AFRL-VA-WP-TR-2001-3054

AERODYNAMIC ANALYSIS FOR THE DESIGN ENVIRONMENT (AANDE)

VOLUME 2: USER'S MANUAL

M. H. LOVE
D. D. EGLE

LOCKHEED MARTIN TACTICAL AIRCRAFT SYSTEMS
AIRFRAME AND INSTALLATION
P.O. BOX 748, MAIL ZONE 2824
FORT WORTH, TX 76101-0748

NOVEMBER 1999

FINAL REPORT FOR PERIOD OF 30 SEPTEMBER 1995 – 30 JUNE 1998

Approved for public release; distribution unlimited.

AIR VEHICLES DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7542

NOTICE

USING GOVERNMENT DRAWINGS, SPECIFICATIONS, OR OTHER DATA INCLUDED IN THIS DOCUMENT FOR ANY PURPOSE OTHER THAN GOVERNMENT PROCUREMENT DOES NOT IN ANY WAY OBLIGATE THE UNITED STATES GOVERNMENT. THE FACT THAT THE GOVERNMENT FORMULATED OR SUPPLIED THE DRAWINGS, SPECIFICATIONS, OR OTHER DATA DOES NOT LICENSE THE HOLDER OR ANY OTHER PERSON OR CORPORATION; OR CONVEY ANY RIGHTS OR PERMISSION TO MANUFACTURE, USE, OR SELL ANY PATENTED INVENTION THAT MAY BE RELATED TO THEM.

THIS REPORT IS RELEASEABLE TO THE NATIONAL TECHNICAL INFORMATION SERVICE (NTIS). AT NTIS, IT WILL BE AVAILABLE TO THE GENERAL PUBLIC, INCLUDING FOREIGN NATIONS.

THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.

Victoria A. Tischler
VICTORIA A. TISCHLER
Aerospace Engineer
Structural Design and Development Branch

Terry M. Harris
TERRY M. HARRIS, Chief
Structural Design and Development Branch
Structures Division

David M. Pratt
DAVID M. PRATT, Technical Advisor
Structures Division
Air Vehicles Directorate

Do not return copies of this report unless contractual obligations or notice on a specific document require its return

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
November 1999

2. REPORT TYPE
Final

3. DATES COVERED *(From - To)*
30 Sep 1995- 30 Jun 1998

4. TITLE AND SUBTITLE
Aerodynamic Analysis for the Design Environment (AANDE)
Volume 2: User's Manual

5a. CONTRACT NUMBER
F33615-95-C-3224

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62201F

5d. PROJECT NUMBER
2401

5e. TASK NUMBER
TI

5f. WORK UNIT NUMBER
05

6. AUTHOR(S)
M.H.Love, D.D. Egle, Lockheed Martin Tactical Aircraft Systems

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lockheed Martin Tactical Aircraft Systems
Airframe and Installation
P.O. Box 748, Mail Zone 2824
Fort Worth, Texas 76101-0748

8. PERFORMING ORGANIZATION REPORT NUMBER
FZM-8538

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Vehicles Directorate
Air Force Research Laboratory
Air Force Materiel Command
Wright Patterson, AFB OH 45433-7542

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-VA-WP-TR-2001-3054

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report is part of the documentations which describe the complete development of the "Aerodynamic ANalysis for the Design Environment (AANDE)" . It is one of three manuals that comprise the final report. The remaining reports consist of the AANDE Theoretical and Applications Studies Manual (Volume I) and the AANDE Programmer's Manual (Volume III).
The objective of the AANDE effort was to establish high quality reliable airloads for multidisciplinary design in ASTROS. This was accomplished by providing: a new steady linear aerodynamic procedure, alternate paths for the import of aerodynamic influence coefficient matrices and nonlinear pressure data, and a general asymmetric maneuver trim procedure.

This user's manual documents updates to the ASTROS Version 12.0 User's Manual as well as user input requirements for QUADPAN. The mofications to the User's manual include: the executive system and MAPOL, the solution control packet, the input data stream, the bulk data packet, the QUADPAN model geometry, the QUADPAN dataset, and panel abutments.

15. SUBJECT TERMS
Aerodynamics, Analysis, Multidisciplinary Design, airloads, ASTROS, trim, asymmetric, QUADPAN, MAPOL

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
U	U	U	SAR	226	Victoria A. Tischler
					19b. TELEPHONE NUMBER *(Include area code)* 937-255-9729

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

FOREWORD

The Air Force Research Laboratory initiated development of the Automated Structural Optimization System, ASTROS, in 1983. Additional development work was conducted and completed in 1995 under contract F33615-87-C-3216. This document presents the user requirements for enhancements to ASTROS developed under the Aerodynamic Analysis for the Design Environment (AANDE) contract F33615-95-C-3224. This contract has been conducted by Lockheed Martin Tactical Aircraft Systems (LMTAS) and their subcontractor Universal Analytics Inc.. Lockheed Martin Aeronautical Systems also provided assistance to LMTAS in the AANDE program. Major contributors to the AANDE program include M.H. Love, the Program Manager, D.D. Egle, and D.K. Barker from LMTAS and R. Coopersmith from LMAS. From Universal Analytics, the major contributors were D.J. Neill, T. Shimko, S. Chen, and J. San Marco.

This report constitutes changes to the ASTROS version 12.0 User's Document (Ref. 1). It is one of four documents generated under the AANDE program.

Dr. Ray Kolonay was the primary Air Force program engineer for the AANDE program. Dr. V.B. Venkayya initiated the program and has provided much of the overall program direction.

1. INTRODUCTION

The unique attributes of ASTROS (Ref. 2) hold great potential for savings in design time, improvements in vehicle performance, and reductions in structural weight in aerospace vehicles. This potential has been limited due to capabilities lacking in modeling and simulation of maneuver loads for design. The overall objective of the Aerodynamic Analysis for the Design Environment contract (F33615-95-C-3224) is to establish high quality, reliable loads simulation in ASTROS. The Lockheed Martin Tactical Aircraft Systems team including Universal Analytics Inc. and Lockheed Martin Aeronautical Systems accomplished this objective by providing a new steady linear aerodynamic procedure, alternate paths for import of aerodynamic influence coefficient matrices and nonlinear pressure data, and a general asymmetric maneuver trim procedure.

The program encompassed three main tasks:

1.0 Phase I - System Specifications
2.0 Phase II - Module Development and Prototyping
3.0 Phase III - "Seamless" Integration and Verification.

In Phase I, changes to the ASTROS modules, paradigms, and data structures were identified, modeled, and tested against realistic scenarios of fighters, bombers, and transport aircraft. The results of these exercises formulated the plans for the software development and verification and are documented in the Software Design Guide (Ref. 3).

In Phase II, individual modules were developed and tested with realistic test cases as well as simple cases used for development. In Phase III, the modules were integrated into ASTROS through the memory manager, database, and MAPOL. Verification studies were performed simulating usage in a preliminary design scenario.

The AANDE contract is documented through four reports, Software Design Guide (Ref. 3), Programmer's Report (this report), the Theoretical and Application Studies Report (Ref. 4), and the User's Report (Ref. 5). The Software Design Guide (SDG) was developed at the end of Phase I of the AANDE program with the intention of defining requirements for ASTROS development beyond the original scope of AANDE but within the scope of air-loads in the design process. Many of the requirements in the SDG are implemented. The remaining documents are supplemental to the ASTROS version 12 documentation (Ref. 6). Areas of modification are documented fully. The Programmer's Report includes new and modified module descriptions and database entities.

The User's Report includes updates to the ASTROS Version 12.0 User's Manual (Ref. 1) as well as user input requirements for QUADPAN. The modifications to User's Manual include:

2. **THE EXECUTIVE SYSTEM AND MAPOL**
3. **THE SOLUTION CONTROL PACKET**
4. **INPUT DATA STREAM**
5. **THE BULK DATA PACKET**
6. **QUADPAN MODEL GEOMETRY**
7. **QUADPAN DATASET**
8. **PANEL ABUTMENTS**

THIS PAGE INTENTIONALLY LEFT BLANK.

2. THE EXECUTIVE SYSTEM AND MAPOL

2.1 CHANGES TO MAPOL

MAPOL was modified in capabilities under the AANDE program. Also, the MAPOL solution sequence was modified extensively. However the overall solution sequence structure was maintained.

New in this version of ASTROS is the concept that multiple databases may be used within a given ASTROS run. This enables the ASTROS user to have aerodynamic data residing on multiple databases and to mix and match data to assemble an aerodynamic model best suited to simulate the aerodynamic characteristics of interest and obtain accurate air loads for multidisciplinary analysis. To enable this, MAPOL needed to handle physical addresses of data independent of the runtime database as well as the data itself.

MAPOL was modified to support the longer entity names implemented in CADDB changes. This was done to facilitate the creation of "indexed" entity names in the aerodynamic model groups. Also, the CHARACTER variable type is supported for the MAPOL language.

Declaration	:	CHARACTER A, B, C;
Assignment	:	A := "string";
Comparison	:	IF A = "QUADPAN" THEN IF A "QUADPAN" THEN
Argument Passing	:	CALL MODULE (A);

CHARACTER data types may be set in a module and passed out to MAPOL with an updated value--just like other number variables.

Then, four new entity name types are listed in Table 2-1.

Table 2-1 New Entity Name Types For MAPOL Character Variables

<option>	Description
GGMEMBER	Group entity type
RGMEMBER	Relational group member
UGMEMBER	Unstructured group member
MGMEMBER	Matrix group member

These are basically entity name "variables" rather than entity name "symbols." The distinction is that the variables can be set in the module and passed through the MAPOL calling sequence. Regular entity names are static - once declared, they become a symbol rather than a variable name. The new feature is used in naming the members of a group. The GGMEMBER type is a RELATION on CADDB, but is denoted separately so that argument passing can perform the appropriate type checking. The other types are just like their non-group counterparts.

