[redacted]

UNCLASSIFIED

REF WBS 4.7.05

NASP Contractor Report 1092

EZDESIT
A Computer Program for Structural Element Sizing and Vehicle Weight Prediction

Jeffrey A. Cerro and C. P. Shore

Contract NAS1-19000

July 1990

[redacted]

[redacted]

Lockheed Engineering & Sciences Company

Langley Program Office
Hampton, Virginia

UNCLASSIFIED

NASP Contractor Report 1092
REF WBS 4.7.05

EZDESIT
A Computer Program for Structural Element Sizing and Vehicle Weight Prediction

Jeffrey A. Cerro

Lockheed Engineering & Sciences Company
Hampton, Virginia

C. P. Shore

Langley Research Center
Hampton, Virginia

Prepared for
NASA Langley Research Center
under Contract NAS1-19000

Printed by
NASA Langley Research Center

July 1990

Lockheed
Engineering & Sciences Company

Langley Program Office
Hampton, Virginia

Disclaimer

The herin distributed programs, EZDESIT and EZBATCH are computer codes in the development phase at Lockheed Engineering and Sciences Corporation. They are being provided at this time to aquaint interested designers with a technique useful for conceptual/ preliminary vehicle weight estimation based on structural analysis. The programs have not been subject to quality control guidelines. Errors of any nature are the responsibility of users to correct at their discretion. Lockheed Corporation will in no event be liable for incidental or consequential damages resulting from the application of these programs.

SUMMARY

A computer program which couples finite element structural analysis to an element sizing code and a graphical pre/post processing code is described. Bar, beam and plate type structural elements are sized to support finite element calculated stress resultants (ie: element running loads). Element sizing criteria consist of minimum gage, buckling, yield and ultimate strength failure mode checks. The program may be run in either an interactive or batch environment. In the interactive mode, the program is menu driven to create a comfortable user interface. VAX FORTRAN and the VMS operating system are required.

To illustrate the element sizing process, a wing similar to that of the space shuttle is sized for multiple load cases that include combined thermal and mechanical loads. All major features of the program are demonstrated in the example problem.

INTRODUCTION

Design of structural systems and aerospace vehicles in particular is inherently multidisciplinary. As observed in reference 1 an optimum vehicle cannot be defined without considering the coupled effects of all design disciplines. At the stage of design where vehicle mold lines, structural concepts, and system layouts are in a state of flux the designer requires performance data to make configuration decisions. Often the right choices can be made only by comparing vehicle systems at equivalent technology levels. In terms of structures, such parametric studies require a knowledge base of structural concepts and associated weights. Vehicle performance codes interrogate the knowledge base and determine structural weight requirements for a given design. For conventional technology this knowledge base is often represented as empirical equations similar to those given in reference 2. As previously pointed out in (ref. 1) advanced technology vehicles require building of the knowledge base using fundamental analysis tools for each discipline.

This paper presents a methodology useful to structural designers tasked with predicting weights of a structural system (typically an aerospace vehicle) for design trade-off studies and preliminary design studies. Finite element structural analysis and structural failure mode sizing of finite elements, are used to calculate component weights. A rigid procedure is defined which implements the same element sizing techniques for any structural system, flexibility exists in that users may input multiple mechanical and thermal load cases, different material properties, and different structural concepts. Rigid procedures are required to fairly assess one structural system against another while flexibility is

necessary to develop a database useful for configuration layout and performance analysis. During preliminary design, element size results can be used as initial conditions for more rigorous mathematical optimization and strength analysis checks.

Failure mode calculated structural weights must be increased to account for what are typically called non-optimum effects. EZDESIT allows the user to input different non-optimum factors for various element construction types but does not address how non-optimum factors should be derived. Inclusion of non-optimum factors is left for users to implement their preferred standards and techniques. An example of using structural element sizing algorithms to predict vehicle weight for purposes of conceptual design is given in reference 3.

Data input requirements, program operation, and program limitations are explained in Appendix A. To maintain order among the various data files, Appendix B defines a standard file naming convention. A complete program execution, illustrating the sizing of a space shuttle type wing, is given in Appendix C. Assuming that EZDESIT has been used to create a parametric set of design data, a technique for using calculated structural weights in a vehicle performance study is explained in Appendix D. Finally, Appendices E and F explain how users may write additional element sizing routines, and also document contents of the distribution tape.

ANALYSIS CAPABILITIES

Program Overview

EZDESIT is a program which ties PATRAN finite element modeling (ref. 4) to the EAL finite element analysis program (ref. 5), for purposes of calculating element sizes (cross sectional dimensions). Interactive menu prompts allow users to translate PATRAN output (finite element model descriptions) to EAL input, structurally size finite elements, and then translate EAL output to PATRAN postprocessing input. Since EAL calculated element internal loads depend on element sizes, and element sizes depend on EAL calculated internal loads, the process is iterative. EZBATCH, a subset of the EZDESIT program, makes it possible to submit a batch process which executes element sizing for a user specified number of iterations. Batch sizing is used once the structural model has been verified by interactive PATRAN preprocessing, and preliminary EAL runs. Both sizing codes are written in VAX FORTRAN and operate on a VAX computer under the VMS operating system (ref. 6). Users must be proficient with PATRAN, EAL, and VMS to run EZDESIT.

Structural modeling is a graphics intensive procedure and numerous preprocessors exist for such purposes. Color graphics and interactive model manipulation have become state of the art for structural analysis. PATRAN provides interactive color graphics modeling capability as well as the ability to color postprocess results. The output of PATRAN is a data file which describes a structural finite element model including loads, constraints and material property information.

EZDESIT has the capability to convert PATRAN neutral file finite element model representations to EAL static solution runstreams. EAL is a general purpose structural analysis computer language. The power of EAL is somewhat diminished when using the structured runstream format created and required by EZDESIT; however, EAL runstreams are saved to a file and analysts may later modify them to perform other analyses. During translation from PATRAN to EAL, model geometry is converted first. Geometry includes nodal locations, element connectivity, and beam reference frames, as well as material property information and preliminary section property data. After model geometry is converted another translation option reads PATRAN node and element loading, responds to user inputs, and creates applied loadsets and static solution processors for EAL.

Figure 1 shows the interrelations among PATRAN, EAL, EZDESIT, and the VAX VMS operating system. All three programs are accessed using commands from VMS. EZDESIT is the central procedure to and from which PATRAN, EAL, and material property information flow. Data flow is primarily in the form of ASCII file access. This helps to modularize various stages of the sizing process, and permits later integration of other analysis programs. The material property data file (figure 1) is a user maintained file of temperature dependent material property information. EZDESIT uses element temperatures obtained from the PATRAN neutral file to interpolate material properties from this file. Temperature dependent allowables are then used for element sizing, and temperature dependent moduli are used to create hot or cold condition element stiffness matrices. EAL uses the runstream created by EZDESIT to produce element stress resultant files, one for each loadcase. EZDESIT then uses these stress resultant files for input to the element sizing subroutines, which calculate new element sizes. When EZDESIT has finished sizing elements the results are stored in PATRAN results file type format. PATRAN can display stress resultants, element weight, element failure mode, and dominant structural loadsets using color graphics and contour plotting capabilities. EZDESIT is used as a post processor to produce tabulated weights of PATRAN named components.

Figure 2. indicates the sequence of events for EZBATCH element sizing. EZBATCH is the batch counterpart to program EZDESIT. During batch sizing, EAL is automatically

executed by spawning a VAX subprocess. This technique performs automatic iterative updating of element stiffnesses with currently calculated element sizes. Updated element stiffnesses are generated by subroutine calls to program STIMAT, (ref. 7). Iteration is required until element sizes do not change significantly between sequential executions of the finite element and element sizing analyses. The loop shown in figure 2 assures element stress resultants will eventually be consistent with element stiffness matrices. When thermal load cases are present looping is required, because thermally generated stress resultants are based on dimensions of the previous iteration. A problem not addressed with EZDESIT or EZBATCH is that of calculating new element temperatures and temperature gradients for resized elements.

Element Sizing Capabilities

Figure 3. shows the types of element internal loads which EZDESIT considers during element sizing. Only two, three and four node elements may be used in the finite element model. Two node elements must be EAL type E23 or E24 which are bar and beam elements respectively. Plate elements must be type E33 or E43, these are typical three and four node type elements. As shown (in the figure) E23 elements support only axial loads. E24 elements represent a planar beam and support axial, shear and moment end loads. Plate elements support running loads and moments in the x, y, and shear direction. For all of the above elements, stress resultants fed to the sizing code as calculated by EAL may be either from a mechanical or thermal loadset. EZDESIT will combine a mechanical and thermal set of stress resultants so users can create a sizing condition where both types of loading are present.

For bar elements, figure 4, the only variable determined during the sizing process is cross sectional area. Planar beam elements have three variables, cap cross sectional area, web height and web thickness. The plate element has a different number of design variables depending upon its construction type. Currently, sizing codes exist for honeycomb and corrugated web type plate elements. For a honeycomb element, facesheet thickness and core height are design variables. Corrugated web elements have only a thickness variable.

When sizing a model for multiple loadcases, files called "worst case files" of element dimensions and unit weights are created. Worst case files are a simplistic way of creating a set of dimensions and element weights that encompass effects from each applied loadcase. Between iterations of element sizing there must be one set of element geometry (ie: dimensions) used to calculate element stiffness matrices. This worst case dimension file is created by finding which loadcase causes each element to be heaviest. Dimensions

determined by that loadcase are used for the next sizing iteration. Also element unit weights, (Lb/Ft for two node, and Lb/Sq Ft for plate elements) are stored in a worst case element unit weight file for postprocessing. When element sizing is completed there are two ways available to view calculated results. Element stress resultants, unit weight, failure mode, and dominant loadset can all be viewed using the color graphic postprocessing capabilities of PATRAN. Also EZDESIT will summarize calculated weights by failure mode, dominant loadset, and element type. Weights for PATRAN named components are determined upon request from EZDESIT. Named components are a useful way of bookkeeping model weights and at least one PATRAN named component must exist for EZDESIT to execute properly.

Program Control

The interactive version of the element sizing code (EZDESIT) is a completely menu driven process. Figure 5 shows the options offered to designers. Element sizing typically requires a rote procedure of stepping through modules available in EZDESIT; however, when proper datafiles exist from previous work, various modules may be used at random to aid in the structural design process. As stated previously, data transfer between modules is done by accessing ASCII files. Only EAL creates machine language libraries of information and these libraries are not accessed by the sizing code. There is a standard set of file names and file types, defined in appendix B, used for keeping track of which files are PATRAN format, which are dimension files and so forth. The interactive process also allows stepping through the element sizing analysis, examining intermediate results, then restarting from various points in the design cycle. Designers can step between EAL execution, PATRAN processing, EAL updates, modeling updates, and material property changes, until design requirements are achieved. When going between EZDESIT and EAL users will be at the VAX VMS level of computing. From VMS, users have the opportunity to examine ASCII files and edit certain required data files if necessary.

The batch version of element sizing (EZBATCH) has more stringent file naming conventions and procedural orders than the interactive environment. Batch execution is used once the designer is satisfied he has a verified structural model, including section property and load definitions. The program will iterate a user specified number of times and create ascending version numbers of the worst case dimension file. Users then compare sequential versions of the dimension file to determine if element geometry convergence has been satisfactorally achieved. Such comparisons are easily made using the VAX DCL file differences command.

Data File Management

No formal database is required to implement EZDESIT or EZBATCH. The library type database system created by executing EAL is used entirely by processors within the EAL runstream and not by other application modules. The system of ASCII files defined in appendix B were useful during program development and may be useful to designers by providing access to model information, such as element geometry, for input to other discipline analysis codes. Figure 6 shows the relationship between data files and the major processors of EZESIT.

The first use of PATRAN is to produce an ASCII neutral file titled PATRAN.OUT. PATRAN.OUT is accessed frequently by the batch and interactive versions of the element sizing process. As has been explained, EZDESIT will create an EAL runstream. EZDESIT will also create element dimension and unit weight files for any specified loadset, as well as worst case dimension and unit weight files. EAL execution creates files of element stress resultants and nodal displacements. Stress resultant files are required input for EZDESIT element sizing. Options in EZDESIT also convert EAL generated stress resultant and nodal displacement files to a results file format consistent with PATRAN file structure requirements. Consistent file naming conventions are key to keeping a model organized. Users should adopt the file naming convention utilized by the batch version of the sizing code defined in appendix B. A temperature dependent material property data file also must be created before element sizing can begin. Typically this file, titled MATPROP.PRN, is maintained by some type of personal computer database program (ref. 8). An ASCII version of the information as described in appendix A must exist in the directory dedicated to a particular finite element model. The types of files created by PATRAN, EZDESIT, and EAL, are described in appendix B.

Application Modules

The major modules in program EZDESIT are, PATTOINT, MENU1, MENU3, MENU4, MENU5, ELSIZER, ELSITODIM, and MENU8. Each modules function and important submodules follow:

PATTOINT: Typical PATRAN neutral file model descriptions are on the order of 10000 to 30000 lines. Sizing requires random access to much of the information in a PATRAN neutral file. For purposes of faster data access, PATRAN.OUT is read into VAX FORTRAN internal file format. Subroutine PATTOINT accomplishes this task and

subsequent processors have total access to the PATRAN neutral file through a common block data structure.

MENU1: As implied by this subroutine name choosing option 1 in EZDESIT figure 5, executes routine MENU1. Interactive prompts are issued to aid the analyst in creating an EAL runstream based on information stored in the PATRAN neutral file.

MENU3: Choosing option 3 from the main menu shown in figure 5 executes subroutine MENU3. MENU3 is similar to MENU1 as it also creates EAL runstream logic from data stored in the PATRAN neutral file. The difference is that MENU3 converts PATRAN applied loadset information to EAL applied loadset syntax. MENU3 also lets the analyst set up inertial load cases, and combine loadcases to form a design loadcase for element sizing. Combining loadcases is useful when it is desired to size for one condition, say an aerodynamic maneuver, but in defining loads for this condition several sets of PATRAN element pressure, and nodal force sets must be utilized. MENU3 also writes processors into the EAL runstream which will generate stress resultant and nodal displacement files.

MENU4: Analysts may want to update EAL runstream element stiffnesses to reflect dimensions of a new dimension file, and/or temperatures of a different PATRAN element temperature set. Figure 5 option 4 performs this task by accessing subroutine MENU4. MENU4 prompts for the name of an existing EAL runstream, dimension file name, and element temperature set, then writes out an updated EAL runstream. This runstream may then be submitted to determine stress resultants based on the modified model. Element stiffness matrix updating is a necessary process, especially when working with thermal loadsets and as was shown in figure 2, is rigorously incorporated into the batch procedure.

MENU5: Figure 5, option 5 is used to translate EAL stress resultant and nodal displacement files to PATRAN results file format. EAL results obtained by submitting an EAL runstream without use of the EZBATCH procedures produce ASCII files of element stress resultants and nodal displacements using the DCU/PRINT process. To examine these results using PATRAN the files have to be converted to PATRAN results file format. Subroutine MENU5 (option 5) accomplishes this task with minimal prompted user input.

ELSIZER: Option 6 of figure 5 calls the interactive version element sizing subroutine, ELSIZER. ELSIZER prepares input parameters for various element sizing routines and then calls execution of appropriate element sizing for each element. First ELSIZER retrieves element applied stress resultants based on the current EAL run. Next it determines what type of element is being sized, bar, beam, honeycomb, or corrugated web, and sets up minimum dimensions and design criteria as specified by the PATRAN neutral file physical

property information. ELSIZER then gets an element temperature and appropriately interpolates material property information from the material property data file so temperature dependent allowables can be passed to the element sizing routine. Four element sizing routines are distributed with EZDESIT. I2FM is for isotropic bar elements, E24ISO is for isotropic material planar beams, HCBNDTH is for isotropic material honeycomb sandwich, and CORWEBTH is for isotropic material corrugated web elements. To show how users can write their own element sizing routine, including the use of composite materials, code for sizing a honeycomb element with laminate facesheet material is described in appendix E.

Element sizing proceeds in an analogous manner for the different types, figure 7 outlines the logic of routine HCBNDTH used for sizing honeycomb sandwich plate elements. Sizing begins with minimum gage defaults read from PATRAN physical property definition data. Failure mode is initially set to indicate minimum gage sizing. Next, element buckling is checked at the minimum gage size. If necessary, panel dimensions are increased to prevent buckling and the element failure mode indicator is updated to indicate a buckling failure mode. If a panel is subject to buckling type failure, it is then checked at ultimate load levels for Von-Mises yielding. This assumes buckling loads must be supported at ultimate conditions, and by definition implies yielding would result in a buckled element. All elements are then checked for Von-Mises yielding at limit load levels. If the element size has to increase to support loads without yielding, the element dimensions are updated and the failure mode indicator is updated accordingly. Finally strength is checked at ultimate load levels. Principal stresses are not allowed to exceed material ultimate allowables, element geometry and failure mode indicator are again updated if ultimate strength sizing requires the addition of material over the yield criteria. Once an element has been sized, element weights, dimensions, and failure mode indicator are written to output ASCII files. Plate element sizing routines include all of the criteria shown in figure 7. Bar elements are subject only to minimum gage, yield and ultimate strength checks. Beam elements are sized only by the minimum gage and yield criteria. Table I shows which failure modes apply to the various elements distributed with EZDESIT. Users should refer to appendix F and element sizing routine source code for more specific information on sizing algorithms.

ELSITODIM: Referring to figure 5, option 7 calls subroutine ELSITODIM. When performing element sizing interactively a dimension file will be produced for each loadset. ELSITODIM will prompt for the names of these files and create the worst case dimension file. The worst case dimension file can then be used to upgrade finite element model stiffness matrices as described in routine MENU4.

MENU8: Interactive element sizing also will produce element unit weight files for each loadset. These are combined into a worst case file by subroutine MENU8 (option 8 of figure 5). MENU8 also prints summaries of model weights by failure mode and loadset based on the worst case unit weight file. PATRAN named component weights will be printed if desired by the user.

EZBATCH contains subroutine calls to many of the same routines used in EZDESIT. Generally users will want to use the EZBATCH process once they have created and verified their model with EZDESIT and PATRAN.

WING DESIGN EXAMPLE

For purposes of illustration, a space shuttle type wing was sized for several thermal and mechanical loadsets using EZDESIT. Most major features of the code are utilized in this descriptive example as explained below.

Design Details

Starting geometry for the sample wing problem was obtained from rough drawings as shown in figure 8. PATRAN phase I modeling techniques were used to define line and surface definitions which are required for later breakup into finite elements. This line and surface geometry description, termed phase I geometry in PATRAN, is shown in figure 9. Figure 10 shows how the phase I model was discretized into a phase II (ie: PATRAN) finite element structural model.

Four load cases were considered for element sizing, they were; a 72.0 degree F. lift condition, a 72.0 degree F. landing condition, a hot lift condition without through thickness thermal gradients, and a hot lift condition with thermal gradients on some elements. Element pressures representing the lift condition are shown using PATRAN vector plots in figure 11. Vector plots indicate the direction of pressure loading. To graphically see assigned pressure magnitudes an element value assignment plot is used. Assignment plots break the total range of displayed values into a set of discrete ranges. Each range is assigned a color (or letter in the case of non-color output devices) and elements are then color coded by range values. Nodal force and constraint plots, figure 12, show the loading used to represent a landing type condition. The forces in figure 12 represent fuselage weight which is reacted through the wing root rib, to supports at node points where landing gear struts would exist. Shown in figure 12 are circles with constrained degrees of freedom indicated. These circles are a display of PATRAN constraints associated with the

landing case. To represent the inertial effect of wing weight on itself, an EAL acceleration loadcase was combined with the illustrated force loadset. Average element temperatures represent the hot condition, as shown in figure 13. A hot lower surface with slightly cooler internal elements and even cooler upper surface elements define the hot condition. To include effects of element through the thickness thermal gradients, surface elements in the wing torquebox were subject to a 100.0 degree F per inch thermal gradient for one loadcase. Figures 11 through 13 illustrate the use of PATRAN to verify thermal/mechanical loading and associated constraint requirements. Other features of the finite element model can also be verified using color graphic capabilities.

The example wing is similar to a space shuttle wing in terms of mold line definition, however it is not an equivalent structural concept. All surface elements are aluminum honeycomb unlike the shuttle wing which has uniaxially stiffened as well as honeycomb skin construction. The internal construction is similar to the space shuttle and employs both corrugated web and truss- type rib and spar construction techniques. Figure 14 shows how PATRAN is used to assign physical property definitions to finite elements. Elements are assigned a property definition by property set number. Data associated with property set 1 (honeycomb) is listed in table II, similar type data is used to define physical properties for the bars (property set 2) and corrugated webs (property set 3). Plots like those in figure 14 are useful model verification tools and show the importance of interactive graphic model preprocessing. An additional entity that can be viewed and verified with PATRAN is nodal weight assignments. Figure 15 shows where subsystem weights are located throughout the structural model.

Study Results

Upon completion of model verification, the EZBATCH process was used to size elements for the sample wing design problem. Post sizing examination of results was done using a combination of PATRAN graphics and EZDESIT tabulations.

Figure 16 shows EAL calculated element stress resultants from the 72.0 degree F. lift condition analysis. Nx loads represent panel running loads in pounds per inch which produce stresses in the EAL element x direction. Ny and Nxy represent y and shear direction in-plane running loads. The running moments Mx, My, and Mxy can also be plotted in PATRAN. Color graphics would permit better visualization than can be realized in black and white, none-the-less load resultant plots are of great value in understanding structural behavior with regard to load path definition. Based on EAL generated stress resultants and application of the sizing routines, calculated element unit weights are shown in figure 17. If actual element geometry information is required for other analysis codes it

can be found in the element dimension file. Portions of an element dimension file are listed in table III. PATRAN plots are also useful for examining element failure modes, figure 18, and controlling (worst case) loadset assignments, figure 19. Figure 18 shows how each of the five failure modes currently considered by EZDESIT design elements in the wing. If users develop logic to size elements based on other failure criteria, that methodology could be incorporated into an element sizing module, and another failure mode indicator defined. For hot structure design it may be useful to size for a creep or fatigue stress limit, or for aerodynamic panels a panel flutter criterion could be added. Figure 19 gives the designer an indication of which loadsets are most influential in his design.

Interpretations of structural performance are quantitatively defined using the interactive mode of EZDESIT. Interactive analysis of calculated element weights will produce results in the form of table IV. Tabular summaries along with graphical post processing, help designers quantify the structural performance of a vehicle design. Load distributions consisting of concentrated areas of intense loading, or large amounts of element bending forces may justify implementing a different structural arrangement or new type of element construction. Choices between high specific strength and high specific stiffness materials can be guided by examining the percentage of weight determined by strength criteria as opposed to stiffness criteria. Particularly constraining loadcases will stand out by driving up structural weight. Structural designers can then discuss vehicle weights with designers from other disciplines to decide if such weights are acceptable, or if possible mission redefinition is warranted to alleviate structural problems.

CONCLUDING REMARKS

A computer program for calculating weights of structural systems based on fundamental failure mode analysis has been presented. The program exists in two versions, EZDESIT runs interactively and EZBATCH which runs in batch mode. Use of these programs define a rigid procedure for creating structural finite element models, executing the finite element analysis for determination of element stress resultants, and sizing of elements based on stress resultants and temperature dependent material properties. Calculated stress resultants may be based on applied thermal as well as mechanical loadsets. Results are summarized qualitatively with color graphic post processing, and quantitatively via component weight tabulations. Iterative capability is provided to maintain the consistency of finite element calculated stress resultants with dimension and temperature dependent element stiffness matrices.

Data input requirements, program operation, and program limitations are explained in Appendix A. A standard file naming convention is given in Appendix B. A complete program execution which illustrates the sizing of a space shuttle type wing is contained in Appendix C. A technique for using calculated structural weights in a vehicle performance study is explained in Appendix D. Appendix E describes how users may write additional element sizing routines, and Appendix F documents contents of the program distribution tape.

The nature of conceptual and preliminary structural design defies any single computer program to have all the flexibility and capability a designer requires. Development of new material systems and structural concepts will continually change assumptions that must be made to perform structural analysis during the vehicle definition phase. Understanding that restriction, the program presented is written in a modular format which encourages users to develop sizing algorithms for structural concepts and wall constructions of specific interest. EZDESIT and EZBATCH are not meant to represent a complete structural design system, but rather a framework which can be built upon to suit particular system study requirements.

Application of calculated structural weights to vehicle performance studies requires inclusion of certain non-optimum factors which account for the weight impact of non-modeled items such as paint, joint build ups, and access facilities. Individual organizations will have to develop a technique for including non-optimum factors based on experience with past designs, and standardized application of the analysis technique. The methods presented represent a system of standard conceptual level structural design techniques, yet provide input options which make it flexible enough to analyze a range of structural systems.

References

1. Hunt, J. L., and Martin, J. G.: Hypersonic Airbreathing Vehicle Conceptual Design (Focus On Aero-Space Plane), Second NASA/Air Force Symposium on Recent Experiences in Multidisciplinary Analysis and Optimization. Sept 28-30, 1988, Hampton. Va.

2. Nicolai L. M: Fundamentals of Aircraft Design. copyright 1975, Distributed by METS, Inc. 6520 Kingsland Court, San Jose California 95120.

3. Bush, L. B.; Lentz, C. A.; Rehder, J. J.; Naftel, J. C.; and Cerro, J. A.: Structural Weights Analysis of Advanced Aerospace Vehicles Using Finite Element Analysis. AIAA-89-2130, August 1989.

4. PDA Engineering: PATRAN Plus User Manual. copyright 1987, PDA Engineering Costa Mesa, California.

5. Whetstone, W. D.: User Instructions EISI-EAL version 312.08. Interim Release, copyright 1985 by Engineering Information Systems Inc. San Jose, California.

6. Digital Equipment Corporation: MicroVMS User's Manual. copyright 1985, Digital Equipment Corporation, Nashua, New Hampshire.

7. Bowman L. M: STIMAT - A Computer Program for Generating Plate Stiffness Matrices of Anisotropic Skin Constructions with Ringframes and Stringers. NASA TM - 100639, (to be published)

8. Borland International: Reflex the Analyst - Users Guide. copyright 1985, Borland/Analytica Inc. Scotts Valley, California.

9. Boeing Company Proprietary: Engineering users manual for the algorithmic mass-factoring method program.

10. Roark R. J. and Young W. C.: Formulas for Stress and Strain. 5th Edition Copyright 1975 by McGraw-Hill Inc. New York, New York.

11. Muha T. J.: User's Manual for the Laminate Point Stress Analysis Computer Program SQ5 as Revised by AFFDL/FBC. Air Force Flight Dynamics Laboratories Technical Memorandum FBC-74-107, July 1974 Wright Patterson Air Force Base, Ohio.

12. Tsai S. W. and Hahn H. T.: Introduction to Composite Materials. copyright 1980 by Technomic Publishing Co. Inc. 265 Post Road West, Westport, Connecticut 06880.

13. Tsai S. W.: Composites Design - 1985. copyright 1985, Think Composites, Box 581 Dayton, Ohio 45419.

TABLE I. - ELEMENT SIZING CRITERIA

Element Type	Failure Mode: Minimum Gage	Global Buckling	Ultimate load with Yield Allowables	Limit load with Yield Allowables	Ultimate Load with Ultimate Allowables
BAR	yes	no	no	yes	yes
BEAM	yes	no	no	yes	no
HONEYCOMB	yes	yes	yes*	yes	yes
Corrugated Web	yes	yes	yes*	yes	yes

* only for elements which otherwise would be sensitive to global buckling

TABLE II - SAMPLE PHYSICAL PROPERTY INPUT

FOR AN ISOTROPIC HONEYCOMB ELEMENT

field	generic name	value	description
1	matnum	6	material number, refer to file matprop.prn (ex. 6 = 2024 aluminum)
2	span	30.	Panel Buckling Span, (in).
3	itype	9	= isotropic honeycomb
4	factor	1.0	factor times input loads to obtain ultimate load
5	exptoth	0.8	panel buckling load knockdown factor (typically 0.8, empirical/theoretical ratio)
6	nof	1.5	Non-optimum factor (1.0 = none)
7	minga	0.01 0	minimum gage of a facesheet, (in.)
8	ymred	1.0	youngs modulus reduction, (1.0 = full value in file matprop.prn)
9	alsred	1.0	allowable strength reduction, (1.0 = full value in file marprop.prn)
10	dummy	0	
11	cormin	.375	minimum core height, (in.)
12	cormax	5.0	maximum core height, (in)
13	rhocor	0.1	core material density, (lb/in**3) (ex: aluminum = 0.1)
14	dummy	.0	
15	dummy	.0	
16	dummy	.0	
17	dummy	.0	
18	dummy	.0	
19	dummy	.0	
20	dummy	.0	
21	dummy	.0	
22	dummy	.0	
23	dummy	.0	
24	dummy	.0	
25	dummy	.0	
26	yiefac	0.67	factor times ultimate load to obtain limit load, (ex: =.67 if using a 1.5 safety factor)
27	dummy	.0	
28	dummy	.0	
29	dummy	.0	
30	dummy	.0	
31	dummy	.0	
32	dummy	.0	
33	dummy	.0	
34	dummy	.0	
35	dummy	.0	

TABLE III. - DIMENSION FILE

Element No. 361 I2fm, area= 0.100Unit wt. is 0.18
Element No. 362 I2fm, area= 0.100Unit wt. is 0.18
Element No. 363 I2fm, area= 0.100Unit wt. is 0.18
Element No. 364 I2fm, area= 0.100Unit wt. is 0.18
Element No. 365 Honeycomb, face sheet gage is 0.010 Height is 0.250 Unit wt. is 0.54
Element No. 366 Honeycomb, face sheet gage is 0.010 Height is 0.250 Unit wt. is 0.54
Element No. 367 corrugated web 0.015Unit wt.is 0.41
Element No. 368 Honeycomb, face sheet gage is 0.010 Height is 0.580 Unit wt. is 0.68
Element No. 369 Honeycomb, face sheet gage is 0.010 Height is 0.635 Unit wt. is 0.71
Element No. 370 Honeycomb, face sheet gage is 0.010 Height is 0.635 Unit wt. is 0.71
Element No. 371 Honeycomb, face sheet gage is 0.011 Height is 0.360 Unit wt. is 0.64

TABLE IV. - WEIGHT TABULATIONS

Minimum Gage	454.
Panel Buckling	490.
Compresive Yield	2818.
Yield	80.
Ultimate	288.

Total weight of E23 elements = 241. Lb
Average unit weight of E23 elements = 0.233 Lb/ft

Total weight of E24 elements = 0.0 Lb
Average unit weight of E24 elements = 0.0 Lb/ft

Total weight of plate elements = 3888.9 Lb
Average unit weight of plate elements = 1.39 p.s.f.

How many load cases were summarized?
5
Weight summarized by load case:

Loadcase Number 1454.
Loadcase Number 2307.
Loadcase Number 3 1893.
Loadcase Number 4237.
Loadcase Number 5 1239.

