in depth.

8. INSTALLATION

WEIGHTS requires about 1500 blocks of disk space for its image, command and help files.

To install WEIGHTS transfer the contents of the media which you have been provided into a directory or subdirectory on disk.

Now move to that directory and execute the INSTALL_WEIGHTS command file by typing

```
@ INSTALL_WEIGHTS
```

You will be prompted for;

the WEIGHTS directory name

the LUSAS directory name

the FINEL directory name

the userdisk name

If you do not support LUSAS or FINEL respond by typing NONE.

You will then be asked for the command string which will invoke WEIGHTS. If you do not make an entry here the default is the string WEIGHTS.

The average user will need at least 20000 blocks when executing WEIGHTS and can easily require 100000 on larger runs if several analyses are concurrent.

The user will need to be able to spawn at least 6 sub processes and open 20 files simultaneously on a single run.

Weights Database Documentation.

Department of Aircraft Design
College Of Aeronautics
Cranfield Institute of Technology

February 1985

APPENDIX B

N.A.D. Murphy

CHAPTER 1

INTRODUCTION

This document contains information about the database used by the program called WEIGHTS developed at Cranfield for predicting and estimating aircraft component weights using finite element techniques. The scope of this document will cover aspects which users and programmers using WEIGHTS would need.

The database was constructed using the philosophy contained in the Software Requirements Document (S.R.D. - Ref. 1.) written for WEIGHTS. The S.R.D. laid down requirements for a sequential access database of the simplest form but potential users expressed a need for a random access database.

Sequential access databases are attractive for program development purposes since they are easy to read and simplify debugging of the programs and the input. On computers such as the VAX 11 used to develop WEIGHTS the drawback of this approach is that if it not to suffer from extreme slowness the data has to be fragmented into several files which can be considered untidy.

Random access databases don't suffer from this problem and can have a speed advantage, but do use up more storage space and are not so easily read, hence not lending themselves to debugging and program development work. They are, however, a neater way of storing data.

Bearing these points and the original program design philosophy in mind, both types of file access database were developed each being a mirror image of the other. The philosophy being that the WEIGHTS program could use either database access method, and during development stages the sequential access method would be requested by the programmer whilst the alternative random access method would be requested by a normal user.

The database control programs have been written as a set of routines which are intended to be accessed as external procedures in a similar way to, for example, 'NAG' or 'GINO' routines. In other words, programmers may make use of standard WEIGHTS database control routines by calling them from their programs and linking their compiled code to the WEIGHTS DATABASE LIBRARY.

CHAPTER 2

VARIABLE PARAMETERS

This chapter describes the variables a programmer must declare in any program written to access WEIGHTS DATABASE ROUTINES. The philosophy adopted when developing the database was to simplify the use of the database as well as to simplify the database itself.

In order to reduce the risk of mistakes in passing parameters between routines, it was decided to keep these transfers to a minimum and instead declare most variables used by this group of routines as global variables. In other words, a variable called NODE would have the same meaning and value in all routines. The importance of this is that once the variable contents have been changed by one routine, any other routine would carry out operations on that variable using the new value.

One drawback of this method is that these global variables must be declared in the program which calls the routines. Another is that the programmer must bear the variables in mind and remember not to misuse them if corruption of the database is to be avoided.

To avoid the necessity of manually typing in the long list of required variables a text file has been created with the required declarations in PASCAL. The programmer may simply paste in this file or, if using a VAX/VMS system, use the include facility to instruct the compiler to refer to the file. The name of this file is WGHTDBASE.VAR.

This file also contains the declarations for all the routines so that if it is used the programmer need only make the correct subroutine calls without diverting attention from the programming task at hand in order to declare external routines.

The text that follows includes a list of variables and a brief description of their contents.