2.2 THE AANDE MAPOL PROGRAM LISTING

The AANDE version MAPOL listing of the standard solution sequence is given in the following pages.

```
STAT LEVL
     1!$***$                                                                   !
     1!$ CSCIID <@(#) MC0083-MAPOLSEQ 12.49 15 JUL 1998 17:01:52> $            !
     1!$***$                                                                   !
     1!$*********************************************************************$!
     1!$                        EXECUTIVE SEQUENCE FOR ASTROS                 $!
     1!$*********************************************************************$!
     1!$*********************************************************************$!
     1!$              CONSTANTS FOR SDCOMP SET SINGULARITY MESSAGES          $!
     1!$*********************************************************************$!
     1!INTEGER   SINGOSET,    SINGASET,    SINGLSET;                           !
     1!$*********************************************************************$!
     1!$              VARIABLE DECLARATION SEGMENT                           $!
     1!$*********************************************************************$!
     1!$                                                                      $!
     1!INTEGER   GSIZE,       NDV,         NITER,       BC,          BCID,     !
     1!          ESIZE(1000), PSIZE(1000), GSIZEB,      LASTITER;              !
     1!REAL      CTL,         CTLMIN;                                          !
     1!LOGICAL   GLBCNVRG,    APPCNVRG,    PFLAG,       GPRINT,      LOADLDV,  !
     1!          LPRINT,      GPUNCH;                                          !
     1!UNSTRUCT  DCENT,       GRIDTEMP,    SMPLOD;                             !
     1!RELATION  DESHIST,     CONST,       MPPARM,      CONVERT,     OCPARM,   !
     1!          MFORM,       GRID,        SPOINT,      EPOINT,      SEQGP,    !
     1!          BGPDT(1000), CSTM,        FORCE,       FORCE1,      MOMENT,   !
     1!          MOMENT1,     PLOAD,       PLOAD2,      PLOAD4,      GRAV,     !
     1!          LOAD,        EIGR,        CONSTORD,    ESAVE,                 !
     1!          TEMP,        TEMPD,       OPNLBUCK,    OEULBUCK,              !
     1!          CORD1C,      CORD1R,      CORD1S,      CORD2C,      CORD2R,   !
     1!          CORD2S,      GPWGGRID,    OGPWG,       GRADIENT;              !
     1!RELATION  IMPORT,      ARCHIVE,     OVERLAY,     ASSEMBLE,    DISASSEM; !
     1!$                                                                      $!
     1!$*********************************************************************$!
     1!$           DECLARATIONS FOR MODULE MKUSET                            $!
     1!$*********************************************************************$!
     1!$                                                                      $!
     1!UNSTRUCT  USET(1000),  GPST(1000);                                      !
     1!RELATION  SPC,         SPC1,        SPCADD,      MPC,         MPCADD,   !
     1!          ASET,        ASET1,       OMIT,        OMIT1,       SUPORT,   !
     1!          JSET,        JSET1,       RBAR,  RBE1, RBE2,  RBE3, RROD;     !
     1!MATRIX    [PGMN(1000)],[PNSF(1000)],[PFOA(1000)],[PARL(1000)],[TMN(1000)], !
     1!          [YS(1000)];                                                   !
     1!MATRIX    [PGMNS(1000)],            [PNSFS(1000)],            [PFOAS(1000)], !
     1!          [PARLS(1000)];                                                !
     1!$                                                                      $!
     1!$*********************************************************************$!
     1!$           DECLARATIONS FOR MODULES MAKEST, EMG AND NLEMG            $!
     1!$*********************************************************************$!
     1!$                                                                      $!
     1!UNSTRUCT  TREF,        DVSIZE,      PCOMPS;                             !
     1!UNSTRUCT  TREFD,       DVSIZED,     DDVSIZE;                            !
     1!IUNSTRUCT KELM,        MELM,        TELM;                               !
     1!IUNSTRUCT KELMD,       MELMD,       TELMD,       DKELM,       DMELM,    !
     1!          DTELM;                                                        !
     1!RELATION  CQDMEM1,     QDMM1EST,    CROD,        CONROD,      RODEST,   !
     1!          CSHEAR,      SHEAREST,    CTRMEM,      TRMEMEST,    CMASS1,   !
     1!          CMASS2,      MASSEST,     CONM1,       CONM1EST,    CONM2,    !
     1!          CONM2EST,    CBAR,        BEAMEST,     CQUAD4,      QUAD4EST, !
     1!          CIHEX1,      IHEX1EST,    CIHEX2,      IHEX2EST,    CIHEX3,   !
     1!          IHEX3EST,    CELAS1,      CELAS2,      ELASEST,               !
     1!          PCOMP,       PQDMEM1,     PROD,        PSHEAR,                !
     1!          PTRMEM,      PMASS,       PELAS,       PBAR,        PBAR1,    !
     1!          PSHELL,      PCOMP1,      PCOMP2,      PIHEX,       MAT1,     !
     1!          MAT2,        MAT8,        MAT9,        CTRIA3,      TRIA3EST, !
     1!          DVTOPTE,     DVTOPTP,     DVTOPTL,     SMATCOL,     NLSMTCOL, !
     1!          EIDTYPE;                                                      !
     1!$                                                                      $!
     1!$*********************************************************************$!
     1!$           DECLARATIONS FOR DESIGN VARIABLES/CONSTRAINTS AND LINKING $!
     1!$*********************************************************************$!
     1!$                                                                      $!
     1!RELATION  DESELM,      DESVARP,     DESVARS,     PLIST,       ELIST,    !
     1!          SHAPE,       PLISTM,      ELISTM,      SHAPEM,      SHPGEN;   !
     1!RELATION  DCONVM,      DCONTW,      DCONEP,      DCONFT,      DCONVMM,  !
     1!          DCONTWM,     DCONEPM,     DCONFTM,     DCONVMP,     DCONTWP,  !
     1!          DCONEPP,     DCONFTP,     DCONALE,     DCONCLA,     DCONFLT,  !
     1!          DCONTRM,     DCONSCP,     DCONF,       DCONSDE,     DCONSDL;  !
     1!RELATION  DCONDSP,     DCONFRQ,     DCONTHK,     DCONTH2,     DCONTH3;  !
     1!RELATION  DCONPMN,     DCONLMN,     DCONLAM;                            !
     1!RELATION  DCONBK,      DCONBKE;                                         !
```

```
1!RELATION    GLBDES,      DESLINK,     TFIXED,      LOCLVAR,     DVCT,          !
1!            DVCTD,       DDVCT;                                                !
1!MATRIX      [PTRANS];                                                          !
1!IMATRIX     [PMINT],     [PMAXT],     [SMAT],      [NLSMAT];                   !
1!$                                                                             $!
1!$*************************************************************************$!
1!$        DECLARATIONS FOR OUTPUT FILE PROCESSING  (EDR/OFP)                    $!
1!$*************************************************************************$!
1!$                                                                             $!
1!RELATION    GRIDLIST,    MODELIST,    ELEMLIST,    FREQLIST,    TIMELIST,      !
1!            ITERLIST,    GDVLIST,     LDVLIST,     DCONLIST,    PLYLIST,       !
1!            DENSLIST,    MACHLIST,    VELOLIST,    CASELIST;                   !
1!$                                                                             $!
1!RELATION    GPFELEM,     EOSUMMRY,    EOBAR,       EOELAS,      EOHEX1,        !
1!            EOHEX2,      EOHEX3,      EOQDMM1,     EOQUAD4,     EOROD,         !
1!            EOSHEAR,     EOTRMEM,     GPFDATA,     EOTRIA3;                    !
1!UNSTRUCT    EODISC;                                                            !
1!$                                                                             $!
1!RELATION    OGRIDLOD,    OGRIDDSP,    OLOCALDV,    OAGRDDSP,    OAGRDLOD;      !
1!MATRIX      [FLUTMODE],  [PTGLOAD],   [PFGLOAD],   [PTHLOAD],   [PFHLOAD];     !
1!$                                                                             $!
1!$*************************************************************************$!
1!$        DECLARATIONS FOR MODULES EMA1, NLEMA1, EMA2 AND GLOBAL                $!
1!$                  MATRIX PARTITION/REDUCTION                                  $!
1!$*************************************************************************$!
1!$                                                                             $!
1!UNSTRUCT    GENEL;                                                             !
1!IUNSTRUCT   DKVI,        DMVI,        DKVI0,       DMVI0,       DKVIG,         !
1!            DMVIG,       DMVID,       DDMVI;                                   !
1!UNSTRUCT    DWGH1,       DDWGH2;                                               !
1!RELATION    GMKCT,       GMMCT,       GMKCT0,      GMMCT0,      GMKCTG,        !
1!            GMMCTG,      GMMCTD,      DGMMCT;                                  !
1!MATRIX      [KGG],       [KNN],       [KFF],       [KAA],       [KLL],         !
1!            [MGG],       [MNN],       [MFF],       [MAA],       [MLL],         !
1!            [MRRBAR],    [MLR],       [KFS],       [KSS],       [KOOINV(1000)], !
1!            [GSUBO(1000)],            [KLLINV(1000)],           [MRR(1000)],   !
1!            [IFM(1000)], [M1GG],      [IFR(1000)], [KRR],       [D(1000)],     !
1!            [KLR],       [K1GG],      [LHS(1000)], [M2GG],      [MOO],         !
1!            [MOA],       [K2GG],      [MAABAR];                                !
1!MATRIX      [TMP1],      [TMP2];                                               !
1!MATRIX      [PG],        [PN],        [PF],        [PA],                       !
1!            [PO],        [PLBAR],     [PR],        [RHS(1000)], [UG(1000)],    !
1!            [UN],        [UF],        [UA],        [UL],        [UM],          !
1!            [AG(1000)],  [AN],        [AF],        [AA],        [AR],          !
1!            [AL],        [UO],        [UOO],       [PS];                       !
1!LOGICAL     M2GGFLAG,    K2GGFLAG;                                             !
1!$                                                                             $!
1!$*************************************************************************$!
1!$     DECLARATIONS FOR SOLUTION CONTROL                                        $!
1!$*************************************************************************$!
1!$                                                                             $!
1!INTEGER     NUMOPTBC,    NBNDCOND,    MAXITER,                                 !
1!            MPS,         MPE,                                                  !
1!            OCS,         OCE,                                                  !
1!            FSDS,        FSDE;                                                 !
1!INTEGER     BLOAD,       BMASS,       BMODES,      BSAERO,      BFLUTR,        !
1!            BDYN,        BDRSP,       BDTR,        BMTR,        BDFR,          !
1!            BMFR,        BGUST,       BBLAST,      NMPC,        NSPC,          !
1!            NOMIT,       NRSET,       DMODES;                                  !
1!REAL        MOVLIM,      WINDOW,      OCMOVLIM,    ALPHA,       CNVRGLIM,      !
1!            NRFAC,       EPS,         FDSTEP,      K6ROT,       TOLVALUE;      !
1!RELATION    JOB,         OPTIMIZE,    CASE;                                    !
1!$                                                                             $!
1!$*************************************************************************$!
1!$     DECLARATIONS FOR SENSITIVITY  EVALUATION                                 $!
1!$*************************************************************************$!
1!$                                                                             $!
1!INTEGER     DDFLG,       NACSD,       NAUS,        NAUA;                       !
1!LOGICAL     ACTBOUND,    ACTFLUT,     ACTDYN,      ACTAERO,     ACTAEFF,       !
1!            ACTUAG,      ACTUAGG,     ACTPNL,      ACTBAR;                     !
1!UNSTRUCT    PCAS,        PRAS,        PCAA,        PRAA,        PCAE;          !
1!RELATION    PDLIST;                                                            !
1!MATRIX      [DFDU],      [PGAS],      [UGA],       [DUG],       [DMUG],        !
1!            [DPFV],      [DPOV],      [DPNV],      [DPAV],      [DUAV],        !
1!            [DUAD],      [DUFV],      [AGA],       [AMAT],      [DKUG],        !
1!            [DPGV],      [DPLV],      [DURD],      [DULD],      [DULV],        !
1!            [DDELDV],    [DPRV],      [DRHS],      [DFDUF],     [PGAA],        !
1!            [DFDUN],     [DMAG],      [DMUN],      [DMUF],      [DMUA],        !
1!            [DMUO],      [DMUL],      [DMUR],      [DMU],       [DP1],         !
1!            [DK1V],      [AUAGC],     [DURV],      [EFFSENS],   [DU1L],        !
1!            [DU1R],      [DU2],       [LHSL],      [LHSU],      [PGAU],        !
1!            [DFSV],      [SENSMT];                                             !
1!IMATRIX     [GLBSIG],    [DPTHVI],    [DPGRVI],    [DPVJ];                     !
1!IMATRIX     [NLGLBSIG],  [DPTHVD],    [DPGRVD],    [DDPTHV],    [DDPGRV];      !
1!$                                                                             $!