Do you wish to list weights for named components? (T/F)
T
Input component name
MODEL
Weight for component MODEL is 4129.9 Lbs
Unit weight for bar elements is 0.23 Lb/ft
for a bar length of 1032.8 ft.
Unit weight for plate elements is 1.39 Lb/sq ft
for a plate area of 2794.53 sq. ft.
another component? (T/F)
T
Input component name
TORQUEBOX
Weight for component TORQUEBOX is1722.3 Lbs
Unit weight for bar elements is 0.21 Lb/ft
for a bar length of 654.8 ft.
Unit weight for plate elements is 1.89 Lb/sq ft
for a plate area of 835.6 sq. ft.
another component? (T/F)
F

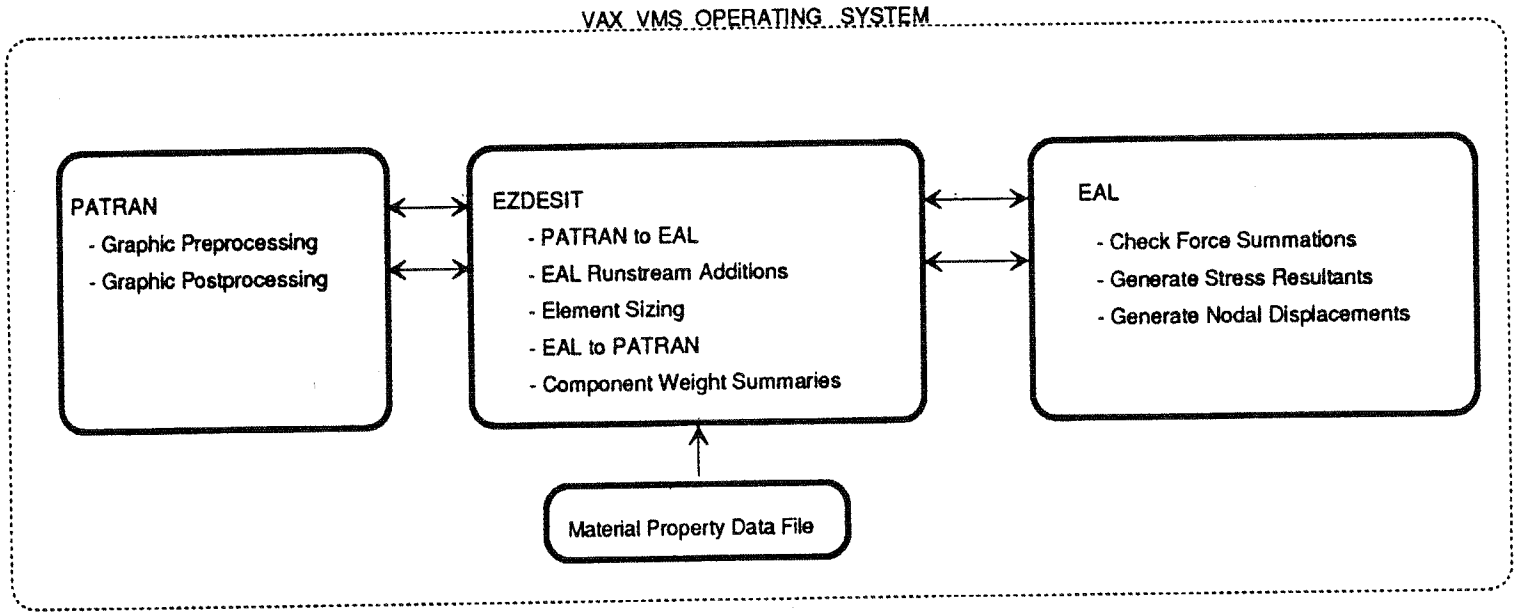


Figure 1. - EZDESIT Interactive, flow chart

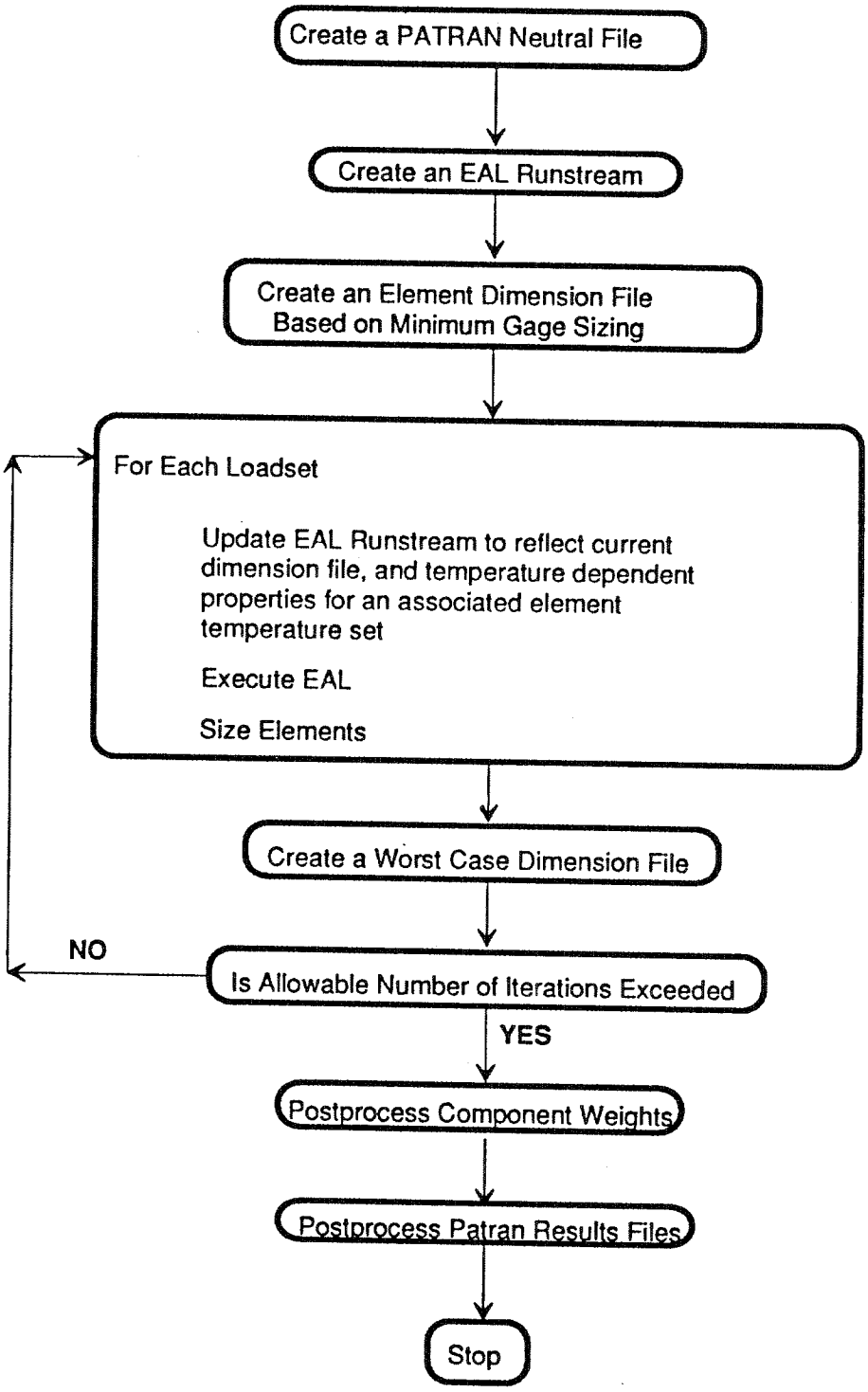


Figure 2. - EZBATCH, flow chart

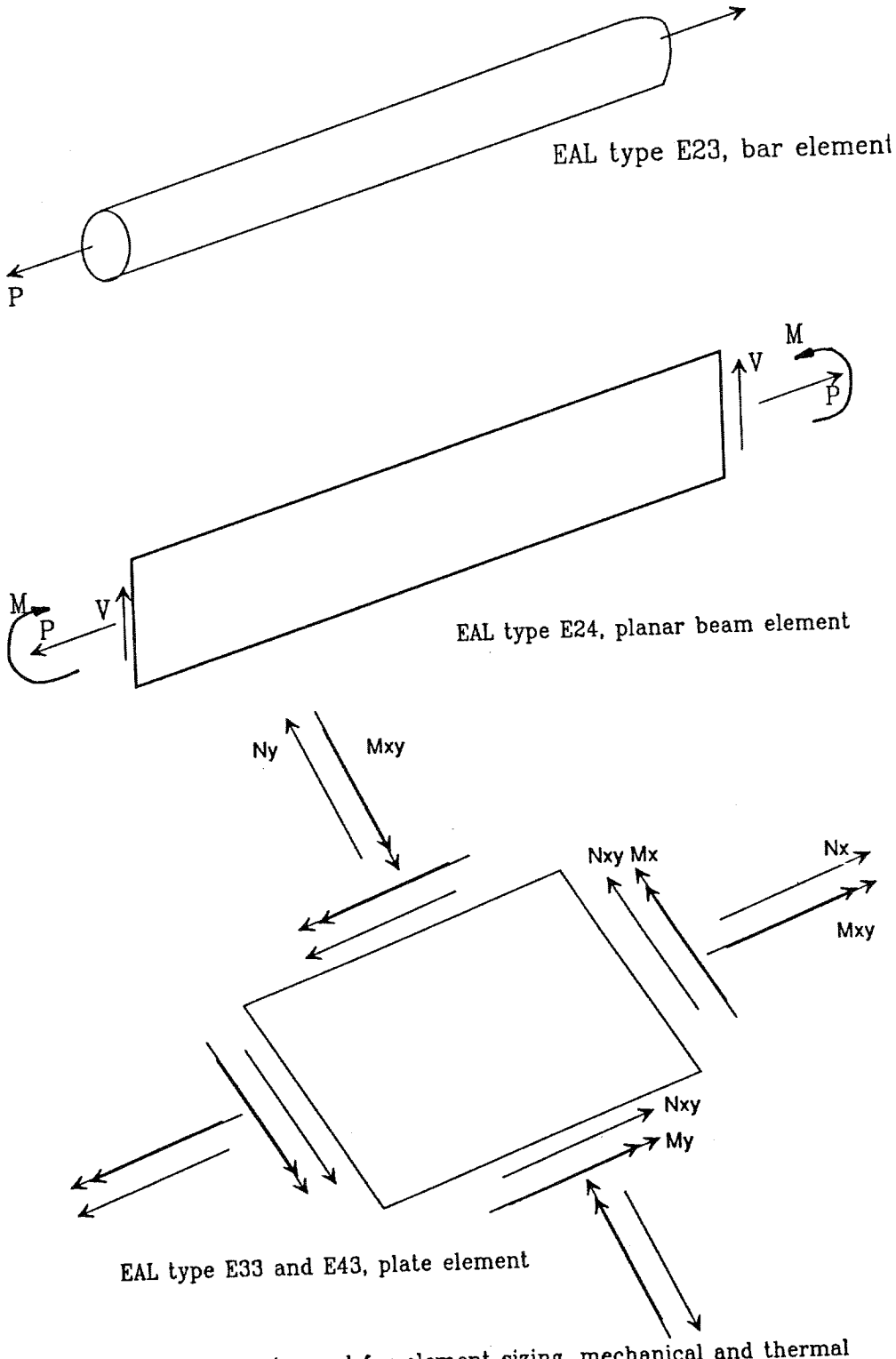


Figure 3.– Stress resultants used for element sizing, mechanical and thermal

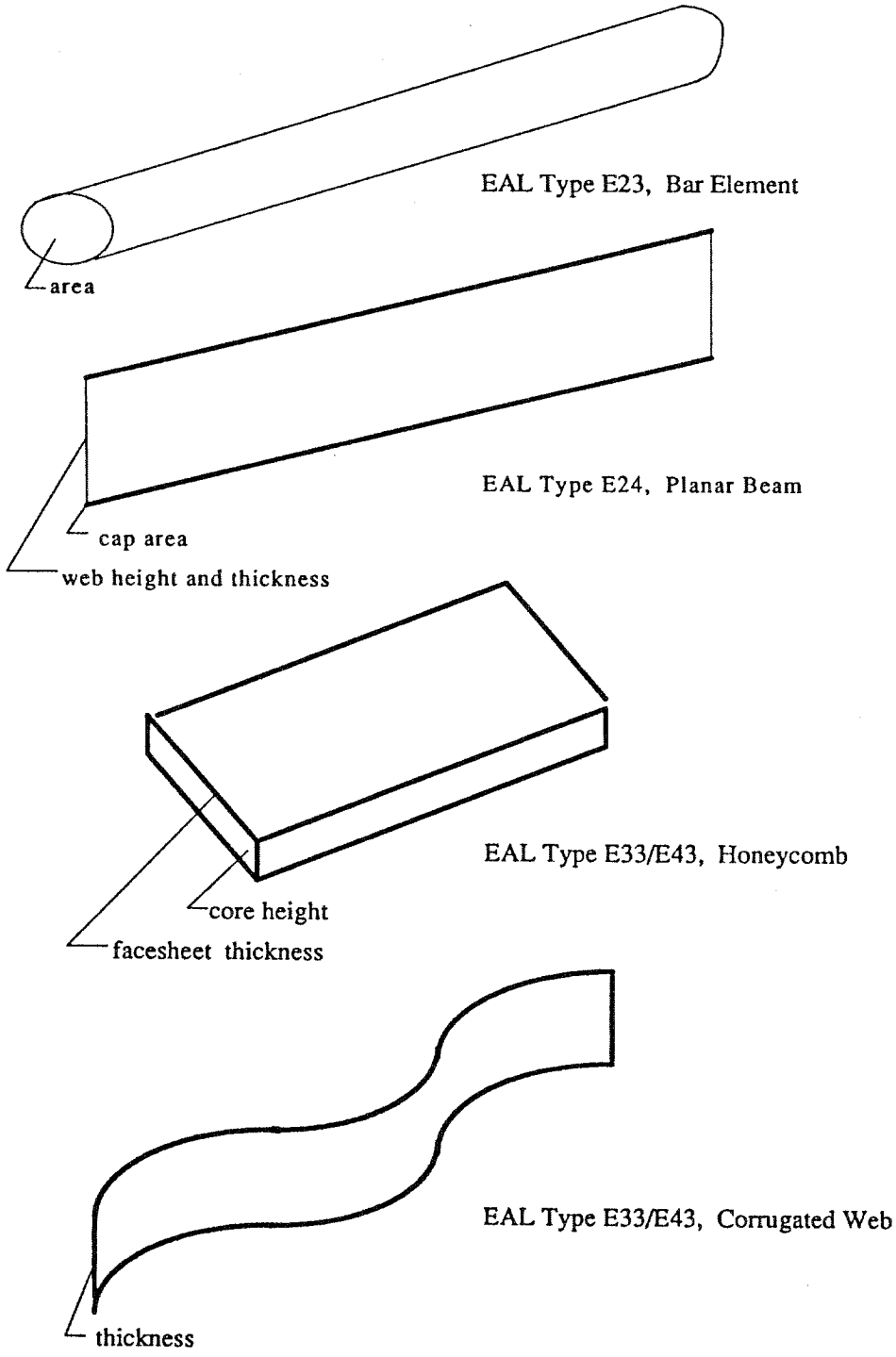


Figure 4. - Element dimensions, sizing code determined

Program EZDESIT

Option

1 - Create an EAL runstream through XQT DRSI

2 - Modify an existing EAL runstream

3 - Create a partial EAL runstream,
for applying/combining loads and constraints

4 - Update EAL element stiffness matrices

5 - Translate EAL results to Patran Results files

6 - Run element sizing code

7 - Create a worst case dimension file

8 - Review calculated component weights

9 - Exit to system

Figure 5. - EZDESIT Program, Main Menu

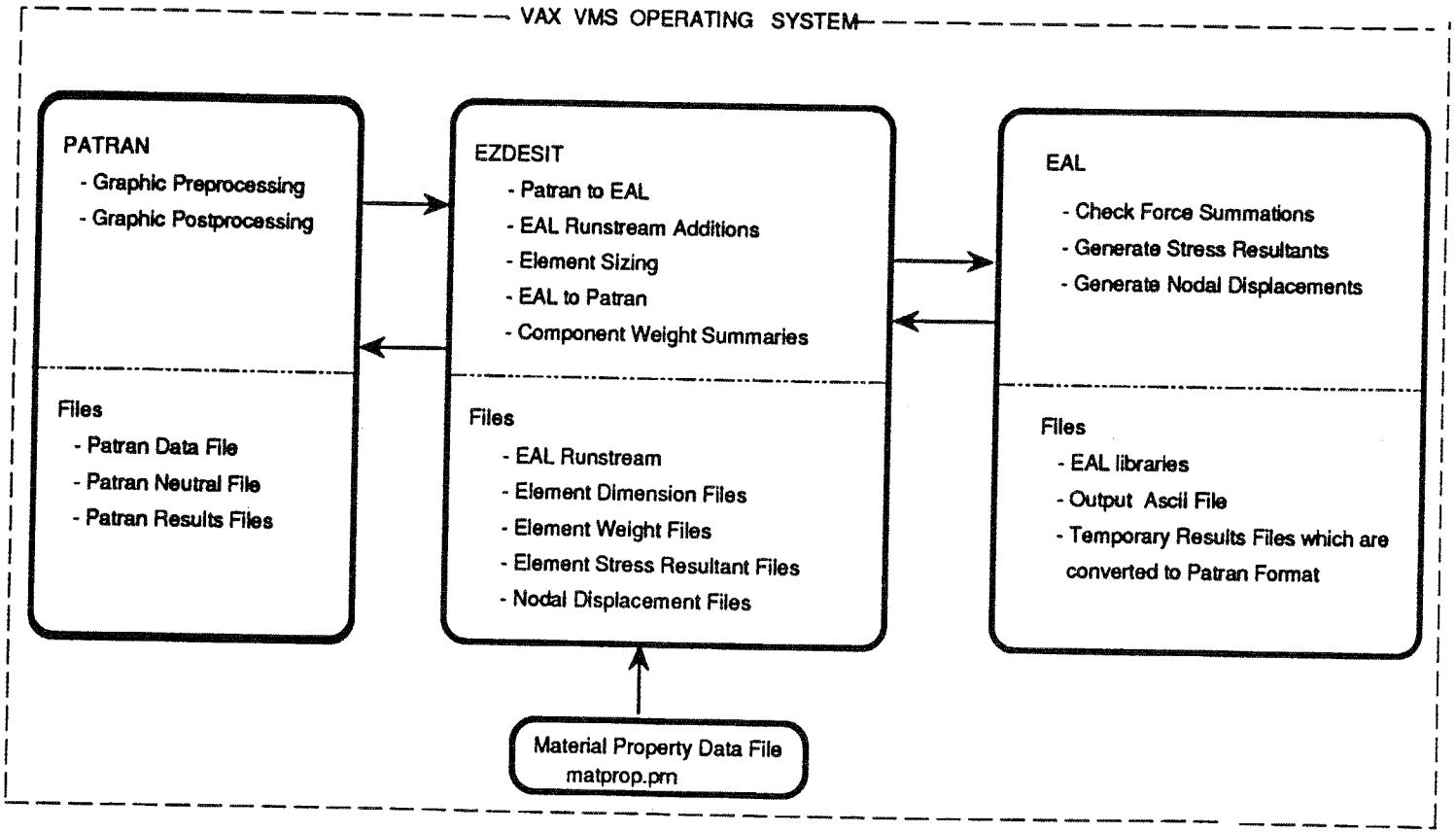


Figure 6. - EZDESIT Interactive, flow chart

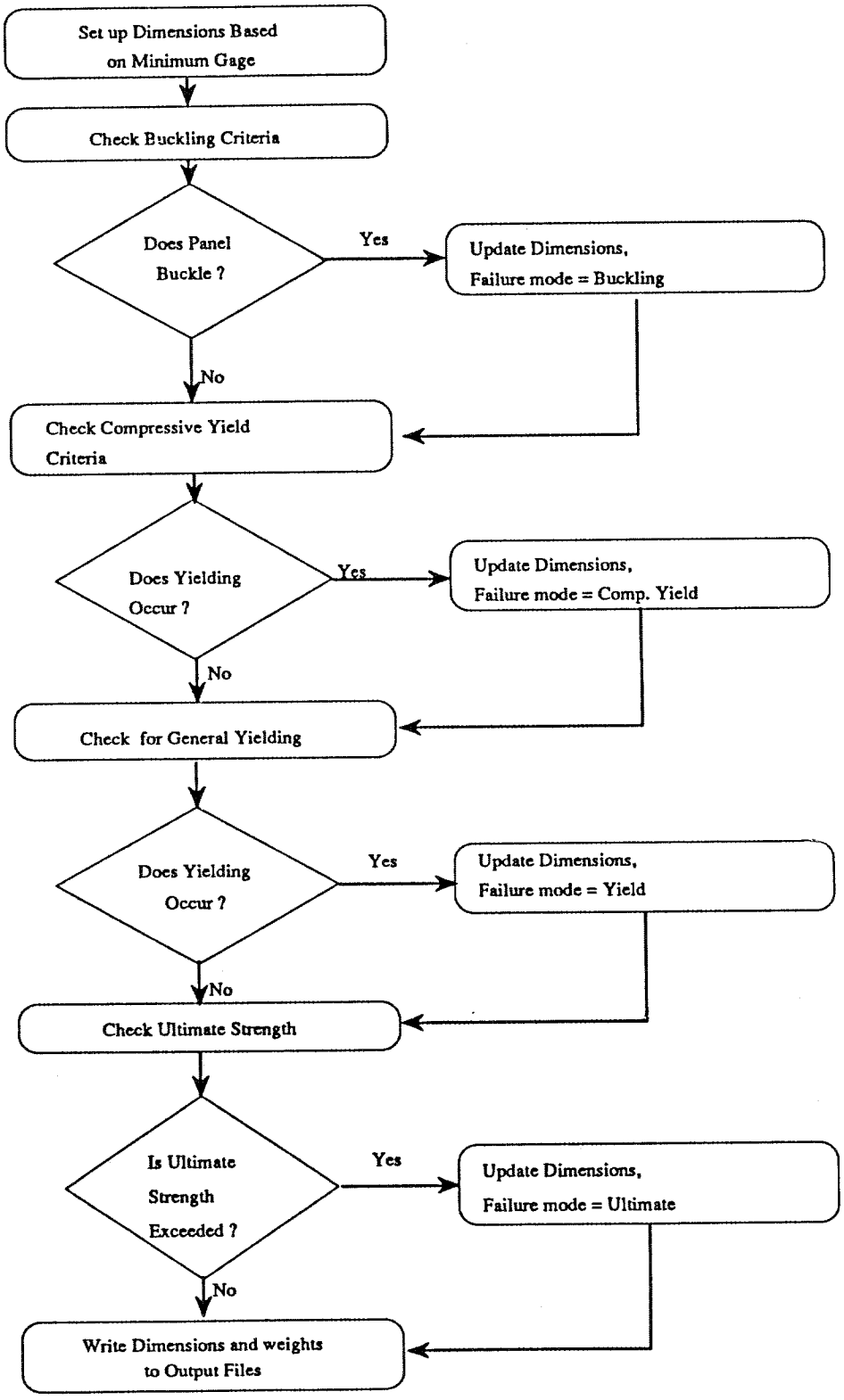


Figure 7. - Element Sizing Logic for an Isotropic Honeycomb Plate Element

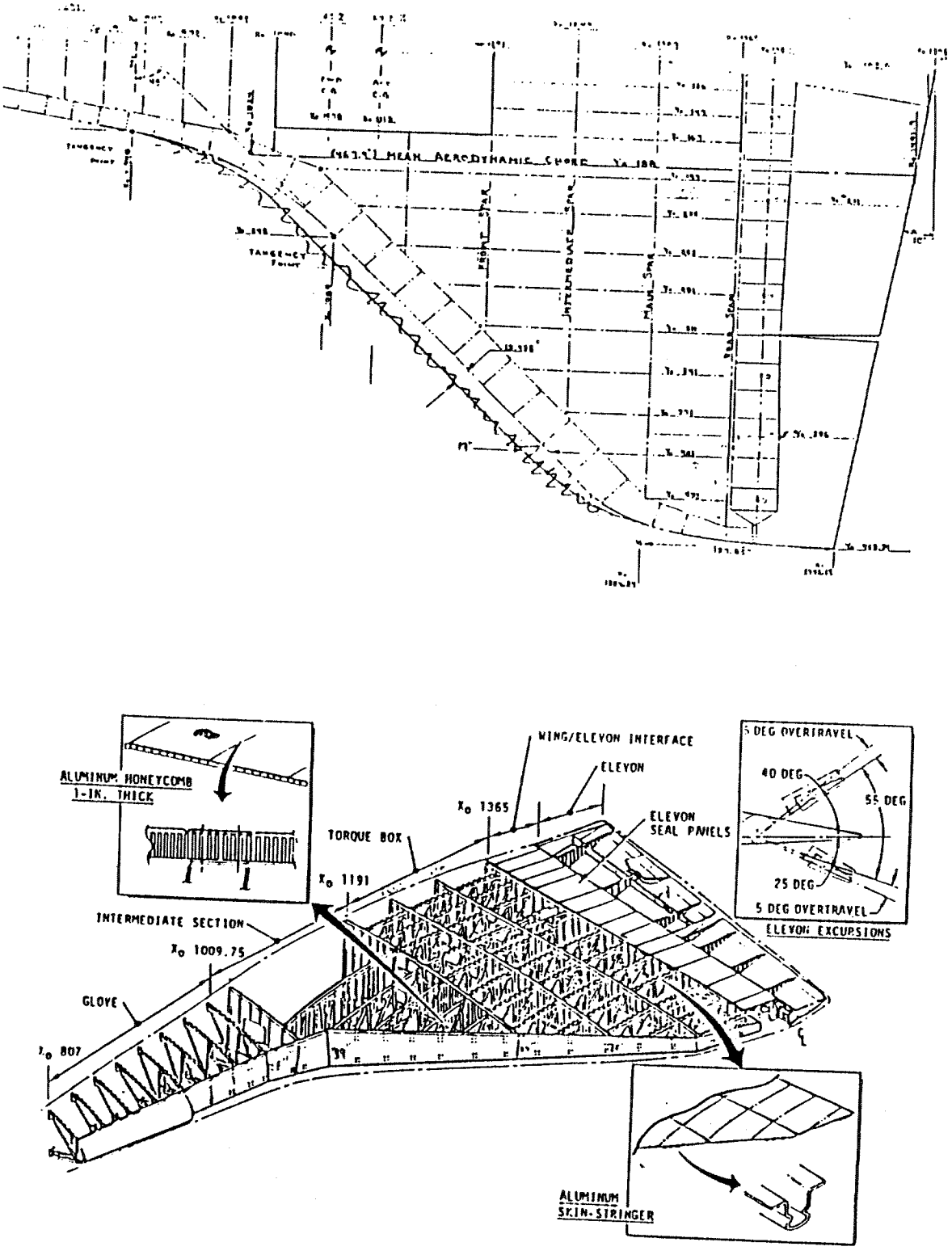
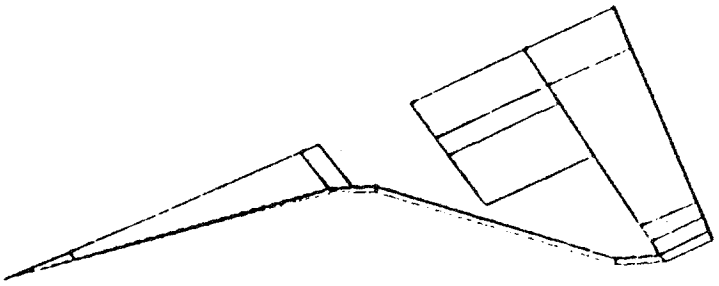
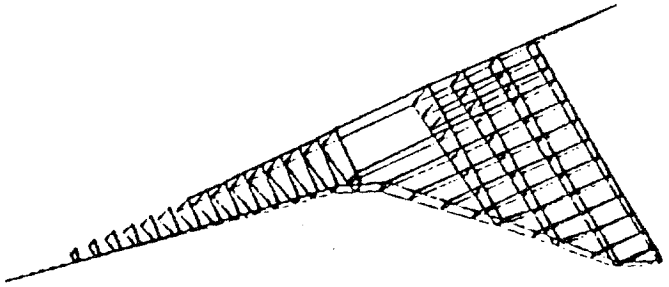


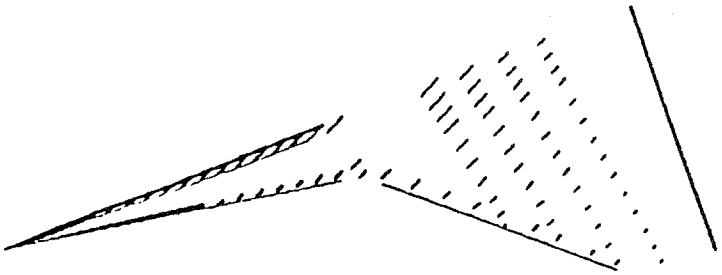
Figure 8 - Example Problem, Structural Layout



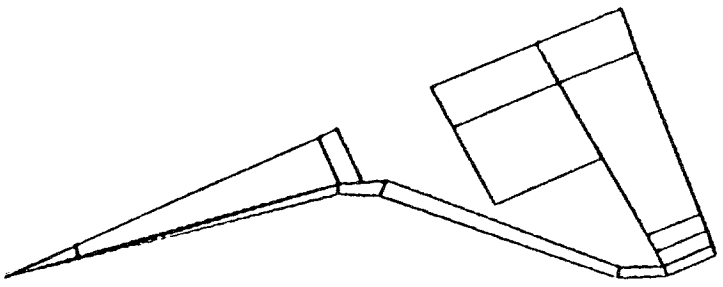
Upper Surface Element Construction Patches



Internal Element Construction Patches

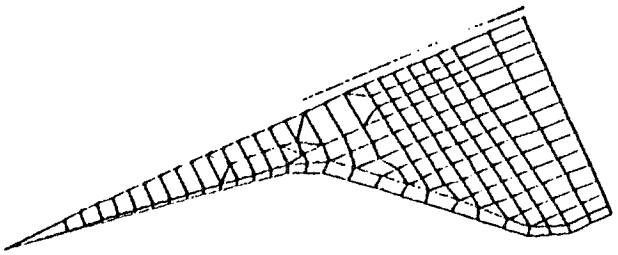


Internal Element Construction Lines

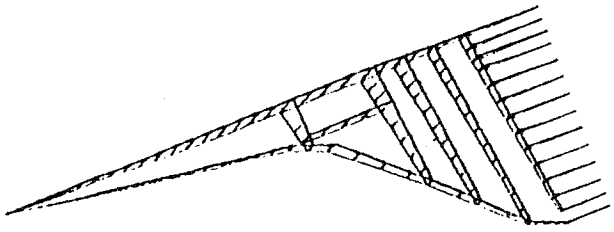


Lower Surface Element Construction Patches

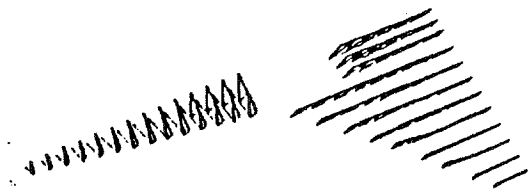
Figure 9 - Patran Phase I Geometry, Construction Lines and Patches