Weights Database Documentation
Variable Parameters

Global Variable List

The simplest method of introducing the variables to the programmer was to include a listing of the declaration PASCAL source code in this text. This follows:

```
TYPE DPRC    = DOUBLE                                          ;
TYPE CH80    = VARYING[80] OF CHAR                             ;
TYPE CH80ARR= ARRAY [1..25] OF CH80                            ;
TYPE INTARR1= ARRAY [1..25] OF INTEGER                         ;
TYPE INTARR2= ARRAY [1..500] OF INTEGER                        ;
TYPE ARR1    = ARRAY[1..4000]                  OF DPRC     ;
TYPE ARR2    = ARRAY[1..4000]                  OF INTEGER  ;
TYPE ARR3    = ARRAY[1..2000]                  OF DPRC     ;
TYPE ARR4    = ARRAY[1..30    ,1..2000]        OF INTEGER  ;
TYPE ARR6    = ARRAY[1..4000  ,1..2]           OF INTEGER  ;
TYPE ARR8    = ARRAY[1..4000  ,1..30]          OF DPRC     ;
TYPE ARR10   = ARRAY[1..100]                   OF INTEGER  ;
TYPE NOTE    = PACKED ARRAY[1..60]             OF CHAR     ;
TYPE ALFA    = ARRAY[1..500] OF VARYING[80]    OF CHAR     ;


VAR AEROREFS                 :   [GLOBAL] INTEGER          ;
VAR C                        :   [GLOBAL] INTEGER          ;
VAR CC                       :   [GLOBAL] INTEGER          ;
VAR CHAPTER_MARKER           :   [GLOBAL] INTARR1          ;
VAR CHAPTER_NO               :   [GLOBAL] INTEGER          ;
VAR CNOCODE                  :   [GLOBAL] INTEGER          ;
VAR CRPCODE                  :   [GLOBAL] INTEGER          ;
VAR CURR                     :   [GLOBAL] INTEGER          ;
VAR DATA_DESCRIPTOR          :   [GLOBAL] CH80ARR          ;
VAR DATA_BASE_NAME           :   [GLOBAL] CH80             ;
VAR DBASE                    :   TEXT                      ;
VAR ELEM                     :   [GLOBAL] ARR4             ;
VAR ELEMS                    :   [GLOBAL] INTEGER          ;
VAR ELNO                     :   [GLOBAL] INTEGER          ;
VAR END_MARKER               :   [GLOBAL] INTARR1          ;
VAR FILE_IN                  :   TEXT                      ;
VAR FILE_OUT                 :   TEXT                      ;
VAR FIXES                    :   [GLOBAL] INTEGER          ;
VAR I                        :   [GLOBAL] INTEGER          ;
VAR ICODE                    :   [GLOBAL] INTEGER          ;
VAR J                        :   [GLOBAL] INTEGER          ;
VAR LINE1,LINE2,LINE3        :   [GLOBAL] NOTE             ;
VAR LINE4,LINE5              :   [GLOBAL] NOTE             ;
VAR LINE6,LINE7,LINE8        :   [GLOBAL] NOTE             ;
VAR LINE9,LINE10,LINE11      :   [GLOBAL] NOTE             ;
VAR LINE20,LINE21            :   [GLOBAL] NOTE             ;
VAR LINE80,LINE81            :   [GLOBAL] NOTE             ;
VAR LINE99,LINE90,LINE93     :   [GLOBAL] NOTE             ;
VAR LOADS                    :   [GLOBAL] INTEGER          ;
VAR NEW_DATA_BASE_NAME       :   [GLOBAL] CH80             ;
```

```
VAR NEW_DBASE                 :    TEXT                              ;
VAR NL                        :    [GLOBAL] ARR2                     ;
VAR NO_OF_TRAILING_LINES      :    [GLOBAL] INTARR1                  ;
VAR NODES                     :    [GLOBAL] INTEGER                  ;
VAR NODFIX                    :    [GLOBAL] ALFA                     ;
VAR OLD_END_MARKER            :    [GLOBAL] INTARR1                  ;
VAR OLD_START_MARKER          :    [GLOBAL] INTARR1                  ;
VAR PRP                       :    [GLOBAL] ARR8                     ;
VAR RETSTAT                   :    [GLOBAL] INTEGER                  ;
VAR SEARCH_NAME               :    [GLOBAL] CH80                     ;
VAR SENTENCE                  :    [GLOBAL] CH80                     ;
VAR START_MARKER              :    [GLOBAL] INTARR1                  ;
VAR TYP                       :    [GLOBAL] ARR6                     ;
VAR VERSION_NO                :    [GLOBAL] INTEGER                  ;
VAR USED                      :    [GLOBAL] INTARR2                  ;
VAR WCODE   :    [GLOBAL] PACKED ARRAY [1..8] OF CHAR                ;
VAR X ,Y ,Z                   :    [GLOBAL] ARR1                     ;
VAR XL , YL , ZL              :    [GLOBAL] ARR1                     ;
```

```
(*****************************************************)
(**                                                 **)
(**     Declare routines in WGHTDBASE.PAS .         **)
(**                                                 **)
(*****************************************************)
[EXTERNAL] PROCEDURE READ_STATISTICS                  ;EXTERN ;
[EXTERNAL] PROCEDURE READ_GEOMETRY_DATA               ;EXTERN ;
[EXTERNAL] PROCEDURE READ_ELEMENT_DATA                ;EXTERN ;
[EXTERNAL] PROCEDURE READ_IN_ELEMENT_TYPE_DATA        ;EXTERN ;
[EXTERNAL] PROCEDURE READ_ELEMENT_PROPERTY_DATA       ;EXTERN ;
[EXTERNAL] PROCEDURE READ_NODAL_FIXATION_DATA         ;EXTERN ;
[EXTERNAL] PROCEDURE READ_NODAL_LOADS_DATA            ;EXTERN ;
[EXTERNAL] PROCEDURE WRITE_NEW_STATS                  ;EXTERN ;
[EXTERNAL] PROCEDURE WRITE_GEOMETRY_DATA              ;EXTERN ;
[EXTERNAL] PROCEDURE WRITE_ELEMENT_DATA               ;EXTERN ;
[EXTERNAL] PROCEDURE WRITE_ELEMENT_TYPE_DATA          ;EXTERN ;
[EXTERNAL] PROCEDURE WRITE_ELEMENT_PROP_DATA          ;EXTERN ;
[EXTERNAL] PROCEDURE WRITE_NODAL_FIXATION_DATA        ;EXTERN ;
[EXTERNAL] PROCEDURE WRITE_NODAL_LOADS_DATA           ;EXTERN ;
```

```
(*****************************************************)
(**                                                 **)
(**     Declare routines found in WGDBASE.PAS .     **)
(**                                                 **)
(*****************************************************)
[EXTERNAL] PROCEDURE OPEN_DBASE                       ;EXTERN ;
[EXTERNAL] PROCEDURE OPEN_NEW_DBASE                   ;EXTERN ;
[EXTERNAL] PROCEDURE READ_INDEX                       ;EXTERN ;
[EXTERNAL] PROCEDURE WRITE_INDEX                      ;EXTERN ;
```

```
[EXTERNAL] PROCEDURE SEEK_CHAPTER ( TITLE:CH80 );EXTERN ;
[EXTERNAL] PROCEDURE FIND_CHAPTER ( TITLE:CH80 );EXTERN ;
[EXTERNAL] PROCEDURE START_A_NEW_CHAPTER
                              (TITLE:CH80   ;
                         LINES : INTEGER )   ;EXTERN ;

[EXTERNAL] PROCEDURE DELETE_A_CHAPTER
                               ( TITLE : CH80 ) ;EXTERN ;

[EXTERNAL] PROCEDURE MAKE_NEW_DBASE_CURRENT      ;EXTERN ;
[EXTERNAL] PROCEDURE EXPAND_DBASE
             ( NO_OF_CHAPTERS              : INTEGER
               CHAPTER_TITLE               : CH80ARR
               NO_OF_TRAILING_LINES        : INTARR1 ) ;
                                               ;EXTERN ;

[EXTERNAL] PROCEDURE COMPRESS_DBASE                ;EXTERN ;



(*************************************************)
(**                                             **)
(**      Declare routines in WGDBASE3.PAS .     **)
(**                                             **)
(*************************************************)
[EXTERNAL] PROCEDURE DREAD_STATISTICS              ;EXTERN ;
[EXTERNAL] PROCEDURE DREAD_GEOMETRY_DATA           ;EXTERN ;
[EXTERNAL] PROCEDURE DREAD_ELEMENT_DATA            ;EXTERN ;
[EXTERNAL] PROCEDURE DREAD_IN_ELEMENT_TYPE_DATA    ;EXTERN ;
[EXTERNAL] PROCEDURE DREAD_ELEMENT_PROPERTY_DATA   ;EXTERN ;
[EXTERNAL] PROCEDURE DREAD_NODAL_FIXATION_DATA     ;EXTERN ;
[EXTERNAL] PROCEDURE DREAD_NODAL_LOADS_DATA        ;EXTERN ;
[EXTERNAL] PROCEDURE DWRITE_NEW_STATS              ;EXTERN ;
[EXTERNAL] PROCEDURE DWRITE_GEOMETRY_DATA          ;EXTERN ;
[EXTERNAL] PROCEDURE DWRITE_ELEMENT_DATA           ;EXTERN ;
[EXTERNAL] PROCEDURE DWRITE_ELEMENT_TYPE_DATA      ;EXTERN ;
[EXTERNAL] PROCEDURE DWRITE_ELEMENT_PROP_DATA      ;EXTERN ;
[EXTERNAL] PROCEDURE DWRITE_NODAL_FIXATION_DATA    ;EXTERN ;
[EXTERNAL] PROCEDURE DWRITE_NODAL_LOADS_DATA       ;EXTERN ;
```