1!$*************************************************************************$!
1!$     SUBSTRUCTURING ENTITIES                                                  $!
1!$*************************************************************************$!
1!$                                                                             $!
1!REAL        RSYM;                                                              !
```

```
1!LOGICAL     STRSYM,       SYMTRN(1000),      ACTUAGGI,          ACSMTR(1000);     !
1!LOGICAL     NEWITER;                                                              !
1!RELATION    RELES;                                                                !
1!MATRIX      [HFREALT(1000)],   [HFIMAGT(1000)];                                   !
1!MATRIX      [HFREAL(1000)],    [HFIMAG(1000)];                                    !
1!MATRIX      [HTKFHR],     [HTKFHI],     [KFFX];                                   !
1!MATRIX      [HTMFHR],     [HTMFHI],     [MFFX];                                   !
1!MATRIX      [HFRTPF],     [HFITPF],     [PFX];                                    !
1!MATRIX      [HRGTKF],     [HRGSTKF],    [HRGPTKF],    [AFX];                      !
1!MATRIX      [HIGTKF],     [HIGSTKF],    [HIGPTKF],    [UFX],        [UAFX];       !
1!MATRIX      [GPTKFX],     [AICS1],      [AICS2],      [AICS3],      [AICS4];      !
1!MATRIX      [GSKFHRT],    [GSKFHIT],    [AICSUM],     [AICDIF],     [AAFX];       !
1!MATRIX      [HRDPFV],     [HIDPFV],     [DPFVX];                                  !
1!MATRIX      [HRDMUF],     [HIDMUF],     [DMUFX];                                  !
1!MATRIX      [PGMNX(1000)],              [PNSFX(1000)],              [UAFCX(1000)];   !
1!MATRIX      [PFOAX(1000)],              [PARLX(1000)],              [AAFCX(1000)];   !
1!MATRIX      [RGBDMG],     [DUFVX],      [DUFVI];                                  !
1!MATRIX    , [UAFI],        [AAFI],        [UAFCI(1000)],        [AAFCI(1000)];    !
1!MATRIX      [UANI],        [AANI],        [UANCI(1000)],        [AANCI(1000)];    !
1!MATRIX      [UAGI(1000)], [AAGI(1000)], [UAGCI(30,33)],         [AAGCI(30,33)];   !
1!MATRIX      [GLBSIGI],     [NLGBSIGI];                                            !
1!MATRIX      [UGAI],       [AGAI],       [AUAGCI],     [AAAGCI];                   !
1!MATRIX      [DKUGI],      [DMAGI],      [DPGVI],      [DMUGI];                    !
1!MATRIX      [DPNVI],      [DMUNI],      [DPFVI],      [DMUFI];                    !
1!MATRIX      [DFDUI],      [DFDUNI],     [DFDUFI],     [DFSVI];                    !
1!UNSTRUCT    USETX(1000);                                                          !
1!UNSTRUCT    PRAAI;                                                                !
1!$                                                                                $!
1!$*****************************************************************************$!
1!$      AERODYNAMIC ENTITIES                                                      $!
1!$*****************************************************************************$!
1!$                                                                                $!
1!CHARACTER  METHOD,       QP,           USS;                                       !
1!CHARACTER  MODEL;                                                                 !
1!GGMEMBER    SAMODEL,      SAEMODEL,     STDYGEOM,     RIGDALOD,     AICMAT,       !
1!            SPLINE,       FLEXLOAD,     RIGDSLOD;                                 !
1!RGMEMBER    REFPARAM,     AEROGRID,     CAEROBOX,     SAGEOM,       SACOMPS;      !
1!MGMEMBER    [AIC],        [AAIC],       [ASAIC],      [AIRFRC],     [GTKG],       !
1!            [GSTKG],      [GPTKG],      [FLXFRC],     [FLXDEF],     [SLPFRC];     !
1!MGMEMBER    [KREALK],     [KIMAGK],     [KIMAGP],     [KIMAGS];                   !
1!MATRIX      [KREALT],     [KIMAGT];                                               !
1!$                                                                                $!
1!INTEGER     SYM,          AICSYM,       MINDEX,       SUB,          S,            !
1!            CASEID;                                                               !
1!INTEGER     SUBF;                                                                 !
1!REAL        QDP,          MACH;                                                   !
1!LOGICAL     LOOP,         GOAERO,       GOSPLINE,     AEFLG(1000);                !
1!LOGICAL     SAROONLY,     NEWMODEL;                                               !
1!UNSTRUCT    ACPT,         UNMK;                                                   !
1!RELATION    AESURF,       AIRFOIL,      AEROS,        AEFACT,       AXSTA,        !
1!            BODY,         SPLINE1,      SET1,         SET2,         ATTACH,       !
1!            TRIM,         AERO,         BLAST,        CAERO6,       PAERO6,       !
1!            GEOMSA,       AECOMPS,      CAERO1,       PAERO1,                     !
1!            CAERO2,       PAERO2,       MKAERO1,      MKAERO2,      FLUTTER,      !
1!            FLFACT,       CLAMBDA,      CONEFFS,      CONLINK,      GEOMUA,       !
1!            AECOMPU,      SPLINE2,      CONEFFF,      AEROGEOM,     CAROGEOM,     !
1!            AERUGEOM,     CAROUGEO,     TRIMDATA;                                 !
1!RELATION    STABCFA,      STABCFS,      SLPARM;                                   !
1!LOGICAL     DOTRMCON;                                                             !
1!LOGICAL     FULAERO;                                                              !
1!$                                                                                $!
1!RELATION    SPLINE3,      PANLST1,      PANLST2;                                  !
1!$-----------------------------------------------------------------------------$!
1!$ PROVISION FOR INCREMENT LOADS CALULATION                                        !
1!LOGICAL     YESAERO,      YESUDEF;                                                !
1!RELATION    TRIMTOC;                                                              !
1!UNSTRUCT    TLABEL;                                                               !
1!$ MATRICES CREATED FOR THE AERODYNAMIC DOMAIN                                     !
1!MATRIX      [AEROLOAD];                                                           !
1!MATRIX      [SAROLOAD],   [ACCELOAD],   [UDFALOAD];                               !
1!$                                                                                 !
1!$ MATRICES CREATED FOR THE STRUCTURAL DOMAIN                                     $!
1!MATRIX      [GPTKN],      [GPTKF];                                                !
1!$                                                                                $!
1!MATRIX      [UDGFORCE],   [UDNFORCE],   [UDFFORCE],   [UDFFORCX],   [ACCFORCE];   !
1!MATRIX      [PAFX];                                                               !
1!MATRIX      [AIRFORCE],                                                           !
1!            [AICS],       [KAFF],       [PAF],        [KAAA],       [PAA],        !
1!            [GASUBO(30,33)],            [SKJ],        [D1JK],       [D2JK],       !
1!            [KARL],       [R11],        [K21(30,33)], [PARBAR],     [PAL],        !
1!            [PAR(30,33)], [K1112(30,33)],             [K22],                      !
1!            [GTKN],       [GTKF],       [GSTKN],                                  !
1!            [GSTKF],      [GSKF],       [UGTKG],      [UGTKN],      [UGTKF],      !
1!            [UGTKA],      [UGTKO],      [UGTKAB],     [AITD],       [KARR],       !
1!            [R12(30,33)], [R22],        [R32(30,33)], [K11],        [K12(30,33)], !
1!            [P1],         [R21(30,33)], [R31(30,33)], [RL11(30,33)],              !
1!            [RU11(30,33)],              [P2],         [MAAA],       [IPMA(30,33)], !
1!            [R13(30,33)], [R33],        [DELC],       [PRIGID],                   !
1!            [AARC],       [AAR],        [AAA(1000)],  [UAA(1000)],  [AAAGC],      !
1!            [PAO(1000)],  [AAFTMP],     [UAFTMP],     [UAN],        [AAN],        !
```

```
1!              [UAG(1000)], [AAG(1000)], [AAL],        [AAF],        [UAF],          !
1!              [KOOL(30,33)],            [KOOU(30,33)],              [LHSA(30,33)],  !
1!              [POARO(30,33)],           [KAO(30,33)], [UAR],        [RHSA(30,33)],  !
1!              [DELTA(1000)],            [PAOC(1000)], [UAAC(1000)], [AAAC(1000)],   !
1!              [UAFC(1000)], [UANC(1000)], [UAGC(30,33)],            [AAFC(1000)],   !
1!              [AANC(1000)], [AAGC(30,33)],            [KL11(30,33)], [KU11(30,33)], !
1!              [R11DPL],     [R11PAL(30,33)],          [R1112(30,33)],               !
1!              [R1113(30,33)],           [KAOT(30,33)],              [UAL];          !
1!MATRIX        [KOOPOA],     [KOOKAO],     [PAG],        [PAGI],                     !
1!              [ULFLX1],     [ULFLX2],     [UAFLX1],     [UAFLX2],                   !
1!              [UOFLX1],     [UOFLX2],     [UFFLX1],     [UFFLX2],     [FLXFRC1],    !
1!              [FLXFRC2];                                                            !
1!MATRIX        [FLXTMP],     [FLXTMP1],    [FLXTMP2],    [FLXTMP3],                  !
1!              [FLXTMPS],    [FLXTMPA],    [FLXKX1],     [FLXKX2],                   !
1!              [GTKGKR],     [GTKGKI],                                               !
1!              [FLXF1R],     [FLXF1I],     [FLXF2R],     [FLXF2I];                   !
1!MATRIX        [UFX1HRT],    [UFX1HIT],    [UFX1SUM],    [UFX1DIF],                  !
1!              [UFX2HRT],    [UFX2HIT],    [UFX2SUM],    [UFX2DIF],                  !
1!              [FLXF1SUM],   [FLXF1DIF],   [FLXF2SUM],   [FLXF2DIF];                 !
1!IMATRIX       [AJJTL],      [QJJL],       [QKKL],       [QHHL];                     !
1!$                                                                                  $!
1!$********************************************************************************$!
1!$     ADDITIONS FOR GENERALIZED TRIM AND TRIM OPTIMIZATION                         $!
1!$********************************************************************************$!
1!$                                                                                  $!
1!INTEGER       SCHITER;                                                              !
1!LOGICAL       SCHCNVG,      TRMRIGD;                                                !
1!RELATION      BMST,         BMST2,        DCONBMST,     RMASS,        SCHEDULE,     !
1!              TCONBMST,     TCONTRM,      TFUNC,        TODVPRM,      TOMPPARM,     !
1!              TRIMOPT,      TRIMR,        TRIMRSLT,     BMSTDATA,     OBMSTLOD;     !
1!$                                                                                  $!
1!$                                                                                  $!
1!$********************************************************************************$!
1!$     DYNAMIC RESPONSE DECLARATIONS                                                $!
1!$********************************************************************************$!
1!$                                                                                  $!
1!INTEGER       HSIZE(1000);                                                          !
1!UNSTRUCT      TFDATA,       ICDATA,       UDLOLY;                                   !
1!RELATION      LAMBDA,       OEIGS,        DLONLY,       DLOAD,        TABLED1,      !
1!              IC,           TLOAD1,       TLOAD2,       RLOAD1,       RLOAD2,       !
1!              TSTEP,        VSDAMP,       TABDMP1,      DLAGS,        TF,           !
1!              DMIG,         GUST,         FREQ,         FREQ1,        FREQ2,        !
1!              FFT,          FLUTREL;                                                !
1!MATRIX        [PHIKH],      [QHJL],       [QKJL],       [PHIA],       [MII],        !
1!              [PHIO],       [PHIF],       [PHIN],       [PHIG(1000)], [KHHT],       !
1!              [KHHF],       [BHH],        [MHH],        [PDT],        [PDF],        !
1!              [KDDT],       [KDDF],       [BDD],        [MDD],        [ICMATRIX],   !
1!              [UTRANA],     [UFREQA],     [UTRANI],     [UFREQI],     [UFREQE],     !
1!              [UTRANE],     [UTRANF],     [UFREQF],     [UTRANN],     [UFREQN],     !
1!              [UTRANG],     [UFREQG],     [MHHFL(30,33)],             [BHHFL(30,33)], !
1!              [QHHLFL(30,33)],            [KHHFL(30,33)];                           !
1!$                                                                                  $!
1!$********************************************************************************$!
1!$  DECLARATIONS FOR GENERALIZED DYNAMIC REDUCTION (GDR)                            $!
1!$********************************************************************************$!
1!$                                                                                  $!
1!INTEGER       LKSET,        LJSET,        NEIV,         GNORM,        NGDR,         !
1!              ASIZE,        LSIZE;                                                  !
1!REAL          FMAX;                                                                 !
1!RELATION      DYNRED;                                                               !