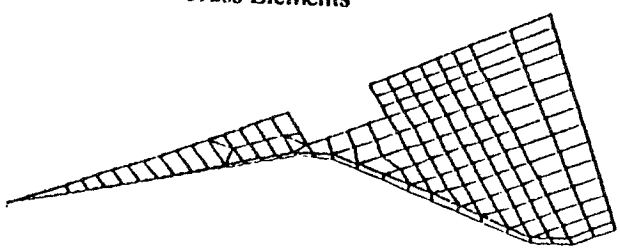
Upper Surface Elements



Internal Spar and Rib Elements

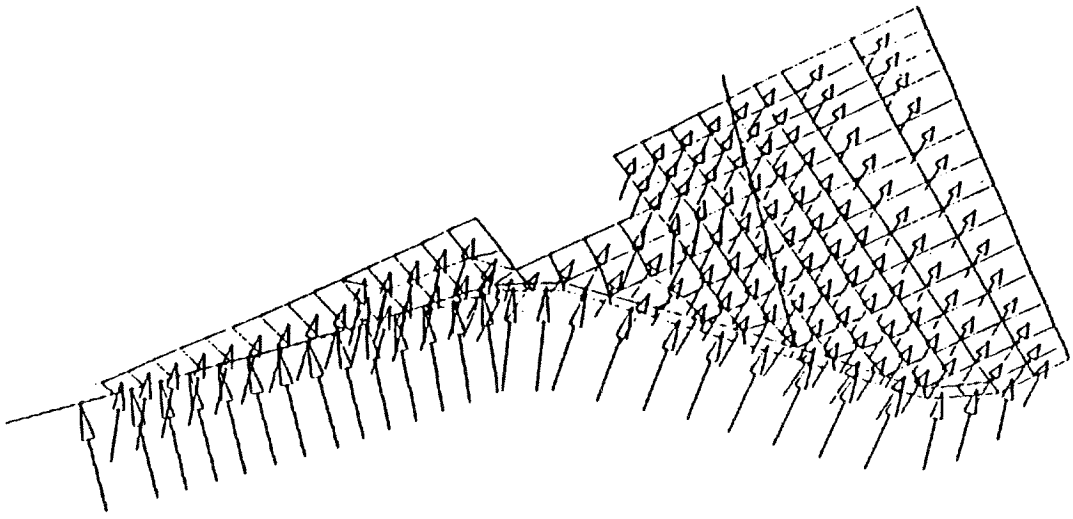


Truss Elements

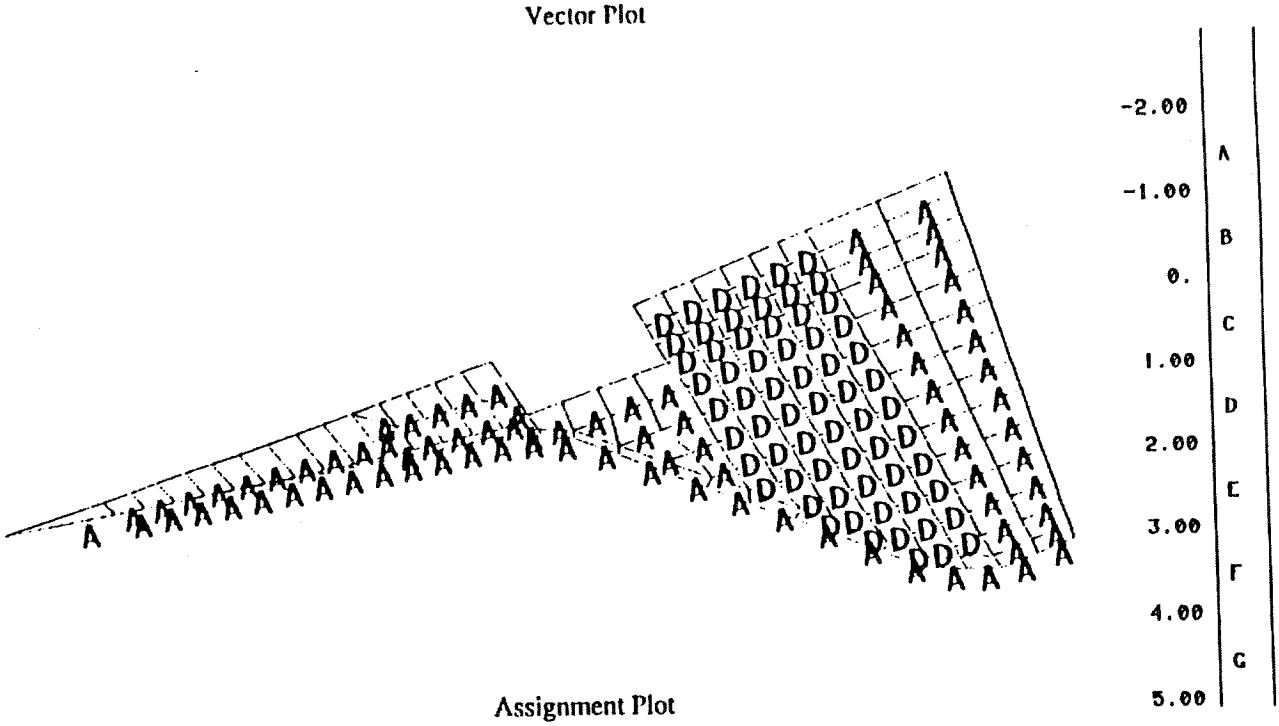


Lower Surface Elements

Figure 10. - Patran Phase II Finite Element Model



Vector Plot



Assignment Plot

Figure 11 - Lift Loads

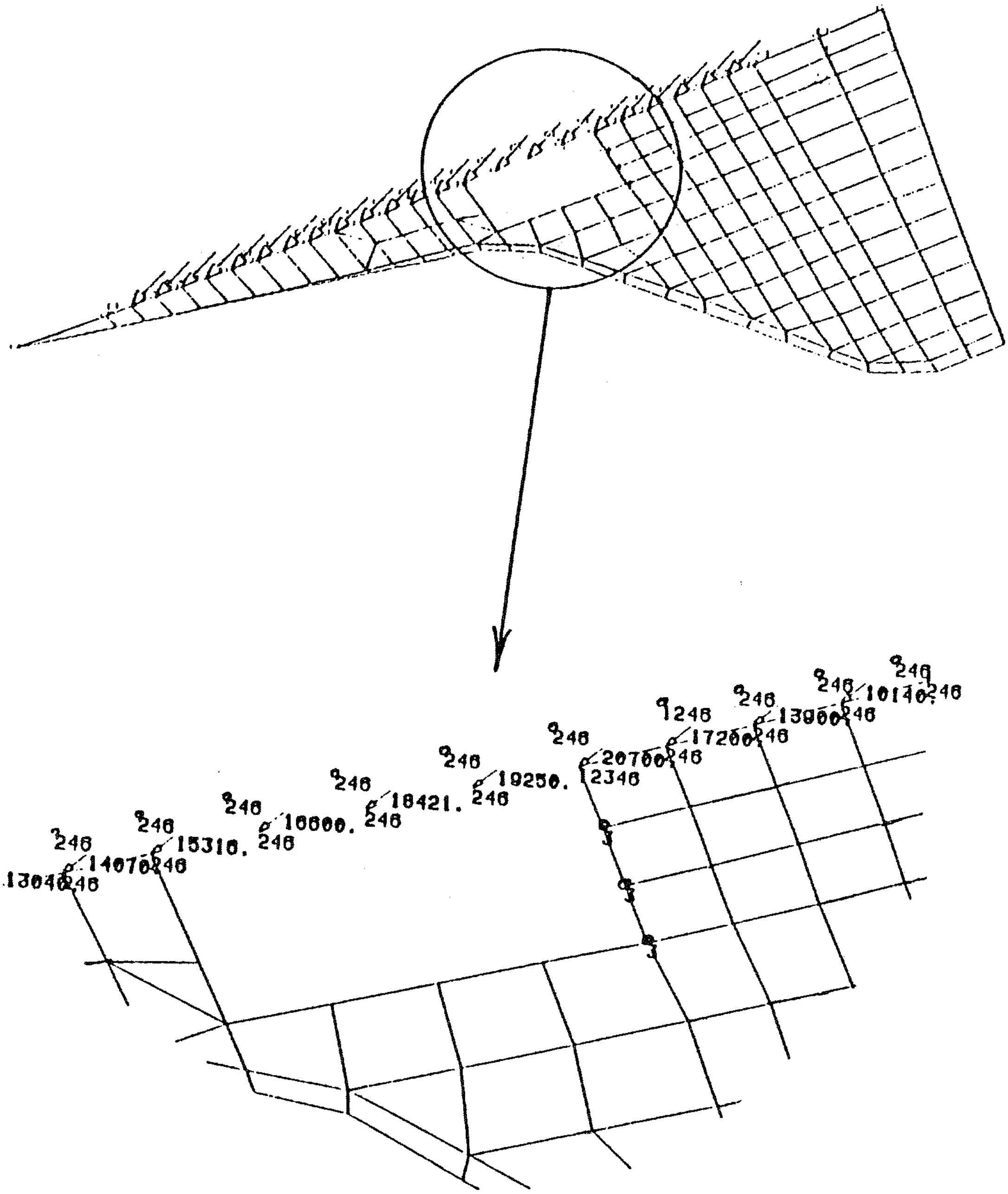


Figure 12. - Body Induced Landing Loads

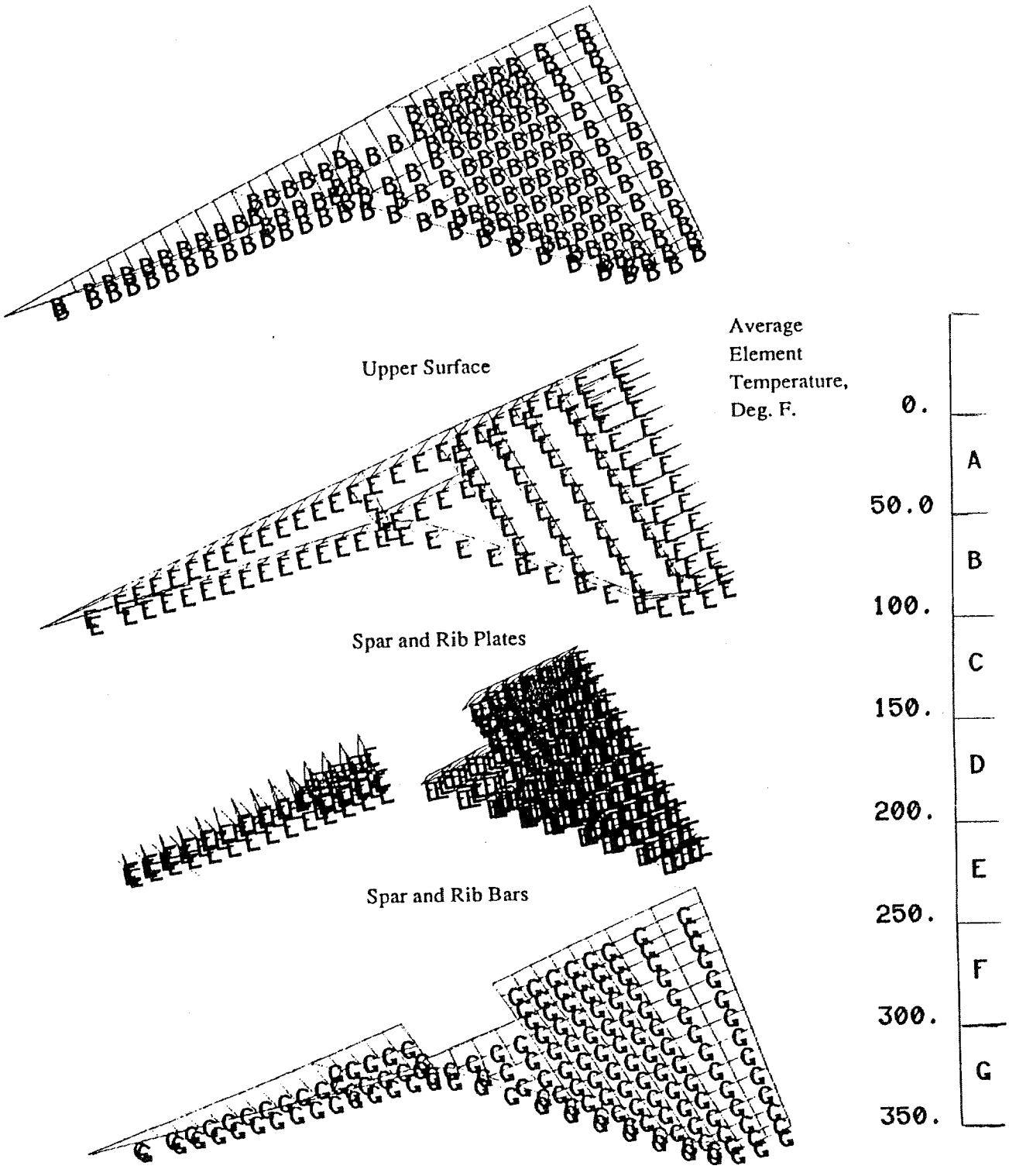


Figure 13. - Hot Condition Element Temperatures

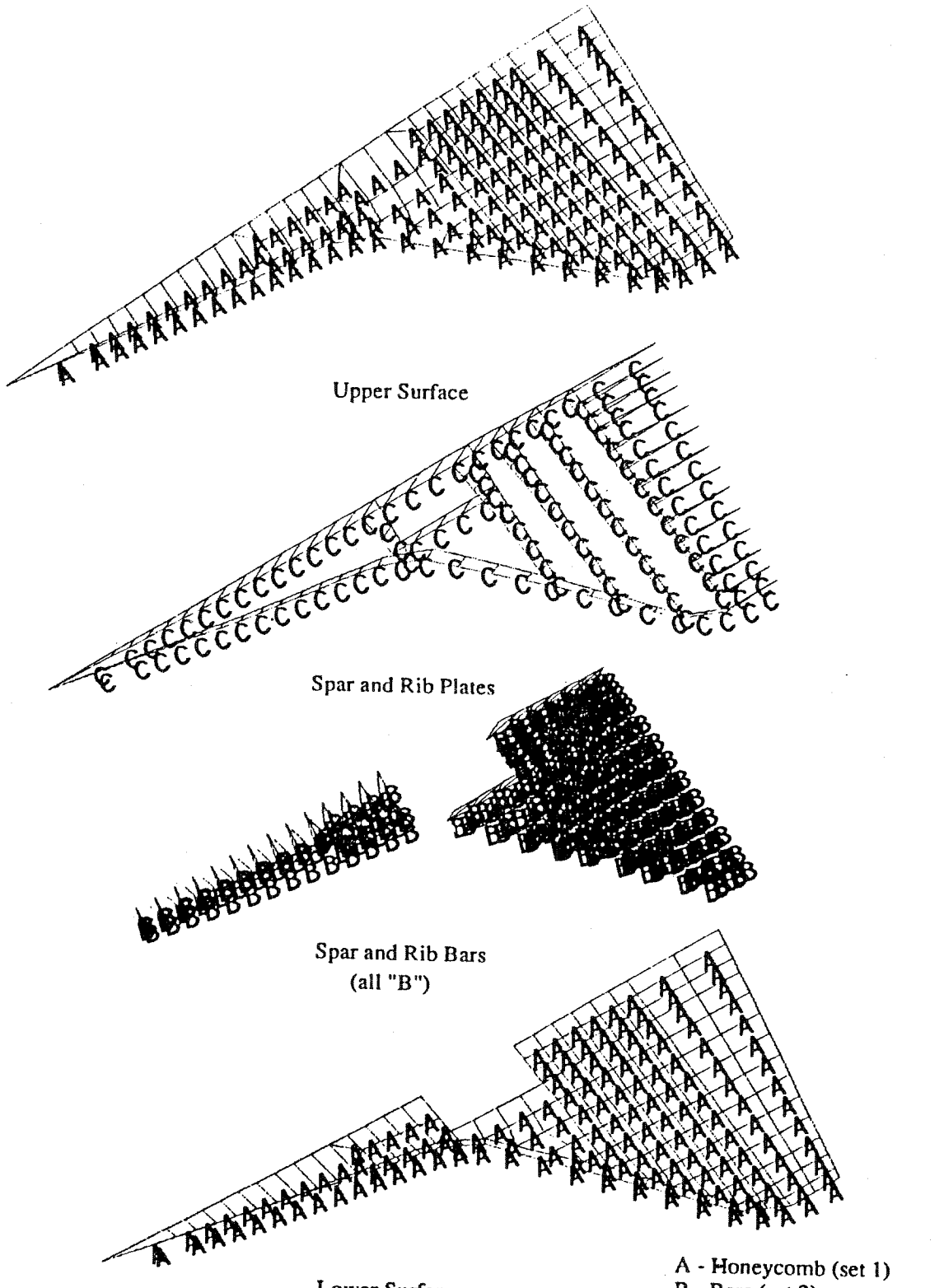


Figure 14. - Property Set Assignment

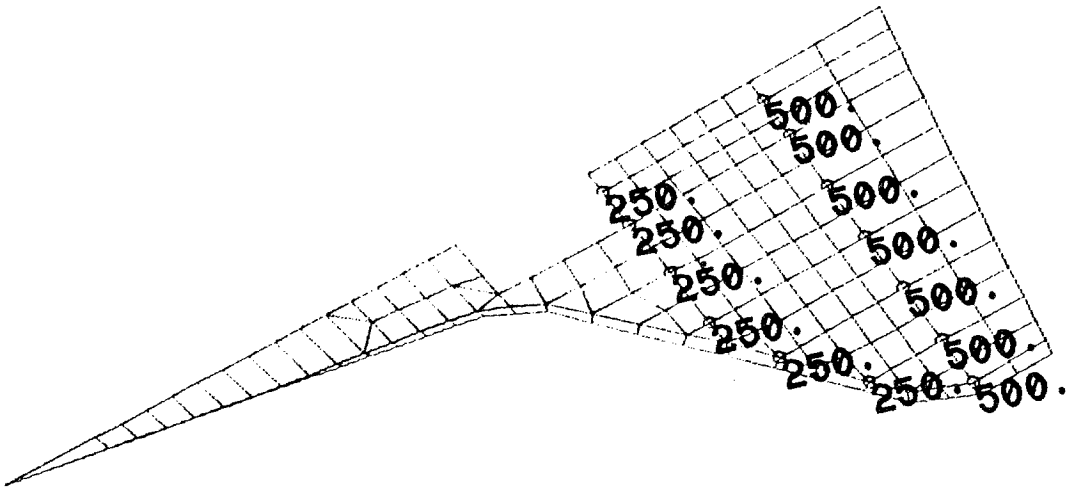


Figure 15. - Subsystem Nodal Weight Assignment

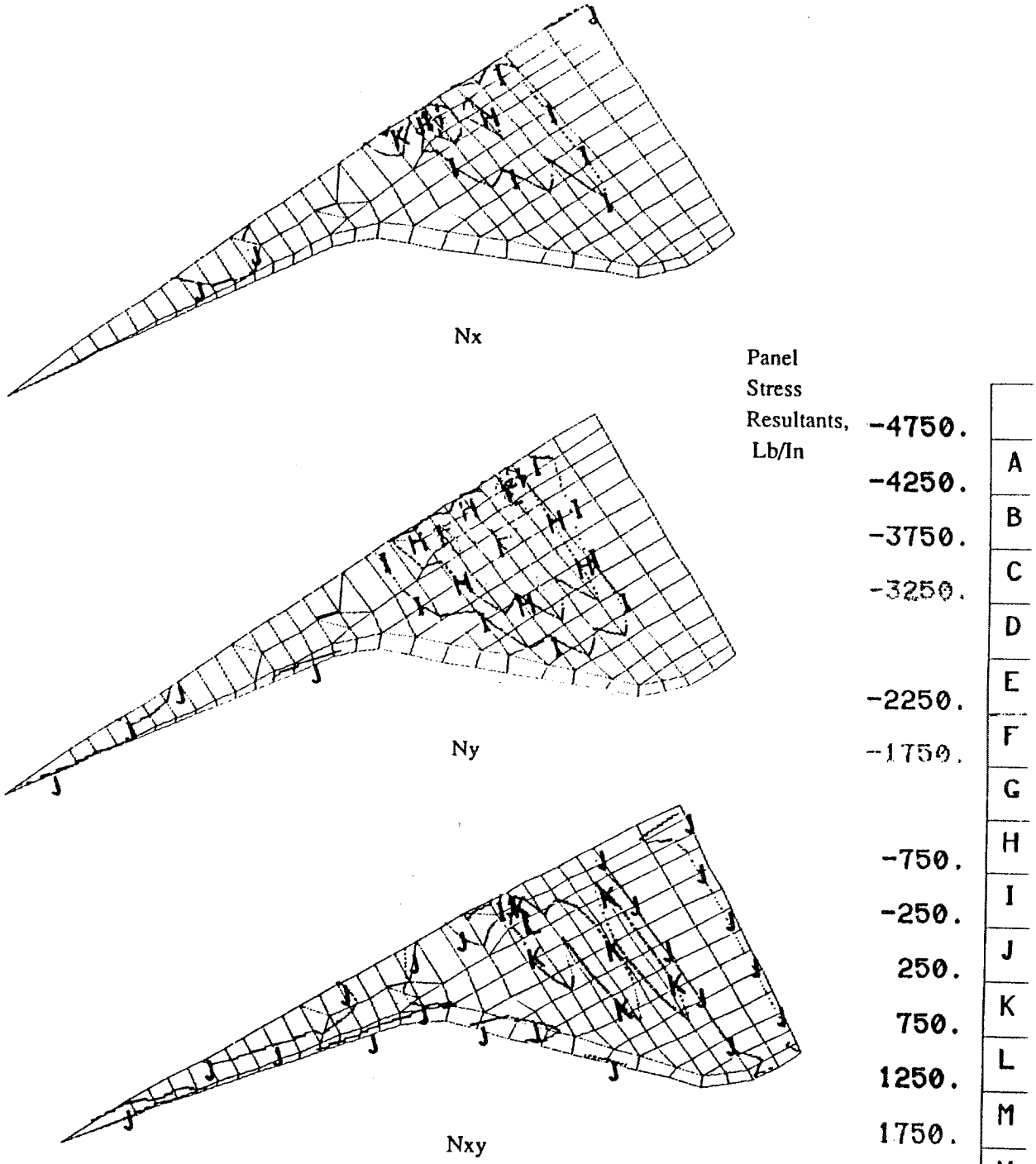


Figure 16. - Upper Surface Stress Resultants, Lift Case

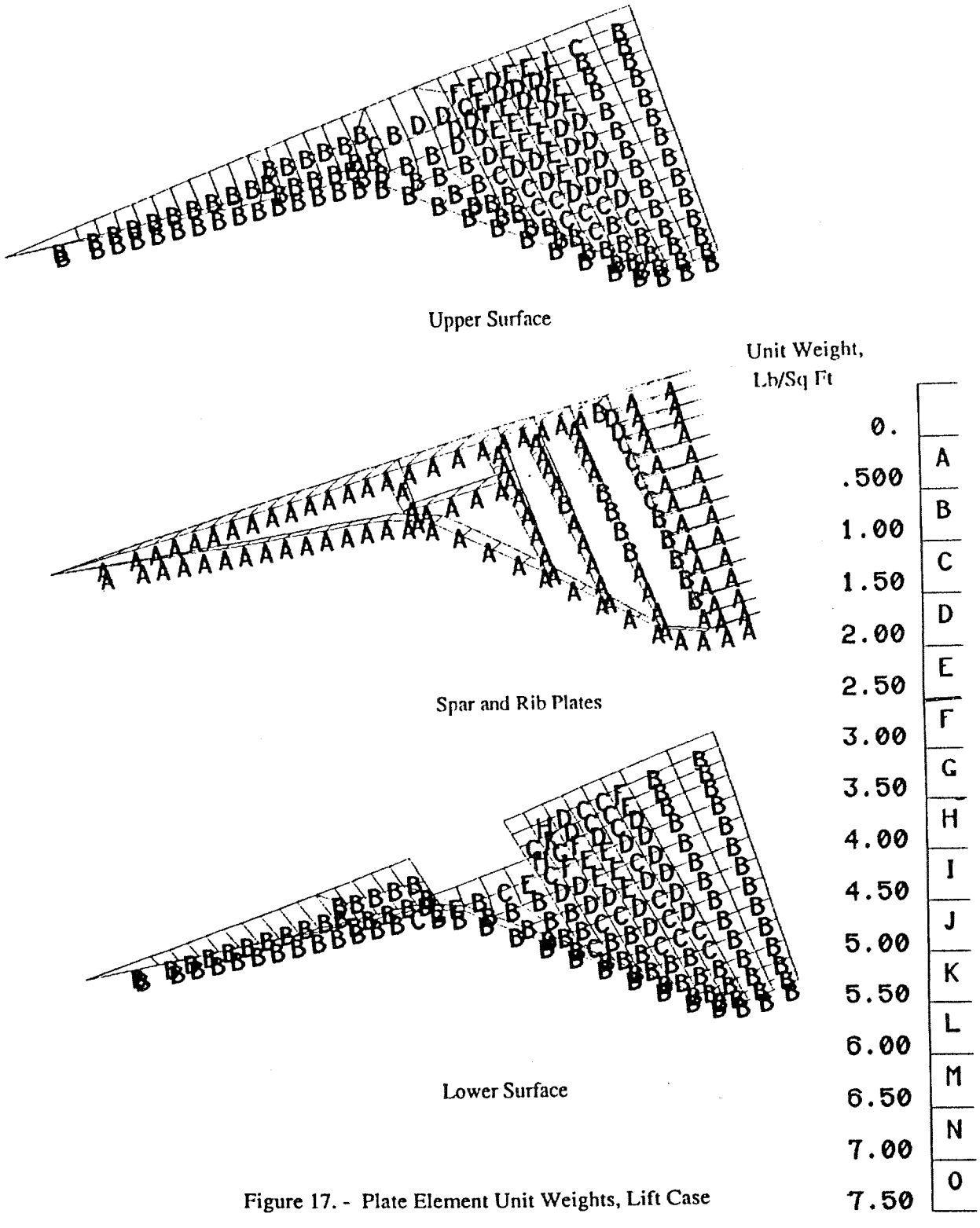


Figure 17. - Plate Element Unit Weights, Lift Case

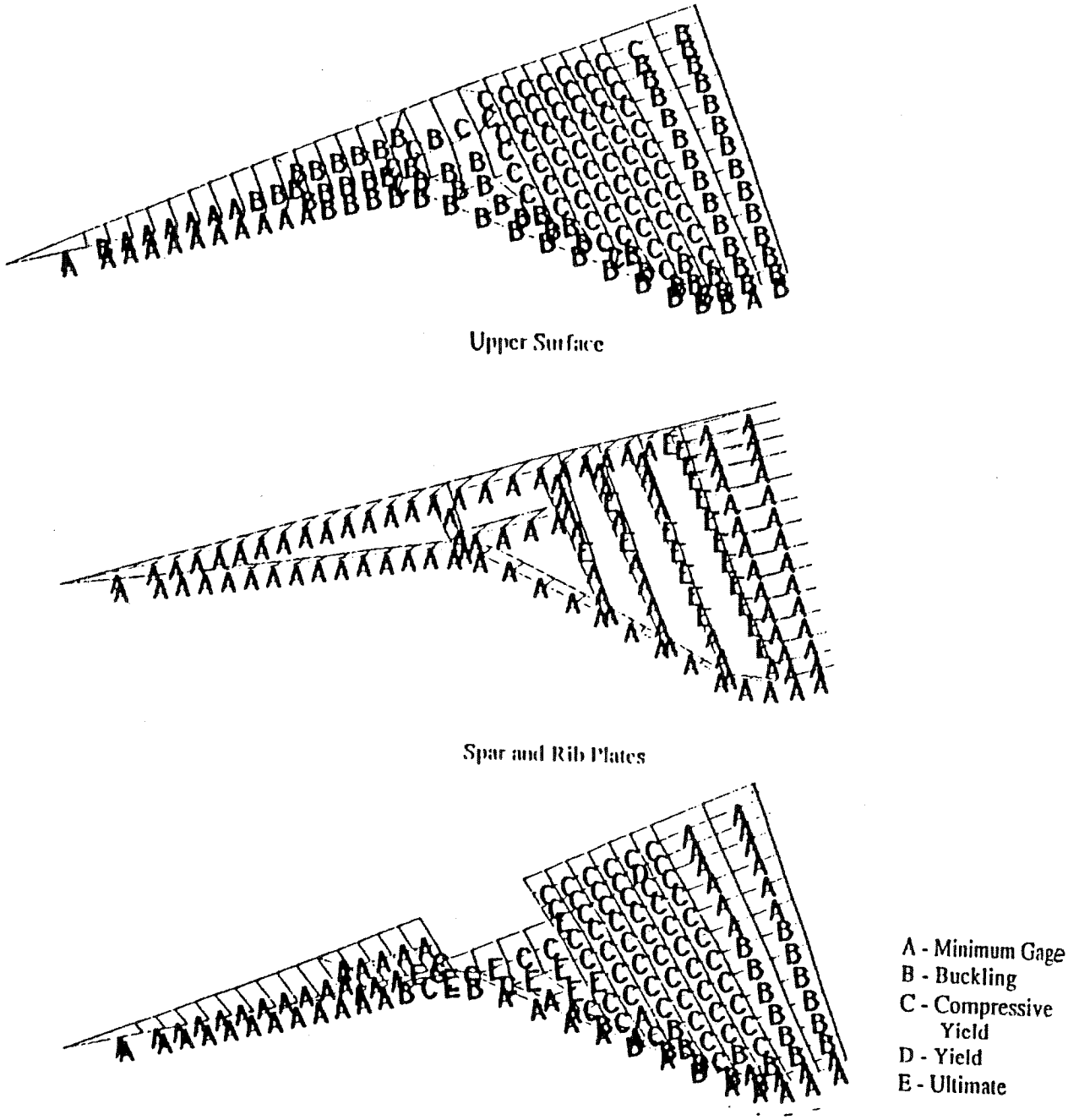
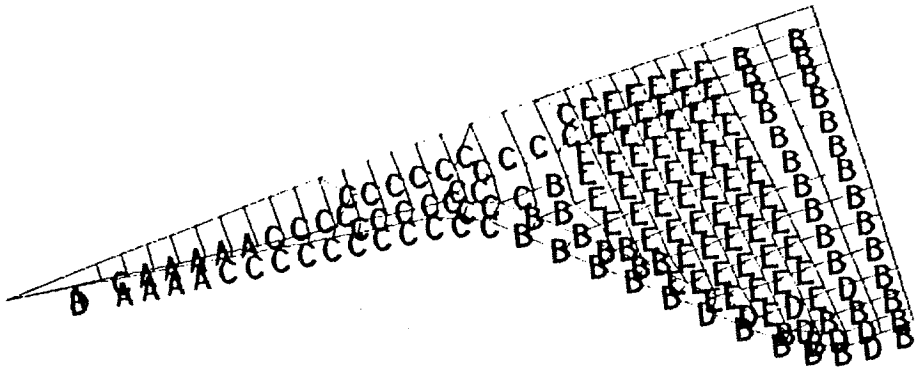
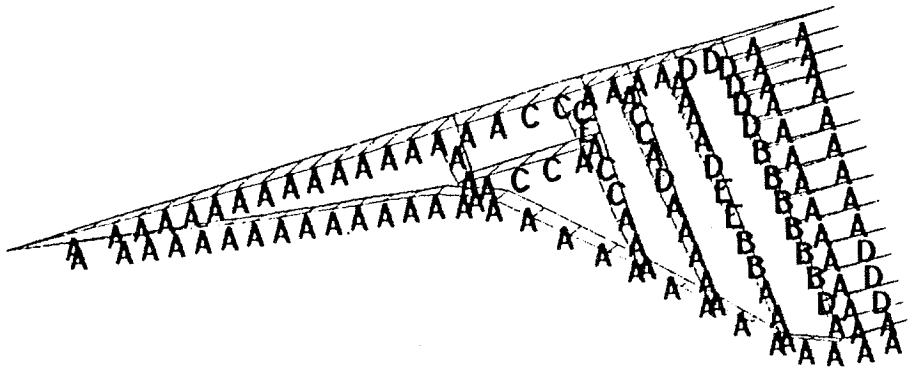


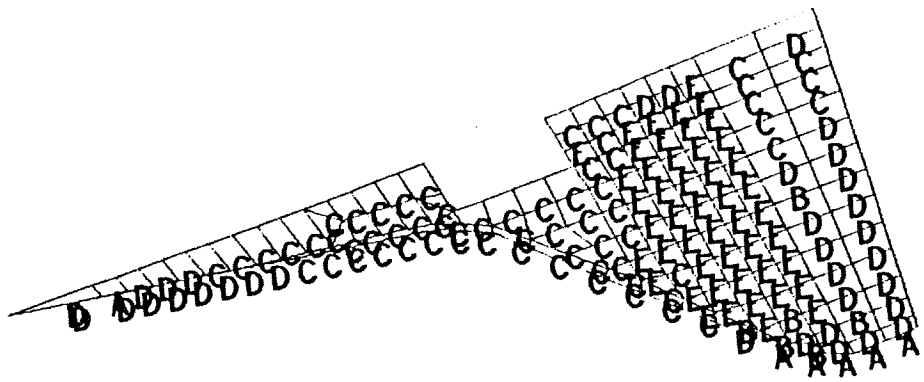
Figure 18. - Failure Mode, Lift Case, Plate Elements



Upper Surface



Spar and Rib Plates



Lower Surface

A - Minimum Gage
B - Lift
C - Landing
D - Hot Lift (
E - Hot Lift (with Ther Gradients

Figure 19. - Dominant Load Case, Plate Elements

Appendix A

User's Guide

This User's Guide describes the basic steps required to build, analyze, and size a model using the EZDESIT and EZBATCH computer programs. Initial user capabilites with PATRAN, EAL, and VAX/VMS are assumed.

The element sizing process can be broken into four major work blocks, see figure A-1. In summary the four blocks involve the use of PATRAN for model making, EZDESIT for conversion to EAL runstream format, VAX/VMS for additional data file creation and batch file execution, then finally use of both EZDESIT and PATRAN for results postprocessing. Specifics of the four blocks are now described.

PATRAN

Modeling Guidelines: Making a PATRAN neutral file and an EAL finite element model compatible with the EZDESIT system requires modeling restrictions. Only two node, three node and four node elements are available to describe a structural problem. For EAL element types, the two node elements must be type E23 (axial bar) and/or type E24 (planar beam). The three node and four node elements must be E33 (triangular plate) and E43 (quadrilateral plate) elements, respectively. For purposes of limiting model size for conceptual design work, a program limit of 2500 elements was chosen. PATRAN phase I and II modeling techniques are typically used to create the PATRAN neutral file (ASCII format) required by the element sizing programs. If planar beam elements are used, they must be oriented to bend about the PATRAN ybeam axis. Type "RUN,YBEAM", in PATRAN to view beam axis orientations. After creating nodal coordinates and element connectivity with PATRAN, the renumbering command must be used to generate the following element numbering scheme. Bar elements must be numbered first, followed by beam elements then triangular, and finally quadrilateral elements. After creating this required element order, element compaction is used to eliminate gaps in the element numbering scheme. Successful renumbering creates a model with elements consecutively numbered, one through to the total number of elements. Similarly, node numbers must be compacted from one to the maximum number of nodes in the model, unused nodes are not allowed. Nodal coordinates must be in units of inches.

Before describing additional modeling guidelines, the material property data file must be explained. On the program distribution tape there is a file titled MATPROP.PRN. This is an ASCII file of temperature dependent material properties. A sample portion of file MATPROP.PRN is shown in table A-1. Each finite element references a material described in this file through PATRAN physical property data sets. Field MATNUM (table A-1) assigns a unique integer value to each material in the material property datafile. For each material there must be at least two complete temperature entries of material properties

which bound element temperatures that will exist as element temperature loadsets in the PATRAN finite element model. Additional temperature dependent inputs are required for piecewise linear interpolation of material property data. The entire typical ten line block of data shown in table A-1 is "repeated" for properties at other temperatures or for new materials. Data is sorted first on MATNUM and secondly on TEMPERATURE, both in an ascending fashion. MATNUM must be of increasing integer value (one value for each material described) beginning at MATNUM = 1 for the first material. MATPROP.PRN is a user maintained file, material property data can be added, deleted or modified at the users discretion.

To size elements with EZDESIT and EZBATCH each finite element in the PATRAN database must be assigned a set of physical property data (table A-2). This data specifies design parameters required to size the various types of elements. PATRAN neutral file packet 04 stores physical property data. With the EZDESIT system there is no use for PATRAN packet 03 information. The most important design parameter associated with physical property data is the variable ITYPE. ITYPE is keyed into field 3 of a PATRAN physical property data set, and tells the sizing program if the element to be sized should be treated as a bar, beam, corrugated web or honeycomb sandwich. Refering to table A-2, note that ITYPE = 9 means the associated elements will be treated as isotropic honeycomb. Other values of ITYPE are reserved for different element types, as seen in tables A-3 through A-5. Additional ITYPES can be defined for user written element sizing routines. The first field of a PATRAN physical property dataset is reserved for the previously described variable MATNUM. MATNUM refers to file MATPROP.PRN, and specifies the material properties to associate with a particular property data set. In turn, any element assigned this property set will be sized using the associated MATNUM material. Assigning physical property data is done with PATRAN commands such as PFEG,P7T12,QUAD,,3 (assigns property set 3 to quad elements on patches 7 through 12). The element MOD command is also useful for assigning property id's to elements not associated with a phase 1 entity. Typically, several actual property sets will be required for sizing a finite element model. For example there could exist one data set for aluminum honeycomb, one for titanium honeycomb, one for steel beams and one for aluminum bars all in a single model. To define a physical property definition the following PATRAN command can be used, PROP,"SET ID",ADD. SET ID represents the property set ID number that will be assigned to appropriate elements with the PFEG command. Tables A-2 through A-5 list the existing property set definitions for types of element constructions that can currently be used with EZDESIT. User written element sizing routines must define a similar set of physical property data. Data manipulation by the EZDESIT program requires that fields 1 and 3 of any user written property set definition contain the generic variables MATNUM and ITYPE respectively. Up to 40 fields can be utilized for defining element physical properties. Values are read into the PIDD array contained in named common block /COMPIDD/ of EZDESIT and EZBATCH.

To define nodal weights (RMASS in EAL), PATRAN nodal temperature loadsets are used to represent nodal weights. Assigning weights as temperatures in PATRAN is

simply a technique used to associate scaler values with nodes. Subsystems and unmodeled non-load bearing structure are typically divided up and distributed to appropriate nodes as input weights which then affect inertial load cases. For purposes of creating temperature loadsets and defining temperatures for material property interpolation, PATRAN element temperature loadsets are used. There must exist at least one element temperature loadset where each element is assigned a temperature value in degrees fahrenheit. Typically the first element temperature loadset is used to define all elements as 72 deg. F. This represents a stress free condition. Various other element temperature data sets can exist to describe the model for other hot or cold design conditions. Element temperature loadsets also are used to represent element thermal gradients as defined in the EAL reference manual, in this instance only the elements with an applied thermal gradient need be called out in the element temperature loadset. When a hot condition will be used in EAL to create a temperature loadset the 72 degree (stress free) temperature will later be subtracted from the specified temperature to create appropriate EAL runstream information. Note that there are basically three uses for element temperature sets defined in PATRAN: 1) to use as an interpolating temperature when the sizing code is asked to create element stiffnesses, 2) to use as an interpolating temperature to obtain material allowables for sizing elements subject to hot condition loadsets, and 3) to be used by EAL for creating applied force sets based on element temperature data.

Other loads used by EAL, which can be input via PATRAN, are element pressures and nodal forces. PATRAN is also used to define constraint sets for later translation to EAL. Currrently, there is a restriction of no more than five degree of freedom constraints for any one node. This is due only to file input conventions. The use of PATRAN for model generation is of immense value in terms of being able to graphically display input geometry, physical property assignments, loadsets and constraint sets. Before an ASCII neutral file can be created for the element sizing codes to access, one more qualification must be met. At least one PATRAN named component must exist. Named components are very useful for examining a large model and have probably been used by the time a neutral file is going to be output. However, if at least one named component does not exist, element temperatures will be incorrectly defined during element sizing. Also note that no neutral file packet 03, 05, 09, or 12 through 20 type data has been generated when creating a model for EZDESIT. If information of this type is written to the neutral file the sizing programs will not operate properly.

Having defined a model using PATRAN as described, and having followed the appropriate modeling guidelines, an ASCII neutral file is then written out using the PATRAN interface menu. The title of this file will be PATRAN.OUT, as defined by PATRAN, and this name must be maintained throughout the element sizing process. The PATRAN prompt to include phase I geometry in the neutral file may be answered with a "no" to help maintain a reasonable size data file.

Menu Options: After a PATRAN neutral file exists in the users default directory, EZDESIT is used to create an EAL runstream suitable for batch execution (see block 2 of figure A-1). A brief review of EZDESIT menu options follows. After a quick review of the main menu, each option will be explained in detail. Typing EZDESIT results in a screen display of the programs main menu.

1 - Create an EAL runstream through XQT DRSI
2 - Modify an existing EAL runstream
3 - Create a partial EAL runstream,
for applying/combining loads and constraints
4 - Update EAL element stiffness matrices
5 - Translate EAL results to PATRAN results files
6 - Run element sizing code
7 - Create a worst case dimension file
8 - Review calculated component weights
9 - Exit to system

Option 1 is used to translate the PATRAN neutral file into an EAL runstream. User prompts determine which of the various EAL processors will be included in the created runstream. Items translated from the PATRAN database by option 1 include, nodal locations, element connectivity, constraint cases, and nodal weights. Option 2 is currently not available and tasks for which it was thought to be useful are accomplished with the VAX editor. Option 3 continues translation of the PATRAN neutral file. PATRAN loadsets are translated to EAL applied force loadsets. Also processors to solve static solution loadset constraint case combinations and print element stress resultants are added with this option. Option 4 reads information about element sizes from an existing dimension file and element temperatures from a requested PATRAN element temperature set. Information is then combined to create new element stiffness matrices for an existing EAL runstream. Element stiffness matrix updating is an integral part of the EZBATCH sizing procedures and is useful in the interactive mode to create initial element stiffnesses based on minimum gage element sizes, or to create model stiffnesses associated with a particular PATRAN element temperature set. Option 5 is used to translate files of element stress resultants and nodal displacements created by EAL's DCU/PRINT statement into PATRAN element and nodal results files. The results of EAL can then be viewed using PATRAN postprocessing capabilities. Option 6 runs the element sizing portion of EZDESIT for a specified mechanical or combined thermal and mechanical set of stress resultants. Material allowables for element sizing are based on a specified PATRAN element temperature set. Element unit weight and dimension files are created by option 6. Unit weight files are in PATRAN element results file format. Option 7 is used to create a worst case element dimension file, (to be described in the POSTPROCESSING section) after dimension files have been created for multiple loadsets. Subsequent element stiffness matrix updating is then based on dimensions from the worst case element dimension file. Option 8 creates a worst case element unit weight file in a manner similiar to the worst case dimension file. Weights are

displayed on screen tabulated by element type, failure mode, and most critical loadset. PATRAN named component weights are also calculated and displayed with option 8. Option 9 returns users to the VAX VMS environment.