Variable Contents

In the following descriptions of the contents of global variables the direct access database specific variables are marked (DAD).

Variable name	Contents
AEROREFS	Number of aerodynamic reference points
C	Incremental counter
CC	Incremental counter
CHAPTER_MARKER	Used by the database expansion routine to contain the chapter numbers of the chapters to be expanded. (DAD)
CHAPTER_NO	Number of chapter currently in use. (DAD)
CNOCODE	Number of chapters not found. (DAD)
CRPCODE	Number of chapters repeated. (DAD)
CURR	Current line number. (DAD)
DATA_DESCRIPTOR	Chapter title or description of contents. (DAD)
DATA_BASE_NAME	Database name. e.g. 'EXAMPLE.' (DAD)
ELEM	Element data. ELEM [I,1] = element number. ELEM [I,2] = number of nodes. ELEM [I,3] = node number. " " " ELEM [I,n] = node number. where n = number of nodes - 2.
ELEMS	Number of elements.
ELNO	Current element number.

END_MARKER	Array of numbers representing the line number in the direct access database file at which the last line of a chapter occurs. (DAD)
FIXES	Number of finite element fixations.
I	Incremental counter.
ICODE	Numerical code. 0 = Chapter not found. 1 = Chapter found. (DAD)
J	Incremental counter.
LINE1 onwards	String .
LOADS	Number of point loads.
NEW_DATA_BASE_NAME	New database name e.g. 'NEW.' (DAD)
NL	Loaded node number.
NO_OF_TRAILING_LINES	Number of lines to be inserted at end of used space in a chapter. (DAD)
NODES	Number of nodes.
NODFIX	Contains node numbers and the direction in which they are restrained.
OLD_END_MARKER	Array of numbers representing the old values of the line number in the direct access database file at which the last line of a chapter occurs. (DAD)
OLD_START_MARKER	Array of numbers representing the old values of the line number in the direct access database file at which the first line of a chapter occurs. (DAD)

PRP	Array of element material property numbers. PRP[I,1] = element number PRP[I,2] = number of properties PRP[I,3] onwards = properties. . These properies include values for thicknesses , modulii , failure stresses , gauge limits , etc.
RETSTAT	Returned status from VAX Run Time Library routines.
START_MARKER	Array of numbers representing the line number in the direct access database file at which the first line of a chapter occurs. (DAD)
TYP	Array of integer representing the element type numbers. e.g. TYP [I,1] = element number TYP [I,2] = element type code. type code 1 = 3 node skin. type code 2 = 4 node skin. type code 3 = 2 node post. type code 4 = 2 node beam.
VERSION_NO	Database version number. e.g. 'STRUT.1' = version 1 'STRUT.12' = version 12 (DAD)
USED	Array containing number of lines used in each chapter. (DAD)
WCODE	Alphanumeric code. 'NOTFOUND' = chapter not found. '***FOUND' = chapter found. (DAD)
X	Array containing finite element nodal X ordinates.
XL	Array containing nodal loads acting in the X direction.
Y	Array containing finite element nodal Y ordinates.

YL	Array containing nodal loads acting in the Y direction.
Z	Array containing finite element nodal Z ordinates.
ZL	Array containing nodal loads acting in the Z direction.

CHAPTER 3

DIRECT ACCESS DATABASE

Introduction

The simplest form of random access file was chosen to reduce the amount of time a programmer would require to become familiar with it and also to simplify implementation of WEIGHTS on different types of computers.

The database consists of one file of the sequentially ordered, direct access type. In other words the data records are organised sequentially, but any specific record may be accessed directly without the need to access other records (as is the case with sequential access files).

The following text describes the database structure and database related WEIGHTS programs.

Weights Database Documentation
Random Access Database

Direct Access Database Structure

The name of the datafile used is specified by the user upon starting WEIGHTS.

The record length specified by default is 500, in other words the physical length allowed for each line of data is 500 characters long. The following file attributes were used:

```
**********************************************************
**                                                      **
**      File variable   = 'DBASE'                       **
**      File name       = variable character string     **
**      Sharing         = 'READWRITE'                   **
**      Organisation    = 'SEQUENTIAL'                  **
**      History         = 'UNKNOWN'                     **
**      Access Method   = 'DIRECT'                      **
**      Record length   = 500                           **
**                                                      **
**********************************************************
```

The file has been structured in the form of chapters the first of which is the contents chapter. This contents chapter contains information about the type of data stored, space used and the position of the begining and end of each chapter.

Contents Chapter

By default this chapter begins on the first line of the file and is 25 lines long.

Each line in this content/index chapter contains the following data records in free format.

start marker, end marker, contents

Where start marker is an integer number refering to the line number in the datafile at which the chapter refered to starts; similarly end marker is an integer number telling us where that chapter ends; contents is an alphanumeric string which may use up all the remaining space in that line and may contain anything the programmer desires. Weight database handling routines use contents to describe the nature of the data, for example, in the case of finite element property data the string, "ELEMENT_PROPERTY_DATA", is stored there.

Standard Chapters

The chapters catered for by Weights Database Routines and their initial sizes are included in the following table.