1!MATRIX        [PGDRG(1000)], [PHIOK],     [KOO],        [GGO],        [KSOO],       !
1!              [KOA],        [LSOO],       [PAJK],       [PFJK],       [UFGDR],      !
1!              [AFGDR],      [UJK],        [GTMP];                                   !
1!$                                                                                  $!
1!$********************************************************************************$!
1!$     BLAST RESPONSE DECLARATIONS                                                  $!
1!$********************************************************************************$!
1!$                                                                                  $!
1!REAL          BQDP;                                                                 !
1!MATRIX        [MPART],      [ID2],        [PHIE],       [PHIR],       [PHIB],       !
1!              [GENM],       [GENK],       [GENF],       [GENQ],       [GENQL],      !
1!              [DTSLP],      [FTF],        [QRE],        [QEE],        [KEQE],       !
1!              [LKQ],        [UKQ],        [GFR],        [GFE],        [BTEM],       !
1!              [BLSTJA],     [BLGTJA],     [BFRC],       [MATTR],      [MATSS],      !
1!              [KEE],        [DELB],       [DELM],       [URDB],       [GENFA],      !
1!              [DWNWSH],     [BLAS],       [SLPMOD],     [QRR],        [UBLASTI],    !
1!              [UBLASTG],    [UBLASTF];                                              !
1!$                                                                                  $!
1!$********************************************************************************$!
1!$                                                                                  $!
1!$                   BEGIN MAPOL SOLUTION SEQUENCE                                  $!
1!$                                                                                  $!
1!$********************************************************************************$!
1!$     PREFACE MODULES                                                              $!
1!$********************************************************************************$!
1!SINGOSET := 1;                                                                      !
1!SINGASET := 2;                                                                      !
1!SINGLSET := 3;                                                                      !
1!$                                                                                  $!
1!QP := "QUADPAN";                                                                    !
```

```
1!USS := "USSAERO";                                                        !
1!FULAERO := FALSE;                                                        !
1!$                                                                       $!
1!$      INITIALIZE SUBSCRIPT VALUES TO "1" TO AVOID RUN TIME PROBLEMS    $!
1!$                                                                       $!
1!SUB      := 1;                                                           !
1!SUBF     := 1;                                                           !
1!PRINT("LOG=('BEGIN PREFACE MODULES')");                                  !
1!CALL SOLUTION ( NUMOPTBC, NBNDCOND, K6ROT, MPS, MPE, OCS, OCE, FSDS, FSDE,  !
1!                MAXITER, MOVLIM, WINDOW, OCMOVLIM, ALPHA, CNVRGLIM,       !
1!                NRFAC, EPS, FDSTEP, TOLVALUE );                           !
1!CALL IFP ( GSIZEB, EIDTYPE );                                            !
1!$                                                                       $!
1!$            PROCESS THE FUNCTIONAL PACKET AND INSTANTIATE THE FUNCTIONS $!
1!$                                                                       $!
1!CALL FPKEVL ( EIDTYPE );                                                 !
1!CALL UTRPRG ( EIDTYPE );                                                 !
1!$                                                                       $!
1!$            GENERATE THE ELEMENT MATRICES                              $!
1!$                                                                       $!
1!PRINT("LOG=('ELEMENT MATRIX GENERATION')");                              !
1!$                                                                       $!
1!CALL MAKEST ( NDV, GLBDES, [PTRANS], [PMINT], [PMAXT], LOCLVAR,          !
1!              TFIXED, DESLINK );                                          !
1!$                                                                       $!
1!CALL EMG ( NDV, GSIZEB, K6ROT, GLBDES, LOCLVAR, [PTRANS], DESLINK, [SMAT],  !
1!           SMATCOL, DVCT, DVSIZE, KELM, MELM, TELM, TREF );               !
1!$                                                                       $!
1!IF NUMOPTBC = 0                                                          !
2!   CALL NLEMG ( 1, NDV, GSIZEB, GLBDES, LOCLVAR, [PTRANS], DESLINK,       !
2!              [NLSMAT], NLSMTCOL, DVCTD, DDVCT, DVSIZED, DDVSIZE, KELMD, DKELM,  !
2!              MELMD, DMELM, TELMD, DTELM, TREFD, FDSTEP );                !
1!$                                                                       $!
1!CALL PFBULK ( GSIZEB, EOSUMMRY, EODISC, GPFELEM );                       !
1!$                                                                       $!
1!$            ASSEMBLE THE ELEMENT MATRICES                              $!
1!$            TO THE SENSITIVITY MATRICES                                $!
1!$                                                                       $!
1!PRINT("LOG=('PHASE 1 ELEM. MATRIX ASSEMBLY')");                          !
1!CALL EMA1  ( NDV, CSTM, GENEL, DVCT, KELM, MELM, GMKCT0, DKVI0,          !
1!             GMMCT0, DMVI0, DWGH1 );                                      !
1!CALL UTUPRG ( KELM, MELM  );                                             !
1!IF NUMOPTBC = 0                                                          !
2!   CALL NLEMA1 ( 1, NDV, GLBDES, DVCTD, DDVCT, KELMD, DKELM, MELMD, DMELM,  !
2!              GMKCT0, DKVI0, GMMCT0, DMVI0, DWGH1, GMKCT, DKVI, GMMCT, DMVI,  !
2!              GMKCTG, DKVIG, GMMCTG, DMVIG, GMMCTD, DMVID, DGMMCT, DDMVI,  !
2!              DDWGH2 );                                                   !
1!$                                                                       $!
1!$            GENERATE THE SIMPLE LOAD VECTORS                           $!
1!$            AND LOAD SENSITIVITIES                                     $!
1!$                                                                       $!
1!PRINT("LOG=('PHASE 1 STATIC LOADS GENER.')");                            !
1!CALL LODGEN ( GSIZEB, GLBDES, DVCT, DVSIZE, GMMCT0, DMVI0, TELM, TREF,    !
1!              SMPLOD, [DPTHVI], [DPGRVI] );                               !
1!CALL UTRPRG ( DVCT  );                                                   !
1!IF NUMOPTBC = 0                                                          !
2!   CALL NLLODGEN ( GSIZEB, GLBDES, DVCTD, DDVCT, DVSIZED, DDVSIZE, GMMCTD,  !
2!                DGMMCT, DMVID, DDMVI, TELMD, DTELM, TREFD, [DPTHVD], [DDPTHV],  !
2!                [DPGRVD], [DDPGRV] );                                     !
1!$                                                                       $!
1!$            GENERATE THE STEADY AERODYNAMIC MODELS                     $!
1!$            BASED ON THE SAERO DISCIPLINES/IMPORT AND OVERLAY OPERATIONS  $!
1!$                                                                       $!
1!CALL TRIMCHEK ( CASE, TRIMDATA );                                        !
1!CALL GPIMPORT;                                                           !
1!LOOP    := TRUE;                                                         !
1!SAROONLY := FALSE;                                                       !
1!MINDEX := 0;                                                             !
1!WHILE LOOP DO                                                            !
2!   MINDEX := MINDEX + 1;                                                  !
2!   CALL AROGNDRV ( MINDEX, CASE, LOOP, GOAERO, CASEID, METHOD, MODEL, MACH,  !
2!                   SYM, AICSYM, SAMODEL, STDYGEOM, RIGDALOD, AICMAT,        !
2!                   AEROGRID, CAEROBOX, SACOMPS, SAGEOM,                     !
2!                   [AIC], [AAIC], [ASAIC],                                  !
2!                   RIGDSLOD, NEWMODEL, SAROONLY );                          !
2!   CALL SFORLD ( , GSIZEB, GLBDES, SMPLOD, [DPTHVI], [DPTHVD], [DPGRVI],   !
2!                   [DPGRVD], RIGDSLOD, [SLPFRC] );                          !
2!   IF GOAERO THEN                                                         !
3!      IF METHOD = QP THEN                                                 !
4!         PRINT("LOG=('QUADPAN AERODYNAMICS')");                           !
4!         CALL QUADPAN ( MODEL, CASEID, CASE, AICSYM, AEROGRID,            !
4!                        CAEROBOX, SACOMPS, SAGEOM, REFPARAM, [AIC], [AAIC],  !
4!                        [ASAIC], [AIRFRC], RIGDALOD, FULAERO );           !
4!      ELSE                                                                !
4!         IF METHOD = USS THEN                                             !
5!            PRINT("LOG=('USSAERO AERODYNAMICS')");                        !
5!            CALL USSAERO ( MINDEX, MODEL, CASEID, MACH, SYM, AICSYM,      !
5!                           SACOMPS, SAGEOM, REFPARAM,                     !
5!                           [AIC], [AAIC], [AIRFRC], RIGDALOD,             !
5!                           AEROGRID, CAEROBOX );                          !
```

```
5!          ELSE                                                                    !
5!             PRINT("(' USER FATAL ERROR: UNRECOGNIZED SAERO METHOD ',15A4)",     !
5!                   METHOD);                                                        !
5!             CALL EXIT;                                                            !
5!          ENDIF;                                                                   !
4!       ENDIF;                                                                      !
3!       CALL LODSAGRP ( SAMODEL, NEWMODEL, METHOD, MACH, SYM, STDYGEOM, RIGDALOD, !
3!                       AICMAT, AEROGRID, CAEROBOX, SACOMPS, SAGEOM,               !
3!                       [AIC], [AAIC], [ASAIC], RIGDSLOD, FULAERO );               !
3!    ENDIF;                                                                         !
2!ENDDO;                                                                             !
1!CALL GRPARCHV;                                                                     !
1!$                                                                                 $!
1!$             GENERATE THE SAEMODEL GROUP AND SPLINE GROUP                         $!
1!$                                                                                 $!
1!LOOP   := TRUE;                                                                    !
1!MINDEX := 0;                                                                       !
1!WHILE LOOP DO                                                                      !
2!    MINDEX := MINDEX + 1;                                                          !
2!    CALL SPLNGNDR ( MINDEX, CASE, LOOP, MODEL, SAEMODEL, SAMODEL,                  !
2!                    SAGEOM, SACOMPS, SPLINE, [GTKG], [GSTKG], [GPTKG],             !
2!                    FLEXLOAD, NEWMODEL, GOSPLINE,                                  !
2!                    [KREALK], [KIMAGK], [KIMAGS],[KIMAGP] );                       !
2!    IF GOSPLINE                                                                    !
3!       CALL SPLINES ( MODEL, GSIZEB, SAGEOM, SACOMPS, AEROS, [GTKG], [GSTKG],      !
3!                      [GPTKG], [KREALK], [KIMAGK], [KIMAGS], [KIMAGP] );           !
2!    CALL LODSPGRP ( NEWMODEL, GOSPLINE, MODEL, SAEMODEL, SPLINE,                   !
2!                    FLEXLOAD, [GTKG], [GSTKG], [GPTKG] );                          !
2!ENDDO;                                                                             !
1!CALL TRIMUPD ( TRIMDATA );                                                         !
1!CALL GRPARCHV;                                                                     !
1!$-----------------------------                                                    $!
1!$                                                                                 $!
1!IF SAROONLY CALL EXIT;                                                             !
1!$                                                                                 $!
1!$      PERFORM TRIM PREFACE OPERATIONS                                            $!
1!$                                                                                 $!
1!CALL PRETRM ( TRIMDATA, TRIMRSLT );                                                !
1!$                                                                                 $!
1!$             GENERATE THE UNSTEADY AIC MATRIX AND THE                            $!
1!$             UNSTEADY SPLINE TRANSFORMATION MATRIX                               $!
1!$                                                                                 $!
1!PRINT("LOG=('UNSTEADY AERODYNAMICS')");                                            !
1!CALL UNSTEADY ( GEOMUA, AECOMPU, [AJJTL], [D1JK], [D2JK], [SKJ],                   !