Creating the First Part of an EAL Runstream: The first prompt from option 1 is for a name of the EAL runstream which will be created. For consistency with the EZBATCH sizing process the name EALINP.COM is recommended. Next follows a series of questions which are self explanatory to users who have created EAL static solution runstreams. EZDESIT will translate joint locations, element connectivity, constraint sets, and PATRAN nodal "temperature" (ie: nodal weight) sets to EAL equivalent runstream format. NSW (EAL non-structural weight) values are used to store element weights, because of this convention the MATC values for material density are set to zero during batch sizing. Dummy information for EAL sections BC, BD, and SA will be written to the runstream, later use of options 6 and 4 will update the dummy information to initially reflect element minimum gage sizes. Constraint set translation includes the ability to add EAL defined symmetry planes, symmetry plane type constraints do not have to be part of the PATRAN model. Care must be used to be certain the EAL default values of parameters LZERO and RZERO are acceptable. Geometry test defaults for processor E may be left as the standard EAL values, changed to more relaxed EZDESIT defaults, or input from the keyboard. For setting up an EAL runstream to be used with EZBATCH at least one execution of DRSI has to be included. Besides being required by the EAL static solution process, the *XQT DRSI line acts as a flag for the batch process which updates the EAL runstream to execute DRSI only for constraint cases requested in the batch process data file. The batch process data file will be explained in a subsequent section.

Updating the EAL Runstream based on Minimum Gage Dimensions: The runstream created by option 1 has dummy information for element section and material properties. Element stiffnesses suitable for first pass execution of the static solution process are produced by updating the EAL runstream to reflect minimum gage dimensions as were keyed in to the PATRAN neutral file physical property data sets. To accomplish this, the first step is creation of a minimum gage dimension file. While still in EZDESIT and after completing option 1, option 6 is chosen. There will be a user prompt for the name of an element stress resultant file to use for element sizing. Typing ZERO instead of a file name runs the sizing routine for a null set of loads, and so produces a dimension file defaulting to previously keyed in physical property minimum gage information. Users should decline options to include a thermal stress resultant file when running the null set of mechanical loads. The name LDSTMIN.DIM is a good name to use when prompted for the name of the output dimension file. Also UWTMIN.OUT is a good name to use for the output element unit weight file. After completion of option 6, choose option 4 to use the minimum gage dimension file as the basis for a new set of element stiffness matrices. Note that option 4 is automatically executed by program EZBATCH during iterative sizing. Interactively, prompts will appear for a dimension file name and a PATRAN element temperature set to use for calculating the new element stiffnesses. LDSTMIN.DIM will typically be the dimension file name. A PATRAN element temperature set where all

elements are set to a stress free temperature (typically 72.0 deg. F) should be chosen for the element temperature input.

Creating the Second Part of the EAL Runstream: Option 3 creates the second part of an EAL runstream. In the rather rigid runstream requirements of the EZBATCH process, this means input for all processors after DRSI. EAL processors AUS, and EQNF are used to define mechanical and thermal loadsets. Processor SSOL is used to obtain static solutions. Processor ES is used to generate tables of element stress resultants and processor DCU is used to print stress resultants and nodal displacements to ASCII files. For a runstream being used in the EZBATCH iteration procedure only processors up to SSOL need to be input. EZBATCH recognizes the flag line *XQT SSOL and writes information for creating appropriate stress resultant files based again on information in the batch process data file. When applied loadsets are created EZDESIT adds a line to call runstream data set FSUM PROG. This data set must exist in library 20 of your EAL model. See appendix B for how to create this runstream dataset.

Option 3 will first prompt for the name given to the first part of your EAL runstream, certain header information is transfered from this file to a new file. A name must be input for this second runstream, LOADS.EAL will be used for discussion purposes. The following menu will then appear:

Run Stream Additions
1 - Add loadsets from the PATRAN database
2 - Apply inertial accelerations
3 - Combine loadsets
4 - Add static solutions
5 - Generate element load files
6 - Generate nodal displacement files
7 - End

The first suboption of Main Menu Option 3 is to translate loads from the PATRAN model to applied loading in the EAL runstream. PATRAN nodal force loadsets, element pressure loadsets, and element temperature loadsets may be converted to EAL loadsets. When converting element temperature loadsets a value to subtract from the PATRAN average element temperature (representing the model stress free temperature) can be subtracted off of the value written to the EAL temperature loadset. Users are also prompted to input a temperature loadset representing element thermal gradients if desired. A single EAL temperature loadset can thus produce stress resultants due to through the thickness element temperature gradients as well as differing average element temperatures and stiffnesses.

Suboption 2 creates an EAL loadset for inertial loading as a z direction acceleration. Users wishing other inertial acceleration components will have to edit the created EAL runstream. Main Menu 3 suboption 3 is used to factor and combine various mechanical

loadsets to create single mechanical loadsets which are then used in the element sizing process. Note that mechanical and thermal loadsets are kept seperate throughout the element sizing process.

Suboption 4, "Add Static Solutions", writes the appropriate EAL SSOL processor information. As with DRSI, when sizing elements using the batch process, only one dummy loadset/constraint SSOL processor need exist in the EAL runstream. Appropriate information will be read from file EZBATCH.DAT.

Suboptions 5 and 6 are useful if the created runstream is to be submitted without going through the EZBATCH process. The EAL processor DCU/PRINT is used to print ASCII files of element stress resultants and nodal displacements. These files can later be translated to PATRAN results file format using Main Menu Option 5. If you choose the option to generate element load files the files will be named FOR#.dat where # is the number of an EAL loadset plus 10. Similarly, displacement files are written to files where # is the number of the EAL loadset plus 100. It is not necessary to run the options which create element load files and nodal displacements if the runstream is to be used in the batch sizing process as these functions are automatically taken care of during batch execution.

VMS

Combining Parts 1 and 2 of the EAL Runstream Into a Single EZBATCH Runstream:
After the files EALINP.COM and LOADS.EAL have been created, Option 9 from the main menu is used to return to the VMS operating system environment (block 3, figure A-1). Setting up an EAL runstream to be used in the iterative EZBATCH process requires that the file LOADS.EAL be appended to EALINP.COM, and some editing changes made. The easiest way to do this is by editing EALINP.COM. Once in the editor, revise the top portion of the file to look like:

```
$DELETE EALINP.;*
$DELETE *.L01;*
$DELETE *.L30;*
$RUN $DISK1:[EAL]EAL312  (or appropriate command for your system)
'EALINP.
*ONLINE=0
*XQT TAB
  - remaining EAL data follows -
```

Now go to the file end, and use the editor include command to bring in file LOADS.EAL. Move up to what was the beginning of LOADS.EAL (now just after the *XQT DRSI line) and delete the 4 lines of header information which is not necessary when making the single runstream for EZBATCH. Move to the end of the file and add the line *XQT EXIT,

save the file as the new version of EALINP.COM.

Creating Data Files for the EZBATCH Process: Before EALINP.COM can be submitted for iterative element sizing two more data files must be created, EZBATCH.DAT and EZBATCH.COM. EZBATCH.COM is a file of three VMS commands and it is the actual file submitted to perform iterative element sizing: The contents EZBATCH.COM are as follows:

```
$SET DEFAULT $DISK3:[CERRO.PATRAN.WINGEX]
(set default directory to that which contains your model)
$ASSIGN EZBATCH.OUT SYS$OUTPUT
(iteration information is written to file EZBATCH.OUT)
$RUN $DISK3:[EZBATCH]EZBATCH
(or whatever directory the ezbatch.exe file exists in)
```

EZBATCH.DAT supplies data to program EZBATCH that defines which EAL loadsets, associated constraint sets, associated thermal loadsets, and associated element temperatures are used for iterative sizing. The following typical EZBATCH.DAT file corresponds to input for the example load and constraint information which will be discussed in table C-2. The record structure must be observed but within a line input is in free field format.

```
@EALINP.COM {name must be ealinp.com}
EALINP.                          {filename of eal library}
364 0 53 397                     {NUME23 NUME24 NUME33 NUME43}
4              {number of loadsets to size for}
1 6 1 1                          {EAL mechanical loadset numbers}
1 2 1 1                          {associated EAL mechanical loadset constraints}
0 0 1 1                          {ITHRMON(I), I=1,# of loadsets}
0 0 3 4                          {EAL temperature loadset numbers}
0 0 1 1                          {associated EAL temperature loadset constraints}
1 1 2 2                          {PATRAN element temperature set for property interpolations}
5              {number of iterations allowed}
```

Where:

nume23 = number of E23 elements in the model
nume24 = number of E24 elements in the model
nume33 = number of E33 elements in the model
nume43 = number of E43 elements in the model
ithrmon = 0 no temperature loadset associated with this mechanical loadset
= 1 there is a temperature loadset associated with this mechanical loadset

After all appropriate data files have been created the iterative element sizing process is initiated by submitting EZBATCH.COM as a batch job. SUBMIT/NOLOG/NOTIFY EZBATCH is a typical VMS command to use.

Checking For Convergence: Completion of EZBATCH sizing will result in a number of worst case dimension files of sequentially increasing version number. If five iterations were permitted, and the first worst case dimension file (LDSTCOMB.DIM) was version one, the most current version would be version six. To check for convergence use the VAX differences command; "DIFFERENCES/output =DIFFERS.OUT LDSTCOMB.DIM". DIFFERS.OUT can be edited to examine how many elements have different dimensions than the previous iteration. Users decide whether a model is converged sufficiently for their analysis purposes.

POSTPROCESSING

Creating a worst case Dimension File: If you have interactively run the sizing code for several loadsets, you will have several LDST#.DIM element dimension files. Main Menu Option 7 is used to look for the heaviest element in each file and use the geometry of that element to write a new dimension file. This is called the worst case element dimension file. A good convention for the naming of the worst case dimension file is LDSTCOMB.DIM. LDSTCOMB.DIM can then be used as input to update model stiffnesses if the user decides to calculate vibration modes or thermal stresses in an early preliminary design analysis. Note that the creation of new LDSTCOMB.DIM files is automatically done during batch program execution.

Graphical Display of Results: Option 5 translates element loads and nodal displacements files created by the EAL processor DCU/PRINT, to PATRAN results file format. Users are prompted for file names created by the EAL runstream (FOR#.dat) and file names they desired for the corresponding PATRAN formatted file. To be consistent with the batch process users should use the filenames PATLD#.OUT, and PATLDTH#.OUT for element load resultant files, where # equals the EAL loadset number. Similarly PATDIS#.OUT should be used for displacement files of the corresponding loadset.

Tabular Display of Results: Choosing Main Menu Option 8 gives the user two subchoices. One to create a worst case element weight file and another to review the calculated element weights. A worst case element unit weight file is created in a manner similiar to the worst case dimension file. A good filename choice for this file is UWTCOMB.OUT, this maintains some consistency with individual loadset unit weight file names. By choosing sub-option 2, calculated structural weights will be written to the users' monitor summarized by failure mode, dominant loadset, and component. It is wise to break your model into several named components within PATRAN so the weights can be reviewed on a component by component basis.

Table A-1

Description of Data Fields in File MATPROP.PRN

Matnum	Matype	Description	
Temperature*	Ex	Ey	GxyVxy
Vyx	Rho	Ftux	FcuxFtuy
Fcuy	Ftyx	Ftyy	Fcyx Fcyy
Fsmat	C	N	Kth Kic
crpco1	cprco2	S11fat	N11fat S12fat
N12fat	S13fat	N13fat	S21fatN21fat
S22fat	N22fat	S23fat	N23fatS31fat
N31fat	S32fat	N32fat	S33fatN33fat
cte1	cte2	0.0	0.00.0

* Temperature corresponds to data field 1, ie: in EZDESIT common block ALWBLS, the array RMPRO holds the above values beginning with temperature as RMPRO(1) and continuing through cte2 as RMPRO(42). Matnum, Matype, and Description are not stored in the RMPRO array.
Field values must use the above record structure, but within one line formatting is not required.

Matnum = Each material is given an identifying number, materials must be put into the the database sorted first by increasing Matnum (integer), and second by increasing Temperature.
Matype = 1 = Isotropic
2 = Organic Composite
3 = Metal Matrix Composite
Description = Text Description of Material described
Temperature = Material temperature, degrees F.
Ex = Youngs Modulus material x direction, psi
Ey = Youngs Modulus material y direction, psi
Gxy = Shear Modulus, psi
Vxy = major (larger) poissons ratio
Vyx = minor (smaller) poissons ratio
Rho = Weight density, Lb/in**3
Ftux = X direction tensile ultimate strength, psi
Fcux = X direction compressive ultimate strength, psi
Ftuy = Y direction tensile ultimate strength, psi
Fcuy = Y direction compressive ultimate strength, psi
Ftyx = X direction tensile yield strength, psi
Ftyy = Y direction tensile yield strength, psi
Fcyx = X direction compressive yield strength, psi
Fcyy = Y direction compressive yield strength, psi
Fsmat = Ultimate shear strength, psi
C = Coefficients in Frac. Mech. eqn: Da/Dn = C*(del K)N
**N = Power in above equation
**Kth = Material threshold fracture toughness (ksi sqrt(in))
**Kic = Material fracture toughness (ksi sqrt(in))
crpco1 = Coefficient in creep curve equation: creep rate (%/hr) = crpco1*(Stress (ksi))crpco2
**crpco2 = Power in above equation
**Sijfat = Allowable nominal stress at fatigue life Nijfat, ksi
**Nijfat = Fatigue life, cycles, for Kt=i; j=1,2,3 breakpoints on fatigue curve
cte1 = coef. of thermal exp., material x direction, in/in/deg f
cte2 = coef. of thermal exp., material y direction, in/in/deg f
** items not required by this version of the sizing program, input 0.0

Table A-2
Physical Property Definition for
An Isotropic Material, Honeycomb Plate

field	generic name	description
1	matnum	material number, refer to file matprop.prn (ex. 6=2024 aluminum)
2	span	Span used to check panel buckling, (in.)
3	itype	9 = isotropic honeycomb
4	factor	factor times input loads to obtain ultimate load (typically 1.0 with EAL runs done at ultimate load)
5	exptoth	panel buckling load knockdown factor (typically .8)
6	nof	Non optimum factor (1.0 = none)
7	minga	minimum gage of a facesheet, (in.)
8	ymred	youngs modulus reduction factor, (1.0 = full value in file matprop.prn)
9	alsred	allowable strength reduction factor, (1.0 = full value in file matprop.prn)
10	dummy	0
11	cormin	minimum core height, (in.)
12	cormaxh	maximum core height, (in.)
13	rhocor	core material density, (lb/in**3) (ex: aluminum = .1, routine hcbndth assumes 2% core factor)
14	dummy	0
15	dummy	0
16	dummy	0
17	dummy	0
18	dummy	0
19	dummy	0
20	dummy	0
21	dummy	0
22	dummy	0
23	dummy	0
24	dummy	0
25	dummy	0
26	yiefac	factor times ultimate load to obtain limit load level (ex: = .67 if using a 1.5 safety factor)
27	dummy	0
28	dummy	0
29	dummy	0
30	dummy	0
31	dummy	0
32	dummy	0
33	dummy	0
34	dummy	0
35	dummy	0

dummy parameters are only placeholders for possible code expansion

Table A-3

Physical Property Definition for
An Isotropic Material, Corrugated Web

field	generic name	description
1	matnum	material number, refer to file matprop.prn (ex. 6=2024 aluminum)
2	span	Panel Buckling Span, (in.)
3	itype	10 = Corrugated Web
4	factor	factor times input loads to obtain ultimate load
5	exptoth	panel buckling load knockdown factor (typically .8 empirical/theoretical ratio)
6	xory	= 1 corrugated along the element x direction = 2 corrugated along the element y direction
7	minga	minimum material gage, (in.)
8	dummy	0
9	dummy	0
10	dummy	0
11	dummy	0
12	dummy	0
13	dummy	0
14	dummy	0
15	dummy	0
16	dummy	0
17	dummy	0
18	yiefac	factor times ultimate load to obtain limit load level (ex: = .67 if using a 1.5 safety factor)
19	dummy	0
20	dummy	0
21	dummy	0
22	dummy	0
23	modred	modulus reduction factor, (1.0 = full value from file matprop.prn)
24	alsred	allowable strength reduction factor, (1.0 = full value from file matprop.prn)
25	nof	non optimum factor, (1.0 = none)

dummy parameters are only placeholders for possible code expansion

Table A-4

Physical Property Definition for a Bar

field	generic name	description
1	matnum	material number, refer to file matprop.prn (ex. 6=2024 aluminum)
2	nof	non optimum factor (1. = none)
3	itype	13 = Bar
4	factor	factor times input loads to obtain ultimate load
5	amin	min. area, (sq. in.)
6	modred	modulus reduction factor, (1.= full value from file matprop.prn)
7	alsred	allowable strength reduction factor, (1.0 = full value from file matprop.prn)
8	dummy	0
9	dummy	0
10	dummy	0
11	yiefac	factor times ultimate load to obtain limit load level (ex: = .67 if using a 1.5 safety factor)
12	dummy	0
13	dummy	0
14	dummy	0
15	dummy	0

dummy parameters are only placeholders for possible code expansion

Table A-5

Physical Property Definition for a Planar Beam

field	generic name	description
1	matnum	material number, refer to file matprop.prn (ex. 6=2024 aluminum)
2	nof	non optimum factor (1. = none)
3	itype	14 = Planar Beam
4	factor	factor times input loads to obtain ultimate load
5	acmin	min. cap area, (sq. in.)
6	bmin	min. web height, (in.)
7	tmin	min. web gauge, (in.)
8	modred	modulus reduction factor (1.0 = full value from file matprop.prn)
9	alsred	allowable strength reduction factor, (1.0 = full value from file matprop.prn)
10	dummy	0
11	dummy	0
12	dummy	0
13	yiefac	factor times ultimate load to obtain limit load level (ex: = .67 if using a 1.5 safety factor)
14	dummy	0
15	dummy	0
16	dummy	0
17	dummy	0

dummy parameters are only placeholders for possible code expansion

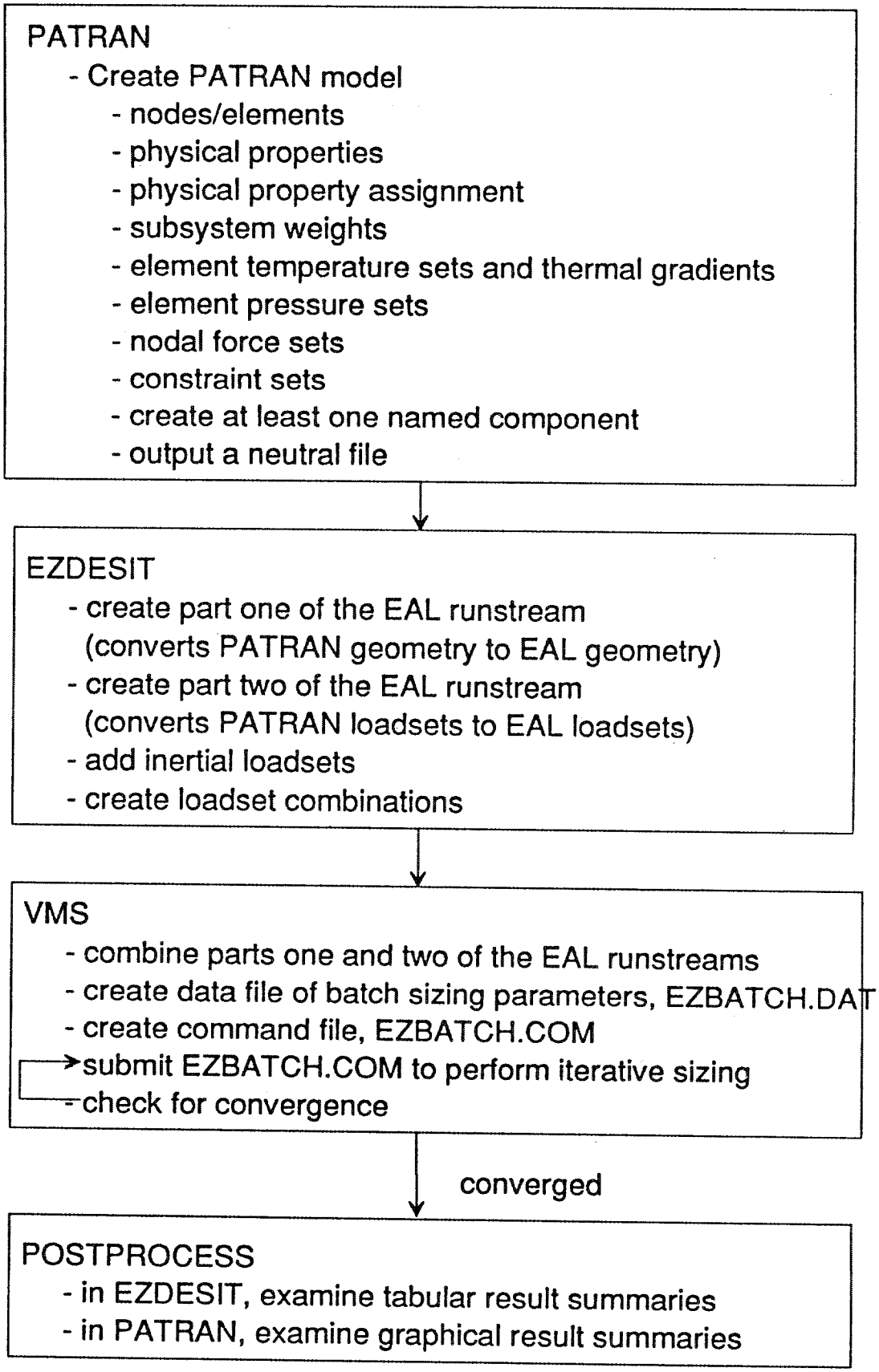


Figure A-1. Process Flow Chart

Appendix B

File Naming Conventions

There are numerous files required and created by the EZDESIT and EZBATCH element sizing programs. The iterative batch process is more restrictive in terms of requiring specific filenames than would be necessary if all sizing was done without iterating. For consistency, a file naming system will be described based on EZBATCH conventions. A preliminary requirement for keeping order, is that all files associated with analyzing a particular model should be kept in a dedicated directory. The file naming convention is as follows:

EALINP. EAL output file, created by submitting EZBATCH.

EALINP.COM EAL input runstream, created by EZDESIT.

EALINP.L01 EAL library 01, created by submitting EZBATCH.

EALINP.L20 EAL library 20, contains the required runstream dataset, "FSUM PROG", must exist before submitting EZBATCH. This runstream can be created by copying and renaming the EAL/EISI supplied runstream "FSUM" distributed by EISI in the EAL utililties source code file. Store the EAL compiled library for this runstream in library EALINP.L20.

EALINP.L30 EAL library 30, created by submitting EZBATCH.

EZBATCH.COM VMS command procedure used to start the EZBATCH sizing program, user created.

EZBATCH.DAT Data file required when submitting EZBATCH, user created.

EZBATCH.OUT Output summary information created by submitting EZBATCH. Useful for debugging and checking program execution clock times.

LDST#.DIM An ASCII file of element dimensions.
= 'MIN', typically used to represent the zero load case sizing results
= 'COMB', typically used to represent the combined loadset, worst case dimension file
= N, typically used to represent the results from mechanical loadset N.
= N_M, typically used to represent the results from mechanical loadset N and thermal loadset M.

Note: N and M correspond to EAL loadset identifiers, not PATRAN

LOADS.EAL Temporary file created by running Option 3 of EZDESIT. Contains EAL runstream data for applying loads, defining static solution parameters, and creating element stress resultant files.

MATPROP.PRN User maintained file of temperature dependent material properties.

PATRAN.DAT User created PATRAN database of the structural model.

PATRAN.OUT Ascii neutral file created from PATRAN.DAT, phase I information not required.

PATDIS#.OUT An ASCII PATRAN displacement results file.
= N, typically used to represent the results from EAL loadset N.

PATLD#.OUT An ASCII PATRAN element results file of EAL calculated element stress resultants. Notation per EAL.
= N, typically used to represent the results from loadset N.
for E23 (BAR) elements:
column 1 = Axial Force, lb
column 2 = Axial Stress, lb/sq in
for E24 (Planar Beam) elements:
column 1 = Axial Force, lb
column 2 = Shear Force, lb
column 3 = Bending Moment, in-lb
for plate elements:
column 1 = Nx, lb/in
column 2 = Ny, lb/in
column 3 = Nxy, lb/in
column 4 = Mx, in lb/in
column 5 = My, in lb/in
column 6 = Mxy, in lb/in

PATLDTH#.OUT Same as PATLD#.OUT but for Thermal load sets

UWT#.OUT An ASCII PATRAN element results file, used to store results of element sizing.

columns 1 thru 3 are NOT used.

column 4 = element unit weights, Lb/FT for 2 node elements, Lb/SQ FT for plates

note: includes non-opt factor specified in the PATRAN physical property data sets.

column 5 = failure mode indicator;

1 - Minimum gage
2 - Panel buckling
3 - Compressive Yield
4 - Yield
5 - Ultimate
6 - Fatigue (not implemented)
7 - Fracture Mech. (not implemented)
8 - Creep (not implemented)
9 - not used
10 - Panel Flutter
11 - not used
12 - Flag, thermal stress or dimension limitations exceeded, unable to size this element, sizes and weights will be set abnormally high as a caution.

column 6 = dominant loadset indicator, this column is created by the interactive sizing code main menu option 8 submenu option 1. A 1 in this column corresponds to the first filename given when entering data for the above option; 2 corresponds to the second file, 3 to the third etc. up to 10 files.

= 'MIN', used to represent the zero load case sizing results

= 'COMB', used to represent the combined loadset, worst case dimension file

= N, typically used to represent the results from mechanical loadset N.

= N_M, typically used to represent the results from mechanical loadset N and thermal loadset M.

Appendix C

Wing Sizing Example and Program Restrictions

Included on the distribution tape for EZDESIT is a VMS backup format saveset titled WINGEX.SAV. This save set is a directory of files which document use of the batch iteration process by sizing a space shuttle type wing subject to multiple load cases, including thermal stresses. Use the VAX VMS backup facility to upload WINGEX.SAV to a directory called WINGEX.DIR. Table C-1 describes the files stored in WINGEX.SAV.

Figure C-1 outlines the steps used to create, analyze, and postprocess this example problem. Initially, (block 1, figure C-1) PATRAN phase 1 geometry was interactively input based on crude sketches. PATRAN GFEG and CFEG options were use to create some of the structural elements, and some were input with repetitive PATRAN phase 2 modeling commands. The model consists of upper and lower surfaces modeled with plate elements. Spars and some ribs are plate elements (EAL E33 and E43 type) with the remaining rib truss structure modeled using bar elements (EAL type E23). No beam elements (EAL type E24) are used. The finite element model is shown in Figure C-2. Table C-2 lists load and constraint cases as defined in the PATRAN and EAL models. Note that each EAL loadset sized for has a distinct loadset identifying number. EAL loadset numbers need not match PATRAN "set id" numbers.

It is desired to size the wing for four loading conditions; lift loads, landing loads, a lift condition with inplane thermal loads, and a lift condition with thermal loads including some element temperature gradients. Users should interactively examine the PATRAN model (PATRAN.DAT) to view applied loads, temperatures, constraint cases and become familiar with the named components. Named components are broken out in two ways, one way separates the wing into five substructures, leading edge, control surface, torque box, landing gear bay and glove. That named component breakdown is:

rootrib	leadedge	controlsurf	torquebox	lgbay	glove
	-lespar	-csus	-tbus	-lgus	-gloveus
	-leus	-csls	-tbls	-lgls	-glovels
	-lels	-csbars	-tbbars	-lgbars	-glovebars
	-lebars	-rearspar	-spar1307	-spar1191	
		-csribs	-spar1249	-spar1010	
				-lgrib	

For purposes of temperature interpolations (performed off line from the EZDESIT system) the following named components were also defined:

model	ls	us	spars	ribs	bars	
-"everything"	-lels	-leus	-lespar	-rootrib	-"all E23's"	
	-csls	-csus	-rearspar	-csribs		
	-tbls	-tbus	-spar1307	-lgrib		
	-lgls	-lgus	-spar1249			
	-glovels	-gloveus	-spar1191			
			-spar1010			

The PATRAN model must also include physical property descriptions and assignment of these physical properties to elements. Three physical property sets were used to define the physical property data required by element sizing routines in the example wing problem. Honeycomb (property set 1) was used for skin elements, bar (property set 2) for truss elements, and corrugated web (property set 3) for plate element ribs and spars. Property set assignments can be viewed by color coding elements to their property id's using the PATRAN RUN,ASSIGN,PID command. Actual physical property data can be examined by giving the PATRAN command "PROP,1T#,SHOW". It is useful to set the PATRAN print option on for this command and later print the file PATRAN.PRT from the VAX command level. After phase II modeling is completed an ASCII neutral file is created from the PATRAN interface menu. PATRAN.OUT (neutral file default name) must be retained as the neutral file name throughout the element sizing process.

Having finished the tasks outlined in block one of figure C-1 work moves onto items accomplished within the EZDESIT interactive environment (block 2, figure C-1), creation of the file EALINP.COM. EALINP.COM is the required name for an EAL runstream within the batch sizing process. Program EZDESIT option 1 creates the first part of an EAL runstream from your existing PATRAN neutral file, (note option 2 is not yet coded). Typical option 1 execution is included in the screen session listing of table C-3. Upon completion of Option 1 you will have an EAL static solution runstream through processor DRSI. Option 3 is then used to create the remaining portion of a runstream, particularly application of loads from the PATRAN neutral file. For this second file to be complete users must request at least one static solution. Doing so writes the line *XQT SSOL in the file which acts as a flag required by the batch sizing process. Appropriate load and constraint case information for each execution of processor SSOL is actually obtained from the batch data file, EZBATCH.DAT. Typically the name EALINP.COM is input in response to Option 1's request for an EAL runstream name. LOADS.EAL is input in response to option 3's request for an output EAL runstream name. Later, when at the VMS level, LOADS.EAL can be copied to the end of file EALINP.COM. While still in EZDESIT, option 6 must be run to create an intial element dimension file which should be named LDSTMIN.DIM. This file is created from a zero load case condition where no prior EAL output is required. EZDESIT can now be left to return to the Vax operating system

(block 3, figure C-1). Using an editor, combine the two portions of the EAL runstream (EALINP.COM and LOADS.EAL) into one file named EALINP.COM, follow the skeleton runstream shown in table C-4 and note that non EAL command header information must be deleted from the top of file LOADS.EAL, as it is being being appended to an existing EAL runstream named EALINP.COM. Adding the first three lines shown in Table C-4 to your EALINP.COM file is very important to minimize disk space requirements during the iterative sizing process. EALINP is output by EZDESIT as the name for your EAL libraries. Also, at the VMS level, copy LDSTMIN.DIM to LDSTCOMB.DIM. This creates a dimension file used by the batch iteration process for the first element sizing iteration. LDSTCOMB.DIM is updated via new version numbers for each iteration of the sizing process. EZBATCH.COM must be created with appropriate directory information and the file EZBATCH.DAT must be created with appropriate file name, model, and loadset requirement information. Table C-5 lists these two files as used for sizing the example wing subject to 4 loadsets and permitting 5 iterations, a total of 20 executions of EAL. The file EALINP.L20 must also exist in the directory to be used for sizing. EALINP.L20 contains the EAL runstream dataset FSUM PROG used to check force summations of applied loadsets.

Having created the following files the user is ready to submit EZBATCH.COM for iterative element sizing (block 3 figure C-1):

EZBATCH.COM	Command file for executing program EZBATCH
EZBATCH.DAT	Data file of element types, and required load/constraint combinations to use in the element sizing process
EALINP.COM	EAL runstream
EALINP.L20	EAL library containing runstream dataset FSUM PROG, calculates force summations of applied load sets.
LDSTCOMB.DIM	First pass element dimension file, usually based on zero load sizing and results in minimum gage stiffnesses used in EAL for the first calculation of element internal loads. A good convention is to use LDSTMIN.DIM as the input requested dimension file name when running EZDESIT option 6 for ZERO load, then copy LDSTMIN.DIM to LDSTCOMB.DIM for input to the iterative batch element sizing. LDSTCOMB.DIM will continually be revised as sizing progresses. Each submittal of EZBATCH.COM iterates a user specified number of times and each iteration produces a new version of LDSTCOMB.DIM. Users can use the VAX differences command to check convergence to within their liking.
MATPROP.PRN	Temperature dependent materials database. The file name matprop.prn must be used for batch element sizing.
PATRAN.DAT	PATRAN database of the finite element model.
PATRAN.OUT	PATRAN neutral file of the finite element model, phase I data not required.

UWTMIN.OUT	A good convention name to use for the requested unit weight file name when running EZDESIT menu option 6 for ZERO load.

Start the batch process with the VMS command "Submit EZBATCH". Upon Completion of EZBATCH.COM the following additional files will reside in the users directory:

EZBATCH.OUT	Output file assigned to SYS$OUTPUT in EZBATCH.COM. Shows progress of the batch process via various write statements, can be deleted.
LDST#.DIM	Dimension files as described in appendix B, typically there will be one for each loadset required to size for, one for a combined (worst case) file, and one for a minimum gage file which was created during the interactive process using EZDESIT.
LDSTCOMB.DIM	Sequentially increasing version numbers of worst case element dimensions, one file for each iteration of the EAL / element sizing solution.
PATLD#.OUT	Element internal loads for mechanical loadset #, as described in appendix B
PATLDTH#.OUT	Element internal loads for thermal loadset #, as described in appendix B
UWT#.OUT	Unit weight file as described in appendix B
EALINP.	Output of the last EAL run.
EALINP.L01	EAL library created from the last EAL run.
EALINP.L30	EAL library created from the last EAL run.

Convergence can be checked based on the last two combined loadset dimension files with the VAX differences command.

ex: differences/match=0/nonumber/output=differs.out ldstcomb.dim

A sample DIFFERS.OUT file is also included in the WINGEX.SAV saveset.

Once the batch process of element sizing is completed the user is ready to post process results, this is block 4 of figure C-1. Post processing in EZDESIT is main menu option 8. Suboption 1, is used to "create a worst case element weights file". UWTCOMB.OUT is the conventional name for this file. The program will prompt for a number of unit weight files to combine and their names. UWTMIN.OUT is input as the first file so that when summarizing unit weights by input file names weights will be governed by the minimum gage condition unless another loadset causes unit weight increases. The sample UWTCOMB.OUT file was created from a summary of 5 loadsets (minimum gage plus the 4 defined EAL loadsets). File names were input in option 8 in the following order, UWTMIN.OUT, UWT01.OUT, UWT06.OUT, UWT01_03.OUT, UWT01_04.OUT. UWTCOMB.OUT is then summarized with the "review calculated component weights" (menu 8 sub option2). Upon exiting EZDESIT the user may wish to examine the results of UWTCOMB.OUT via PATRAN's element color coding capabilities.

Note that elements of unusually high unit weight will result when the batch sizing process is unable to reduce thermal stresses to an acceptable value. For honeycomb elements the facesheet gages are set to one inch thickness to act as a flag for this condition.