Contents	Lines allocated
INDEX	25
STATISTICS	10
GEOMETRY	1000
ELEMENT_DATA	3000
ELEMENT_TYPE_DATA	3000
ELEMENT_PROPERTY_DATA	3000
FIXATIONS	1000
NODAL_LOADS	1000

Weights Database Documentation
Random Access Database

Direct Access WEIGHT Database Routines

The following routines have been written to carry out some of the most common tasks to the direct access database.

COMPRESS_DBASE

DELETE_A_CHAPTER

DREAD_ELEMENT_DATA

DREAD_ELEMENT_PROPERTY_DATA

DREAD_GEOMETRY_DATA

DREAD_IN_ELEMENT_TYPE_DATA

DREAD_NODAL_FIXATION_DATA

DREAD_NODAL_LOADS_DATA

DREAD_STATISTICS

DWRITE_ELEMENT_DATA

DWRITE_ELEMENT_PROP_DATA

DWRITE_ELEMENT_TYPE_DATA

DWRITE_GEOMETRY_DATA

DWRITE_NEW_STATS

DWRITE_NODAL_FIXATION_DATA

DWRITE_NODAL_LOADS_DATA

EXPAND_DBASE

FIND_CHAPTER

MAKE_NEW_DBASE_CURRENT

OPEN_DBASE

OPEN_NEW_DBASE

READ_INDEX

SEEK_CHAPTER

Weights Database Documentation
Random Access Database

START_A_NEW_CHAPTER

WRITE_INDEX

The text that follows is a description of the processes carried out by each of these routines and instructions on how to use them.

COMPRESS DATABASE

The database must be in existence and open before this routine can be used. The blank spaces in all the chapters are found and the data is rearranged such that there are no more empty lines. The new compressed data is stored in a new version of the database which then becomes the current database. The old versions are not deleted unless the user requests this.

This routine could be useful if space storage on your system is at a premium as it reduces the amount of space used. But it does mean that the database would have to be expanded if more data is to be added to it, hence leading to increased data processing. It is suggested that this routine be used at the end of a "run" rather than during one.

It requires no formal parameters but, as with all the routines in this document, the variables described in chapter 2 must be declared as described in that chapter.

SUMMARY

The procedure name is: COMPRESS_DBASE

Database must be open.

Database must have contents.

Variables in chapter 2 must be declared.

Creates new compressed database.

The new database is made the default.

New database is left in a write state.

DELETE A CHAPTER

Given the chapter contents string as input this procedure overwrites its contents with empty lines and deletes its entry in the index chapter. It then allocates the space gained to the preceeding chapter.

It requires a value parameter containing the chapter contents string and, as with all the routines in this document, the variables described in chapter 2 must be declared as described in that chapter.

SUMMARY

Procedure name : DELETE_A_CHAPTER

Pascal example : DELETE_A_CHAPTER ('GEOMETRY')

Database must be open.

Variables in chapter 2 must be declared.

Deletes contents of chapter.

Deletes contents to chapter in index chapter.

Increases space allowed for preceding chapter.

Database is left in a write state.

DREAD ELEMENT DATA

This routine reads standard WEIGHTS finite element topology data from the current database. No parameters need to be passed but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DREAD_ELEMENT_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents read (ELEM[J,I]).

Database must contain element topology data.

Database is left in the read state.

DREAD ELEMENT PROPERTY DATA

This routine reads standard WEIGHTS finite element property data from the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DREAD_ELEMENT_PROPERTY_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents read (PRP[I,J]).

Database must contain element property data.

Database is left in the read state.

DREAD GEOMETRY DATA

This routine reads standard WEIGHTS finite element geometry data from the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DREAD_GEOMETRY_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents read (I , X[I] , Y[I] , Z[I]).

Database must contain geometry data.

Database is left in the read state.

DREAD IN ELEMENT TYPE DATA

This routine reads standard WEIGHTS finite element type data from the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DREAD_IN_ELEMENT_TYPE_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents read (TYP[I,J]).

Database must contain element type data.

Database is left in the read state.

DREAD NODAL FIXATION DATA

This routine reads standard WEIGHTS finite element nodal fixation data from the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DREAD_NODAL_FIXATION_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents read (NODFIX [I]).

Database must contain nodal fixation data.

Database is left in the read state.

DREAD NODAL LOADS DATA

This routine reads standard WEIGHTS finite element nodal loads data from the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DREAD_NODAL_LOADS_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents read (NL[I], XL[I], YL[I], ZL[I]).

Database must contain nodal load data.

Database is left in the read state.

DREAD STATISTICS DATA

This routine reads standard WEIGHTS finite element statistics data from the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DREAD_STATISTICS_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents read (<text> , <variable>).
One line per <variable>.

<variable> = NODES , ELEMS , AEROREFS , LOADS, FIXES

<text> = LINE1 , LINE2 , LINE80 , LINE81 , LINE93

Database must contain geometry data.

Database is left in the read state.

DWRITE ELEMENT DATA

This routine writes standard WEIGHTS finite element topology data to the current database. No parameters need to be passed but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DWRITE_ELEMENT_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents written (ELEM[J,I]).

Database must contain element topology data.

Database is left in the write state.

DWRITE ELEMENT PROPERTY DATA

This routine writes standard WEIGHTS finite element property data to the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DWRITE_ELEMENT_PROPERTY_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents written (PRP[I,J]).

Database must contain element property data.

Database is left in the write state.

DWRITE ELEMENT TYPE DATA

This routine writes standard WEIGHTS finite element type data to the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DWRITE_ELEMENT_TYPE_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents written (TYP[I,J]).

Database must contain element type data.

Database is left in the write state.

DWRITE GEOMETRY DATA

This routine writes standard WEIGHTS finite element geometry data to the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DWRITE_GEOMETRY_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents written (I , X[I] , Y[I] , Z[I]).

Database must contain geometry data.

Database is left in the write state.

DWRITE NEW STATS

This routine writes standard WEIGHTS finite element statistics data to the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DWRITE_NEW_STATS

Database must be open.

Variables in chapter 2 must be declared.

Contents written (<text> , <variable>).
One line per <variable>.

<variable> = NODES , ELEMS , AEROREFS , LOADS, FIXES

<text> = LINE1 , LINE2 , LINE80 , LINE81 , LINE93

Database must contain geometry data.

Database is left in the write state.