1!                AERUGEOM, CAROUGEO );                                              !
1!CALL AMP ( [AJJTL], [D1JK], [D2JK], [SKJ], [QKKL], [QKJL], [QJJL] );               !
1!CALL SPLINEU ( GSIZEB, GEOMUA, AECOMPU, AERO, [UGTKG] );                           !
1!$                                                                                 $!
1!$*******************************************************************************$!
1!$        BEGIN OPTIMIZATION LOOP                                                  $!
1!$*******************************************************************************$!
1!$                                                                                 $!
1!IF NUMOPTBC > 0 THEN                                                               !
2!    PRINT("LOG=('*****************************')");                                !
2!    PRINT("LOG=('BEGIN OPTIMIZATION')");                                           !
2!$                                                                                 $!
2!$   INITIALIZE MAPOL PARAMETERS                                                   $!
2!$                                                                                 $!
2!    GLBCNVRG := FALSE;                                                             !
2!    APPCNVRG := FALSE;                                                             !
2!$                                                                                 $!
2!$   BEGIN CONVERGENCE LOOP                                                        $!
2!$                                                                                 $!
2!    WHILE NOT GLBCNVRG AND NITER <= MAXITER DO                                     !
3!$                                                                                 $!
3!$      ASSEMBLE THE GLOBAL MATRICES                                               $!
3!$                                                                                 $!
3!       NITER := NITER + 1;                                                         !
3!       PRINT("LOG=('----------------------------')");                              !
3!       PRINT("LOG=('   DESIGN ITERATION ',I3)",NITER);                             !
3!       CALL APFLUSH;                                                               !
3!$                                                                                 $!
3!$      FLUSH LARGE ENTITIES TO REDUCE GROWTH IN DATABASE                          $!
3!$                                                                                 $!
3!       CALL UTRPRG ( CONST );                                                      !
3!       CALL UTRPRG ( GPFELEM, EOBAR, EOELAS, EOHEX1, EOHEX2, EOHEX3, EOQDMM1 );    !
3!       CALL UTRPRG ( EOQUAD4, EOROD, EOSHEAR, EOTRMEM, GPFDATA, EOTRIA3 );         !
3!       CALL UTRPRG ( OPNLBUCK, OEULBUCK, PDLIST );                                 !
3!       CALL UTRPRG ( STABCFA );                                                    !
3!       CALL UTMPRG ( [GLBSIG], [NLGLBSIG] );                                       !
3!       CALL UTMPRG ( [GLBSIGI], [NLGBSIGI] );                                      !
3!       NEWITER := TRUE;                                                            !
3!$                                                                                 $!
3!       CALL ITERINIT ( NITER, CONST, LAMBDA );                                     !
3!       CALL DVMOVLIM ( NITER, NDV, GLBDES, MOVLIM );                               !
3!       GPRINT := FALSE;                                                            !
3!       GPUNCH := FALSE;                                                            !
3!       LOADLDV := FALSE;                                                           !
3!       LPRINT := FALSE;                                                            !
3!       CALL GDVPRINT ( NITER, NDV, GLBDES, MOVLIM, , GPRINT );                     !
```

```
3!        CALL GDVPUNCH ( NITER, NDV, GLBDES, GPUNCH );                          !
3!        CALL LDVLOAD ( GLBDES, LOCLVAR, [PTRANS], OLOCALDV, NITER, NDV, ,      !
3!                       LOADLDV );                                              !
3!        CALL LDVPRINT ( OLOCALDV, NITER, , LPRINT );                           !
3!        CALL GDVRESP ( NITER, NDV, GLBDES );                                   !
3!        CALL MSWGRESP ( NITER, NDV, GLBDES, DESLINK );                         !
3!        CALL TCEVAL ( NITER, NDV, MOVLIM, WINDOW, GLBDES, LOCLVAR, [PMINT],    !
3!                      [PMAXT], [PTRANS], TFIXED, CONST );                      !
3!        CALL BCEVAL ( NITER, NDV, GLBDES, LOCLVAR, [PTRANS], CONST );          !
3!        CALL LAMINCON ( NITER, NDV, DCONLAM, DCONLMN, DCONPMN, TFIXED, GLBDES, !
3!                        LOCLVAR, [PTRANS], CONST );                            !
3!$                                                                             $!
3!        CALL NLEMG ( NITER, NDV, GSIZEB, GLBDES, LOCLVAR, [PTRANS], DESLINK,   !
3!                     [NLSMAT], NLSMTCOL, DVCTD, DDVCT, DVSIZED, DDVSIZE, KELMD, !
3!                     DKELM, MELMD, DMELM, TELMD, DTELM, TREFD, FDSTEP );       !
3!$                                                                             $!
3!        CALL NLEMA1 ( NITER, NDV, GLBDES, DVCTD, DDVCT, KELMD, DKELM,          !
3!              MELMD, DMELM, GMKCT0, DKVI0, GMMCT0, DMVI0, DWGH1, GMKCT, DKVI,  !
3!              GMMCT, DMVI, GMKCTG, DKVIG, GMMCTG, DMVIG, GMMCTD, DMVID,        !
3!              DGMMCT, DDMVI, DDWGH2 );                                         !
3!$                                                                             $!
3!        CALL NLLODGEN ( GSIZEB, GLBDES, DVCTD, DDVCT, DVSIZED, DDVSIZE,        !
3!                        GMMCTD, DGMMCT, DMVID, DDMVI, TELMD, DTELM, TREFD,     !
3!                        [DPTHVD], [DDPTHV], [DPGRVD], [DDPGRV] );              !
3!$                                                                             $!
3!        CALL EMA2 ( NITER, NDV, GSIZEB, GLBDES, GMKCTG, DKVIG, [K1GG],         !
3!                                              GMMCTG, DMVIG, [M1GG] );         !
3!$                                                                             $!
3!$       BEGIN BOUNDARY CONDITION LOOP FOR OPTIMIZATION                        $!
3!$                                                                             $!
3!        FOR BC = 1 TO NUMOPTBC DO                                              !
4!           CALL BCIDVAL ( BC, CASE, BCID );                                    !
4!           PRINT("LOG=('       BOUNDARY CONDITION ',I8)",BCID);                !
4!$                                                                             $!
4!$          ESTABLISH THE BASE USET AND PARTITIONING DATA FOR THE BC           $!
4!$          THIS DATA MUST BE RECREATED EACH ITERATION SINCE GDR CAN CHANGE IT $!
4!$                                                                             $!
4!           CALL MKUSET( BCID, GSIZEB, [YS(BC)], [TMN(BC)], [PGMN(BC)], [PNSF(BC)],!
4!                        [PFOA(BC)], [PARL(BC)], USET(BC) );                    !
4!$                                                                             $!
4!$          MAKE B.C.-DEPENDENT BGPDT FROM BASE, ADDING THE EXTRA POINTS FOR   $!
4!$          THIS B.C.                                                          $!
4!$                                                                             $!
4!           CALL BCBGPDT( BCID , GSIZEB , BGPDT(BC) , ESIZE(BC) );              !
4!           GSIZE      := GSIZEB;                                               !
4!           PSIZE(BC) := ESIZE(BC) + GSIZE;                                     !
4!$                                                                             $!
4!           IF NITER < 2   CALL AROSYMCK (CASE, BGPDT(BC), USET(BC), RELES, BC, !
5!                                         TOLVALUE, STRSYM );                   !
4!$                                                                             $!
4!$          PROCESS MATRICES, TRANSFER FUNCTIONS, AND INITIAL CONDITIONS FOR   $!
4!$          THIS B.C.                                                          $!
4!$                                                                             $!
4!           CALL BCBULK( BCID , PSIZE(BC) , BGPDT(BC) , USET(BC) );             !
4!$                                                                             $!
4!           CALL BOUND  ( BCID, GSIZE, ESIZE(BC), USET(BC), BLOAD, BMASS, DMODES, !
4!                         BMODES, BSAERO, BFLUTR, BDYN, BDRSP, BDTR, BMTR, BDFR, !
4!                         BMFR, BGUST, BBLAST, NMPC, NSPC, NOMIT, NRSET, NGDR ); !
4!$                                                                             $!
4!$          DETERMINE IF ANY M2GG/K2GG INPUT DATA ARE TO BE ADDED              $!
4!$                                                                             $!
4!           CALL NULLMAT ( [KGG], [MGG] );                                      !
4!           CALL MK2GG ( BCID, GSIZEB, [M2GG], M2GGFLAG, [K2GG], K2GGFLAG );    !
4!           IF M2GGFLAG THEN                                                    !
5!              [MGG] := [M1GG] + [M2GG];                                        !
5!           ELSE                                                                !
5!              [MGG] := [M1GG];                                                 !
5!           ENDIF;                                                              !
4!           IF K2GGFLAG THEN                                                    !
5!              [KGG] := [K1GG] + [K2GG];                                        !
5!           ELSE                                                                !
5!              [KGG] := [K1GG];                                                 !
5!           ENDIF;                                                              !
4!$                                                                             $!
4!$          CALL THE GRID POINT WEIGHT GENERATOR FOR THIS BOUNDARY CONDITON    $!
4!$                                                                             $!
4!           CALL GPWG ( NITER, BCID, GPWGGRID, [MGG], OGPWG );                  !
4!$                                                                             $!
4!           IF BLOAD <> 0 CALL GTLOAD (NITER, BCID, GSIZE, BGPDT(BC), GLBDES,   !
5!                                      SMPLOD, [DPTHVI], [DPTHVD],              !
5!                                      [DPGRVI], [DPGRVD], [PG], OGRIDLOD);     !
4!$                                                                             $!
4!$          PARTITION-REDUCTION OF GLOBAL MATRICES                             $!
4!$                                                                             $!
4!           IF NUMOPTBC > 1  CALL NULLMAT ( [KNN], [PN], [MNN],                 !
5!                                 [GTKN], [GSTKN], [GPTKN], [UGTKN] );          !
4!           IF NMPC <> 0 THEN                                                   !
5!$                                                                             $!
5!$             PERFORM MPC REDUCTION                                           $!
5!$                                                                             $!
```

```
5!         PRINT("LOG=('          MPC REDUCTION')");                                     !
5!         CALL GREDUCE ( [KGG], [PG], [PGMN(BC)], [TMN(BC)], [KNN], [PN] );             !
5!         IF BMASS <> 0  CALL GREDUCE ( [MGG],,[PGMN(BC)], [TMN(BC)], [MNN] );!
5!         IF BSAERO <> 0 THEN                                                           !
6!            CALL SPLINFND ( BCID, CASE, MODEL, SPLINE, [GTKG], [GSTKG],                !
6!                            [GPTKG] );                                                 !
6!            CALL GREDUCE (, [GTKG], [PGMN(BC)], [TMN(BC)], , [GTKN]);                  !
6!            CALL GREDUCE (, [GSTKG], [PGMN(BC)], [TMN(BC)], , [GSTKN]);                !
6!            CALL GREDUCE (, [GPTKG], [PGMN(BC)], [TMN(BC)], , [GPTKN]);                !
6!         ENDIF;                                                                        !
5!         IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0                                   !
6!            CALL GREDUCE (, [UGTKG], [PGMN(BC)], [TMN(BC)], ,[UGTKN] );                !
5!      ELSE                                                                             !
5!$                                                                                     $!
5!$        NO MPC REDUCTION                                                             $!
5!$                                                                                     $!
5!         [KNN] := [KGG];                                                               !
5!         IF BLOAD <> 0 [PN] := [PG];                                                   !
5!         IF BMASS <> 0 [MNN] := [MGG];                                                 !
5!         IF BSAERO <> 0 THEN                                                           !
6!            [GTKN]  := [GTKG];                                                         !
6!            [GSTKN] := [GSTKG];                                                        !