Table C-1

Directory $DISK3:[CERRO.PATRAN.WINGEX]

DIFFERS.OUT;1	{Output of vax differences command on ldstcomb.dim}
EALINP.BKP;1	{backup copy of a version of ealinp.com}
EALINP.COM;1	{EAL runstream for the batch process}
EZBATCH.COM;16	{Command file to submit to perform iterative sizing}
EZBATCH.DAT;6	{Data file required with EZBATCH.COM submittal}
EZBATCH.OUT;1	{History of iterative sizing code execution - used only for debugging
purposes}	
HARDCOPY.SUB;5	{PATRAN required file if using the HCPLOT command}
LDST01.DIM;1	{Resulting dimension file for Eal mechanical loadset 1}
LDST01_03.DIM;1	{Resulting dimension file for Eal mechanical loadset 1 and thermal loadset 3}
LDST01_04.DIM;1	{Resulting dimension file for EAL mechanical loadset 1 and thermal loadset
4}	
LDST06.DIM;1	{Resulting dimension file for EAL mechanical loadset 6}
LDSTCOMB.DIM;6	{Succesively created combined loadset dimension -}
LDSTCOMB.DIM;5	{files, resulting from batch iterative sizing. -}
LDSTCOMB.DIM;4	{LDSTCOMB.DIM;1 is a copy of LDSTMIN.DIM used -}
LDSTCOMB.DIM;3	{to obtain an initial element stiffness matrix -}
LDSTCOMB.DIM;2	{for the first sizing iteration -}
LDSTCOMB.DIM;1	
LDSTMIN.DIM;1	{A minimum gage dimension file created by running EZDESIT for a ZERO
load	condition}
MATPROP.PRN;23	{The material property database}
PATLD01.OUT;3	{Eal calculated element loads for loadset 1}
PATLD06.OUT;1	{" " " " " " 6}
PATLDTH03.OUT;2	{" " " " " " 3,thermal}
PATLDTH04.OUT;1	{" " " " " " 4,thermal}
PATRAN.BKP;3	{Backup of PATRAN.DAT}
PATRAN.DAT;1	{The PATRAN database for this problem}
PATRAN.OUT;19	{The PATRAN neutral file, ASCII format}
UWT01_03.OUT;1	{Unit weight results file, same loadset -}
UWT01.OUT;1	{notation as used above for ldst#.dim}
UWT01_04.OUT;1	
UWT06.OUT;1	
UWTCOMB.OUT;2	{Unit weight results file, created after completion of iterative sizing by running EZDESIT option 8, sub option 1}
UWTMIN.OUT;1	{Unit weight results file, created before running the iterative sizing process
using	EZDESIT option 6 Zero load option}

Table C-2

PATRAN Loadsets

Element Pressure	Set 1 - Lift Load
Nodal Force	Set 1 - Landing Loads
Element Temperature	Set 1 - 72 deg F
Element Temperature	Set 2 - Hot Condition, No element gradients
Element Temperature	Set 3 - Torque Box Elements, temperature gradient 100 deg/in

PATRAN and Eal Constraint Sets

For Lift Load	constraint set 1
For Landing Load	constraint set 2

PATRAN Physical Property Data

Honeycomb	- Property set 1
Bars	- Property set 2
Sine Wave Webs	- Property set 3

PATRAN Nodal Temperature Sets, used to apply Nodal Weights

Rear Spar Systems	- Nodal Temperature Set 1
Front Spar Systems	- Nodal Temperature Set 2

Eal Loadsets

Lift Load	- APPL FORC 1
Landing Load Nodal Forces	- APPL FORC 2
Hot condition with no gradients	- APPL FORC 3
Hot condition with gradients on Torque Box	- APPL FORC 4
Landing Load, Inertial Acceleration	- APPL FORC 5
Design Landing Load Set, (2 and 5 combined)	- APPL FORC 6

EAL Loadset Constraint Set Combinations
for Static Solution Execution

Mechanical Loadset	Mechanical Constraint Set	Temperature Loadset	Temperature Constraint Set	*Temperature Set for Analysis
1	1	-	-	1
6	2	-	-	1
1	1	3	3	2
1	1	4	3	2

* Used In EAL to reduce element stiffnesses based on element temperature, and in the element sizing process to degrade allowables based on element temperature

TABLE C-3

EZDESIT Prompts and Responses
Working in the VMS Operating System
User Response is Underlined

dir/size {files required to begin the element sizing process}

Directory $DISK3:[CERRO.PATRAN.WINGEX]

EALINP.L20;1	12/12
EZBATCH.COM;16	1/3
EZBATCH.DAT;7	1/3
HARDCOPY.SUB;6	1/3
MATPROP.PRN;23	23/24
PATRAN.BKP;5	2236/2238
PATRAN.DAT;1	2236/2238
PATRAN.OUT;21	970/972

Total of 8 files, 5480 blocks.
EZDESIT
Welcome to the EZDESIT element sizing program

Block 2,
Figure C-1

Make sure caplock is on or the program
will not operate properly
Current Caveat List Includes but is not
restricted to the following:

1) At least 1 named component must exist
   or element temperatures won't be right
   No packets between 11 and 21 allowed
2) Fij stress recoverys are not used
3) packet 2 ADATA not allowed in patran
4) Current version is user hostile
   No typo's allowed
5) model must be in inches
6) nice to have the eal element ref frame
   for plates with x paralleling
   the vehicle length axis
7) note: pure thermal stress cases would
   result in minimum gage structure,
   as there is no mech. load to size for

Hit enter to continue

Wait, reading in file patran.out ...
Begin packet 1
End packet 1, Begin packet 2
End packet 2, Begin packet 3
End packet 3, Begin packet 4
End packet 4, Begin packet 5
End packet 5, Begin packet 6

End packet 6, Begin packet 7
End packet 7, Begin packet 8
End packet 8, Begin packet 10
End packet 10, Begin packet 11
End packet 11, Begin packet 21
Neutral file read in
Input number of E23, E24, E33 and E43 elements
364 0 53 397

Program EZDESIT

1 - Create an EAL Runstream through XQT DRSI

2 - Modify an Existing EAL Runstream

3 - Create a partial EAL Runstream
for applying/combining loads & constraints

4 - Update EAL element stiffness matrices

5 - Translate EAL results to Patran Results Files

6 - Run Element Sizing Code

7 - Create Worst Case Dimension File

8 - Review Calculated Component Weights

9 - Exit to System

1 {For creating the first portion of an EAL runstream}
Name your eal runstream, ex: test.eal
EALINP.COM
Do you wish to write header info? (Y/N)
Y
Input your default directory pathname
Ex: $DISK3:[ROOT.SUB1.SUB2]
$DISK3:[CERRO.PATRAN.WINGEX] {path to your working directory}
Do you wish to write start and title info.? (Y/N)
Y
Do you wish to write dummy MATC info.? (Y/N)
Y {Will be iteratively updated}
Do you wish to write dummy NSW info.? (Y/N)
Y {Element weight is accounted for via the NSW property in EAL}
Do you wish to write JLOC info.? (Y/N)
Y
Do you wish to write dummy BC, BD, & SA info.? (Y/N)
Y {Will be iteratively updated}
Do you wish to write CON info.? (Y/N)
Y
Input a patran constraint set to translate
1

What Constraint cas will this be in EAL?
1
Is there an associated symmetry plane?
N
Another Constraint Set Translation?
Y
Input a patran constraint set to translate
2
What Constraint cas will this be in EAL?
2
Is there an associated symmetry plane?
N
Another Constraint Set Translation?
N
Do you wish to set up a symmetry plane con case?
N
What set id do you want translated to mass data?
Input 0 for no rmass output
1
another set ?
Y
What set id do you want translated to mass data?
Input 0 for no rmass output
2
another set ?
N
Do you wish to write ELD info.? (Y/N)
Y {NSECT, NMAT, and NNSW EAL pointers will be iteratively updated}
Do you wish to write E info.? (Y/N)
Y
Any Resets Required for processor E?
N
Do you wish to Override the EAL geometry test defaults?
Y
Do you wish to override the EzEAL defaults?
Y
Input t1, t2, t3, t4, t5, t6, t7, t8
1.E-20 -.1 .1 .1 28. .01 .01 .01
Do you wish to write EKS info.? (Y/N)
Y
Do you wish to write SEQ info.? (Y/N)
Y
Do you wish to write TAN info.? (Y/N)
Y
Do you wish to write K info.? (Y/N)
Y
Do you wish to sum CEM and RMAS into MAS ? (Y/N)
Y
Do you wish to write DRSI info.? (Y/N)
Y
Input a constraint case for obtaining a constrained stiffness matrix, (DRSI)
1
Create Another Constrained Stiffness Matrix?

N {for iterative sizing only 1 constraint case need be}
{flagged, additional required constraint cases are}
{defined in file EZBATCH.DAT}

Program EZDESIT

1 - Create an EAL Runstream through XQT DRSI

2 - Modify an Existing EAL Runstream

3 - Create a partial EAL Runstream
for applying/combining loads & constraints

4 - Update EAL element stiffness matrices

5 - Translate EAL results to Patran Results Files

6 - Run Element Sizing Code

7 - Create Worst Case Dimension File

8 - Review Calculated Component Weights

9 - Exit to System

3 {to create the applied loading portion of an EAL runstream}
Input the name of your existing EAL runstream
EALINP.COM
Input the name for your new EAL runstream
LOADS.EAL

Run Stream Additions

1 - Add Loadsets from the Patran Database

2 - Apply Inertial Accelerations

3 - Combine Loadsets

4 - Add Static Solutions

5 - Generate Element load files

6 - Generate Nodal Displacement files

7 - End

1

Loadset Translation

1 - Translate Patran Force Loadsets (tbc)

2 - Translate Patran Pressure Loadsets

3 - Translate Patran Element Temperature Loadsets

4 - End Loadset Translation

1
Input a patran nodal force loadset
1
What will be the EAL nodal force loadset?
2
Input a loadset title for EAL
LANDING LOADS
Translate another patran nodal force loadset?
N

Loadset Translation

1 - Translate Patran Force Loadsets (tbc)

2 - Translate Patran Pressure Loadsets

3 - Translate Patran Element Temperature Loadsets

4 - End Loadset Translation

2
Which patran loadset is to be translated?
1
Which loadset will this be in EAL?
1
Input a title for this loadset
LIFT PRESSURE
Translate another pressure set ?
N

Loadset Translation

1 - Translate Patran Force Loadsets (tbc)

2 - Translate Patran Pressure Loadsets

3 - Translate Patran Element Temperature Loadsets

4 - End Loadset Translation

3
Input a patran average el. temperature loadset
0 for no average temperature
2
Input a stress free temperature to be subtracted
from the average element temperature
72.
Input a patran el. gradient temperature loadset
0 for no gradient
0
What will be the EAL temperature loadset?
3
Input a loadset title for EAL
AVG TEMP
Translate another patran temperature loadset?
Y
Input a patran average el. temperature loadset
0 for no average temperature
2
Input a stress free temperature to be subtracted
from the average element temperature
72.
Input a patran el. gradient temperature loadset
0 for no gradient
3
What will be the EAL temperature loadset?
4
Input a loadset title for EAL
AVG AND GRADIENT
Translate another patran temperature loadset?
N

Loadset Translation

1 - Translate Patran Force Loadsets (tbc)

2 - Translate Patran Pressure Loadsets

3 - Translate Patran Element Temperature Loadsets

4 - End Loadset Translation

4

Run Stream Additions

1 - Add Loadsets from the Patran Database

2 - Apply Inertial Accelerations

3 - Combine Loadsets

4 - Add Static Solutions

5 - Generate Element load files

6 - Generate Nodal Displacement files

7 - End

2
Input Joint Number to accelerate about
1
Input acceleration ,(1=1g)
-2.5 {Note: this is acceleration in the - Z direction}
Input an eal loadset identifier
5
Input a loadset title for EAL
LANDING ACCELERATION

Run Stream Additions

1 - Add Loadsets from the Patran Database

2 - Apply Inertial Accelerations

3 - Combine Loadsets

4 - Add Static Solutions

5 - Generate Element load files

6 - Generate Nodal Displacement files

7 - End

3
What will be the EAL loadset number of this combined loadset?
6 {combine loadsets 2 and 5 into 6, a hot lift condition}
How many loadsets do you wish to combine?
2
Input a loadset title for EAL
LANDING
Input a loadset number and factor
2 1.
Input a loadset number and factor

5 1,
Do you wish to create another loadset combination?
N

Run Stream Additions

1 - Add Loadsets from the Patran Database

2 - Apply Inertial Accelerations

3 - Combine Loadsets

4 - Add Static Solutions

5 - Generate Element load files

6 - Generate Nodal Displacement files

7 - End

4
Input the constraint case and loadset for this static solution
1 1 {As with DRSI, this acts as flag during iterative sizing}
{additonal loadset/constraint definitions are in file}
{EZBATCH.DAT}
Do you wish to perform another static solution?
N

Run Stream Additions

1 - Add Loadsets from the Patran Database

2 - Apply Inertial Accelerations

3 - Combine Loadsets

4 - Add Static Solutions

5 - Generate Element load files

6 - Generate Nodal Displacement files

7 - End

7

Program EZDESIT

1 - Create an EAL Runstream through XQT DRSI

2 - Modify an Existing EAL Runstream

3 - Create a partial EAL Runstream
for applying/combining loads & constraints

4 - Update EAL element stiffness matrices

5 - Translate EAL results to Patran Results Files

6 - Run Element Sizing Code

7 - Create Worst Case Dimension File

8 - Review Calculated Component Weights

9 - Exit to System

6 {Size elements for zero load first, this results in}
{a dimension file based on minimum gage data}
Do you want a summary file of element geometry ? (t/f)
T
Input material property database name
MATPROP.PRN
Input Loads file name, (type ZERO for no mechanical load case)
ZERO
Do you wish to superimpose a thermal load case ? (T/F)
F

Name your output file, ex: uwt(ldst).out
UWTMIN.OUT
Name your element dimension file, ex: LDST(ldst).dim
LDSTMIN.DIM

What PATRAN neutral file temperature set id
do you want to use for this analysis ?
1
Input title for results file
MINIMUM GAGE SIZING
Input First Subtitle for results file

Input Second Subtitle for results file

element 100
element 200
element 300
element 400
element 500
element 600
element 700
element 800
Do you wish to analyze another load case (y/n)?
N

Program EZDESIT

1 - Create an EAL Runstream through XQT DRSI

2 - Modify an Existing EAL Runstream

3 - Create a partial EAL Runstream
for applying/combining loads & constraints

4 - Update EAL element stiffness matrices

5 - Translate EAL results to Patran Results Files

6 - Run Element Sizing Code

7 - Create Worst Case Dimension File

8 - Review Calculated Component Weights

9 - Exit to System

4 {use the minimum gage dimension file to update}
{the initial dummy EAL element stiffnesses}
Input material property database name
MATPROP.PRN
Input dimension file name (ex: LDSTCOMB.DIM)
LDSTMIN.DIM
What PATRAN neutral file temperature set id
do you want to use for this analysis ?
1
element 100
element 200
element 300
element 400
element 500
element 600
element 700
element 800
Property compaction process...
Input the name of your existing EAL runstream
out name will be ealinp.com
EALINP.COM

Program EZDESIT

1 - Create an EAL Runstream through XQT DRSI

2 - Modify an Existing EAL Runstream

3 - Create a partial EAL Runstream
for applying/combining loads & constraints

4 - Update EAL element stiffness matrices

5 - Translate EAL results to Patran Results Files

6 - Run Element Sizing Code

7 - Create Worst Case Dimension File

8 - Review Calculated Component Weights

9 - Exit to System

9

DIR/SIZE {directory after the first execution of EZDESIT}

Block 3
Figure C-1

Directory $DISK3:[CERRO.PATRAN.WINGEX]

EALINP.COM;1	133/135	{New}
EALINP.L20;1	12/12	
EZBATCH.COM;16	1/3	
EZBATCH.DAT;7	1/3	
HARDCOPY.SUB;6	1/3	
LDSTMIN.DIM;1	113/114	{New}
LOADS.EAL;1	243/243	{New}
MATPROP.PRN;23	23/24	
PATRAN.BKP;5	2236/2238	
PATRAN.DAT;1	2236/2238	
PATRAN.OUT;21	970/972	
UWTMIN.OUT;1	138/138	{New}

Total of 14 files, 6492/6510 blocks.

{ NOW EDIT EALINP.COM TO REVISE HEADER INFORMATION AND }
{ INCLUDE LOADS.EAL, SAVE THE EDITED FILE AS THE NEW }
{ VERSION OF EALINP.COM }

{ COPY LDSTMIN.DIM TO LDSTCOMB.DIM, FOR ITERATIVE SIZING }
$ COPY LDSTMIN.DIM LDSTCOMB.DIM
{ SUBMIT THE ITERATIVE SIZING PROGRAM TO THE BATCH QUEUE }
$ SUBMIT/NOLOG/NOTIFY EZBATCH

DIRS {Directory after completion of EZBATCH}

Directory $DISK3:[CERRO.PATRAN.WINGEX]

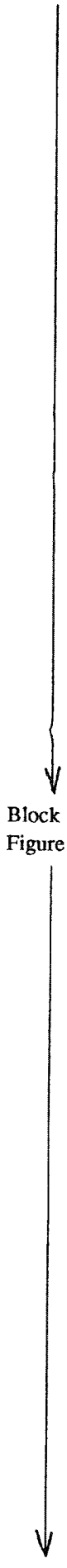
EALINP.;20	1218/1218	
EALINP.BKP;1	625	
EALINP.COM;1	961/963	
EALINP.L01;1	8402/8403	{New}
EALINP.L20;1	12/12	
EALINP.L30;1	8/9	{New}
EZBATCH.COM;16	1/3	
EZBATCH.DAT;8	1/3	
EZBATCH.OUT;1	37/39	{New}

```
HARDCOPY.SUB;6          1/3
LDST01.DIM;1          113/114      {New}
LDST01_03.DIM;1       113/114      {New}
LDST01_04.DIM;1       113/114      {New}
LDST06.DIM;1          113/114      {New}
LDSTCOMB.DIM;6          113/114    {New}
LDSTCOMB.DIM;5          113/114    {New}
LDSTCOMB.DIM;4          113/114    {New}
LDSTCOMB.DIM;3          113/114    {New}
LDSTCOMB.DIM;2          113/114    {New}
LDSTCOMB.DIM;1          113/114
LDSTMIN.DIM;1         113/114
MATPROP.PRN;23          23/24
PATLD01.OUT;3         156/156      {New}
PATLD01.OUT;2         156/156      {New}
PATLD01.OUT;1         156/156      {New}
PATLD06.OUT;1         156/156      {New}
PATLDTH03.OUT;1         156/156    {New}
PATLDTH04.OUT;1         156/156    {New}
PATRAN.BKP;5          2236/2238
PATRAN.DAT;1          2236/2238
PATRAN.OUT;21          970/972
UWT01.OUT;1           137/138      {New}
UWT01_03.OUT;1        137/138      {New}
UWT01_04.OUT;1        137/138      {New}
UWT06.OUT;1           137/138      {New}
UWTMIN.OUT;1           138/138
Total of 36 files, 18973/19008 blocks.
$ EZDESIT
Welcome to the EZDESIT element sizing program

     Make sure caplock is on or the program
     will not operate properly
     Current Caveat List Includes but is not
     restricted to the following:

        1) At least 1 named component must exist
           or element temperatures won't be right
           No packets between 11 and 21 allowed
        2) Fij stress recoverys are not used
        3) packet 2 ADATA not allowed in patran
        4) Current version is user hostile
              No typo's allowed
        5) model must be in inches
        6) nice to have the eal element ref frame
           for plates with x paralleling
           the vehicle length axis
        7) note: pure thermal stress cases would
           result in minimum gage structure,
           as there is no mech. load to size for

     Hit enter to continue
```



Block 4
Figure C-1

Wait, reading in file patran.out ...
Begin packet 1
End packet 1, Begin packet 2
End packet 2, Begin packet 3
End packet 3, Begin packet 4
End packet 4, Begin packet 5
End packet 5, Begin packet 6
End packet 6, Begin packet 7
End packet 7, Begin packet 8
End packet 8, Begin packet 10
End packet 10, Begin packet 11
End packet 11, Begin packet 21
Neutral file read in
Input number of E23, E24, E33 and E43 elements
364 0 53 397

Program EZDESIT

1 - Create an EAL Runstream through XQT DRSI

2 - Modify an Existing EAL Runstream

3 - Create a partial EAL Runstream
for applying/combining loads & constraints

4 - Update EAL element stiffness matrices

5 - Translate EAL results to Patran Results Files

6 - Run Element Sizing Code

7 - Create Worst Case Dimension File

8 - Review Calculated Component Weights

9 - Exit to System

8

1 - Create a Worst Case Element Weights File

2 - Review Component Weights

3 - End

1
How many load Files do you wish to sort ?
5
Input name of file # 1
UWTMIN.OUT
Input name of file # 2
UWT01.OUT

Input name of file # 3
UWT06.OUT
Input name of file # 4
UWT01_03.OUT
Input name of file # 5
UWT01_04.OUT
Name your output file
UWTCOMB.OUT

1 - Create a Worst Case Element Weights File

2 - Review Component Weights

3 - End

2
Input the file name to be post processed
(ex: UWTCOMB.OUT)
UWTCOMB.OUT
Weight Summarized by Failure Mode

Minimum Gage	447.
Panel Buckling	553.
Compresive yield	2295.
Yield	93.
Ultimate	335.
Fatigue	0.
Fracture Mechanics	0.
Creep	0.
Acoustic Fatigue	0.
Panel Flutter	0.
Thermal Stress Exceeded	0.

Total weight of E23 elements = 257.9705 Lb.
Average unit weight of E23 elements = 0.2497751 Lb/Ft

Total weight of E24 elements = 0.0000000E+00 Lb.
Average unit weight of E24 elements = 0.0000000E+00 Lb/Ft

Total weight of plate elements = 3464.810 Lb.
Average unit weight of plate elements = 1.239852 p.s.f.

How many load cases were summarized ?
5
Weight summarized by load case

Loadcase Number	1	447.
Loadcase Number	2	180.
Loadcase Number	3	1622.
Loadcase Number	4	413.
Loadcase Number	5	1061.

Do you wish to list weights for named components ? (T/F)
T

Input component name
MODEL
Weight for component MODEL is 3722.78 LBS.
Unit weight for bar elements is 0.24978lb/ft
for a bar length of 1032.81104 ft.
Unit weight for plate elements is 1.23985lb/sq ft
for a plate area of 2794.53491 sq. ft.
Another component ? (t/f)
T
Input component name
SPARS
Weight for component SPARS is 244.78 LBS.
Unit weight for plate elements is 0.54346lb/sq ft
for a plate area of 450.40601 sq. ft.
Another component ? (t/f)
T
Input component name
RIBS
Weight for component RIBS is 206.65 LBS.
Unit weight for plate elements is 0.46560lb/sq ft
for a plate area of 443.83682 sq. ft.
Another component ? (t/f)
T
Input component name
LS
Weight for component LS is 1570.10 LBS.
Unit weight for plate elements is 1.66504lb/sq ft
for a plate area of 942.98047 sq. ft.
Another component ? (t/f)
T
Input component name
US
Weight for component US is 1473.34 LBS.
Unit weight for plate elements is 1.46099lb/sq ft
for a plate area of 1008.45288 sq. ft.
Another component ? (t/f)
T
Input component name
TORQUEBOX
Weight for component TORQUEBOX is 1553.32 LBS.
Unit weight for bar elements is 0.23151lb/ft
for a bar length of 654.75360 ft.
Unit weight for plate elements is 1.67757lb/sq ft
for a plate area of 835.57593 sq. ft.
Another component ? (t/f)
T
Input component name
LGBAY
Weight for component LGBAY is 847.87 LBS.
Unit weight for bar elements is 0.42230lb/ft
for a bar length of 124.13235 ft.
Unit weight for plate elements is 2.17095lb/sq ft
for a plate area of 366.40527 sq. ft.

Another component ? (t/f)
T
Input component name
GLOVE
Weight for component GLOVE is 462.66 LBS.
Unit weight for bar elements is 0.22257lb/ft
for a bar length of 193.99196 ft.
Unit weight for plate elements is 1.48182lb/sq ft
for a plate area of 283.08737 sq. ft.
Another component ? (t/f)
F
end

1 - Create a Worst Case Element Weights File

2 - Review Component Weights

3 - End

3

Program EZDESIT

1 - Create an EAL Runstream through XQT DRSI

2 - Modify an Existing EAL Runstream

3 - Create a partial EAL Runstream
for applying/combining loads & constraints

4 - Update EAL element stiffness matrices

5 - Translate EAL results to Patran Results Files

6 - Run Element Sizing Code

7 - Create Worst Case Dimension File

8 - Review Calculated Component Weights

9 - Exit to System

9
TSB2$DIR/SIZE

Directory $DISK3:[CERRO.PATRAN.WINGEX]

EALINP.;20 1289

```
EALINP.BKP;2          625
EALINP.COM;1         1030
EALINP.L01;1         8412
EALINP.L20;2           12
EALINP.L30;1            8
EZBATCH.COM;16          1
EZBATCH.DAT;11          1
EZBATCH.OUT;1          42
HARDCOPY.SUB;6          1
LDST01.DIM;1          113
LDST01_03.DIM;1       113
LDST01_04.DIM;1       113
LDST06.DIM;1          113
LDSTCOMB.DIM;6          113
LDSTCOMB.DIM;5          113
LDSTCOMB.DIM;4          113
LDSTCOMB.DIM;3          113
LDSTCOMB.DIM;2          113
LDSTCOMB.DIM;1          113
LDSTMIN.DIM;1         113
MATPROP.PRN;23         23
PATLD01.OUT;3         156
PATLD01.OUT;2         156
PATLD01.OUT;1         156
PATLD06.OUT;1         156
PATLDTH03.OUT;1         156
PATLDTH04.OUT;1         156
PATRAN.BKP;1         2236
PATRAN.DAT;1         2236
PATRAN.OUT;21         970
UWT01.OUT;1           137
UWT01_03.OUT;1        137
UWT01_04.OUT;1        137
UWT06.OUT;1           137
UWTCOMB.OUT;1           156
UWTMIN.OUT;1          138

Total of 37 files, 19907 blocks.
TSB2$
```

Table C-4

EAL Skeleton Runstream for Batch Sizing

```
$DELETE EALINP.;*              #Add this line to your EAL runstream
$Delete *.l01;*               #"    "    "    "  "    "    "
$Delete *.l30;*               #"    "    "    "  "    "    "
$RUN $DISK1:[EAL]EAL312
'EALINP.
*ONLINE=0
*XQT TAB
START   339
TITLE 'WING TEST CASE
MATC
    1   10000000. 0.33 0.000 0.130687E-04 0.130687E-04  0.

  192    9450911. 0.33 0.000 0.132647E-04 0.132647E-04  0.
NSW
    1     0.01500

   74     0.00576
JLOC
      1            776.89789          -105.00000            22.29386

    339            633.34973          -122.66966             0.00000
MREF
FORMAT=2
    1    1    0.0    0.0    0.0
BC
    1     0.100

   15     0.236
SA
FORMAT=COUPLED
NMAT=1
    1  3.68       3.68       3.68      0.000E+00  0.000E+00  0.000E+00 >
   3.68       3.68       3.68       363.       363.       363.     >
   3.68       3.68       3.68      -363.      -363.      -363.
  0.236E+06  0.778E+05  0.236E+06  0.000E+00  0.000E+00  0.790E+05  0.000E+00 >
  0.000E+00  0.000E+00  0.402E+04  0.000E+00  0.000E+00  0.000E+00  0.133E+04 >
  0.402E+04  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.135E+04

NMAT=192
  135  2.30       2.30       2.30      0.000E+00  0.000E+00  0.000E+00 >
   2.30       2.30       2.30       241.       241.       241.     >
   2.30       2.30       2.30      -241.      -241.      -241.
  0.216E+06  0.714E+05  0.216E+06  0.000E+00  0.000E+00  0.725E+05  0.000E+00 >
  0.000E+00  0.000E+00  0.977E+04  0.000E+00  0.000E+00  0.000E+00  0.322E+04 >
  0.977E+04  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.327E+04
CON=  1
ZERO  2:     1

ZERO  6:   314
```

```
CON=  2
ZERO  2:      1

ZERO  6:   314
RMASS
*XQT ELD
E23
NMAT=    1: NSECT=      1: NNSW=    1
    1    3    0    0 #el no      1

E33
NMAT=   48: NSECT=      1: NNSW=   16
   24   73   22    0 #el no    365

E43
NMAT=   76: NSECT=      3: NNSW=   18
   65   67   68   66 #el no    418

*XQT E
T= -.100E-19  -0.10  0.100  0.100  28.0000  0.01  0.01  0.01
*XQT EKS
*XQT SEQ
*XQT TAN
*XQT K
*XQT M
*XQT AUS
MAS=SUM(CEM,RMAS)
*XQT DRSI
RESET CON=     1              # End of information created by ezdesit option 1
*XQT AUS                      # remaining portion created by ezdesit option 3
ALPHA: CASE TITL      2
1'LAND
SYSVEC: APPL FORC      2
J=      2:  0.00E+00 0.00E+00 -5381.  0.00E+00 0.000E+00 0.000E+00

J=    313:  0.00E+00 0.00E+00 -0.1925E+05 0.00E+00 0.000E+00 0.00E+00
!QLIB=1: !JNT=1: !SET=     2
*CALL (20 FSUM PROG)
*XQT AUS
ALPHA: CASE TITL      1
1'LIFT PRESSURE
ELDATA: PRES E33      1
G=1: E=    1:   -.29600E+01  -.29600E+01  -.29600E+01

ELDATA: PRES E43      1
G=1: E=   11:   -.20000E+01  -.20000E+01  -.20000E+01  -.20000E+01

G=1: E=  380:   -.20000E+01  -.20000E+01  -.20000E+01  -.20000E+01
*XQT EQNF
RESET SET=     1
*XQT DCU
CHANGE 1 EQNF FORC     1 1 APPL FORC     1 1
!QLIB=1: !JNT=1: !SET=     1
```

```
*CALL (20 FSUM PROG)
*XQT AUS
ALPHA: CASE TITL     3
1'TEMP HOT CASE
ELDATA: TEMP E23     3
G=1: E=   1:   212.342     0.000000E+00 0.000000E+00

G=1: E= 364:   228.000     0.000000E+00 0.000000E+00
ELDATA: TEMP E33     3
G=1: E=   1:   273.114      273.114      273.114      0.000000E+00

G=1: E=  53:   48.9259      48.9259      48.9259      0.000000E+00
ELDATA: TEMP E43     3
G=1: E=   1:   182.520      182.520      182.520      182.520      0.00E+00

G=1: E= 397:   180.079      180.079      180.079      180.079      0.00E+00
*XQT EQNF
RESET SET=     3
*XQT DCU
CHANGE 1 EQNF FORC     3 1 APPL FORC     3 1
!QLIB=1: !JNT=1: !SET=     3
*CALL (20 FSUM PROG)
*XQT AUS
ALPHA: CASE TITL     4
1'AVGANDGRAD
ELDATA: TEMP E23     4
G=1: E=   1:   212.342     0.000000E+00 0.000000E+00

G=1: E= 364:   228.000     0.000000E+00 0.000000E+00
ELDATA: TEMP E33     4
G=1: E=   1:   273.114      273.114      273.114      0.100000E-01

G=1: E=  53:   48.9259      48.9259      48.9259      0.100000E-01
ELDATA: TEMP E43     4
G=1: E=   1:   182.520      182.520      182.520      182.520      0.10E-01

G=1: E= 397:   180.079      180.079      180.079      180.079      0.10E-01
*XQT EQNF
RESET SET=     4
*XQT DCU
CHANGE 1 EQNF FORC     4 1 APPL FORC     4 1
!QLIB=1: !JNT=1: !SET=     4
*CALL (20 FSUM PROG)
*XQT SSOL
*XQT EXIT
```

Table C-5

Additional Data Files Required for Batch Sizing

File EZBATCH.DAT:

```
@ealinp.com                {name must be ealinp.com}
EALINP.                    {filename of eal library}
364 0 53 397               {nume23 nume24 nume33 nume43}
4                          {number of loadsets to size}
1 6 1 1                    {mechanical loadsets}
1 2 1 1                    {mech. loadset constraints}
0 0 1 1                    {Ithrmon(i), i=1,# loadsets}
0 0 3 4                    {temperature loadsets}
0 0 1 1                    {temp. loadset constraints}
1 1 2 2                    {temp. set for properties}
5                          {number of iterations allowed}
```

File EZBATCH.COM:

```
$SET DEFAULT $DISK3:[CERRO.PATRAN.WINGEX]
$ASSIGN EZBATCH.OUT SYS$OUTPUT
$run $disk3:[cerro.patran.transls.ezbatch]ezbatch
```

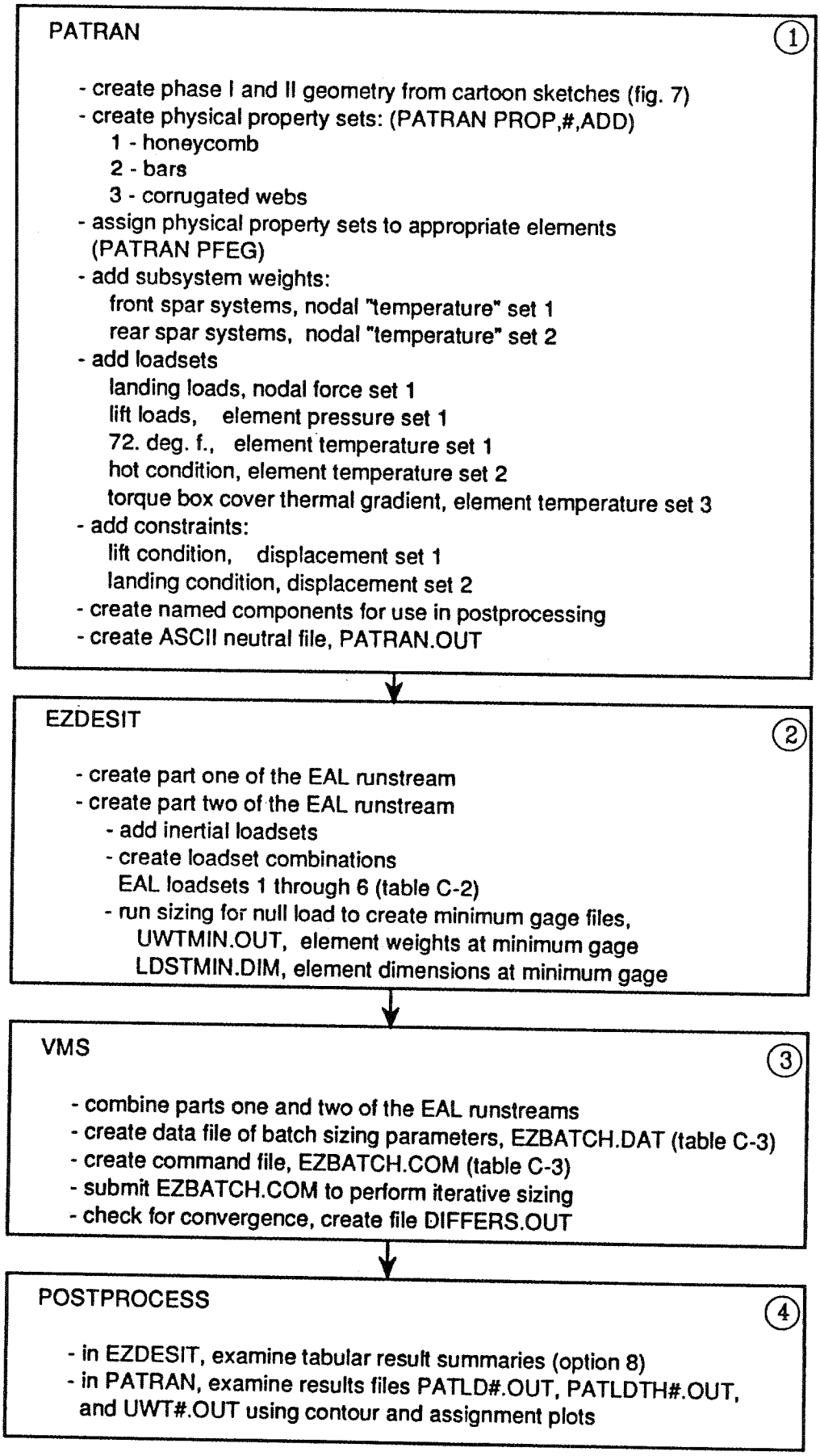


Figure C-1. Process Flow Chart

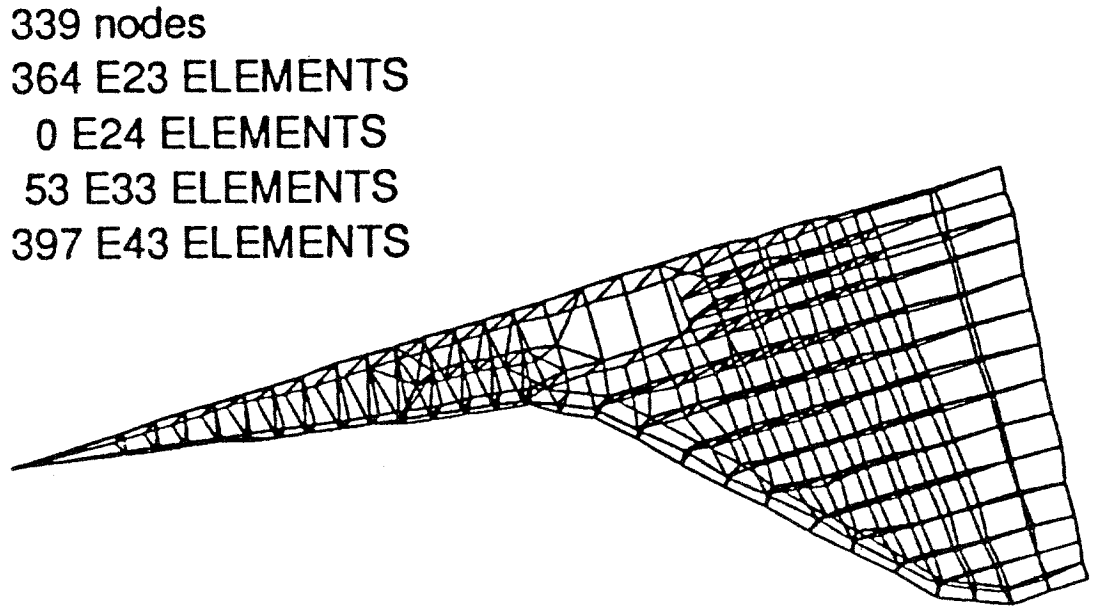


Figure C-2. - Wing finite element model

APPENDIX D

Using EZDESIT to Create a Structural Weight Database For Conceptual Design

Consider a vehicle design team given the mission to define a least gross weight single stage to orbit horizontal takeoff vehicle. Approaching this problem, the configurator has a relatively small number of mission fixed design paramaters. In this case they are the single stage to orbit and horizontal takeoff requirements. Using past knowledge the configurator will choose some other non varying parameters say, liquid hydrogen fuel and a turbo/ram/scram/rocket propulsion cycle. Structural parameters he might like to vary include fuselage cross section, fuselage aspect ratio, wing aspect ratio, wing thickness, and wingsweep. Sizes of all components will also have to vary to support the rubberized type sizing requirements of a vehicle performance analysis code. The following typical structural database requirements can be defined by the configurator.