DWRITE NODAL FIXATION DATA

This routine writes standard WEIGHTS finite element nodal fixation data to the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DWRITE_NODAL_FIXATION_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents written (NODFIX [I]).

Database must contain nodal fixation data.

Database is left in the write state.

DWRITE NODAL LOADS DATA

This routine writes standard WEIGHTS finite element nodal loads data to the current database. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : DWRITE_NODAL_LOADS_DATA

Database must be open.

Variables in chapter 2 must be declared.

Contents written (NL[I], XL[I], YL[I], ZL[I]).

Database must contain nodal load data.

Database is left in the write state.

EXPAND DBASE

The purpose of this routine is to increase or decrease to amount of space allocated to a particular chapter. The routine may expand one or more chapters at a time and makes the necessary changes to the index file after it has done so.

SUMMARY

Procedure name : EXPAND_DBASE

Pascal example :
EXPAND_DBASE (CHAPTERS , TITLES , LINES)

Variable CHAPTERS is an integer giving number of chapters to be expanded. eg. CHAPTERS := 2

Variable TITLES is a character string array containing the chapter names to be expanded.
eg. TITLES [1] := 'GEOMETRY'
TITLES [2] := 'ELEMENT_DATA'

Variable LINES is an integer array containing the incremental increase or decrease in lines to be allocated to those chapters listed in TITLES.
eg. LINES [1] := 500
LINES [2] := -26

Database must be open.

Variables in chapter 2 must be declared.

A new 'expanded' database is created.

The new database is made the default.

If a chapter requested is not found then the variable CNOCODE is set to the number of 'lost' chapters. The process is carried out for the chapters that have been located.

If one or more chapters are repeated in the request the variable CRPCODE is set to the number of repetitions. The process is carried out ignoring the repetitions

The database is left in the write mode.

FIND CHAPTER

Given the title of a chapter this routine set the file pointer to the first line of that chapter.

SUMMARY

Procedure name : FIND_CHAPTER

Pascal example : FIND_CHAPTER ('GEOMETRY')

Variables in chapter 2 must be declared.

The database must be open.

If the chapter is not found the variables
WCODE is set to 'NOTFOUND'
ICODE is set to 0

If the chapter is found the variables
WCODE is set to '***FOUND'
ICODE is set to 1

The variable CHAPTER NO is set to the integer value representing the position of the chapter in the database index.

The database is read enabled.

The database pointer is set to the first line of the required chapter.

MAKE NEW DBASE CURRENT

When called this routine closes current and new database files. It then reopens the new database file as the default file. It can only function if a new database exists.

SUMMARY

Procedure name : MAKE_NEW_DBASE_CURRENT

Variables in chapter 2 must be declared.

Closes current database.

Closes new database.

Opens new database as default (DBASE).

The database is left uninitialised.
ie. it is not read or write enabled.

Weights Database Documentation
Random Access Database

OPEN DBASE

This routine opens a database file with the required characteristics using the database name stored in the variable DATABASENAME. The routine FINDDATABASENAME has to be run at least once before this routine can be executed. The database is left disabled.

SUMMARY

Procedure name : OPEN_DBASE

Variables in chapter 2 must be declared.

File variable : DBASE

Example of filename : 'WING.1;3'
where in this case 1 is the database version number and 3 is the file version number.

OPEN NEW DBASE

This routine manipulates the variable DATABASENAME such that a new version of the name is stored in NEWDATABASENAME. It then opens a new database file with the required characteristics.

SUMMARY

Procedure name: OPEN_NEW_DBASE

Variables in chapter 2 must be declared.

File variable : NEW_DBASE

Example of old file name : 'WING.4'
The new file name would be : 'WING.5'

The new database is left disabled.

READ INDEX

This routine reads the contents of the database index .

SUMMARY

Procedure name : READ_INDEX

Variables in chapter 2 must be declared.

Database must be open.

Contents read :
(START_MARKER[I], END_MARKER[I], USED[I])

Database is left read enabled.

SEEK CHAPTER

Given the title of a chapter, this routine examines the index chapter of the database for a matching title and stores the information about the position and contents of that chapter. This routine is similar to the SEEK_CHAPTER routine except that it does not move the file pointer to the head of the chapter.

SUMMARY

Procedure name : SEEK_CHAPTER

Pascal example : SEEK_CHAPTER ('GEOMETRY')

Variables in chapter 2 must be declared.

The database must be open.

If the chapter is not found the variables
WCODE is set to 'NOTFOUND'
ICODE is set to 0

If the chapter is found the variables
WCODE is set to '***FOUND'
ICODE is set to 1

The variable CHAPTER_NO is set to the integer value representing the position of the chapter in the database index.

The database is read enabled.

START A NEW CHAPTER

Given the title and size of the chapter this routine makes an appropriate entry into the database index chapter.

SUMMARY

Procedure name : START_A_NEW_CHAPTER

Pascal example :
START_A_NEW_CHAPTER ('STATISTICS' , 100)
Where the space allocated is 100 lines.

Database must be open.

Variables in chapter 2 must be declared.

Database is left write enabled.

WRITE INDEX

This routine writes the current database contents and size information into the index chapter of the database.

SUMMARY

Procedure name : WRITE_INDEX

Variables in chapter 2 must be declared.

Database must be open.

Contents written :
(START_MARKER[I], END_MARKER[I], USED[I])

Database is left write enabled.

CHAPTER 4

SEQUENTIAL ACCESS DATABASE

Introduction

This is the simplest possible form of database where data is read to a datafile in the same order it is written and data records follow each other sequentially. This is why this type of data processing can be very slow, to site an example, if there is a data file containing 500 data records stored in this fashion and the 500th data record is the one we wish to read, it would be necessary to read the 499 preceeding records first.

The only way to get over this problem is to keep different types of data in different files, hence reducing the size of the file to be read. This is no real problem since we usually have a fair idea of the type of data we want to read, for example, we would know if we wanted to access nodal geometry data or element topology data. This is the method used by WEIGHTS.

The following text describes the datafiles used and their formats as well as WEIGHTS DATABASE ROUTINES (W.D.R) available for handling them.

Weights Database Documentation
Sequential Access Database

Sequential Access Database Structure

This database consists of many seperate sequential access files stored in the directory or sub-directory specified by the user as the database directory. Each of these files may be printed on hardcopy terminals or on V.D.U.s without any prior processing. This means that they may also be editted.