6!            [GPTKN] := [GPTKG];                                                        !
6!         ENDIF;                                                                        !
5!         IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0 [UGTKN]  := [UGTKG];              !
5!      ENDIF;                                                                           !
4!$                                                                                     $!
4!$     PERFORM AUTOSPC CALCULATIONS ON THE KNN MATRIX                                  $!
4!$                                                                                     $!
4!      PRINT("LOG=('          AUTOSPC COMPUTATIONS')");                                 !
4!      CALL GPSP ( NITER, BCID, NGDR, [KNN], BGPDT(BC), [YS(BC)],                       !
4!                  USET(BC), GPST(BC) );                                                !
4!      CALL MKPVECT ( USET(BC), [PGMN(BC)], [PNSF(BC)],                                 !
4!                               [PFOA(BC)], [PARL(BC)] );                               !
4!      CALL BOUNDUPD ( BCID, GSIZE, ESIZE(BC), USET(BC), NSPC, NOMIT, NRSET );!
4!$                                                                                     $!
4!$     FOR SENSITIVITY ANALYSIS, SAVE A COPY OF THE PRE-GDR PART. VECTS.               $!
4!$                                                                                     $!
4!      CALL MKPVECT ( USET(BC), [PGMNS(BC)], [PNSFS(BC)],                               !
4!                               [PFOAS(BC)], [PARLS(BC)] );                             !
4!$                                                                                     $!
4!      IF NUMOPTBC > 1 CALL NULLMAT ( [KFF], [PF], [MFF], [GTKF], [GSTKF],              !
5!                                     [GPTKF], [UGTKF], [KFFX], [PFX],                  !
5!                                     [MFFX] );                                         !
4!      IF NSPC <> 0 THEN                                                                !
5!$                                                                                     $!
5!$        PERFORM SPC REDUCTION                                                        $!
5!$                                                                                     $!
5!         PRINT("LOG=('          SPC REDUCTION')");                                     !
5!         CALL NREDUCE ( [KNN], [PN], [PNSF(BC)], [YS(BC)], [KFF], [KFS],               !
5!                        [KSS], [PF], [PS] );                                           !
5!         IF BMASS <> 0 CALL NREDUCE ( [MNN], , [PNSF(BC)], , [MFF] );                  !
5!         IF BSAERO <> 0 THEN                                                           !
6!            CALL NREDUCE ( , [GTKN], [PNSF(BC)], , , , , [GTKF] );                     !
6!            CALL NREDUCE ( , [GSTKN], [PNSF(BC)], , , , , [GSTKF] );                   !
6!            CALL NREDUCE ( , [GPTKN], [PNSF(BC)], , , , , [GPTKF] );                   !
6!         ENDIF;                                                                        !
5!         IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0                                   !
6!            CALL NREDUCE (,[UGTKN], [PNSF(BC)],,,,, [UGTKF]);                          !
5!      ELSE                                                                             !
5!$                                                                                     $!
5!$        NO SPC REDUCTION                                                             $!
5!$                                                                                     $!
5!         [KFF] := [KNN];                                                               !
5!         IF BLOAD <> 0 [PF] := [PN];                                                   !
5!         IF BMASS <> 0 [MFF] := [MNN];                                                 !
5!         IF BSAERO <> 0 THEN                                                           !
6!            [GTKF]  := [GTKN];                                                         !
6!            [GSTKF] := [GSTKN];                                                        !
6!            [GPTKF] := [GPTKN];                                                        !
6!         ENDIF;                                                                        !
5!         IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0 [UGTKF]  := [UGTKN];              !
5!      ENDIF;                                                                           !
4!$                                                                                     $!
4!$     ADD IN THE NEW MODULE TO GENERATE THE H MATRIX FOR SYM-TRAN                     $!
4!$                                                                                     $!
4!      CALL SAERODRV (BCID, 1, LOOP, MINDEX, SYM, MACH, QDP,                            !
4!                     TRIMDATA, TRIMRSLT, METHOD );                                     !
4!      SYMTRN(BC) := FALSE;                                                             !
4!      ACSMTR(BC) := FALSE;                                                             !
4!      IF METHOD=USS THEN                                                               !
5!         IF STRSYM THEN                                                                !
6!            IF SYM=0 THEN                                                              !
7!               SYMTRN(BC) := TRUE;                                                     !
7!            ENDIF;                                                                     !
6!         ENDIF;                                                                        !
5!      ELSE                                                                             !
5!         IF METHOD=QP THEN                                                             !
6!            IF STRSYM THEN                                                             !
7!               IF SYM=0 OR SYM=-1 THEN                                                 !
```

```
8!                  SYMTRN(BC) := TRUE;                                          !
8!                  ACSMTR(BC) := TRUE;                                          !
8!               ENDIF;                                                          !
7!            ENDIF;                                                             !
6!         ENDIF;                                                                !
5!      ENDIF;                                                                   !
4!      IF SYMTRN(BC) THEN                                                       !
5!         CALL RBMGEN ( BGPDT(BC), 20, [RGBDMG] );                              !
5!         CALL AROHGEN (CASE, BGPDT(BC), USET(BC), RELES, BC, TOLVALUE,         !
5!                       [HFREALT(BC)], [HFIMAGT(BC)], USETX(BC) );              !
5!         CALL MKPVECT ( USETX(BC), [PGMNX(BC)], [PNSFX(BC)],                   !
5!                        [PFOAX(BC)], [PARLX(BC)] );                            !
5!         CALL TRNSPOSE ( [HFREALT(BC)], [HFREAL(BC)] );                        !
5!         CALL TRNSPOSE ( [HFIMAGT(BC)], [HFIMAG(BC)] );                        !
5!         IF ACSMTR(BC) THEN                                                    !
6!            CALL TRNSPOSE ( [KREALK], [KREALT] );                              !
6!            CALL TRNSPOSE ( [KIMAGK], [KIMAGT] );                              !
6!         ENDIF;                                                                !
5!         [HTKFHR]  := [HFREALT(BC)] * [ [KFF] * [HFREAL(BC)] ];                !
5!         [HTKFHI]  := [HFIMAGT(BC)] * [ [KFF] * [HFIMAG(BC)] ];                !
5!         [KFFX]    := [HTKFHR] + [HTKFHI];                                     !
5!         IF BMASS <> 0 THEN                                                    !
6!            [HTMFHR] := [HFREALT(BC)] * [ [MFF] * [HFREAL(BC)] ];              !
6!            [HTMFHI] := [HFIMAGT(BC)] * [ [MFF] * [HFIMAG(BC)] ];              !
6!            [MFFX]   := [HTMFHR] + [HTMFHI];                                   !
6!         ENDIF;                                                                !
5!         IF BSAERO = 0 THEN                                                    !
6!            [HFRTPF]  := [HFREALT(BC)] * [PF];                                 !
6!            [HFITPF]  := [HFIMAGT(BC)] * [PF];                                 !
6!            [PFX]     := [HFRTPF] + [HFITPF];                                  !
6!         ELSE                                                                  !
6!            IF ACSMTR(BC) THEN                                                 !
7!               [HRGTKF]   := [HFREALT(BC)] * [ [GTKF] * [KREALK] ];            !
7!               [HIGTKF]   := [HFIMAGT(BC)] * [ [GTKF] * [KIMAGK] ];            !
7!               [HRGSTKF]  := [HFREALT(BC)] * [ [GSTKF] * [KREALK] ];           !
7!               [HIGSTKF]  := [HFIMAGT(BC)] * [ [GSTKF] * [KIMAGS] ];           !
7!               [HRGPTKF]  := [HFREALT(BC)] * [ [GPTKF] * [KREALK] ];           !
7!               [HIGPTKF]  := [HFIMAGT(BC)] * [ [GPTKF] * [KIMAGP] ];           !
7!            ELSE                                                               !
7!               [HRGTKF]   := [HFREALT(BC)] * [GTKF];                           !
7!               [HIGTKF]   := [HFIMAGT(BC)] * [GTKF] ;                          !
7!               [HRGSTKF]  := [HFREALT(BC)] * [GSTKF];                          !
7!               [HIGSTKF]  := [HFIMAGT(BC)] * [GSTKF] ;                         !
7!               [HRGPTKF]  := [HFREALT(BC)] * [GPTKF];                          !
7!               [HIGPTKF]  := [HFIMAGT(BC)] * [GPTKF] ;                         !
7!            ENDIF;                                                             !
6!         ENDIF;                                                                !
5!      ELSE                                                                     !
5!         [KFFX]      := [KFF];                                                 !
5!         [MFFX]      := [MFF];                                                 !
5!         [PFX]       := [PF];                                                  !
5!         [GPTKFX]    := [GPTKF];                                               !
5!         USETX(BC)  := USET(BC);                                               !
5!         [PGMNX(BC)] := [PGMN(BC)];                                            !
5!         [PNSFX(BC)] := [PNSF(BC)];                                            !
5!         [PFOAX(BC)] := [PFOA(BC)];                                            !
5!         [PARLX(BC)] := [PARL(BC)];                                            !
5!      ENDIF;                                                                   !
4!$                                                                             $!
4!      IF NUMOPTBC > 1 CALL NULLMAT ( [KAA], [PA], [MAA],                       !
5!                                     [KAAA], [PAA], [UGTKA] );                 !
4!$                                                                             $!
4!      IF NGDR <> 0 THEN                                                        !
5!$                                                                             $!
5!$        PERFORM THE GENERAL DYNAMIC REDUCTION WHICH IS DISCIPLINE            $!
5!$        INDEPENDENT.  THE RESULTING [GSUBO] MATRIX WILL BE USED BY           $!
5!$        ALL DISCIPLINES                                                      $!
5!$                                                                             $!
5!         PRINT("LOG=('          DYNAMIC REDUCTION')");                         !
5!$                                                                             $!
5!$        OBTAIN THE OMITTED DOF PARTITION OF KFF AND MFF                      $!
5!$                                                                             $!
5!         CALL PARTN ( [KFFX], [KOO], , [KOA], , [PFOAX(BC)] );                 !
5!         CALL PARTN ( [MFFX], [MOO], , , , [PFOAX(BC)] );                      !
5!         ASIZE := GSIZE - NMPC - NSPC - NOMIT;                                 !
5!         LSIZE := ASIZE - NRSET;                                               !
5!         CALL GDR1 ( [KOO], [MOO], [KSOO], [GGO], LKSET, LJSET, NEIV,          !
5!                     FMAX, BCID, BGPDT(BC), USETX(BC), NOMIT, LSIZE );         !
5!$                                                                             $!
5!$        LKSET            MEANING                                             $!
5!$              <> 0       APPROX. MODE SHAPES SELECTED                        $!
5!$               = 0       NO APPROX. MODE SHAPES IN GDR                       $!
5!$                                                                             $!
5!         IF LKSET <> 0  THEN                                                   !
6!            CALL SDCOMP ( [KSOO], [LSOO], USETX(BC), SINGOSET );               !
6!            CALL GDR2   ( [LSOO], [MOO], [PHIOK], LKSET, LJSET,                !
6!                          NEIV,   FMAX,  BCID );                               !