Circular body	Lenticular body
10 < length/diameter < 20	1.5 < Width/Height < 3.0
10K < Volume < 100K	10 < Length/effective diameter < 20
	10K < Volume < 100K

Wave Rider	Delta Wing
50 < Le sweep < 65	1. < Aspect ratio < 3.
5 < Max Sect. W/H < 15	50 < Weight/Sexp < 200
10K < Volume < 100K	2000 < Sref < 10000

Given these database requirements, the structures group may proceed with component analyses. As an example EZDESIT could be used to produce the following parametric data, from structural analysis of a delta wing.

Aspect Ratio	Wing loading	Sref	Wetted Area Unit Weight
0.8	140	2500	2.0
0.8	140	5000	3.0
0.8	70	2500	1.5
0.8	70	5000	2.5
1.6	140	2500	2.5
1.6	140	5000	3.5
1.6	70	2500	2.0
1.6	70	5000	3.0

A vehicle sizing code could interrogate the component database using power law equations (typical at the conceptual design stage). For n independent variables at least 2 to the n data points would be required. That is for a wing with varying aspect ratio, wing loading, and plan area 2 to the third actural analyses are required. The Pascal routine GETWEIGHT, in table D-1, shows how the wing

database can be accessed by a call to procedure MULVARDBINTRP.

Typical results from Getweight are:

Input			Output, Wetted Area
Aspect Ratio	Wing loading	Sref	Unit Weight
0.8	140	2500	2.00
0.8	140	3750	2.54
1.6	60	2500	1.90
5.0	150	2500	3.65*
5.0	150	8000	5.30*

* (Input Aspect Ratio questionable)

Similar databases would have to be created for the various fuselage types and other structural pieces which will be subject to analysis. The performance group can then access a calculated weight structural database within the vehicle sizing code, which would also have data generated by the propulsion, aerodynamic, and subsystem groups.

Table D-1

Pascal Listing, Program GetWeight

```
Program getweight;
{for checking procedure mvardbin.pas, an include file}
Type
  Rvec6 = array[1..6] of real;
  Rvec64 = array[1..64] of real;
  rarray64b6 = array[1..64,1..6] of real;
Var
  x: rarray64b6;
  y: rvec64;
  n: integer;
  xin: rvec6;
  ans: real;
  ch: char;
  i: integer;

{function x to the y for integers}
function intfxtoy(x,y: integer): integer;
Var
  rx,ry,rans: real;
begin
  rx:=x;
  ry:=y;
  rans:=exp(ry*Ln(rx));
  intfxtoy:=trunc(rans);
end; {of function intfxtoy}

{$I mvardbin.pas}
Begin

 Repeat
   n:=3;
   x[1,1]:=0.8; x[1,2]:=140.; x[1,3]:=2500.;    y[1]:=2.0;
   x[2,1]:=0.8; x[2,2]:=140.; x[2,3]:=5000.;    y[2]:=3.0;
   x[3,1]:=0.8; x[3,2]:=70.;  x[3,3]:=2500.;    y[3]:=1.5;
   x[4,1]:=0.8; x[4,2]:=70.;  x[4,3]:=5000.;    y[4]:=2.5;
   x[5,1]:=1.6; x[5,2]:=140.; x[5,3]:=2500.;    y[5]:=2.5;
   x[6,1]:=1.6; x[6,2]:=140.; x[6,3]:=5000.;    y[6]:=3.5;
   x[7,1]:=1.6; x[7,2]:=70.; x[7,3]:=2500.;     y[7]:=2.0;
   x[8,1]:=1.6; x[8,2]:=70.; x[8,3]:=5000.;     y[8]:=3.0;
   For i:=1 to n Do
   Begin
     Writeln('Input x[',i,']');
     Readln(xin[i]);
   End; { of for}
   MulVardbIntrp(n,x,y,xin,ans);
   Writeln('ans = ',ans);
   Writeln('another case y/n?');
   readln(ch);
 Until upcase(ch) = 'N';
```

```
End.

      {file mvardbin.pas}
      Procedure Pwrlval(x1,y1,x2,y2,xin: real; Var yout: real);

{*********************************************************************}
{ input pair of x,y values and an x value interpolation pt            }
{ return yout at the value of xin based on a Power Law Interpolation}
{ requires function fxtoy in mathlib.pas                             }
{          J. A. Cerro     1/27/88                                    }
{*********************************************************************}

CONST
  e = 2.718281828;                {base of natural log}
VAR
  c,n:                        REAL; {calculated power law parameters}
BEGIN
   n:=ln(y2/y1)/ln(x2/x1);
   c:=ln(y2)-ln(x2)*n;
   c:=fxtoy(e,c);
   yout:=c*fxtoy(xin,n);
END; {of procedure Pwrlval}


Procedure MulVarDBIntrp(n: integer; x: rarray64b6; y: rvec64;
                        xin: rvec6;
                        Var ans: real);

{*********************************************************************}
{  Used to interpolate the multidimensionsal data                     }
{              y=f(x1,x2,x3...x to the n                              }
{  x is a 2 to the n by n dimensional array of independent var.       }
{   y is a vector of 2 to the n dependent variables                   }
{  n is the number of independent variables                           }
{   xin is a vector of n input independent variables which will be    }
{        interpolated upon by power law equations to arrive at        }
{         the answer ans                                              }
{  requires intfxtoy and pwrlval procedures                           }
{                    J. A. Cerro  1/27/88                             }
{*********************************************************************}

 Const
   jinc = 1; kinc = 2;
 Var
   i,j,k,iinc,it,m,p: integer;
   test1,test2: integer;
 Begin
   it:=0;
   Repeat
     it:=it+1;
     iinc:=intfxtoy(2,it);
     m:=n-it+1;
```

```
    p:=intfxtoy(2,it-1);
    i:=1-iinc; k:=-1; j:=1-jinc;
    Repeat
      i:=i+iinc; k:=k+kinc; j:=j+jinc;
      pwrlval(x[i,m],y[k],x[i+p,m],y[k+1],xin[m],y[j]);
      test1:=(intfxtoy(2,n)-intfxtoy(2,it)+1);
    Until i = test1;
  Until 1 = (intfxtoy(2,n)/intfxtoy(2,it));
  ans:=y[1];
End; {of MulVarDBIntrp}
```

Appendix E

Writing Your Own Element Sizing Code

EZDESIT and EZBATCH are modular, this allows users to write sizing codes for alternate element constructions. To code a new element sizing routine the following steps must be followed:

1) In subroutine SIZEL, (source file SIZEL.FOR) add the calling statement to your new element sizing routine. ex: if(int(pidd(3)).eq.15) Call MYELEMENT(...) Note, when setting up PATRAN property set fields for your new routine field 1 must be the material number reference, field 3 must be the element type indicator.

2) Write the subroutine for sizing your element type, include the object code in the linking procedures MAKEZDESIT.COM, and MAKEZBATCH.COM. In your element sizing routine the logical variable ICALCIJ will be checked to determine if element stiffnesses need to be calculated based on known dimensions (ICALCIJ = true), or the element must be sized based on known internal loads (ICALCIJ = false). Subroutine STIFFJ, reference 7, may be used to calculate plate element stiffness matrices. Bar and beam type elements use dimensions as opposed to element stiffnesses written to files as required respectively for E23 and E24 EAL element types. The parameter list of your element sizing code must return values for variables unitwt and indfm to subroutine SIZEL. Unitwt must be in pounds per foot for a two node element and pounds per square foot for a plate element. INDFM must be as defined in appendix B. Element dimensions must be written to unit 12 if ILIST is true, follow the logic of subroutine HCBNDTH.

3) Add logic to subroutine ELSITODIM which will read the formatted dimension file per your format statement in subroutine MYELEMENT.

4) To include your new element sizing routine in the batch process modify subroutine elsdimb in the same manner ELSITODIM was modified in Step 3.

5) Compile all modified source code, add your object module name to the command procedures MAKEZDESIT.COM and MAKEZBATCH.COM described in appendix F. Use these command procedures to create the new executable images for EZDESIT and EZBATCH.

As an example suppose we wish to have a sizing routine for a composite facesheet honeycomb sandwich element construction. Failure modes of minimum gage, plate buckling due to inplane loads, and ultimate strength are required design conditions. Various design default values for the element sizing routine must be assigned along with a material number and an indicator telling EZDESIT that a particular element is to be sized as composite honeycomb. The first step is to define a PATRAN physical property description that can be assigned to the new element type. The chosen convention for physical property descriptions of composite honeycomb elements follows.

field	generic name	description
1	matnum	material number, refer to file matprop.prn matnum is required to be in field 1
2	span	Panel Buckling Span, (in.)
3	itype	12 = composite honeycomb itype is required to be in field 3
4	factor	factor times input loads to obtain ultimate load
5	exptoth	panel buckling load knockdown factor (typically .8 empirical/theoretical ratio)
6	tlamina	lamina gage, in.
7	nof	non optimum factor, (1. = none)
8	ymred	youngs modulus reduction factor, (1.0 = full value in file matprop.prn)
9	alsred	allowable strength reduction factor, (1.0 = full value in file matprop.prn)
10	dummy	0
11	cormin	minimum core height, (in.)
12	cormaxh	maximum core height, (in.)
13	rhocor	core material density, (lb/in**3) (ex: aluminum= .1)
14	fibrangl	Angle from element reference frame x axis to laminate material x axis, (degrees) Input 999.0 to use PATRAN matang values.
15	dummy	0
16	dummy	0
17	dummy	0
18	dummy	0
19	dummy	0
20	dummy	0
21	dummy	0
22	dummy	0
23	dummy	0
24	dummy	0
25	dummy	0
26	yiefac	factor times ultimate load to obtain limit load level (ex: = .67 if using a 1.5 safety factor)
27	dummy	0
28	dummy	0
29	dummy	0
30	dummy	0
31	dummy	0
32	dummy	0
33	minnum0	min. number of 0 degree lamina, el. ref. frame
34	minnum90	min. number of 90 degree lamina, el. ref. frame
35	minnum45	min. number of 45 degree lamina, el. ref. frame, must be an even number.

The following element sizing routine follows the logic of routine hcbndth and works for sizing composite honeycomb plates. Various called subroutines are explained in Appendix F.

```
C Subroutine to size composite material honeycomb
C considers in plane and bending mechanical loads
C failure modes of min. gage,
C panel buckling and ultimate strength implemented
C no credit given for matrix strength or stiffness
C in directions other than along the fiber

Subroutine Hcmmc(indel,xl,yl,xyl,xm,ym,xym,xlth,ylth,xylth,xmth,
+                ymth,xymth,dmy,hin,nm0,nm90,nm45,uwt,indfm)

Dimension Pidd(40),rmpro(50),S(3),Cij(6,6),Fij(6,3),taray(100)
Dimension Fatsig(3,3),Fatlif(3,3),TCij(6,6),TFij(6,3),thetary(100)
Logical Ilist,Icalcij

Common /ComPidd/Pidd
Common /Alwbls/Rmpro,matype
Common /Comlog/Ilist,Icalcij
Common /Comcnt/nume23,nume24,nume33,nume43,num2n,num3n,num4n

!apply material knockdowns
rmpro2=rmpro(2)*pidd(8)
rmpro3=rmpro(3)*pidd(8)
rmpro4=rmpro(4)*pidd(8)
rmpro8=rmpro(8)*pidd(9)
rmpro9=rmpro(9)*pidd(9)
rmpro10=rmpro(10)*pidd(9)
rmpro11=rmpro(11)*pidd(9)
rmpro12=rmpro(12)*pidd(9)
rmpro13=rmpro(13)*pidd(9)
rmpro14=rmpro(14)*pidd(9)
rmpro15=rmpro(15)*pidd(9)
rmpro16=rmpro(16)*pidd(9)

!get fibrangl
fibrangl=pidd(14)
if (int(fibrangl).eq.999) Call getmatang(indel,fibrangl)

!Do stiffness or sizing depending on icalcij

If(icalcij) then
 Read(7,908) t1f,ht,num0,num90,num45
 nlayr1=num0+num90+num45
 Call datfil12(rmpro2,rmpro3,rmpro(5),rmpro4,rmpro(7),num0,num90,
+              num45,nlayr1,pidd(6),ht,TCij,TFij)
 uwt=t1f*2.*rmpro(7)+.02*ht*pidd(13)
  !transform thru fibrangl, material axis to e.r.f
  si=sind(-fibrangl)
  co=cosd(-fibrangl)
  call trnsfrmr(co,si,tcij,cij)
  call trnsfr2(co,si,tfij,fij)
  keal=indel-num2n
  Write(13,909) indel
  Write(13,901) keal,(fij(i,1),i=1,6)
  Write(13,902) (fij(i,2),i=1,6)
  write(13,903) (fij(i,3),i=1,6)
  Write(13,906) cij(1,1),cij(1,2),cij(2,2),cij(1,3),cij(2,3),cij(3
+,3),cij(1,4)
  Write(13,906) cij(2,4),cij(3,4),cij(4,4),cij(1,5),cij(2,5),cij(3
+,5),cij(4,5)
  Write(13,907) cij(5,5),(cij(i,6),i=1,6)
```

```
      Write(14,904) indel,uwt
      !Calculate composite alpha's note alph12 not used in EAL
      Do 8 I=1,nlayr1
    8 taray(i)=pidd(6) !to set up taray, equal thick lamina
      Call Layitup(num0,num90,num45,thetary)
      Call Calphi2(rmpro2,rmpro3,rmpro(5),rmpro(6),rmpro4,rmpro(41),rm
     +pro(42),nlayr1,taray,thetary,alph1,alph2,alph3)
      Write (16,905) indel,rmpro2,rmpro(5),rmpro(7),alph1,alph2,
     +                fibrangl
      Else

C Minimum gage setup
      ht=pidd(11)
      num0=pidd(33)
      num90=pidd(34)
      num45=pidd(35)
      t1f=(num0+num90+num45)*pidd(6)
      uwt=t1f*rmpro(7)*2.+ht*.02*pidd(13)
      indfm=1

      !Transform loads from element ref. frame to material axis
      !coordinate system and factor up to ultimate condition
      rm=cosd(fibrangl)
      rn=sind(fibrangl)
      call tsaita1(xl,yl,xyl,rm,rn,xlp,ylp,xylp)
      call tsaita1(xm,ym,xym,rm,rn,xmp,ymp,xymp)
      call tsaita1(xlth,ylth,xylth,rm,rn,xlthp,ylthp,xylthp)
      call tsaita1(xmth,ymth,xymth,rm,rn,xmthp,ymthp,xymthp)
      xl=xlp*pidd(4)
      yl=ylp*pidd(4)
      xyl=xylp*pidd(4)
      xm=xmp*pidd(4)
      ym=ymp*pidd(4)
      xym=xymp*pidd(4)
      xlth=xlthp
      ylth=ylthp
      xylth=xylthp
      xmth=xmthp
      ymth=ymthp
      xymth=xymthp

      !add thermal and mech for buckling check if the thermal loads
      !are detrimental, note that this
      !increases buckling resistance without increasing thermal
      !load which is anticonservative, however for now it is more
      !conservative than not including thermal load at all (comp case)
      xlb=.0
      ylb=.0
      xylb=.0
      if (xlth.lt..0) then
        xlb=xl+xlth
      else
        xlb=xl
      endif
      if (ylth.lt..0) then
        ylb=yl+ylth
      else
        ylb=yl
      endif
      if ((xyl*xylth).gt..0) then
        xylb=xyl+xylth
      else
        xylb=xyl
```

```
      endif

C Buckling check at ultimate load
      ctr0=.0
      ctr90=.0
      ctr45=.0
      t0b=num0*pidd(6)
      t90b=num90*pidd(6)
      t45b=num45*pidd(6)
      t1fb=t1f
      htb=ht
   10 Ind=1
      Do While (Ind.eq.1)
        vf0=t0b/t1f
        vf90=t90b/t1f
        vf45=t45b/t1f
        Call Whatsnu(rmpro(6),rmpro2,rmpro3,v12)
        Call Qonaxis(Rmpro2,Rmpro3,Rmpro4,Rmpro(6),v12,Qxx,Qyy,Qxy,Qss)
        Call Uonaxis(Qxx,Qyy,Qxy,Qss,U1,U2,U3,U4,U5)
        call astar(vf0,vf90,vf45,u1,u2,u3,u5,a11star,a22star,a66star)
        Riperin=t1fb*htb**2/2.
        aonb=1.1 !later add this data to pidd
        bona=1./aonb
        b=pidd(2)
        a=aonb*b
        if (int(xl).ge.0) then
          tmpxl=.0
          rnxcr=1.
        else
          call rt35c1a(aonb,b,A11Star,riperin,rmpro(6),rnxcr)
          if(int(rnxcr).eq.0) rnxcr=.000001
          tmpxl=.0-xl
        endif
        if (int(yl).ge.0) then
          tmpyl=.0
          rnycr=1.
        else
          call rt35c1a(bona,a,A22Star,riperin,v12,rnycr)
          if(int(rnycr).eq.0) rnycr=.000001
          tmpyl=.0-yl
        endif
        if (int(xyl).eq.0) then
          tmpxyl=.0
          rnxycr=1.
        else
          v=(rmpro(6)+v12)/2.
          e=abs(A66Star*2.*(1.+v))
          call rt35c4a(aonb,b,e,riperin,v,rnxycr)
          if(int(rnxycr).eq.0) rnxycr=.000001 !check these 3 ifs
          tmpxyl=xyl
        endif
        test=tmpxl/rnxcr/pidd(5)+tmpyl/rnycr/pidd(5)+(tmpxyl/rnxycr/pidd
     +(5))**2
        ind=1
        if(test.lt.1.) ind=0
        If (Ind.eq.1) then
          ind2=0
          totload=tmpxl+tmpyl+2.0*tmpxyl
          Do While(ind2.eq.0)
            ctr0=ctr0+tmpxl/totload*pidd(6)
            ctr90=ctr90+tmpyl/totload*pidd(6)
            ctr45=ctr45+tmpxyl/totload*pidd(6)
            If(ctr0.ge.pidd(6)) then
```

```
            t0b=t0b+pidd(6)
            ctr0=.0
            ind2=1
          Endif
          If(ctr90.ge.pidd(6)) then
            t90b=t90b+pidd(6)
            ctr90=.0
            ind2=1
          Endif
          If(ctr45.ge.pidd(6)) then
            t45b=t45b+pidd(6)
            ctr45=.0
            ind2=1
          Endif
        End Do
        num0b=max(nint(pidd(33)),nint(t0b/pidd(6)))
        num90b=max(nint(pidd(34)),nint(t90b/pidd(6)))
        num45b=max(nint(pidd(35)),nint(t45b/pidd(6)))
        if (mod(num45b,2).ne.0) num45b=num45b+1
        t1fb=t0b+t90b+t45b
        htb=4.*t1fb*Rmpro(7)/.02/Pidd(13)
        If(htb.lt.Pidd(11)) htb=Pidd(11)
        If(htb.gt.Pidd(12)) htb=Pidd(12)
      Endif
      uwtb=t1fb*rmpro(7)*2.+.02*htb*pidd(13)
      If(uwtb.gt.uwt) then
        ht=htb
        num0=num0b
        num45=num45b
        num90=num90b
        t1f=t1fb
        uwt=t1f*rmpro(7)*2.+.02*ht*pidd(13)
        Indfm=2
      Endif
    End Do

C Compressive yield check at limit load, indfm=3, not
C implemented for composite honeycomb

C Yield at Limit load level check, indfm=4, not implemented
C for composite honeycomb

C Ultimate Strength check
  60 htmin=ht
     !don't use flamoff here, let thermal loads
     !remain as calculated by EAL
     txlth=xlth
     tylth=ylth
     txylth=xylth
     txmth=xmth
     tymth=ymth
     txymth=xymth

     !reduce allowables by amount of thermal stress present from last
     !iteration, note the batch process is required so element dimensions
     !and thermal stress calculations can become consistent

     !upper surface first
     xlthcomb = txlth/2.+txmth/hin
     sigxth = xlthcomb/(nm0*pidd(6))
     if (sigxth .ge. .0) then
       f11tx = rmpro8-sigxth
       f11cx = rmpro9
```

```
else
  f11tx = rmpro8
  f11cx = rmpro9-sigxth
endif

ylthcomb = tylth/2.+tymth/hin
sigyth = ylthcomb/(nm90*pidd(6))
if (sigyth .ge. .0) then
  f11ty = rmpro8-sigyth
  f11cy = rmpro9
else
  f11ty = rmpro8
  f11cy = rmpro9-sigyth
endif

xylthcomb = abs(txylth/2. + txymth/hin)
sigxyth = xylthcomb/(nm45*pidd(6))
f11txy = rmpro8-sigxyth
f11cxy = rmpro9+sigxyth

!check to see if thermal stresses exceed allowables
!kick out if necessary and set indfm to 12
if (f11tx  .le. .0  .or.
+    f11cx  .ge. .0  .or.
+    f11ty  .le. .0  .or.
+    f11cy  .ge. .0  .or.
+    f11txy .le. .0  .or.
+    f11cxy .ge. .0      ) then
     uwt=99.0/pidd(7)/144.0
     indfm=12
     go to 80
endif

Call Ophccm2(xl/2.,yl/2.,xyl/2.,xm,ym,xym,htmin,pidd(12),pidd(6)
+                ,f11tx,f11cx,f11ty,f11cy,f11txy,f11cxy,
+                rmpro(7),pidd(13),num0u,num90u,num45u,htu)
if (num0u.lt.pidd(33)) num0u=pidd(33)
if (num90u.lt.pidd(34)) num90u=pidd(34)
if (num45u.lt.pidd(35)) num45u=pidd(35)
t1fu=(num0u+num90u+num45u)*pidd(6)
uwtu=t1fu*2.*rmpro(7)+.02*htu*pidd(13)
if (uwtu .gt. uwt) then
  num0=num0u
  num90=num90u
  num45=num45u
  ht=htu
  t1f=t1fu
  uwt=uwtu
  indfm=5
Endif


!now lower surface
xlthcomb = txlth/2.-txmth/hin
sigxth = xlthcomb/(nm0*pidd(6))
if (sigxth .ge. .0) then
  f11tx = rmpro8-sigxth
  f11cx = rmpro9
else
  f11tx = rmpro8
  f11cx = rmpro9-sigxth
endif
```

```
      ylthcomb = tylth/2.-tymth/hin
      sigyth = ylthcomb/(nm90*pidd(6))
      if (sigyth .ge. .0) then
        f11ty = rmpro8-sigyth
        f11cy = rmpro9
      else
        f11ty = rmpro8
        f11cy = rmpro9-sigyth
      endif

      xylthcomb = abs(txylth/2. - txymth/hin)
      sigxyth = xylthcomb/(nm45*pidd(6))
      f11txy = rmpro8-sigxyth
      f11cxy = rmpro9+sigxyth

      !check to see if thermal stresses exceed allowables
      !kick out if necessary and set indfm to 12
      if (f11tx  .le. .0  .or.
     +    f11cx  .ge. .0  .or.
     +    f11ty  .le. .0  .or.
     +    f11cy  .ge. .0  .or.
     +    f11txy .le. .0  .or.
     +    f11cxy .ge. .0      ) then
          uwt=99.0/pidd(7)/144.0
          indfm=12
          go to 80
      endif

      Call Ophccm2(xl/2.,yl/2.,xyl/2.,-xm,-ym,-xym,htmin,
     +             pidd(12),pidd(6)
     +               ,f11tx,f11cx,f11ty,f11cy,f11txy,f11cxy,
     +               rmpro(7),pidd(13),num0u,num90u,num45u,htu)
      if (num0u.lt.pidd(33)) num0u=pidd(33)
      if (num90u.lt.pidd(34)) num90u=pidd(34)
      if (num45u.lt.pidd(35)) num45u=pidd(35)
      t1fu=(num0u+num90u+num45u)*pidd(6)
      uwtu=t1fu*2.*rmpro(7)+.02*htu*pidd(13)
      if (uwtu .gt. uwt) then
        num0=num0u
        num90=num90u
        num45=num45u
        ht=htu
        t1f=t1fu
        uwt=uwtu
        indfm=5
      Endif


C Fatigue Life Check, indfm=6, not implemented for composite honeycomb

C Fracture Mechanics life check, indfm=7, not implemented

C Creep Life Check, indfm=8, not implemented

C Acoustic Fatigue Life Check, indfm=9, not implemented

C Panel Flutter Check, indfm=10, on hold
   80   uwt=uwt*pidd(7)*144.   !multipy by non-opt factor
        If(Ilist)Write(12,900) indel,t1f,ht,num0,num90,num45,uwt
      Endif !icalcij test
  900 Format('El ',i5,' t1f=',f6.3,' ht=',f6.2,' # of 0/90/45=',2(i2,'/'
     +),i2,' uwt=',f7.2)
  `01 Format (i5,6g11.4,' >')
```

```
902 Format (6g11.4,' >')
903 Format (6g11.4)
904 Format ('W(',i4,'):',f10.5)
905 Format (i5,f12.0,f5.2,f6.3,1X,f13.10,1X,F13.10,f6.0)
906 format (7g11.4,' >')
907 format (7g11.4)
908 Format (13x,f6.3,4x,f6.2,14x,2(i2,1x),i2,5x,f7.2)
909 format ('NMAT= ',i5)
    Return
    End
```

APPENDIX F

Distribution Tape Contents, Program Structure and Subroutine Documentation

The distribution tape for EZDESIT contains four savesets, UTIL1.SAV, EZDESIT.SAV, EZBATCH.SAV, and WINGEX.SAV. The files in these savesets should be uploaded to directories of the same respective filename but of course with the VAX filetype extension of DIR. The contents of WINGEX.DIR are discussed in appendix C. EZBATCH.DIR contains all source and executable code for the batch version of the EZDESIT program. EZDESIT.DIR contains all source and executable code for the menu driven interactive mode. UTIL1.DIR contains various math and strength of materials subroutines called by EZDESIT and EZBATCH. A command file titled Makezdesit.com exists in directory EZDESIT.DIR. MAKEZDESIT.COM, shown below, is used to create the executable file "EZDESIT.EXE" by linking the required object modules.

```
Link/debug/executable=ezdesit.exe -
 ezdesit.obj, elsizer.obj, sizel.obj, hcbndth.obj, corwebth.obj,-
 i2fm.obj,  e24iso.obj, compactor.obj,-
 eldtopld.obj,  edispdis.obj,-
 elsitodim.obj, dimtoeal.obj,-
 fndldst.obj,  datfils.obj,-
 elsipost.obj,  [cerro.util1]somsub.obj,-
 [CERRO.util1]mathlib.obj,-
 [CERRO.util1]stiffj.obj,-
 [CERRO.UTIL1]PATTOINT.OBJ
```

What follows is an outline structure of subroutine calls for the interactive version of the code EZDESIT, and a description of all subroutines.

Subroutine Call Hierarchy

```
EZEAL
  MAINMEN
    PATTOINT
    MENU1
    MENU2
    MENU3
      MEN3OP1
        MEN3OP1S1
        MEN3OP1S2
        MEN3OP1S3
      MEN3OP2
      MEN3OP3
      MEN3OP4
      MEN3OP5
      MEN3OP6
    MENU4
      DIMTOEAL
        GETPID
        GETPDAT
        GETELTEMP
        GETRMPRO
        SIZEL
          HCBNDTH*
          CORWEBTH*
          E24ISO*
          I2FM*
        GETNODES
        COMPACTOR
    MENU5
      ELDTOPLD
      EDISPDIS
    ELSIZER
      GETLOAD
      GETPID
      GETPDAT
      GETELTEMP
      GETRMPRO
      SIZEL
        HCBNDTH*
        CORWEBTH*
        E24ISO*
        I2FM*
    ELSITODIM
      GETPID
      GETPDAT
    MENU8
      FNDLDST
      ELSIPOST
        GETARLN
          AREATRI
```

*Element Sizing Routines

```
HCBNDTH
  RT35C1A
    INTRPL1
  RT35C4A
    INTRPL1
  OPTHC
  HCVM2TH
    CALCVM
    BISECT2
      CALCVM
    BISECT4
      CALCVM
  PRINCIP
  VONMISE
  FATINTR
    SEMILOG
    INTRPL1
  FMLIFE
    PRINCIP
    DADNCRV
  ACOUSFAT
  PFLUT
    FOFM
```

```
CORWEBTH
 RT1C19
 RT35C1A
   INTRPL1
 PRINCIP
 VONMISE
   FATINTR
     SEMILOG
   INTRPL1
```

```
  E24ISO
BEAMS1
  SECANT3
```

```
  I2FM
  PRINCIP
  VONMISE
FATINTR
 SEMILOG
   INTRPL1
```

AREATRI:
Source File Name: MATHLIB.FOR
Declaration: AreaTri(x1,y1,z1,x2,y2,z2,x3,y3,z3,area)
Input Parameters:
x1,y1,z1,x2,y2,z2,x3,y3,z3: real;
Spacial coordinates of triangle vertices
Output Parameters:
area: real; area of triangle in units as per input
Named Common: none
File Management: none
Calls: none
Description: From the three input spacial coordinates,
calculates the area of the connecting triangle.

ASTAR:
Source File Name: SOMSUB.FOR
Declaration Astar(vf0,vf90,vf45,U1,U2,U3,U5,A11Star,A22Star,
A66Star)
Input Parameters:
vf0: real; volume fraction of 0 degree plies
vf90: real; volume fraction of 90 degree plies
vf45: real; volume fraction of 45 degree plies
U1: real; Linear combination of On-Axis moduli
U2: real; Linear combination of On-Axis moduli
U3: real; Linear combination of On-Axis moduli
U5: real; Linear combination of On-Axis moduli
Output Parameters:
A11star: real; In-plane Stiffness
A22star: real; In-plane Stiffness
A66star: real; In-plane Stiffness
Named Common: none
File Management: none
Calls: none
Description: Reference 12, Table 7.2 page 7-8.
Determines in-plane stiffnesses for 0/90/45
degree laminates.

ASUBIJ:
Source File Name: SOMSUB.FOR
Declaration: Asubij(Ex,Ey,vx,vy,Es,numlay,taray,thetary,Aij)
Input Parameters:
Ex: real; Lamina modulus in material x direction
Ey: real; Lamina modulus in materila y direction
vx: real; Lamina major (larger) poissons ratio
vy: real; Lamina minor (smaller) poissons ratio
Es: real; Lamina shear modulus
numlay: integer; Number of laminae in the layup
taray: real(100); Array of laminae thicknesses
thetary: real(100); Array of laminae angles from
material x direction, (degrees).
Output Parameters:
Aij: real(3,3); In-plane laminate modulus, not normalized
Named Common: none
File Management: none
Calls: QONAXIS, TSAIT32
Description: Calculates A matrix for input laminate.
Reference 11, equation 4.14, page 119.

BEAMS1:
Source File Name: SOMSUB.FOR

Declaration: Beams1(p,v,rm,pth,vth,rmth,bmin,tmin,acmin,
rhoc,rhow,sigvma,ain,tin,bin,ac,bw,tw)
Input Parameters:
p: real; mechanical beam axial load (Lb.)
v: real; mechanical beam shear load (Lb.)
rm: real; mechanical beam bending moment (In. Lb.)
pth: real; thermal beam axial load (Lb.)
vth: real; thermal beam shear load (Lb.)
rmth: real; thermal beam bending moment (In. Lb.)
bmin: real; minimum allowable web height, (In.)
tmin: real; minimum allowable web thickness, (In.)
acmin: real; minimum allowable cap area, (Sq. In.)
rhoc: real; density of cap material, (Lb/In**3)
rhow: real; density of web material, (Lb/In**3)
sigvma: real; allowable Von-Mises stress, (psi)
ain: real; cap area used when thermal loads were
calculated, (sq in)
tin: real; web thickness used when thermal loads
were calculated, (in)
bin: real; web height used when thermal loads were
calculated, (in)
Output Parameters:
ac: real; cap area, (sq in)
bw: real; web height, (in)
tw: real; web thickness, (in)
Named Common: none
File Management: none
Calls: SECANT3
Description: Calculates planar beam optimum dimensions.
Considers mechanical and thermal loads.