The most commonly used file variables used in the WEIGHTS DATABASE ROUTINES are 'FILE_IN' and 'FILEOUT' for input and output respectively. With the exception of file names and record lengths the file attributes are the default values assigned by the VAX File Management System.

The text that follows gives the names and record lengths of the files used and briefly describes the contents.

File name	Record Length	Contents
ELEMENT.DAT	(200)	Finite element, element geometry data.
ELPROP2.DAT	(500)	Finite element, element property data.
ELTYPE.DAT	(40)	Finite element, element type data.
GEOMETRY.DAT	(80)	Finite element nodal geometry data.
NODFIX.DAT	(default)	Finite element nodal fixation data.
PNTLOADS.DAT	(default)	Finite element nodal load data.
STATS.DAT	(80)	WEIGHTS statistical data. Statistics about the finite element model and other anlyses as well as run time statistics about computer usage.

Sequential Access WEIGHTS Database Routines

The following routines have been written to carry out some of the most common tasks to the sequential access database.

READ_ELEMENT_DATA

READ_ELEMENT_PROPERTY_DATA

READ_GEOMETRY_DATA

READ_IN_ELEMENT_TYPE_DATA

READ_NODAL_FIXATION_DATA

READ_NODAL_LOADS_DATA

READ_STATISTICS

WRITE_ELEMENT_DATA

WRITE_ELEMENT_PROP_DATA

WRITE_ELEMENT_TYPE_DATA

WRITE_GEOMETRY_DATA

WRITE_NEW_STATS

WRITE_NODAL_FIXATION_DATA

WRITE_NODAL_LOADS_DATA

The text that follows describes the processes carried out by the routines and gives instructions on their use.

READ ELEMENT DATA

This routine reads standard WEIGHTS finite element topology data from the data file 'ELEMENT.DAT' . No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : READ_ELEMENT_DATA

Data file 'ELEMENT.DAT' must exist
and contain element data.

Variables in chapter 2 must be declared.

Contents read (ELEM[J,I]).

Database must contain element topology data.

READ ELEMENT PROPERTY DATA

This routine reads standard WEIGHTS finite element property data from the data file 'ELPROP2.DAT' . No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : READ_ELEMENT_PROPERTY_DATA

Data file 'ELPROP2.DAT' must exist and contain element property data.

Variables in chapter 2 must be declared.

Contents read (PRP[I,J]).

READ GEOMETRY DATA

This routine reads standard WEIGHTS finite element geometry data from the file 'GEOMETRY.DAT' . No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : READ_GEOMETRY_DATA

Data file 'GEOMETRY.DAT' must exist and contain geometry data.

Variables in chapter 2 must be declared.

Contents read (I , X[I] , Y[I] , Z[I]).

READ IN ELEMENT TYPE DATA

This routine reads standard WEIGHTS finite element type data from the data file 'ELTYPE.DAT'. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : READ_IN_ELEMENT_TYPE_DATA

Data file 'ELTYPE.DAT' must exist
and contain element type data.

Variables in chapter 2 must be declared.

Contents read (TYP[I,J]).

READ NODAL FIXATION DATA

This routine reads standard WEIGHTS finite element nodal fixation data from the data file 'NODFIX.DAT'. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : READ_NODAL_FIXATION_DATA

Data file 'NODFIX.DAT' must exist and contain nodal fixation data.

Variables in chapter 2 must be declared.

Contents read (NODFIX [I]).

READ NODAL LOADS DATA

This routine reads standard WEIGHTS finite element nodal loads data from the data file 'PNTLOADS.DAT'. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : READ_NODAL_LOADS_DATA

Data file 'PNTLOADS.DAT' must exist and contain nodal load data.

Variables in chapter 2 must be declared.

Contents read (NL[I], XL[I], YL[I], ZL[I]).

READ STATISTICS DATA

This routine reads standard WEIGHTS finite element statistics data from the data file 'STATS.DAT'. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : READ_STATISTICS_DATA

Data file 'STATS.DAT' must exist
and contain WEIGHTS statistics data.

Variables in chapter 2 must be declared.

Contents read (<text> , <variable>).
One line per <variable>.

<variable> = NODES , ELEMS , AEROREFS ,
LOADS, FIXES

<text> = LINE1 , LINE2 , LINE80 ,
LINE81 , LINE93

WRITE ELEMENT DATA

This routine writes standard WEIGHTS finite element topology data to the data file 'ELEMENT.DAT'. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : WRITE_ELEMENT_DATA

Variables in chapter 2 must be declared.

Contents written (ELEM[J,I]).

Data file 'ELEMENT.DAT' created.

WRITE ELEMENT PROPERTY DATA

This routine writes standard WEIGHTS finite element property data to the data file 'ELPROP2.DAT'. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : WRITE_ELEMENT_PROPERTY_DATA

Data file 'ELPROP.DAT' is created.

Variables in chapter 2 must be declared.

Contents written (PRP[I,J]).

WRITE ELEMENT TYPE DATA

This routine writes standard WEIGHTS finite element type data to the data file 'ELTYPE.DAT'. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : WRITE_ELEMENT_TYPE_DATA

Data file 'ELTYPE.DAT' is created.

Variables in chapter 2 must be declared.

Contents written (TYP[I,J]).

WRITE GEOMETRY DATA

This routine writes standard WEIGHTS finite element geometry data to the data file 'GEOMETRY.DAT'. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : WRITE_GEOMETRY_DATA

Data file 'GEOMETRY.DAT' is created.

Variables in chapter 2 must be declared.

Contents written (I , X[I] , Y[I] , Z[I]).

WRITE NEW STATS

This routine writes standard WEIGHTS finite element statistics data to the data file 'STATS.DAT'. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : WRITE_NEW_STATS

Data file 'STATS.DAT' is created.

Variables in chapter 2 must be declared.

Contents written (<text> , <variable>).
One line per <variable>.

<variable> = NODES , ELEMS , AEROREFS , LOADS, FIXES

<text> = LINE1 , LINE2 , LINE80 , LINE81 , LINE93

WRITE NODAL FIXATION DATA

This routine writes standard WEIGHTS finite element nodal fixation data to the data file 'NODFIX.DAT'. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : WRITE_NODAL_FIXATION_DATA

Data file 'NODFIX.DAT' is created.