6!         ENDIF;                                                                !
5!         CALL GDR3 ( [KOO], [KOA], [MGG], [PHIOK], [TMN(BC)], [GGO],           !
5!                     [PGMNX(BC)], [PNSFX(BC)], [PFOAX(BC)], [GSUBO(BC)],       !
```

```
5!                BGPDT(BC), USETX(BC),                                      !
5!                LKSET, LJSET, ASIZE, GNORM, BCID );                        !
5!        CALL GDR4 ( BCID, GSIZE, PSIZE(BC), LKSET, LJSET,                  !
5!                [PGMNX(BC)], [TMN(BC)], [PNSFX(BC)], [PFOAX(BC)],          !
5!                [PARLX(BC)], [PGDRG(BC)], [PAJK], [PFJK], BGPDT(BC),       !
5!                USETX(BC) );                                               !
5!     ENDIF;                                                                !
4!$                                                                         $!
4!     IF BLOAD <> 0 OR BMODES <> 0 OR BFLUTR <> 0 OR BDYN <> 0 THEN         !
5!$                                                                         $!
5!$       REDUCE THE MATRICES WITHOUT AEROELASTIC CORRECTIONS               $!
5!$                                                                         $!
5!        IF NGDR <> 0 THEN                                                  !
6!$                                                                         $!
6!$          PERFORM THE GENERAL DYNAMIC REDUCTION                          $!
6!$                                                                         $!
6!           PRINT("LOG=('          SYMMETRIC DYNAMIC REDUCTION')");         !
6!$                                                                         $!
6!           [MAA] := TRANS ( [GSUBO(BC)] ) * [ [MFFX] * [GSUBO(BC)] ];      !
6!           [KAA] := TRANS ( [GSUBO(BC)] ) * [ [KFFX] * [GSUBO(BC)] ];      !
6!           IF BLOAD <> 0  [PA]    := TRANS ( [GSUBO(BC)] ) * [PFX];        !
6!           IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0 THEN                !
7!              [TMP1]   := TRANS ( [UGTKF] ) * [GSUBO(BC)];                 !
7!              CALL TRNSPOSE ( [TMP1], [UGTKA] );                           !
7!           ENDIF;                                                          !
6!        ELSE                                                               !
6!           IF NOMIT <> 0 THEN                                              !
7!$                                                                         $!
7!$             PERFORM THE STATIC REDUCTION                                $!
7!$                                                                         $!
7!              PRINT("LOG=('          STATIC CONDENSATION')");              !
7!$                                                                         $!
7!              CALL FREDUCE ( [KFFX], [PFX], [PFOAX(BC)], ,[KOOINV(BC)], , , !
7!                             [GSUBO(BC)], [KAA], [PA], [PO], USETX(BC) );  !
7!$                                                                         $!
7!              IF BMASS <> 0 THEN                                           !
8!$                                                                         $!
8!$                PERFORM GUYAN REDUCTION OF THE MASS MATRIX               $!
8!$                                                                         $!
8!                 CALL PARTN ( [MFFX], [MOO], , [MOA], [MAABAR],            !
8!                              [PFOAX(BC)] );                               !
8!                 [MAA] := [MAABAR] + TRANS([MOA]) * [GSUBO(BC)] +          !
8!                          TRANS([GSUBO(BC)]) * [MOA] +                     !
8!                          TRANS([GSUBO(BC)]) * [ [MOO] * [GSUBO(BC)] ];    !
8!                 IF NRSET <> 0 [IFM(BC)] := [MOO] * [GSUBO(BC)] + [MOA];   !
8!              ENDIF;                                                       !
7!              IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0 THEN             !
8!                 CALL ROWPART ( [UGTKF], [UGTKO], [UGTKAB], [PFOAX(BC)] ); !
8!                 [TMP1]   := TRANS( [UGTKO] ) * [GSUBO(BC)];               !
8!                 CALL TRNSPOSE ( [TMP1], [TMP2] );                         !
8!                 [UGTKA] := [UGTKAB] + [TMP2];                             !
8!              ENDIF;                                                       !
7!           ELSE                                                            !
7!$                                                                         $!
7!$             NO F-SET REDUCTION                                          $!
7!$                                                                         $!
7!              [KAA] := [KFFX];                                             !
7!              IF BLOAD <> 0 [PA] := [PFX];                                 !
7!              IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0 [UGTKA]:=[UGTKF]; !
7!              IF BMASS <> 0 [MAA] := [MFFX];                               !
7!           ENDIF;                                                          !
6!        ENDIF;                                                             !
5!$                                                                         $!
5!        IF NRSET <> 0 THEN                                                 !
6!$                                                                         $!
6!$          PERFORM THE SUPPORT SET REDUCTION                              $!
6!$                                                                         $!
6!           PRINT("LOG=('          SUPPORT REDUCTION')");                   !
6!           IF NITER = 1 THEN                                               !
7!              CALL PARTN  ( [KAA], [KRR], [KLR], , [KLL], [PARLX(BC)] );   !
7!              CALL SDCOMP ( [KLL], [KLLINV(BC)], USETX(BC), SINGLSET );    !
7!              CALL FBS ( [KLLINV(BC)], [KLR], [D(BC)], -1 );               !
7!              CALL RBCHECK ( BCID, USETX(BC), BGPDT(BC), [D(BC)], [KLL],   !
7!                             [KRR], [KLR] );                               !
7!           ELSE                                                            !
7!              IF BLOAD <> 0 THEN                                           !
8!                 CALL PARTN  ( [KAA], , [KLR], , [KLL], [PARLX(BC)] );     !
8!                 CALL SDCOMP ( [KLL], [KLLINV(BC)], USETX(BC), SINGLSET ); !
8!              ENDIF;                                                       !
7!           ENDIF;                                                          !
6!$                                                                         $!
6!$          CALCULATE THE REDUCED MASS MATRIX                              $!
6!$                                                                         $!
6!           CALL PARTN ([MAA], [MRRBAR], [MLR], , [MLL], [PARLX(BC)]);      !
6!           [IFR(BC)] := [MLL] * [D(BC)] + [MLR];                           !
6!           [MRR(BC)] := [MRRBAR] + TRANS ( [MLR] )    * [D(BC)] +          !
6!                                   TRANS ( [D(BC)] ) * [IFR(BC)];          !
6!           [R22]     := TRANS ( [D(BC)] ) * [MLR] + [MRRBAR];              !
6!$                                                                         $!
6!           IF BLOAD <> 0 THEN                                              !
```

```
7!$                                                                      $!
7!$             PROCESS STATICS WITH INERTIA RELIEF                      $!
7!$                                                                      $!
7!              PRINT(                                                    !
                 "LOG=('         >>>DISCIPLINE: STATICS(INERTIA RELIEF)')"); !
7!              CALL ROWPART ( [PA], [PR], [PLBAR], [PARLX(BC)] );        !
7!              [LHS(BC)] := [MRR(BC)];                                   !
7!              [RHS(BC)] := TRANS([D(BC)]) * [PLBAR] + [PR];             !
7!              CALL INERTIA ( [LHS(BC)], [RHS(BC)], [AR] );              !
7!              [AL]      := [D(BC)] * [AR];                              !
7!              CALL ROWMERGE ( [AA], [AR], [AL], [PARLX(BC)] );          !
7!              [RHS(BC)] := [PLBAR] - [IFR(BC)] * [AR];                  !
7!              CALL FBS ( [KLLINV(BC)], [RHS(BC)], [UL] );               !
7!              CALL YSMERGE ( [UA], , [UL], [PARLX(BC)] );               !
7!           ENDIF;                                                       !
6!           IF BMODES <> 0 THEN                                          !
7!              PRINT("LOG=('         >>>DISCIPLINE: NORMAL MODES')");    !
7!              CALL REIG ( NITER, BCID, USETX(BC), [KAA], [MAA], [MRR(BC)], !
7!                          [D(BC)], LAMBDA, [PHIA], [MII], HSIZE(BC) );  !
7!              CALL OFPMROOT ( NITER, BCID, LAMBDA );                    !
7!              CALL FCEVAL ( NITER, BCID, LAMBDA, CONST );               !
7!           ENDIF;                                                       !
6!        ELSE                                                            !
6!$                                                                      $!
6!$          NO SUPPORT SET REDUCTION                                    $!
6!$                                                                      $!
6!           IF BLOAD <> 0 THEN                                           !
7!              PRINT("LOG=('         >>>DISCIPLINE: STATICS')");         !
7!              CALL SDCOMP ( [KAA], [KLLINV(BC)], USETX(BC), SINGASET  ); !
7!              CALL FBS ( [KLLINV(BC)], [PA], [UA] );                    !
7!           ENDIF;                                                       !
6!           IF BMODES <> 0 THEN                                          !
7!              PRINT("LOG=('         >>>DISCIPLINE: NORMAL MODES')");    !
7!              CALL REIG ( NITER, BCID, USETX(BC), [KAA], [MAA], , , LAMBDA, !
7!                          [PHIA], [MII], HSIZE(BC) );                   !
7!              CALL OFPMROOT ( NITER, BCID, LAMBDA );                    !
7!              CALL FCEVAL ( NITER, BCID, LAMBDA, CONST );               !
7!           ENDIF;                                                       !
6!        ENDIF;                                                          !
5!     ENDIF;                                                             !
4!     IF BSAERO <> 0 THEN                                                !
5!$                                                                      $!
5!$       PERFORM STATIC AEROELASTIC ANALYSES                            $!
5!$                                                                      $!
5!        PRINT("LOG=('         SAERO INITIALIZATION')");                 !
5!        CALL TRNSPOSE ( [GSTKF], [GSKF] );                              !
5!        IF SYMTRN(BC) THEN                                              !
6!           CALL TRNSPOSE ( [HRGSTKF], [GSKFHRT] );                      !
6!           CALL TRNSPOSE ( [HIGSTKF], [GSKFHIT] );                      !
6!        ENDIF;                                                          !
5!        LOOP := TRUE;                                                   !
5!        SUB  := 0;                                                      !
5!        WHILE LOOP DO                                                   !
6!           SUB := SUB + 1;                                              !
6!           CALL FTRIMDRV ( BCID, SUB, TRIMDATA, METHOD, MODEL, MACH,    !
6!                           SYM, SAMODEL, SAEMODEL, STDYGEOM, RIGDALOD,  !
6!                           RIGDSLOD, FLEXLOAD, AICMAT, AEROGRID,        !
6!                           CAEROBOX, SACOMPS, SAGEOM, [AIC], [AAIC],    !
6!                           [ASAIC] );                                   !
6!           CALL SAERODRV (BCID, SUB, LOOP, MINDEX, SYM, MACH, QDP,      !
6!                          TRIMDATA, TRIMRSLT, METHOD, 1 );              !
6!$                                                                      $!
6!$          ADJUST THE KFF MATRIX AND DETERMINE THE RIGID AIR LOADS     $!
6!$                                                                      $!
6!           IF SYMTRN(BC) THEN                                           !
7!             IF ACSMTR(BC) THEN                                         !
8!                [AICS1]  := [HRGTKF]*[TRANS([ASAIC])*[GSKFHRT]];        !
8!                [AICS2]  := [HRGTKF]*[TRANS([ASAIC])*[GSKFHIT]] + [AICS1]; !
8!                [AICS3]  := [HIGTKF]*[TRANS([ASAIC])*[GSKFHRT]] + [AICS2]; !
8!                [AICS]   := [HIGTKF]*[TRANS([ASAIC])*[GSKFHIT]] + [AICS3]; !
8!             ELSE                                                       !
8!                [AICSUM] := (0.5) [AIC] + (0.5)  [AAIC];                !
8!                [AICDIF] := (0.5) [AIC] - (0.5)  [AAIC];                !
8!                [AICS1]  := [HRGTKF]*[TRANS([AICSUM])*[GSKFHRT]];       !
8!                [AICS2]  := [HRGTKF]*[TRANS([AICDIF])*[GSKFHIT]] + [AICS1]; !
8!                [AICS3]  := [HIGTKF]*[TRANS([AICDIF])*[GSKFHRT]] + [AICS2]; !
8!                [AICS]   := [HIGTKF]*[TRANS([AICSUM])*[GSKFHIT]] + [AICS3]; !
8!              ENDIF;                                                    !
7!           ELSE                                                         !
7!             IF SYM =  1 [AICS]:= [GTKF]*[TRANS([AIC])*[GSKF]];         !