BCKTST1:
Source File Name: SOMSUB.FOR
Declaration: BckTst1(a,b,aforny,Ex,Ey,Eforshear,v,xl,yl,xyl,
riperinx,riperiny,riperinxy,exptoth,test)
Input Parameters:
a: real; Panel length, x direction, (in)
b: real; Panel width, y direction, (in)
aforny: real; Panel length to be used for Ny load,
could be different than b if trying
to check a local phenomena, (in)
Ex: real; Youngs modulus, panel x direction, (psi)
Ey: real; Youngs modulus, pahel y direction, (psi)
Eforshear: real; Youngs modulus used when checking for
shear buckling, not a shear modulus, psi
v: real; poisons ratio
xl: real; x direction running load, (lb/in)
yl: real; y direction running load, (lb/in)
xyl: real; shear running load, (lb/in)
riperinx: real; moment of inertia per inch supporting xl,
(in**3)
riperiny: real; moment of inertia per inch supporting yl,
(in**3)
riperinxy: real; moment of inertia per inch supporting xyl,
(in**3)
exptoth: real; Knockdown factor on allowable moment supported
1.0 = full theoretical critical load supported
Output Parameters:
test: real; if test is less than 1. then buckling will
not occur, else panel will buckle.
Named Common: none
File Management:none
Calls: RT35C1A, RT35C4A

Description: Based on input panel in-plane loads and geometry, calculates the parameter "test" to determine if the given panel will buckle in a global manner. Uses roark equations for for buckling in the x, y and shear directions in an independent manner then combines the effects via an interaction equation: Rx + Ry +Rxy**2 = test. If test is less than 1 the panel will not buckle.

BCKTST2:

Source File Name: SOMSUB.FOR

Declaration: BckTst1(a,b,aforny,Ex,Ey,Eforshear,v,xl,yl,xyl, riperinx,riperiny,riperinxy,exptoth,test, rireqx,rireqy,rireqxy)

Input Parameters:

a: real; Panel length, x direction, (in)

b: real; Panel width, y direction, (in)

aforny: real; Panel length to be used for Ny load, could be different than b if trying to check a local phenomena, (in)

Ex: real; Youngs modulus, panel x direction, (psi)

Ey: real; Youngs modulus, pahel y direction, (psi)

Eforshear: real; Youngs modulus used when checking for shear buckling, not a shear modulus, psi

v: real; poisons ratio

xl: real; x direction running load, (lb/in)

yl: real; y direction running load, (lb/in)

xyl: real; shear running load, (lb/in)

riperinx: real; moment of inertia per inch supporting xl, (in**3)

riperiny: real; moment of inertia per inch supporting yl, (in**3)

riperinxy: real; moment of inertia per inch supporting xyl, (in**3)

exptoth: real; Knockdown factor on allowable moment supported 1.0 = full theoretical critical load supported

Output Parameters:

test: real; if test is less than 1. then buckling will not occur, else panel will buckle.

rireqx: real; approx. required moment of inertia, x

rireqy: real; approx. requried moment of inertia, y

rireqxy: real; approx. req. mom. of inertia in shear

Named Common: none

File Management:none

Calls: RT35C1A, RT35C4A

Description: Similar to BCKITST1 but the three additional returned parameters give an indication of required running moments of inertia required to preclude buckling.

BISECT2:

Source File Name: SOMSUB.FOR

Declaration Bisect2(xl,yl,xyl,sigvma,t,a,b,eps)

Input Parameters:

xl: real; x direction running load, (lb/in)

yl: real; y direction running load, (lb/in)

xyl: real; shear running load, (lb/in)

sigvma: real; allowable Von-Mises stress, (psi)

a: real; low guess at face sheet gage, (in)

b: real; high guess at face sheet gage, (in)

eps: real; numerical tolerance

Output Parameters:

t: real; calculated face sheet gage, (in)

Named Common: none

File Management: none

Calls: CALCVM
Description: Calculates honeycomb facesheet thickness that satisfies allowable von-mises stress constraint for the in-plane loading. Bisection root finding technique.

BISECT4:
Source File Name: SOMSUB.FOR
Declaration Bisect4(xl,yl,xyl,xm,ym,xym,xlth,ylth,xylth, xmth,ymth,xymth,sigvma,t,h,a,b,eps)
Input Parameters:
xl: real; mechanical x direction running load, (lb/in)
yl: real; mechanical y direction running load, (lb/in)
xyl: real; mechanical shear running load, (lb/in)
xm: real; mechanical x direction moment, (in-lb/in)
ym: real; mechanical y direction moment, (in-lb/in)
xym: real; mechanical shear moment, (in-lb/in)
xlth: real; thermal x direction running load, (lb/in)
ylth: real; thermal y direction running load, (lb/in)
xylth: real; thermal shear running load, (lb/in)
xmth: real; thermal x direction moment, (in-lb/in)
ymth: real; thermal y direction moment, (in-lb/in)
xymth: real; thermal shear moment, (in-lb/in)
sigvma: real; allowable Von-Mises stress, (psi)
h: real; height of honeycomb, (in)
a: real; low guess at face sheet thickness, (in)
b: real; high guess at face sheet thickness, (in)
eps: real; numerical tolerance
Output Parameters:
t: real; facesheet thickness, (in)
Named Common: none
File Management: none
Calls: CALCVM
Description: Calculates honeycomb facesheet thickness that satisfies allowable von mises stress constraint for in-plane plus bending mechanical and thermal loads. Bisection root finding technique.

CALCVM:
Source File Name: SOMSUB.FOR
Declaration: CalcVM(xl,yl,xyl,xm,ym,xym,t,h,sigvm)
Input Parameters:
xl: real; x direction running load, (lb/in)
yl: real; y direction running load, (lb/in)
xyl: real; shear running load, (lb/in)
xm: real; x direction running moment, (in-lb/in)
ym: real; y direction running moment, (in-lb/in)
xym: real; shear running moment, (in-lb/in)
t: real; facesheet thickness, (in)
h: real; honeycomb core height, (in)
Output Parameters:
sigvm: real; Von-Mises stress, (psi)
Named Common: none
File Management: none
Calls: none
Description: Calculates Von Mises stress in a honeycomb panel subject to in-plane and bending loads.

CALNUM:
Source File Name: SOMSUB.FOR
Declaration: CalNum(xl,F11t,F11c,tlamina,num)
Input Parameters:
xl: real; x direction running load, (lb/in)
F11t: real; lamina tensile ultimate strength, (psi)

F11c: real; lamina compressive ultimate strength, (psi)
tlamina: real; lamina thickness, (in)
Output Parameters:
num: integer; number of required lamina to support xl
Named Common: none
File Management: none
Calls: none
Description: Calculates the number of lamina required to support running load xl.

CALPHI2:
Source File Name: SOMSUB.FOR
Declaration: Calphi2(Ex,Ey,vx,vy,Es,ax,ay,numlay,taray, thetary,alph1,alph2,alph3)
Input Parameters:
Ex: real; Lamina modulus in material x direction
Ey: real; Lamina modulus in materila y direction
vx: real; Lamina major (larger) poissons ratio
vy: real; Lamina minor (smaller) poissons ratio
Es: real; Lamina shear modulus
ax: real; lamina coeff. of expansion, x direction
ay: real; lamina coeff. of expansion, y direction
numlay: integer; Number of laminae in the layup
taray: real(100); Array of laminae thicknesses
thetary: real(100); Array of laminae angles from material x direction, (degrees).
Output Parameters:
alph1: real; laminate coeff. of exp. 0 deg. direction
alph2: real; laminate coeff. of exp. 90 deg. direction
alph3: real; laminate in-plane shear coeff. of expansion
Named Common: none
File Management: none
Calls: ASUBIJ, TSAE365, QONAXIS,
Description: Based on reference 10, determines inplane laminate thermal expansion coefficients.

COMPACTOR:
Source File Name: COMPACTOR.FOR
Declaration: Compactor(NumE23,NumE24,NumE33,NumE43)
Input Parameters:
nume23: Integer; Number of EAL E23 type elements
nume24: Integer; Number of EAL E24 type elements
nume33: Integer; Number of EAL E33 type elements
nume43: Integer; Number of EAL E43 type elements
Output Parameters: none
Named Common: none
File Management: Opens matcfile.eal, swfile.eal, eldfile.eal, and cijfile.eal. File info is read into record structures and input files are closed and deleted. Compacted information is written to new files of the same four names.
Calls: none
Description: compacts material property, non structural weight, element definition, and element stiffness EAL files to eliminate duplicate information.

CORWEBTH:
Source File Name: CORWEBTH.FOR
Declaration: Corwebth(indel,xl,yl,xyl,xm,ym,xym,xlth,ylth,xylth, xmth,ymth,xymth,tin,unitwt,indfm)
Input Parameters:
indel: integer; element number

xl: real; mechanical x direction running load, (lb/in)
yl: real; mechanical y direction running load, (lb/in)
xyl: real; mechanical shear running load, (lb/in)
xm: real; dummy parameter, not utilized
ym: real; dummy parameter, not utilized
xym: real; dummy parameter, not utilized
xlth: real; thermal x direction running load, (lb/in)
ylth: real; thermal y direction running load, (lb/in)
xylth: real; thermal shear running load, (lb/in)
xmth: real; dummy parameter, not utilized
ymth: real; dummy parameter, not utilized
xymth: real; dummy parameter, not utilized
tin: real; thickness upon which thermal stress resultants were based, (in)

Output Parameters:
unitwt: real; panel unit weight, (lb/sq ft)
indfm: integer; failure mode indicator, see text pg. 15

Named Common: /compidd/ pidd: real(40)
/alwbls/ rmpro: real(50); matype: integer
/comlog/ ilist,icalcij: logical
/comcnt/ nume23,nume24,nume33,nume43,num2n, num3n,num4n: integer

File Management: none, but can write dimension info to unit 12.

Calls: RT1C19, RT35C1A, RT35C4A, PRINCIP, VONMISE

Description: An element sizing routine for isotropic material corugated web type elements with inplane mechanical loads, and inplane thermal loads. Sizes for mechanical loads reducing at temperature allowables by the amount of thermal stress present based on the input tin dimensions and thermal stress resultants. Euler type buckling is the only buckling failure mode. (need to check into a more appropriate shear buckling criteria, local). Yield checks at ultimate load for buckling sensitive panels, and at limit loads for all panels. Ultimate strength checks and yield strength checks include the thermal stress effects. When including thermal stress load cases the batch sizing process should be used to assure convergence of the element dimensions and resulting thermal stresses.

DATFIL2:

Source File Name: DATFILS.FOR

Declaration: DatFil2(E,v,G,rho,t,cij,fij)

Input Parameters:
E: real; Youngs modulus, (psi)
v: real; poissons ratio,
G: real; Shear modulus, (psi)
rho: real; material density, (lb/cu in)
t: real; thickness, (in)

Output Parameters:
cij: real(6,6); modulus matrix, ref. EAL
fij: real(6,3); stress recovery matrix, ref. EAL (not utilized)

Named Common: none

File Management: none

Calls: none

Description: calculates plate element EAL Cij matrix for corugated web elements. (temporary till STIFFJ routine is utilized)

DATFIL7:

Source File Name: DATFILS.FOR

Declaration: DatFil7(t1f,ht,Ex,Ey,v,Es,rho,cij,fij,wa)
Input Parameters:
t1f: real; facesheet thickness, (in)
ht: real; honeycomb core height, (in)
ex: real; Youngs modulus, (psi)
ey: real; Youngs modulus, (psi)
v: real; poissons ratio
es: real; Shear modulus, (psi)
rho: real; material density, (lb/cu in)
Output Parameters:
cij: real(6,6); modulus matrix, ref. EAL
fij: real(6,3); stress recovery matrix, ref. EAL
(not utilized)
wa: real; dummy parameter
Named Common: /comdf/ file: character(100)*80;
File Management: none
Calls: STIFFJ
Description: calculates plate element EAL Cij matrix for
isotropic honeycomb elements.

DATFIL12:
Source File Name: DATFILS.FOR
Declaration: DatFil12(Ex,Ey,v,Es,rho,num0,num90,num45,nlayr,
tlamina,ht,cij,fij)
Input Parameters:
Ex: real; lamina modulus x direction
Ey: real; lamina modulus y direction
v: real; lamina poissons ratio (major)
Es: real; lamina shear modulus
rho: real; density
num0: integer; number of 0 deg plys in one facesheet
num90: integer; number of 90 deg plys in one facesheet
num45: integer; number of 45 deg plys in one facesheet
numlayr: integer; number of plies in one facesheet
tlamina: real; lamina gage
ht: real; core height
Output Parameters:
cij: real(6,6); modulus matrix, ref. EAL
fij: real(6,3); stress recovery matrix, ref. EAL
(not utilized)
Named Common: /comdf/ file: character(100)*80;
File Management: none
Calls: STIFFJ
Description: calculates plate element EAL Cij matrix for
composite honeycomb elements.

DIMTOEAL:
Source File Name: DIMTOEAL.FOR
Declaration: DimToEal(NumE23p,NumE24p,NumE33p,NumE43p)
Input Parameters:
nume23p: integer; Number of EAL E23 elements
nume24p: integer; Number of EAL E24 elements
nume33p: integer; Number of EAL E33 elements
nume43p: integer; Number of EAL E43 elements
Output Parameters: none
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
/comlog/ ilist,icalcij: logical
/comcnt/ nume23,nume24,nume33,nume43,num2n,
num3n,num4n: integer
/compidd/pidd: real(40)
/alwbls/ rmpro: real(50); matype: integer
File Management: Opens a user input element dimension file as

unit 7. Opens a user input material properties
data file as unit 9. Also opens:
unit 13 as Cijfile.eal
unit 14 as SWfile.eal
unit 15 as eldfile.eal
unit 16 as Matcfile.eal
Closes 7,9,13 t 16 upon completion leaving
the files required for compactor to sort
through and compress.

Calls: SIZEL,GETPID,GETPDAT,GETELTEMP,GETRMPRO,GETNODES,COMPACTOR

Description: Basically the same code as Elsizer but with the sizing routines taken out and dimensions coming from the input dimension file. Icalcij is true so Element stiffness matrices, unit weights, connectivities and material prop (matc) information is written to files. Done by calls to datfil# procedures and subsequent calls to STIFFJ (reference 7).

EDISPDIS:

Source File Name: EDISPDIS.FOR

Declaration: EdisPdis(numnod)

Input Parameters:

numnod: integer; Number of nodes in the finite element model

Output Parameters: none

Named Common: none

File Management: Opens a user input file of EAL nodal displacements (must have been printed via dcu print), also opens a user input file for the PATRAN nodal displacement results file to be output. Closes both files upon completion.

Calls: none

Description: The user input EAL nodal displacements are translated to PATRAN nodal displacement results file format.

ELDTOPLD:

Source File Name: ELDTOPLD.FOR

Declaration: EldToPld(NumE23,NumE24,NumE33,NumE43)

Input Parameters:

nume23: integer; Number of EAL E23 elements
nume24: integer; Number of EAL E24 elements
nume33: integer; Number of EAL E33 elements
nume43: integer; Number of EAL E43 elements

Output Parameters: none

Named Common: none

File Management: Opens a user input file of EAL element loads (must have been printed via dcu print), also opens a user input file for the PATRAN element results file to be output. Closes both files upon completion.

Calls: none

Description: The user input element loads file is translated to PATRAN element results file format.

ELSIPOST:

Source File Name: ELSIPOST.FOR

Declaration: ElsiPost(NumE23,NumE24,NumE33,NumE43)

Input Parameters:

nume23: integer; Number of EAL E23 elements
nume24: integer; Number of EAL E24 elements
nume33: integer; Number of EAL E33 elements
nume43: integer; Number of EAL E43 elements

Output Parameters: none
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File Management: Opens and closes a user input file of summarized unit weights, (typically output from fndldst).
Calls: GETARLN
Description: Summarizes unit weights times element lengths and areas. Summary printed to screen by failure mode loadset and component. Lengths and areas determined from dimensions in the PATRAN database.

ELSITODIM:
Source File Name: ELSITODIM.FOR
Declaration: ElsiToDim(NumE23,NumE24,NumE33,NumE43)
Input Parameters:
nume23: integer; Number of EAL E23 elements
nume24: integer; Number of EAL E24 elements
nume33: integer; Number of EAL E33 elements
nume43: integer; Number of EAL E43 elements
Output Parameters: none
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File Management: Asks for a number of loadset dimension files, (ldst*.dim typ.). Opens these files as units n+11. Opens an output file named 'LDSTCOMB.DIM' as logical unit 7. Sorts through each loadset dimension file for each element finding the heaviest element and writing that size to a new file called ldstcomb.dim. Closes 7,8 and the n+11 files upon completion.
Calls: GETPID,GETPDAT
Description: See file management statement.

ELSIZER:
Source File Name: ELSIZER.FOR
Declaration: Elsizer(NumE23,NumE24,NumE33,NumE43)
Input Parameters:
nume23: integer; Number of EAL E23 elements
nume24: integer; Number of EAL E24 elements
nume33: integer; Number of EAL E33 elements
nume43: integer; Number of EAL E43 elements
Output Parameters: none
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
/comlog/ ilist,icalcij: logical
/compidd/ pidd: real(40)
/alwbls/ rmpro: real(50); matype: integer
File Management: Opens user input element loads files as unit 7, a user input material properties data file as unit 9, and a user input name unit weight output file as unit 11. Optionally opens an output element dimension file as unit 12, and if including thermal stresses in the element sizing, an input element dimension file (based on a previous sizing iteration or possibly the minimum gage dimension file) as unit 13. Closes units 7,9,11,12, and 13 upon completion.
Calls: GETLOAD, SIZEL, GETPID, GETPDAT, GETELTEMP, GETRMPRO
Description: Interactively prompts for a stress resultant file name to size elements for, and if including thermal stresses, a stress resultant file from the

thermal load case analysis. Also prompts for the names of output weight and dimension files and the material datafile to be used for mechanical properties. An associated PATRAN element temperature set to be used as average element temperatures for obtaining allowables is interactively input. Calls routine SIZEL to size each element and write info to the weight and dimension files. Prompts for reexecution to size another load case.

EQUIVT:
Source File Name: SOMSUB.FOR
Declaration: Equivt(t1f,ht,eqt)
Input Parameters:
t1f: real; facesheet thickness, (in)
ht: real; honeycomb core height, (in)
Output Parameters:
eqt: real; equivalent inertia based thickness, (in)
Named Common: none
File Management: none
Calls: none
Description: Returns equivalent thickness to get the inertia of a piece of honeycomb with dimensions t1f and ht.

EZDESIT:
Source File Name: EZDESIT.FOR
Declaration: EzDesit
Input Parameters: none
Output Parameters: none
Named Common: none
File Management: none
Calls: MAINMEN
Description: Main program name, starts the interactive codes by calling MAINMEN. Lists some program restrictions and has section of known problems listed as comments.

E24ISO:
Source File Name: E24ISO.FOR
Declaration: E24Iso(indel,p,v,rm,pth,vth,rmth,ain,tin,bin, unitwt,indfm)
Input Parameters:
indel: integer; Element number
p: real; mechanical axial force, (lb)
v: real; mechanical shear force, (lb)
rm: real; mechanical bending moment, (in lb)
pth: real; thermal axial force, (lb)
vth: real; thermal shear force, (lb)
rmth: real; thermal bending moment, (in lb)
ain: real; cap area upon which thermal forces were calculated, (sq in)
tin: real; web thickness upon which thermal forces were calculated, (in)
bin: real; web height upon which thermal forces were calculated, (in)
Output Parameters:
unitwt: real; panel unit weight, (lb/sq ft)
indfm: integer; failure mode indicator, see text pg. 15
Named Common: /compidd/ pidd: real(40)
/alwbls/ rmpro: real(50); matype: integer
/comlog/ ilist,icalcij: logical
/comcnt/ nume23,nume24,nume33,nume43,num2n,

num3n,num4n: integer
File Management: none, but can write dimension info to unit 12.
Calls: BEAMS1, LOWLOAD
Description: Mechanical loads, and thermal loads are used. Sizes for mechanical loads reducing at temperature allowables by the amount of thermal stress present based on the input ain, tin, and bin dimensions and thermal stress resultants. No Stability checks are performed. Sizes only for a limit load yield criterion. When including thermal stress load cases the batch sizing process should be used to assure convergence of the element dimensions and resulting thermal stresses.

FLAMOFF:
Source File Name: SOMSUB.FOR
Declaration: Flamoff(xl,yl,xyl,xm,ym,xym,xlth,ylth,xylth, xmth,ymth,xymth,txlth,tylth,txylth, txmth,tymth,txymth)
Input Parameters:
xl: real; mechanical x direction running load, (lb/in)
yl: real; mechanical y direction running load, (lb/in)
xyl: real; mechanical shear running load, (lb/in)
xm: real; mechanical x dir. running moment, (in lb/in)
ym: real; mechanical y dir. running moment, (in lb/in)
xym: real; mechanical shear running moment, (in lb/in)
xlth: real; thermal x direction running load, (lb/in)
ylth: real; thermal y direction running load, (lb/in)
xylth: real; thermal shear running load, (lb/in)
xmth: real; thermal x dir. running moment, (in lb/in)
ymth: real; thermal y dir. running moment, (in lb/in)
xymth: real; thermal shear running moment, (in lb/in)
Output Parameters:
txlth: real; thermal x direction running load, (lb/in)
tylth: real; thermal y direction running load, (lb/in)
txylth: real; thermal shear running load, (lb/in)
txmth: real; thermal x dir. running moment, (in lb/in)
tymth: real; thermal y dir. running moment, (in lb/in)
txymth: real; thermal shear running moment, (in lb/in)
Named Common: none
File Management: none
Calls: none
Description: sets thermal loads to zero if not of same sign as mechanical loads. Not considered a good practice if thermal stresses dominate mechanical stresses.

FNDLDST:
Source File Name: FNDLDST.FOR
Declaration: FndLdst(NumE23,NumE24,NumE33,NumE43)
Input Parameters:
nume23: integer; Number of EAL E23 elements
nume24: integer; Number of EAL E24 elements
nume33: integer; Number of EAL E33 elements
nume43: integer; Number of EAL E43 elements
Output Parameters: none
Named Common: none
File Management: Opens user input unit weight files output from elsizer as a sequential number of units starting with 7. Also opens an output file as the next higher unit No. Closes all Units before returning control to calling routine.
Calls: none
Description: Looks at the input unit weight files and produces

a similarly structured output file consisting of data
from the heaviest element of the input files.

GETARLN:
Source File Name: PATTOINT.FOR
Declaration: GetArLn(NumE23,NumE24,NumE33,NumE43,id,area)
Input Parameters:
nume23: integer; Number of EAL E23 elements
nume24: integer; Number of EAL E24 elements
nume33: integer; Number of EAL E33 elements
nume43: integer; Number of EAL E43 elements
id: integer; element number
Output Parameters: area: real
Named common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File Management: none
Calls: areatri
Description: Uses the PATRAN neutral file and the input element
id No. to get the area (3 and 4 node elements) or
length (2 node elements) of the input element.

GETBMANG:
Source File Name: PATTOINT.FOR
Declaration: Getbmang(indel,x1,x2,x3)
Input Parameters:
indel: integer; PATRAN el. id. no.
Output Parameters: x1,x2,x3: real; coordinates of third point
for beam reference
Named common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File Management: none
Calls: none
Description: Returns the PATRAN neutral file beam orientation
third point coordinates for the input indel el. no.

GETELTEMP:
Source File Name: PATTOINT.FOR
Declaration: GetElTemp(idel,itsid,tempdf)
Input Parameters:
idel: integer; PATRAN element id number
itsid: integer; PATRAN element temp. set id
Output Parameters: tempdf: real; element temperature
Named common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File Management: none
Calls: areatri
Description: reads PATRAN element temperatures from the
neutral file, must be in the form packet 11
followed directly by packet 21. returns -999.
if a problem occurs.

GETLOAD:
Source File Name: SIZEL.FOR
Declaration: GetLoad(numel,aryload)
Input Parameters:
numel: integer; Number of elements in the fea model
Output Parameters:
aryload: real(2500,6); matrix of 6 load components for
up to 2500 elements
Named Common: none
File management: none but reads unit 7 a PATRAN element results
file of element internal loads.
Calls: none
Description: Stores the loads from file 7 into the array

aryload(i,j) where i=element id no. and j = 1
through 6 load component.

GETMATANG:
Source File Name: PATTOINT.FOR
Declaration: GetMatang(indel,fibrangl)
Input Parameters:
indel: integer; PATRAN element id
Output Parameters:
fibrangl: real; PATRAN value of matang, degrees
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File management: none
Calls: none
Description: Returns the PATRAN neutral file matang, defining
material axis frames for plate elements

GETNODES:
Source File Name: PATTOINT.FOR
Declaration: GetNodes(id,j1,j2,j3,j4)
Input Parameters:
id: integer; PATRAN element id
Output Parameters:
j1,j2,j3,j4: integer; nodes defining element connectivity
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File management: none
Calls: none
Description: Returns element connectivity for the indicated element
j1,j2: for 2 node elements; j1,j2,j3 for three; etc.

GETPDAT:
Source File Name: PATTOINT.FOR
Declaration: GetPdat(ipid,field)
Input Parameters:
ipid: integer; id number of a PATRAN property set
Output Parameters:
field: real(40); values associated with above property set
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File management: none
Calls: none
Description: Reads patfile to obtain the field value associated
with filed number ifield of property set ipid.

GETPID:
Source File Name: PATTOINT.FOR
Declaration: GetPid(indel,ipid)
Input Parameters:
indel: integer; element number
Output Parameters:
ipid: integer; property set number assigned to
above element
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File management: none
Calls: none
Description: Reads patfile to obtain the property id number
assigned to element indel.

GETRMPRO:
Source File Name: PATTOINT.FOR

Declaration: GetRmpro(lun,matnum,tempdf,rmpro)
Input Parameters:
lun: integer; logical unit no. of material data file
matnum: integer; material id no. in material data file
tempdf: real; temperature to use for interpolation
Output Parameters:
rmpro: real(50); array containing interpolated material properties, defined in appendix A
Named Common: none
File management: none
Calls: Intrpl1
Description: Reads lun (material property datafile) to find temperatures for material matnum which surround temperature tempdf, then linearly interpolates to get properties at temperature tempdf.

HCBNDTH:
Source File Name: HCBNDTH.FOR
Declaration: HcBndTh(indel,xl,yl,xyl,xm,ym,xym,xlth,ylth,xylth, xmth,ymth,xymth,tin,hin,unitwt,indfm)
Input Parameters:
indel: integer; element number
xl: real; mechanical x direction running load, (lb/in)
yl: real; mechanical y direction running load, (lb/in)
xyl: real; mechanical shear running load, (lb/in)
xm: real; mechanical x direction moment, (in-lb/in)
ym: real; mechanical y direction moment, (in-lb/in)
xym: real; mechanical shear moment, (in-lb/in)
xlth: real; thermal x direction running load, (lb/in)
ylth: real; thermal y direction running load, (lb/in)
xylth: real; thermal shear running load, (lb/in)
xmth: real; thermal x direction moment, (in-lb/in)
ymth: real; thermal y direction moment, (in-lb/in)
xymth: real; thermal shear moment, (in-lb/in)
tin: real; thickness upon which thermal stress resultants were based, (in)
hin: real; honeycomb core height upon which thermal stress resultants were calculated, (in)
Output Parameters:
unitwt: real; panel unit weight, (lb/sq ft)
indfm: integer; failure mode indicator, see text pg. 15
Named Common: /compidd/ pidd: real(40)
/alwbls/ rmpro: real(50); matype: integer
/comlog/ ilist,icalcij: logical
/comcnt/ nume23,nume24,nume33,nume43,num2n, num3n,num4n: integer
File Management: none
Calls: RT35C1A, RT35C4A, OPTHC, HCVM2TH, PRINCIP, VONMISE, LOWLOAD,
(FATINTR, FMLIFE, ACOUSFAT, PFLUT, not currently)
Description: An element sizing routine for isotropic material honeycomb elements with inplane and bending mechanical loads, and inplane and bending thermal loads. If icalcij is true routine datfil7 is called to calculate element stiffnesses based on dimensions read from logical unit 7. Writes EAL formatted data for MATC, NSW, and SA procedures to units 16, 14, and 13 respectively. Alternately, if icalcij is false, new element dimensions are calculated. The element is sized for mechanical loads reducing at temperature allowables by the amount of thermal stress present based on the input tin, hin dimensions and thermal stress resultants. Buckling checks for in-plane load

resultants only. Euler type buckling is the only buckling failure mode. Yield checks at ultimate load for buckling sensitive panels, and at limit loads for all panels. Ultimate strength checks and yield strength checks include the thermal stress effects. When including thermal stress load cases the batch sizing process should be used to assure convergence of the element dimensions and resulting thermal stresses.

HCVM2TH:
Source File Name: SOMSUB.FOR
Declaration: HcVm2th(xl,yl,xyl,xm,ym,xym,xlth,ylth,xylth,
xmth,ymth,xymth,tming,rhof,rhoc,vc,reqi,
sigvma,corminh,cormaxh,tin,hin,t,h)
Input Parameters:
xl: real; mechanical x direction running load, (lb/in)
yl: real; mechanical y direction running load, (lb/in)
xyl: real; mechanical shear running load, (lb/in)
xm: real; mechanical x direction moment, (in-lb/in)
ym: real; mechanical y direction moment, (in-lb/in)
xym: real; mechanical shear moment, (in-lb/in)
xlth: real; thermal x direction running load, (lb/in)
ylth: real; thermal y direction running load, (lb/in)
xylth: real; thermal shear running load, (lb/in)
xmth: real; thermal x direction moment, (in-lb/in)
ymth: real; thermal y direction moment, (in-lb/in)
xymth: real; thermal shear moment, (in-lb/in)
tming: real; minimum facesheet thickness, (in)
rhof: real; density of facesheet material, (lb/cu in)
rhoc: real; density of core material, (lb/cu in)
vc: real; core density factor, (0.02 = 2% core)
reqi: real; required moment of inertia, (in**3)
sigvma: real; allowable Von-Mises stress, (psi)
corminh: real; minimum core height, (in)
cormaxh: real; maximum core height, (in)
tin: real; thickness upon which thermal stress resultants were based, (in)
hin: real; honeycomb core height upon which thermal stress resultants were calculated, (in)
Output Parameters:
t: real; facesheet thickness, (in)
h: real; honeycomb core height, (in)
Named Common: none
File Management: none
Calls: CALCVM, BISECT2, BISECT4
Description: Calculates optimum dimensions for a honeycomb panel subject to inplane loads, bending loads, thermal loads, gage constraints, and inertia requirements.

IBSS:
Source File Name: SOMSUB.FOR
Declaration: IBss(tsh,sp,tst,ht,riperin)
Input Parameters: tsk,sp,tst,ht: real;
Output Parameters: riperin: real;
Named Common: none
File Management: none
Calls: none
Description: Returns moment of inertia for a blade stiffened panel. (Not currently utilized)

INCLUP:

Source File Name: SOMSUB.FOR

Declaration: IncLup(tmpxl,tmpyl,tmpxyl,tlamina,ctr0,ctr90,
ctr45,num0,num90,num45)

Input Parameters:

tmpxl: real; x direction running load, (lb/in)
tmpyl: real; y direction running load, (lb/in)
tmpxyl: real; shear running load, (lb/in)
tlamina: real; thickness of a lamina, (in)
ctr0: real; proportion of 0 degree load to total load
ctr90: real; proportion of 90 degree load to total load
ctr45: real; proportion of shear load to total load

Output Parameters:

ctr0: real; proportion of 0 degree load to total load
ctr90: real; proportion of 90 degree load to total load
ctr45: real; proportion of shear load to total load
num0: real; number of 0 degree laminae
num90: real; number of 90 degree laminae
num45: integer; number of +- 45 degree laminae

Named Common: none

File Management: none

Calls: none

Description: Input facesheet inplane running loads, and a running proportion of current fraction that each lamina is towards being fully incremented to a new full lamina thickness. Used by calling program to increment lamina gages in a proportion matching the inplane load magnitudes. Ctr0, ctr90, and ctr45 are within a sizing loop in the calling program.

INTRPL1:

Source File Name: MATHLIB.FOR

Declaration: Intrpl1(x1,y1,x2,y2,x,y)

Input Parameters:

x1: real; first independent value
y1: real; first dependent value
x2: real; second independent value
y2: real; second dependent value
x: real; requested independent value

Output Parameters:

y: real; calculated dependent value

Named Common: none

File Management: none

Calls: none

Description: Used to linearly interpolate between points (x1,y1) and (x2,y2), x is the independent variable.

I2FM:

Source File Name: I2FM.FOR

Declaration: I2fm(indel,xl,xlth,ain,unitwt,indfm)

Input Parameters:

indel: integer; element number
xl: real; mechanical axial force, (lb)
xlth: real; thermal axial force, (lb)
ain: real; area upon which xlth was calculated, (sq in)

Output Parameters:

unitwt: real; bar unit weight, (lb/ft)
indfm: integer; failure mode indicator, see text pg. 15

Named Common: /compidd/ pidd: real(40)
/alwbls/ rmpro: real(50); matype: integer
/comlog/ ilist,icalcij: logical

/comcnt/ nume23,nume24,nume33,nume43,num2n,
num3n,num4n: integer
File Management: none, but can write dimension info to unit 12.
Calls: PRINCIP, VONMISE, LOWLOAD
Description: If icalcij is true EAL formatted data for MATC, NSW, and BC procedures are written to units 16, 14, and 13 respectively. Alternately (icalcij = false) calculates element sizes for isotropic material bar (EAL E23 type) elements. Includes mechanical and thermal loads. Sizes for mechanical loads reducing at temperature allowables by the amount of thermal stress present based on the input tin, dimensions and thermal stress resultants. No stabality checks are performed. Ultimate strength checks and yield strength checks include the thermal stress effects. When including thermal stress load cases the batch sizing process should be used to assure convergence of the element dimensions and resulting thermal stresses.

LAYITUP:
Source File Name: SOMSUB.FOR
Declaration: LayItUp(num0,num90,num45,theta)
Input Parameters:
num0: integer; number of 0 degree plies in the laminate
num90: integer; number of 90 degree plies in the laminate
num45: integer; number of +-45 degree plies in the laminate
Output Parameters:
theta: real(100); array of ply orientations, (deg)
Named Common: none
File Management: none
Calls: none
Description: knowing the number of 0, 90, and +-45 degree lamina in a laminate, layitup will output an array of lamina angles, balanced symmetric when possible.