Variables in chapter 2 must be declared.

Contents written (NODFIX [I]).

WRITE NODAL LOADS DATA

This routine writes standard WEIGHTS finite element nodal loads data to the data file 'PNTLOADS.DAT'. No parameters need to be passed, but variables must be declared as shown in chapter 2.

SUMMARY

Procedure name : WRITE_NODAL_LOADS_DATA

Data file 'PNTLOADS.DAT' is created.

Variables in chapter 2 must be declared.

Contents written (NL[I], XL[I], YL[I], ZL[I]).

A P P E N D I X C

Example of WEIGHTS

output

C1

Rib 1
Number of elements = 8
Number of active elements = 0
Component weight = 1.075346-01
Rib 2
Number of elements = 8
Number of active elements = 0
Component weight = 2.108805E-01
Rib 3
Number of elements = 8
Number of active elements = 0
Component weight = 1.074606E-01
Rib 4
Number of elements = 8
Number of active elements = 0
Component weight = 1.074606E-01
Rib 5
Number of elements = 8
Number of active elements = 0
Component weight = 1.074606E-01
Rib 6
Number of elements = 8
Number of active elements = 0
Component weight = 1.074606E-01
Rib 7
Number of elements = 8
Number of active elements = 0
Component weight = 1.074606E-01
Rib 8
Number of elements = 8
Number of active elements = 0
Component weight = 1.074606E-01
Rib 9
Number of elements = 8
Number of active elements = 0
Component weight = 1.074606E-01
Rib 10
Number of elements = 8
Number of active elements = 0
Component weight = 1.074606E-01
Rib 11
Number of elements = 8
Number of active elements = 0
Component weight = 1.074606E-01
Rib 12
Number of elements = 7
Number of active elements = 0
Component weight = 8.261655E-02
Rib 13
Number of elements = 6
Number of active elements = 0
Component weight = 7.842105E-02
Rib 14
Number of elements = 5
Number of active elements = 1
Component weight = 3.859105E-02
Rib 15
Number of elements = 4
Number of active elements = 0
Component weight = 5.328155E-02

```
 Rib 16
Number of elements          =     3
Number of active elements =     3
Component weight            =   2.12303E-01
 Top Skin Bay 1
Number of elements          =     7
Number of active elements =     0
Component weight            =   7.41202E-01
 Top Skin Bay 2
Number of elements          =     8
Number of active elements =     0
Component weight            =   7.24331E-01
 Top Skin Bay 3
Number of elements          =     6
Number of active elements =     0
Component weight            =   7.99805E-01
 Top Skin Bay 4
Number of elements          =     6
Number of active elements =     0
Component weight            =   9.15756E-01
 Top Skin Bay 5
Number of elements          =     6
Number of active elements =     0
Component weight            =   9.33443E-01
 Top Skin Bay 6
Number of elements          =     6
Number of active elements =     6
Component weight            =   4.67976E-01
 Top Skin Bay 7
Number of elements          =     8
Number of active elements =     0
Component weight            =   3.07056E-01
 Top Skin Bay 8
Number of elements          =     8
Number of active elements =     0
Component weight            =   2.73061E-01
 Top Skin Bay 9
Number of elements          =     6
Number of active elements =     0
Component weight            =   4.79871E-01
 Top Skin Bay 10
Number of elements          =     8
Number of active elements =     0
Component weight            =   4.44673E-01
 Top Skin Bay 11
Number of elements          =     9
Number of active elements =     1
Component weight            =   3.19717E-01
 Top Skin Bay 12
Number of elements          =     7
Number of active elements =     1
Component weight            =   2.73730E-01
 Top Skin Bay 13
Number of elements          =     6
Number of active elements =     1
Component weight            =   1.94838E-01
 Top Skin Bay 14
Number of elements          =     5
Number of active elements =     1
Component weight            =   3.79501E-01
```

Top Skin Bay 15
Number of elements = 4
Number of active elements = 2
Component weight = 2.57209E+00
Bottom Skin bay 1
Number of elements = 9
Number of active elements = 0
Component weight = 1.09237E+00
Bottom Skin Bay 2
Number of elements = 8
Number of active elements = 0
Component weight = 7.77220E-01
Bottom Skin Bay 3
Number of elements = 8
Number of active elements = 0
Component weight = 8.37650E-01
Bottom Skin Bay 4
Number of elements = 8
Number of active elements = 0
Component weight = 9.55780E-01
Bottom Skin Bay 5
Number of elements = 8
Number of active elements = 0
Component weight = 9.74276E-01
Bottom Skin Bay 6
Number of elements = 8
Number of active elements = 0
Component weight = 4.83358E-01
Bottom Skin Bay 7
Number of elements = 8
Number of active elements = 0
Component weight = 3.14787E-01
Bottom Skin Bay 8
Number of elements = 8
Number of active elements = 0
Component weight = 2.30183E-01
Bottom Skin Bay 9
Number of elements = 8
Number of active elements = 0
Component weight = 4.83320E-01
Bottom Skin Bay 10
Number of elements = 8
Number of active elements = 0
Component weight = 4.35466E-01
Bottom Skin Bay 11
Number of elements = 8
Number of active elements = 0
Component weight = 3.22262E-01
Bottom Skin Bay 12
Number of elements = 7
Number of active elements = 0
Component weight = 2.37686E-01
Bottom Skin Bay 13
Number of elements = 6
Number of active elements = 0
Component weight = 1.95541E-01
Bottom Skin Bay 14
Number of elements = 5
Number of active elements = 0
Component weight = 1.94906E-01

```
 Bottom Skin Bay 15
Number of elements          =     4
Number of active elements   =     3
Component weight            =   2.11872E+00
 Rear Spar
Number of elements          =    15
Number of active elements   =     3
Component weight            =   1.17953E+00
 Front Spar
Number of elements          =    15
Number of active elements   =     5
Component weight            =   2.63501E+00
 Ribs Sub Total
Number of elements          =   113
Number of active elements   =     4
Component weight            =   1.75070E+00
 Stringers
Number of elements          =   250
Number of active elements   =     7
Component weight            =   1.07822E+01
 Top Skin Total
Number of elements          =   109
Number of active elements   =     6
Component weight            =   9.25207E+00
 Bottom Skin Total
Number of elements          =   111
Number of active elements   =     3
Component weight            =   9.75452E+00
 Wing Weight
Number of elements          =   613
Number of active elements   =    28
Component weight            =   3.59339E+01
```

```
 Rib 1
Number of elements          =     8
Number of active elements =     0
Component weight            =  1.07534E-01
 Rib 2
Number of elements          =     8
Number of active elements =     0
Component weight            =  2.03301E-01
 Rib 3
Number of elements          =     8
Number of active elements =     0
Component weight            =  1.07460E-01
 Rib 4
Number of elements          =     8
Number of active elements =     0
Component weight            =  1.07460E-01
 Rib 5
Number of elements          =     8
Number of active elements =     0
Component weight            =  1.07460E-01
 Rib 6
Number of elements          =     8
Number of active elements =     0
Component weight            =  1.07460E-01
 Rib 7
Number of elements          =     8
Number of active elements =     0
Component weight            =  1.07460E-01
 Rib 8
Number of elements          =     8
Number of active elements =     0
Component weight            =  1.07460E-01
 Rib 9
Number of elements          =     8
Number of active elements =     0
Component weight            =  1.07460E-01
 Rib 10
Number of elements          =     8
Number of active elements =     0
Component weight            =  1.07460E-01
 Rib 11
Number of elements          =     8
Number of active elements =     0
Component weight            =  1.07460E-01
 Rib 12
Number of elements          =     7
Number of active elements =     0
Component weight            =  8.26165E-02
 Rib 13
Number of elements          =     6
Number of active elements =     1
Component weight            =  7.22731E-02
 Rib 14
Number of elements          =     5
Number of active elements =     1
Component weight            =  3.85910E-02
 Rib 15
Number of elements          =     4
Number of active elements =     0
Component weight            =  4.04204E-02
```

```
 Rib 16
Number of elements          =     3
Number of active elements =     3
Component weight            =   2.12303E-01
 Top Skin Bay 1
Number of elements          =     7
Number of active elements =     0
Component weight            =   5.49560E-01
 Top Skin Bay 2
Number of elements          =     6
Number of active elements =     0
Component weight            =   5.24361E-01
 Top Skin Bay 3
Number of elements          =     8
Number of active elements =     0
Component weight            =   5.60319E-01
 Top Skin Bay 4
Number of elements          =     8
Number of active elements =     0
Component weight            =   6.51507E-01
 Top Skin Bay 5
Number of elements          =     8
Number of active elements =     0
Component weight            =   7.63643E-01
 Top Skin Bay 6
Number of elements          =     8
Number of active elements =     0
Component weight            =   4.36126E-01
 Top Skin Bay 7
Number of elements          =     8
Number of active elements =     1
Component weight            =   2.87169E-01
 Top Skin Bay 8
Number of elements          =     8
Number of active elements =     0
Component weight            =   2.55131E-01
 Top Skin Bay 9
Number of elements          =     8
Number of active elements =     1
Component weight            =   4.57714E-01
 Top Skin Bay 10
Number of elements          =     8
Number of active elements =     1
Component weight            =   4.30621E-01
 Top Skin Bay 11
Number of elements          =     8
Number of active elements =     0
Component weight            =   3.09407E-01
 Top Skin Bay 12
Number of elements          =     7
Number of active elements =     0
Component weight            =   2.58630E-01
 Top Skin Bay 13
Number of elements          =     6
Number of active elements =     0
Component weight            =   1.95555E-01
 Top Skin Bay 14
Number of elements          =     5
Number of active elements =     1
Component weight            =   3.79684E-01
```

```
 Top Skin Bay 15
Number of elements          =     4
Number of active elements =     1
Component weight            =   2.57540E+00
 Bottom Skin Bay 1
Number of elements          =     9
Number of active elements =     0
Component weight            =   8.39269E-01
 Bottom Skin Bay 2
Number of elements          =     8
Number of active elements =     0
Component weight            =   5.85409E-01
 Bottom Skin Bay 3
Number of elements          =     8
Number of active elements =     0
Component weight            =   6.07935E-01
 Bottom Skin Bay 4
Number of elements          =     8
Number of active elements =     0
Component weight            =   6.68580E-01
 Bottom Skin Bay 5
Number of elements          =     8
Number of active elements =     0
Component weight            =   7.70763E-01
 Bottom Skin Bay 6
Number of elements          =     8
Number of active elements =     0
Component weight            =   4.33537E-01
 Bottom Skin Bay 7
Number of elements          =     8
Number of active elements =     0
Component weight            =   2.84691E-01
 Bottom Skin Bay 8
Number of elements          =     8
Number of active elements =     0
Component weight            =   2.53778E-01
 Bottom Skin Bay 9
Number of elements          =     8
Number of active elements =     0
Component weight            =   4.43747E-01
 Bottom Skin Bay 10
Number of elements          =     8
Number of active elements =     1
Component weight            =   4.14245E-01
 Bottom Skin Bay 11
Number of elements          =     8
Number of active elements =     1
Component weight            =   3.12791E-01
 Bottom Skin Bay 12
Number of elements          =     7
Number of active elements =     1
Component weight            =   2.66628E-01
 Bottom Skin Bay 13
Number of elements          =     6
Number of active elements =     1
Component weight            =   1.93285E-01
 Bottom Skin Bay 14
Number of elements          =     5
Number of active elements =     0
Component weight            =   1.59386E-01
```

```
 Bottom Skin Bay 15
Number of elements          =     4
Number of active elements =     3
Component weight            =   2.13381E+00
 Rear Spar
Number of elements          =    15
Number of active elements =     5
Component weight            =   1.13643E+00
 Front Spar
Number of elements          =    15
Number of active elements =     5
Component weight            =   1.93078E+00
 Ribs Sub Total
Number of elements          =   113
Number of active elements =     5
Component weight            =   1.72418E+00
 Stringers
Number of elements          =   250
Number of active elements =     5
Component weight            =   2.15644E+01
 Top Skin Total
Number of elements          =   109
Number of active elements =     5
Component weight            =   8.03528E+00
 Bottom Skin Total
Number of elements          =   111
Number of active elements =     7
Component weight            =   8.42287E+00
 Wing Weight
Number of elements          =   613
Number of active elements =    30
Component weight            =   4.34639E+01
```