LOWLOAD:
Source File Name: Sizel.for
Declaration: LowLoad(a,b,c,d,e,f,p,q,r,s,t,u,imin)
Input Parameters:
a,b,c,d,e,f,p,q,r,s,t,u: real; load values
Output Parameters:
imin: integer; low load indicator
Named Common: none
File Management: none
Calls: none
Description: if (a,b,c,d,e,f,p,q,r,s,t, and u) are less than the value 2, imin is returned as 1 typicaly meaning the 6 input loads are sufficiently low enough to permit minimum gage element design. Else imin is set to 0.

MAINMEN:
Source File Name: EZDESIT.FOR
Declaration: MainMen
Input parameters: none
Output parameters: none
Named common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File Management: Opens and closes patran.out as unit 8, while open the entire file is read into patfile via

routine pattoint. If patran.out is not found
execution continues with a warning.

Calls: PATTOINT, MENU1, MENU3, MENU4, MENU5, ELSIZER, ELSITODIM, MENU8

Description: Displays the main menu, options 1 through 9, routes control based on interactive input of an option #.

MENU1:

Source File Name: EZDESIT.FOR

Declaration: Menu1(NumE23,NumE24,NumE33,NumE43)

Input parameters:

nume23: integer; number of EAL E23 elements
nume24: integer; number of EAL E24 elements
nume33: integer; number of EAL E33 elements
nume43: integer; number of EAL E43 elements

Output parameters: none

Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)

File Management: Opens a file as unit 7, this file is the name of an EAL input runstream to be created. Closes unit 7 before the return.

Calls: none

Description: Prompts for interactive input of directory default, and EAL runstream information through processor DRSI. Information is written to the file opened as unit 7. Dummy MATC properties are written to be upgraded by other routines, similarly for NSW, MREF, BC, BD, and SA. PATRAN constraint cases are translated, the option for symmetry in the EAL runstream is prompted for. PATRAN nodal temperature sets may be translated to EAL RMASS data. Processor E resets are prompted for and the EZDESIT program has defaults which are more lax than EAL if desired. Note, the option to sum RMASS and CEM matrices should be chosen even if no RMASS data is present, just to maintain default EAL runstream requirements.

MENU3:

Source File Name: EZDESIT.FOR

Declaration: Menu3(NumNode,NumE23,NumE24,NumE33,NumE43)

Input Parameters:

numnode: integer; number of nodes in the fe model
nume23: integer; number of EAL E23 elements
nume24: integer; number of EAL E24 elements
nume33: integer; number of EAL E33 elements
nume43: integer; number of EAL E43 elements

Output Parameters: none

Named Common: none

File Management: Opens user input file names for old and new EAL runstreams as units 8 and 9 respectively. Also closes units 8 and 9.

Calls: MEN3OP1, MEN3OP2, MEN3OP3, MEN3OP4, MEN3OP5, MEN3OP6

Description: Displays menu 3 and prompts for an old EAL runstream to get header info from and a new EAL runstream to write info to. Routes processing to the appropriate routine base on the user input menu choice.

MENU4:

Source File Name: EZDESIT.FOR

Declaration: Menu4(NumE23,NumE24,NumE33,NumE43)

Input Parameters:

nume23: integer; number of EAL E23 elements

nume24: integer; number of EAL E24 elements
nume33: integer; number of EAL E33 elements
nume43: integer; number of EAL E43 elements

Output Parameters: none
Named Common: none
File Management: Opens a user input file of an existing EAL runstream as unit 13. Opens a user input file of a new EAL runstream as unit 8. Opens working files cijfile.eal, swfile.eal, matcfile.eal, and eldfile.eal as units 9, 10, 11, and 12 respectively. Units 8 and 13 are closed before the return and units 9 through 12 are closed and deleted before the return.
Calls: DIMTOEAL,
Description: uses routine dimtoeal to convert dimension files to new element stiffness matrices based on an input temperature set for material property interpolation. Updates the new EAL runstream appropriately. Also updates NMAT, NSECT, and NSW indicators in ELD, and creates appropriate data for these indicators.

MENU5:
Source File Name: EZDESIT.FOR
Declaration: Menu5(Numnode,NumE23,NumE24,NumE33,NumE43)
Input Parameters:
numnode: integer; number of nodes in the fe model
nume23: integer; number of EAL E23 elements
nume24: integer; number of EAL E24 elements
nume33: integer; number of EAL E33 elements
nume43: integer; number of EAL E43 elements

Output Parameters: none
Named Common: none
File Management: none
Calls: ELDTOPLD, EDISPDIS
Description: Displays Menu 5 for translating EAL force and displacement files to PATRAN results file format. Routes processing to the appropriate routine based on the interactive input menu choice.

MENU8:
Source File Name: EZDESIT.FOR
Declaration: Menu8(NumE23,NumE24,NumE33,NumE43)
Input Parameters:
nume23: integer; number of EAL E23 elements
nume24: integer; number of EAL E24 elements
nume33: integer; number of EAL E33 elements
nume43: integer; number of EAL E43 elements

Output Parameters: none
Named Common: none
File Management: none
Calls: FNDLDST, ELSIPOST
Description: Displays Menu 8 for creating worst case element weight files and summarizing component weights. Routes processing to the appropriate routine based on the interactive input menu choice.

MEN3OP1:
Source File Name: EZDESIT.FOR
Declaration: Men3Op1(Numnode,NumE23,NumE24,NumE33,NumE43)
Input parameters:
numnode: integer; number of nodes in the fe model
nume23: integer; number of EAL E23 elements

nume24: integer; number of EAL E24 elements
nume33: integer; number of EAL E33 elements
nume43: integer; number of EAL E43 elements
Output parameters: none
Named Common: none
File Management: none
Calls: M3OP1S1, M3OP1S2, M3OP1S3
Description: Displays the Submenu from Main Menu option 3, loadset translation. Routes processing to the appropriate routine based on an interactive user menu choice.

MEN3OP2:
Source File Name: EZDESIT.FOR
Declaration: Men3Op2
Input parameters: none
Output parameters: none
Named Common: none
File Management: none, but writes to unit 9.
Calls: none
Description: Writes the EAL runstream info to apply inertial acceleration. Limited as it only applys z direction acceleration. Could easily be upgraded to permit applied accelerations and rotations in all 6 d.o.f.

MEN3OP3:
Source File Name: EZDESIT.FOR
Declaration: Men3Op3
Input Parameters: none
Output Parameters: none
Named Common: none
File Management: none, but writes to unit 9.
Calls: none
Description: Writes the EAL runstream information to combine applied force loadsets. Used to define a loadset for element sizing which is a combination of various other mechanical loadsets.

MEN3OP4:
Source File Name: EZDESIT.FOR
Declaration; Men3Op4
Input Parameters: none
Output Parameters: none
Named Common: none
File Management: none, but writes to unit 9.
Calls: none
Description: Writes the EAL runstream information for processor SSOL asking for appropriate load and constraint set id's.

MEN3OP5:
Source File Name: EZDESIT.FOR
Declaration: Men3Op5(NumE23,NumE24,NumE33,NumE43)
Input Parameters:
nume23: integer; number of EAL E23 elements
nume24: integer; number of EAL E24 elements
nume33: integer; number of EAL E33 elements
nume43: integer; number of EAL E43 elements
Output Parameters: none
Named Common: none
File Management: none, but writes to unit 9.
Calls: none
Description: Writes the EAL runstream information to create

and print load resultant files.

MEN3OP6:
Source File Name: EZDESIT.FOR
Declaration: Men3Op6
Input Parameters: none
Output Parameters: none
Named Common: none
File Management: none, but writes to unit 9.
Calls: none
Description: Writes the EAL runstream information to
print displacement files.

MREFY:
Source File Name: MREFY.FOR
Declaration: Mrefy(nume23,nume24,nume33,nume43,numnods)
Input Parameters:
nume23: integer; number of EAL E23 elements
nume24: integer; number of EAL E24 elements
nume33: integer; number of EAL E33 elements
nume43: integer; number of EAL E43 elements
numnode: integer; number of nodes in the fe model
Output Parameters: none
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File Management: Opens existing file ealinp.com as unit 13,
and temp.eal (a new file) as unit 8. Closes
both files upon completion.
Calls: GETPID,GETPDAT,GETBMANG,NEWTST,LIB$RENAME_FILE
Description: Uses information in patfile to update ealinp.com by
adding MREF information, (EAL format 2). File temp.eal
is the working copy of the updated EAL runstream.
Temp.eal is renamed to ealinp.com before exiting the
subroutine.

M3OP1S1:
Source File Name: EZDESIT.FOR
Declaration: M3Op1S1(Numnode,NumE23,NumE24,NumE33,NumE43)
Input Parameters:
numnode: integer; number of nodes in the fe model
nume23: integer; number of EAL E23 elements
nume24: integer; number of EAL E24 elements
nume33: integer; number of EAL E33 elements
nume43: integer; number of EAL E43 elements
Output Parameters: none
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File Management: None, but writing is done to unit 9.
Calls: none
Description: Used to translate PATRAN nodal force data to EAL
runstream format. Output is written to unit 9.
Interactive prompting is done for loadset, title,
etc.

M3OP1S2:
Source File Name: EZDESIT.FOR
Declaration: M3Op1S2(NumE23,NumE24,NumE33,NumE43)
Input Parameters:
nume23: integer; number of EAL E23 elements
nume24: integer; number of EAL E24 elements
nume33: integer; number of EAL E33 elements
nume43: integer; number of EAL E43 elements
Output Parameters: none
Named Common: /compnf/ patfile: character(30000)*80

kntreca: integer(45)
File Management: None, but writing is done to unit 9.
Calls: none
Description: Used to translate PATRAN distributed force data to EAL runstream format. Output is written to unit 9. Interactive prompting is done for loadset, title, etc.

M3OP1S3:
Source File Name: EZDESIT.FOR
Declaration: M3Op1S3(NumE23,NumE24,NumE33,NumE43)
Input Parameters:
nume23: integer; number of EAL E23 elements
nume24: integer; number of EAL E24 elements
nume33: integer; number of EAL E33 elements
nume43: integer; number of EAL E43 elements
Output Parameters: none
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File Management: None, but writing is done to unit 9.
Calls: none
Description: Used to translate PATRAN element temperature data to EAL runstream format. Output is written to unit 9. Interactive prompting is done for loadset, title, etc. Options are programed to allow use of one PATRAN element temperature set for average EAL average element temperature, and another for the appropriate gradient. Both average element temperature and gradient may be input as 0. Also the stress free temperature may be subtracted from the average element temperature, required for thermal stress work.

NEWTST:
Source File Name: MREFY.FOR
Declaration: Newtst(mref,numrefs,x1,x2,x3,inew)
Input Parameters:
mref: 500 element array structure: corresponds to and EAL line of MREF data for the format = 2 option
k: integer; an EAL id. no.
il: integer; an EAL axis orientation indicator
x1,x2,x3: real; coordinates of EAL third point
numrefs: integer; number of reference frames defined in the passed in mref structure
x1,x2,x3: real; third point coordinates
Output Parameters: inew: integer; returns MREF indicator number or -1 if the third point indicator is new
Named Common: none
File Management: none
Calls: none
Description: Checks to see if the input third point has allready been used to define an mref record in the mref structure, if so, returns the indicator number, if not returns a -1

OPTHC:
Source File Name: SOMSUB.FOR:
Declaration: OptHc(tmin,corminh,cormaxh,rhoc,rhof,ri,t1f,ht)
Input Parameters:
tmin: real; minimum facesheet thickness, (in)
corminh: real; minimum core height, (in)
cormaxh: real; maximum core height, (in)
rhoc: real; density of core material, (lb/cu in)

(ex: 0.1 = aluminum)
rhof: real; density of facesheet material, (lb/cu in)
ri: real; required moment of inertia, (in**3)
Output Parameters:
t1f: real; facesheet thickness, (in)
ht: real; core height, (in)
Named Common: none
File Management: none
Calls: none
Description: Calculates honeycomb panel optimum dimensions subject to minimum gage and inertia constraints.

OPTHCCOM:
Source File Name: SOMSUB.FOR:
Declaration: OptHcCom(xl,yl,xyl,xm,ym,xym,hmin,hmax,tlamina, F11t,F11c,rhof,rhoc,num0,num90,num45,hfinal)
Input Parameters:
xl: real; x direction running load, (lb/in)
yl: real; y direction running load, (lb/in)
xyl: real; shear running load, (lb/in)
xm: real; x direction moment, (in-lb/in)
ym: real; y direction moment, (in-lb/in)
xym: real; shearing moment, (in-lb/in)
hmin: real; minimum core height, (in)
hmax: real; maximum core height, (in)
tlamina: real; lamina thickness, (in)
F11t: real; lamina longitudinal tensile ult. strength, (psi)
F11c: real; lamina transverse tensile ult. strength, (psi)
rhof: real; facesheet material density, (lb/cu in)
rhoc: real; core material density, (lb/cu in)
(ex: 0.1 = aluminum)
Output Parameters:
num0: integer; number of 0 degree lamina
num90: integer; number of 90 degree lamina
num45: integer; number of +- 45 degree lamina
hfinal: real; core height, (in)
Named Common: none
File Management: none
Calls: CALNUM
Description: Sizes a composite honeycomb panel to be lightest weight for a given set of mechanical inplane and bending loads based on ultimate strength of the lamina.

PATTOINT:
Source File Name: PATTOINT.FOR
Declaration: PatToInt(lun)
Input Parameters:
lun: integer; logical unit number of PATRAN neutral file
Output Parameters: none
Named Common: /compnf/ patfile: character(30000)*80
kntreca: integer(45)
File Management: Reads from PATRAN neutral file lun.
writes results to internal file patfile.
Calls: none
Description: Used to put PATRAN neutral file information into system memory so it can be randomly accessed without disk read/writes. Kntreca, keeps track of which record begins a PATRAN packet type, ie: kntreca(2) equals the record number of the first line of packet 2 information, etc. etc. Currently somewhat limited as no ADATA can exist in packet 2, and constraint cases can not have more than 5 constraint conditions per node.

PRINCIP:
Source File Name: SOMSUB.FOR:
Declaration: Princip(sigx,sigy,sigxy,S)
Input Parameters:
sigx: real; x direction stress
sigy: real; y direction stress
sigxy: real; shear stress
Output Parameters:
S: real(3); in-plane principal stresses 1 and 2
position 3 holds Taumax
Named Common: none
File Management: none
Calls: none
Description: Calculates in-plane principal stresses, S(3)
used to hold Taumax.

PSANGLE:
Source File Name: SOMSUB.FOR:
Declaration: Psangle(sx,sy,sxy,theta)
Input Parameters:
sx: real; x direction stress
sy: real; y direction stress
sxy: real; shear stress
Output Parameters:
theta: real; angle from principal stress axis to sx
direction, (degrees)
position 3 holds Taumax
Named Common: none
File Management: none
Calls: none
Description: calculates theta per above definition, ref.
"Advanced Strength and Applied Stress Analysis",
Budynas, p83

QOFFAXIS:
Source File Name: SOMSUB.FOR:
Declaration: QoffAxis(Ex,Ey,vx,vy,Es,theta,j,k,Qjk)
Input Parameters:
Ex: real; lamina longitudinal youngs modulus, (psi)
Ey: real; lamina transverse youngs modulus, (psi)
vx: real; major (larger) poissons ratio
vy: real; minor (smaller) poissons ratio
Es: real; lamina longitudinal shear modulus, (psi)
theta: real; off axis angle, (degrees)
j: integer; Qjk subscript
k: integer; Qjk subscript
Output Parameters:
Qjk: real; value of Q (off axis stiffness)
for array component (j,k)
Named Common: none
File Management: none
Calls: TSAIT32
Description: Determines off axis orthotropic modulus constants
from lamina engineering constants. Uses TSAIT32.

QONAXIS:
Source File Name: SOMSUB.FOR:
Declaration: QonAxis(Ex,Ey,Es,vx,vy,Qxx,Qyy,Qxy,Qss)
Input Parameters:
Ex: real; lamina longitudinal youngs modulus, (psi)
Ey: real; lamina transverse youngs modulus, (psi)
Es: real; lamina longitudinal shear modulus, (psi)

vx: real; major (larger) poissons ratio
vy: real; minor (smaller) poissons ratio

Output Parameters:
Qxx: real; on axis modulus
Qyy: real; on axis modulus
Qxy real; on axis modulus
Qss: real; on axis modulus

Named Common: none
File Management: none
Calls: none
Description: Determines on axis orthotropic modulus constants from lamina engineering constants. Reference 12, equation 4.12, page 4-4.

QUADROOT:
Source File Name: SOMSUB.FOR:
Declaration: QuadRoot(A,B,C,r1,r2,Ifail)
Input Parameters:
A: real; coefficient in quadratic equation
B: real; coefficient in quadratic equation
C: real; coefficient in quadratic equation

Output Parameters:
r1: real; larger root
r2: real; smaller root
ifail: integer; failure parameter

Named Common: none
File Management: none
Calls: none
Description: Calculates roots to the quadratic equation. ifail=0, ok; ifail=1, error;

RT1C19:
Source File Name: SOMSUB.FOR:
Decleration: RT1C19(r,t,alph,ri,y1b)
Input Parameters:
r: real; sector radius, (in)
t: real; thickness, (in)
alph: real; sector half angle, (radians)

Output Parameters:
ri: real; moment of inertia about neutral axis, (in**4)
y1b: real; neutral axis offset, (in)

Named Common: none
File Management: none
Calls: none
Description: Reference 9, table 1, case 19. Returns inertia and neutral axis for a corrugated plate.

RT35C1A:
Source File Name: SOMSUB.FOR:
Declaration: RT35C1A(aonb,b,E,riperin,v,rncr)
Input Parameters:
aonb: real;panel length to width ratio
b: real; panel width, (in)
E: real; youngs modulus, (psi)
riperin: real; moment of inertia per inch, (in**3)
v: real; poissons ratio

Output Parameters:
rncr: real; critical running load, (lb/in)

Named Common: none
File Management: none
Calls: none
Description: Reference 9, table 35, case 1a. Returns buckling

critical load for a plate subject to uniaxial compression.

RT35C4A:

Source File Name: SOMSUB.FOR:
Declaration: RT35C4A(aonb,b,E,riperin,v,rncr)
Input Parameters: aonb,b,e,riperin,v: real;
aonb: real;panel length to width ratio
b: real; panel width, (in)
E: real; youngs modulus, (psi)
riperin: real; moment of inertia per inch, (in**3)
v: real; poissons ratio
Output Parameters:
rncr: real; critical running load, (lb/in)
Named Common: none
File Management: none
Calls: none
Description: Reference 9, table 35, case 4a. Returns buckling critical load for a plate subject to shear load.

RT35C5A:

Source File Name: SOMSUB.FOR:
Declaration: RT35C5A(t,b,v,E,sx,sy,txy)
Input Parameters:
t,b,v,e,sx,sy,txy: real;
Output Parameters: ind: integer;
Named Common: none
File Management: none
Calls: none
Description: Reference 9, table 35, case 5a. Returns buckling indicator for a plate with in-plane loads. (not currently utilized)

SAMERN:

Source File Name: MATHLIB.FOR:
Declaration: SameRN(rn1,rn2,match)
Input Parameters:
rn1,rn2: real; real numbers to compare
Output Parameters:
match: integer; match=0, not same real number
match=1, rn1, and rn2 are the same
Named Common: none
File Management: none
Calls: none
Description: none required

SECANT3:

Source File Name: SOMSUB.FOR:
Declaration: Secant3(p,rm,bw,tw,sigvma,x0,x00,ac)
Input Parameters:
p: real; axial load, (lb)
rm: real; bending moment, (in-lb)
bw: real; web height, (in)
tw: real; web thickness, (in)
sigvma: real; allowable Von-Mises Stress, (psi)
x0: real; dummy parameter not utilized
x00: real; dummy parameter not utilized
Output Parameters:
ac: real; cap area, (sq in)
Named Common: none
File Management: none
Calls: none
Description: Calculates planar beam cap area to satisfy yield stress constraint. Secant iteration method.

SEMILOG:

Source File Name: MATHLIB.FOR

Declaration: SemiLog(x1,y1,x2,y2,a,b)

Input Parameters:

x1: real; first independent variable

y1: real; first dependent variable

x2: real; second independent variable

y2: real; second dependent variable

Output Parameters:

a: real; see description

b: real; see description

Named Common: none

File Management: none

Calls: none

Description: calculates the a, b coefficients in the semi-log equation, y=a*(e**bx).

SIZEL:

Source File Name: ELSIZER.FOR

Declaration: Sizel(indel,ithrmon,isize,
xl,yl,xyl,xm,ym,xym,xlth,ylth,xylth,
xmth,ymth,xymth,unitwt,indfm)

Input Parameters:

indel: integer; element number

ithrmon: integer; indicator to include thermal stresses
=1 yes; =0 no;

isize: integer; indicator to determine sizing or
stiffness matrix calculation,
=1 do sizing; =0 calculate stiff. matrix

xl: real; mechanical x direction running load, (lb/in)

yl: real; mechanical y direction running load, (lb/in)

xyl: real; mechanical shear running load, (lb/in)

xm: real; mechanical x direction moment, (in-lb/in)

ym: real; mechanical y direction moment, (in-lb/in)

xym: real; mechanical shear moment, (in-lb/in)

xlth: real; thermal x direction running load, (lb/in)

ylth: real; thermal y direction running load, (lb/in)

xylth: real; thermal shear running load, (lb/in)

xmth: real; thermal x direction moment, (in-lb/in)

ymth: real; thermal y direction moment, (in-lb/in)

xymth: real; thermal shear moment, (in-lb/in)

Output Parameters: unitwt: real; indfm: integer

unitwt: real; element unit weight, (plate = psf; bar = lb/in)

indfm: integer; failure mode indicator, ref. text pg. 15

Named Common: /compidd/ pidd: real(40)
/alwbls/ rmpro: real(50); matype: integer
/compnf/ patfile: character(30000)*80
kntreca: integer(45)
/comlog/ ilist,icalcij: logical
/comcnt/ nume23,nume24,nume33,nume43,
num2n,num3n,num4n: integer

File Management: none, but reads unit 9 for material properties as a function of temperature.

Calls: Element Sizing Routines, ex: HCBNDTH, E24ISO, I2FM, CORWEBTH, or user written element sizing routines

Description: Routes element sizing control to the appropriate sizing routine depending on the value of pidd(3) (itype in the PATRAN physical property description). Depending on the value of isize, calls the element sizing routine to either calculate element stiffnesses or calculate new element dimensions. Returns unit weight and

failure mode to the calling procedure. This is where calls to the user written element sizing routines are added.

STIFFJ:
Source File Name: STIFFJ.FOR
Declaration: STIFFJ(pascij,pasfij,paswto)
Input Parameters: none
Output Parameters:
pascij: real*8(6,6); EAL constituitive stiffness matrix
pasfij: real*8(6,3); EAL stress recovery matrix, (dummy values, not utilized)
paswto: real*8; plate unit weight
Named Common: Many common blocks peculiar to program STIMAT reference 7, not required to know these to access STIFFJ.
/comdf/myfile: character(100)*80
File Management: none
Calls: all specific to program STIMAT
Description: A modification of program STIMAT to work with internal files as opposed to using disk files which would require i/o operations for each element. Uses data passed by the internal file myfile to calculate an element stiffness matrix.

TRNSFRMR:
Source File Name: SOMSUB.FOR
Declaration: Trnsfrmr(rm,rn,cijp,cij)
Input Parameters:
rm: real; cosine theta
rn: real; sine theta
cijp: real(6,6); input stiffness matrix
Output Parameters:
cij: real(6,6); transformed stiffness matrix
Named Common: none
File Management: none
Calls: TSAITA7
Description: Breaks a Cij matrix into A,B,D components and transforms it through angle theta via reference 11 table A.7 page 440.

TRNSFR2:
Source File Name: SOMSUB.FOR
Declaration: Trnsfr2(rm,rn,cijp,cij)
Input Parameters:
rm: real; cosine theta
rn: real; sine theta
cijp: real(6,3); input stress recovery matrix
Output Parameters:
cij: real(6,3); transformed stress recovery matrix
Named Common: none
File Management: none
Calls: TSAITA7
Description: Similiar to TRNSFRMR but used for transforming EAL Fij matrices as opposed to Cij matrices.

TSAE365:
Source File Name: SOMSUB.FOR
Declaration: TsaE365(q,s)
Input Parameters:
q: real(3,3); modulus matrix
Output Parameters:

s: real(3,3); compliance matrix
Named Common: none
File Management: none
Calls: none
Description: Reference 11 equation 3.65 page 97.
Determines compliance matrix from modulus matrix.

TSAIE417:
Source File Name: SOMSUB.FOR
Declaration: TsaiE417(a,r1,r2,r3,eps)
Input Parameters:
a: real(3,3); compliance matrix
r1: real; x direction running load
r2: real; y direction running load
r3: real; shear running load
Output Parameters:
eps: real(3); x, y, and shear strain
Named Common: none
File Management: none
Calls: none
Description: Reference 11 equation 4.17.
Multiplies compliance by in-plane stress
resultants to obtain laminate strains.

TSAIQSP:
Source File Name: SOMSUB.FOR
Declaration: TsaiQsp(X,Xpr,Y,Ypr,S,Fxx,Fxy,Fyy,Fss,Fx,Fy)
Input Parameters:
X: real; lamina longitudinal tensile ult. strength
Xpr: real; lamina longitudinal comp. ult. strength
Y: real; lamina transverse tensile ult. strength
Ypr: real; lamina transverse comp. ult. strength
S: real; lamina longitudinal shear strength
Output Parameters:
Fxx,Fxy,Fyy,Fss,Fx,Fy: real(3); Quadratic strength parameters
Named Common: none
File Management: none
Calls: none
Description: Reference 11 page 281. Calculates parameters
required for the quadratic strength allowable
stress check.

TSAITA1:
Source File Name: SOMSUB.FOR
Declaration: TsaiTa1(sig1,sig2,sig6,rm,rn,sig1p,sig2p,sig6p)
Input Parameters:
Sig1,Sig2,Sig6: real; input stress state
rm: real; cosine theta
rn: real; sine theta
Output Parameters:
Sig1p,Sig2p,Sig6p: real; transformed stress state
Named Common: none
File Management: none
Calls: none
Description: Reference 11 Appendix A table a.1
Transformation of stress to primed axis.

TSAITA7:
Source File Name: SOMSUB.FOR
Declaration: TsaiTa7(rm,rn,qp,q)
Input Parameters:
rm: real; cosine theta
rn: real; sine theta

qp: real(3,3); input stiffness matrix
Output Parameters:
q: real(3,3); transformed stiffness matrix
Named Common: none
File Management: none
Calls: none
Description: Reference 11 Table A7, page 440.
transformation of stiffness from one off axis
orientation to another through the angle theta.

TSAIT32:
Source File Name: SOMSUB.FOR
Declaration: TsaiT32(rm,rn,qxx,qyy,qxy,qss,qoff)
Input Parameters:
rm: real; cosine theta
rn: real; sine theta
qxx,qyy,qxy,qss: real; on axis moduli
Output Parameters:
qoff: real(3,3); off axis modulus matrix
Named Common: none
File Management: none
Calls: none
Description: Reference 11 Table 3.2, page 69.
transformation of stiffness from on axis
orientation to off axis orientation through
the angle theta.

TSAQSNP:
Source File Name: SOMSUB.FOR
Declaration: TsaQsnp(Qxx,Qyy,Qxy,Qss,Fxx,Fyy,Fxy,Fss,Fx,Fy,
Gxx,Gyy,Gxy,Gss,Gx,Gy)
Input Parameters:
Qxx,Qyy,Qxy,Qss; real; on axis moduli
Fxx,Fyy,Fxy,Fss,Fx,Fy: real; quadratic strength parameters
Output Parameters:
Gxx,Gyy,Gxy,Gss,Gx,Gy: real; above parameters in strain space
Named Common: none
File Management: none
Calls: none
Description: Reference 11 equation 7.28. Calculates
lamina allowable strain parameters based
on allowable strength parameters for the
quadratic strength check.

TSAT76A:
Source File Name: SOMSUB.FOR
Declaration: TsaT76a(rm,rn,Gxx,Gyy,Gxy,Gss,
G11,G22,G12,G66,G16,G26)
Input Parameters:
rm: real; cosine theta
rn: real; sine theta
Gxx,Gyy,Gxy,Gss: real; On axis strength parameters in
strain space
Output Parameters:
G11,G22,G12,G66,G16,G26: real; Off axis parameters in
strain space
Named Common: none
File Management: none
Calls: none
Description: Reference 11 Table 7.6(a). Transform of
quadratic strength parameters in strain space.

TSAT76B:

Source File Name: SOMSUB.FOR
Declaration: TsaT76b(rm,rn,Gx,Gy,G1,G2,G6)
Input Parameters:
rm: real; cosine theta
rn: real; sine theta
Gx,Gy: real; on axis linear strength parameters in strain space
Output Parameters:
G1,G2,G6: real; off axis parameters in strain space
Named Common: none
File Management: none
Calls: none
Description: Reference 11 Table 7.6(b). Transform of linear strength parameters in strain space.

UONAXIS:
Source File Name: SOMSUB.FOR
Declaration: UonAxis(Qxx,Qyy,Qxy,Qss,U1,U2,U3,U4,U5)
Input Parameters:
Qxx,Qyy,Qxy,Qss: real; on axis stiffness moduli
Output Parameters:
U1,U2,U3,U4,U5: real; linear combinations of above
Named Common: none
File Management: none
Calls: none
Description: Determines linear combinations of on axis stiffness moduli based on reference 12 table 6.3 page 6-4.

VONMISE:
Source File Name: SOMSUB.FOR
Declaration: VonMise(S,sigvm)
Input Parameters:
S: real(3); principal stresses, (S(3)=0 for biaxial)
Output Parameters:
sigvm: real; Von-Mises stress
Named Common: none
File Management: none
Calls: none
Description: Calculates Von Mises stress, used to check against a material yield allowable.

WHATSNU:
Source File Name: SOMSUB.FOR
Declaration Stmt: WHATSNU(v21,E1,E2,v12)
Input Parameters: v21,e1,e2: real;
Output Parameters: v12: real;
Named Common: none
File Management: none
Calls: none
Description: Calculates major poisson ratio based on minor and modulus values for a lamina. Could also be used in reverse by inverting input.

Subroutine Call Hierarchy - Program EZBATCH

A linking command file called MAKEEZBATCH.COM (similar to MAKEZDESIT.COM) exists in the ezbatch directory. The following routines are used to create the batch version of element sizing, and also exist in the ezbatch directory.

EZBATCH

- PATTOINT, Reads patran.out into internal file format. Same routine as used in EZEAL.

- DIMEALB, Reads element dimension files and converts dimensions to stiffnesses for use in updating the EAL runstream based on new element sizes and element average temperatures. Structured very much like ELSIZER but uses element dimensions as read from a file as opposed to calculating new ones.

- EALUPDT, Updates the EAL runstream for execution of the next required loadset.

- LIB$SPAWN, Submits an updated EAL runstream

- ELDPLDB, Batch version of ELDTOPLD

- ELSIZEB, Batch version of ELSIZER

- ELSDIMB, Reads output files from ELSIZEB, finds the heaviest element for each file and uses that element geometry to write the worst case element dimension file, LDSTCOMB.DIM.

- CHKERVEC, Stops batch iteration after element geometry convergence, currently only set to allow a fixed input number of iterations.

Major Subroutines of EZBATCH:

DIMEALB:

Source File Name: DIMEALB.FOR
Declaration Stmt: DIMEALB(nume23,nume24,nume33,nume43,dimfile,itsid)
Input Parameters: nume23,nume24,nume33,nume43: integers
dimfile: character*15;
itsid: integer;
Output Parameters: none
Named Common: /Compnf/ patfile: character(30000)*80;
kntreca: integer(45);
/Comlog/ ilist,icalcij: logical;
File Management: Opens unit 7 as dimfile (typically dimfile="ldstcomb.dim"). Opens units 13, 14, 15, and 15 as Cijfile.eal, SWfile.eal, ELDfile.eal, and MATCfile.eal respectively. These are EAL runstream segments to be compacted by subroutine COMPACTOR. Also used the material property data file matprofil as unit 9, (typically matprofil='Matprop.prn").
Calls: SIZEL2, COMPACTOR
Description: Basically a duplicate of DIMTOEAL for the batch process. Reads the latest dimension file for element dimensions to input to the stiffness matrix calculation subroutine. Creates stiffness, weight matc and connectivity files. (cijfile.eal, swfile.eal, matcfile.eal, eldfile.eal) these files are compacted by subroutine COMPACTOR.

Ealupdt:

Source File Name: EALUPDT.FOR
Declaration Stmt: EALUPDT(nume23,nume24,nume33,nume43, modstif,ealfno,ldst,ldstth,iconm,iconth)
Input Parameters: nume23,nume24,nume33,nume43: integer;
modstif: integer;
ealfno: character*15;
ldst,ldstth,iconm,iconth: integer;
Output Parameters: none
Named Common: none
File Management: Opens matcfile.eal, swfile.eal, eldfile.eal, cijfile.eal, and ealfno. Ealfno (an input EAL runstream) is updated with info from the other four files. The other four files are then closed and deleted. The original ealfno file is deleted but a new one is written with the updated information.
Description: per file management

EldPldB:

Source File Name: ELDPLDB.FOR
Declaration Stmt; eldpldb(nume23,nume24,nume33,nume43, ldsfile,outfile)
Input Parameters; nume23,nume24,nume33,nume43: integer;
ldsfile: character*15
Output Parameters; outfile: character*15
Named Common; none
File Management; Opens unit 17 as the input stress resultant file (EAL DCU PRINT format) and unit 18 as the

output stress resultant file (PATRAN element results file format). closes both files upon completion deleting the input stress resultatn file.

Description: ldsfile is read in and converted to PATRAN element results file format. ldsfile is subsequently deleted and outfile is saved.

ELSDIMB:

Source File Name: ELSDIMB.FOR

Declaration Stmt; Elsdimb(nume23,nume24,nume33,nume43, numfils,elsisum)

Input Parameters; nume23,nume24,nume33,nume43: integer; numfils: integer; elsisum: character(20)*15;

Output Parameters; none

Named Common; /Compnf/ patfile: character(30000)*80; kntreca: integer(45);

File Management; Opens dimension files as units n+11. where n is a counter from 1 to the number of files being searched. Closes these files upon subroutine completion.

Calls: GETPID, GETPDAT

Description: The summary files are compared to find which has the heaviest unit weight for each element. A new summary file 'LDSTCOMB.DIM' is written consisting of this heaviest element geometry information. No files are deleted in this process. File 'patran.out' is used for information purposes, this program should be upgraded to use named common and the internal form of patran.out

ELSIZEB:

Declaration Stmt; Elsizeb(nume23,nume24,nume33,nume43, imechon,ldsfile,ithrmon,ldsfilt,eloutfil, ellisfil,itsid)

Input Parameters; nume23,nume24,nume33,nume43,imechon: integer; ldsfile: character*15; ithrmon: integer; ldsfilt: character*15; itsid: integer;

Output Parameters; eloutfil: character*15; ellisfil: character*15;

Named Common: /Compnf/ patfile: character(30000)*80; kntreca: integer(45); /Comlog/ ilist,icalcij: logical;

File Management;

Calls: GETLOAD, SIZEL

Description: Sizes elements based on loads from ldsfile. ldsfile is not deleted when closed. Output consists of the two files eloutfil and ellisfil.

[redacted]