7!             IF SYM = -1 [AICS]:= [GTKF]*[TRANS([AAIC])*[GSKF]];        !
7!$                                                                      $!
7!$    ADD IN OPTION FOR ASYMMETRIC AIC                                  $!
7!$                                                                      $!
7!             IF SYM =  0 [AICS] := [GTKF]*[TRANS([ASAIC])*[GSKF]];      !
7!           ENDIF;                                                       !
6!$                                                                      $!
6!$                                                                      $!
6!$    DEFINE ZERO LOAD VECTOR FOR ACCELERATION PARAMETERS               $!
6!$                                                                      $!
```

```
6!            CALL ACCFGEN (NITER, BCID, SUB, SYMTRN(BC), RIGDALOD, RIGDSLOD,  !
6!                          STDYGEOM, TRIMDATA, CONLINK, TRIMTOC, [KFFX],       !
6!                          TLABEL, [ACCFORCE], [ACCELOAD], MACH);              !
6!$                                                                            $!
6!$                                                                            $!
6!$     USER DEFINED LOADS FROM STATIC LOAD PARAMETER DEFINITION               $!
6!$                                                                            $!
6!            CALL UDEFGEN (NITER, BCID, SUB, SYMTRN(BC), RIGDSLOD, [UDGFORCE],!
6!                       [UDFALOAD], GSIZE, TLABEL, TRIMDATA, STDYGEOM,         !
6!                       TRIMTOC, MACH, YESUDEF);                               !
6!$                                                                            $!
6!            IF NMPC <> 0 THEN                                                 !
7!               CALL GREDUCE (, [UDGFORCE], [PGMN(BC)],[TMN(BC)],,[UDNFORCE]);!
7!            ELSE                                                              !
7!               [UDNFORCE] := [UDGFORCE];                                      !
7!            ENDIF;                                                            !
6!$                                                                            $!
6!            CALL NREDUCE ( , [UDNFORCE], [PNSF(BC)], , , , , [UDFFORCE] );    !
6!$                                                                            $!
6!            IF SYMTRN(BC) THEN                                                !
7!               CALL UDEFTRAN (NITER, BCID, SUB, [UDFFORCE], TRIMTOC,          !
7!                             [HFREALT(BC)], [HFIMAGT(BC)], [UDFFORCX] );      !
7!            ELSE                                                              !
7!               [UDFFORCX] := [UDFFORCE];                                      !
7!            ENDIF;                                                            !
6!$                                                                             !
6!$                                                                            $!
6!$     NEW AIR FORCE MERGE ROUTINE                                            $!
6!$                                                                            $!
6!            IF SYMTRN(BC) THEN                                                !
7!               CALL ARFMRG (NITER, BCID, SUB, SYMTRN(BC), TRIMDATA, TLABEL,   !
7!                          RIGDALOD, STDYGEOM, [HRGPTKF], [HIGPTKF], CASE,     !
7!                          [AEROLOAD], [AIRFORCE], TRIMTOC, MACH, YESAERO);    !
7!            ELSE                                                              !
7!               CALL ARFMRG (NITER, BCID, SUB, SYMTRN(BC), TRIMDATA, TLABEL,   !
7!                          RIGDALOD, STDYGEOM, [GPTKF], [HIGPTKF], CASE,       !
7!                          [AEROLOAD], [AIRFORCE], TRIMTOC, MACH, YESAERO);    !
7!            ENDIF;                                                            !
6!$                                                                            $!
6!$        MERGE LOADS IN THE AERODYNAMIC DOMAIN FOR STABILITY DERIVATIVES     $!
6!$                                                                            $!
6!            [SAROLOAD] := [ACCELOAD];                                         !
6!            IF YESUDEF THEN                                                   !
7!               CALL APPEND ( [UDFALOAD], [SAROLOAD] );                        !
7!            ENDIF;                                                            !
6!            IF YESAERO THEN                                                   !
7!               CALL APPEND ( [AEROLOAD], [SAROLOAD] );                        !
7!            ENDIF;                                                            !
6!            CALL RIGDSTAB ( BCID, SUB, TRIMTOC, [SAROLOAD], STDYGEOM,         !
6!                            BGPDT(BC), QDP, STABCFA );                        !
6!$                                                                            $!
6!$        MERGE LOADS IN THE STRUCTURAL DOMAIN FOR AEROLELATIC SOLUTION       $!
6!$                                                                            $!
6!            [PAF]   := [ACCFORCE];                                            !
6!            IF YESUDEF THEN                                                   !
7!               CALL APPEND ( [UDFFORCX], [PAF] );                             !
7!            ENDIF;                                                            !
6!            IF YESAERO THEN                                                   !
7!               [PAFX]  := (QDP) [AIRFORCE];                                   !
7!               CALL APPEND ( [PAFX], [PAF] );                                 !
7!            ENDIF;                                                            !
6!               CALL UTMPRG ( [PAFX] );                                        !
6!               [KAFF] := [KFFX] - (QDP) [AICS];                               !
6!$           CALL UTMPRG ( [AICS] );                                           !
6!$                                                                            $!
6!$           REDUCE THE MATRICES WITH AEROELASTIC CORRECTIONS                 $!
6!$           SAVE THE SUBCASE/BC DEPENDENT DATA FOR SENSITIVITY ANALYSIS      $!
6!$                                                                            $!
6!            IF NGDR <> 0 THEN                                                 !
7!$                                                                            $!
7!$              PERFORM THE GENERAL DYNAMIC REDUCTION                         $!
7!$                                                                            $!
7!               PRINT("LOG=('          SAERO DYNAMIC REDUCTION')");            !
7!               [MAAA] := TRANS ( [GSUBO(BC)] ) * [ [MFFX] * [GSUBO(BC)] ];    !
7!               [KAAA] := TRANS ( [GSUBO(BC)] ) * [ [KAFF] * [GSUBO(BC)] ];    !
7!               [PAA]  := TRANS ( [GSUBO(BC)] ) * [PAF];                       !
7!            ELSE                                                              !
7!               IF NOMIT <> 0 THEN                                             !
8!$                                                                            $!
8!$                 PERFORM THE STATIC REDUCTION                               $!
8!$                                                                            $!
8!                  PRINT("LOG=('          SAERO STATIC CONDENSATION')");       !
8!$                                                                            $!
8!                  IF NITER = 1  AND SUB = 1 AND NRSET <> 0 AND BLOAD = 0 AND !
9!                     BMODES = 0 AND BFLUTR = 0 AND BDYN = 0 THEN              !
9!$                                                                            $!
9!$                    FORM [KAA] ON FIRST PASS SO [D] CAN BE FORMED           $!
9!$                                                                            $!
9!                     CALL FREDUCE ([KFFX], ,[PFOAX(BC)], ,[KOOINV(BC)], , ,  !
9!                                   [GSUBO(BC)], [KAA], , , USETX(BC) );       !
```

```
9!                   ENDIF;                                                !
8!$                                                                       $!
8!                   CALL FREDUCE ( [KAFF], [PAF], [PFOAX(BC)], BSAERO,    !
8!                                  [KOOL(BC,SUB)], [KOOU(BC,SUB)],        !
8!                                  [KAO(BC,SUB)], [GASUBO(BC,SUB)], [KAAA], !
8!                                  [PAA], [POARO(BC,SUB)], USETX(BC));    !
8!$                                                                       $!
8!                   IF BMASS <> 0 THEN                                    !
9!$                                                                       $!
9!$                     PERFORM GUYAN REDUCTION OF THE MASS MATRIX        $!
9!$                                                                       $!
9!                      CALL PARTN ( [MFFX], [MOO], , [MOA], [MAABAR],     !
9!                                   [PFOAX(BC)] );                        !
9!                      [MAAA] := [MAABAR] + TRANS([MOA]) * [GASUBO(BC,SUB)] + !
9!                                TRANS([GASUBO(BC,SUB)]) * [MOA] +        !
9!                                TRANS([GASUBO(BC,SUB)]) * [[MOO] *       !
9!                                [GASUBO(BC,SUB)]];                       !
9!                      IF NRSET <> 0                                      !
10!                        [IFMA(BC,SUB)] := [MOO]*[GASUBO(BC,SUB)]+[MOA]; !
9!                   ENDIF;                                                !
8!                ELSE                                                     !
8!$                                                                       $!
8!$                  NO F-SET REDUCTION                                   $!
8!$                                                                       $!
8!                   IF NITER = 1  AND SUB = 1 AND NRSET <> 0 AND BLOAD = 0 AND !
9!                      BMODES = 0 AND BFLUTR = 0 AND BDYN = 0 THEN        !
9!$                                                                       $!
9!$                     FORM [KAA] ON FIRST PASS SO [D] CAN BE FORMED     $!
9!$                                                                       $!
9!                      [KAA] := [KFFX];                                   !
9!                   ENDIF;                                                !
8!                   [KAAA] := [KAFF];                                     !
8!                   [MAAA] := [MFFX];                                     !
8!                   [PAA]  := [PAF];                                      !
8!                ENDIF;                                                   !
7!             ENDIF;                                                      !
6!$                                                                       $!
6!             IF NRSET <> 0 THEN                                          !
7!$                                                                       $!
7!$               PERFORM THE SUPPORT SET REDUCTION                       $!
7!$                                                                       $!
7!                PRINT("LOG=('        SAERO SUPPORT REDUCTION')");        !
7!$                                                                       $!
7!                IF NITER = 1  AND SUB  = 1 AND BLOAD = 0 AND BMODES = 0 AND !
8!                   BFLUTR = 0 AND BDYN = 0 THEN                          !
8!$                                                                       $!
8!$                  [D] WAS NOT COMPUTED FOR NON-SAERO DISCIPLINES SO    $!
8!$                  NEED TO COMPUTE IT NOW                               $!
8!$                                                                       $!
8!                   CALL PARTN  ( [KAA], [KRR], [KLR], , [KLL], [PARLX(BC)] ); !
8!                   CALL SDCOMP ( [KLL], [KLLINV(BC)], USETX(BC), SINGLSET ); !
8!                   CALL FBS ( [KLLINV(BC)], [KLR], [D(BC)], -1 );        !
8!                   CALL RBCHECK ( BCID, USETX(BC), BGPDT(BC), [D(BC)], [KLL], !
8!                                  [KRR], [KLR] );                        !
8!                ENDIF;                                                   !
7!$                                                                       $!
7!$               CALCULATE THE REDUCED MASS MATRIX                       $!
7!$                                                                       $!
7!                CALL PARTN ([MAAA], [MRRBAR], [MLR], , [MLL], [PARLX(BC)]); !
7!                [R13(BC,SUB)] := [MLL] * [D(BC)] + [MLR];                !
7!                [R33]         := [MRRBAR] + TRANS ( [MLR] ) * [D(BC)] +  !
7!                                 TRANS ( [D(BC)] ) * [R13(BC,SUB)];      !
7!                [R22]         := TRANS ( [D(BC)] ) * [MLR] + [MRRBAR];   !
7!                CALL TRNSPOSE ( [R13(BC,SUB)], [R21(BC,SUB)] );          !
7!$                                                                       $!
7!$               PROCESS STEADY AEROELASTIC DISCIPLINE                   $!
7!$                                                                       $!
7!                PRINT("LOG=('        >>>DISCIPLINE: STEADY AERO')");     !
7!                CALL PARTN ( [KAAA], [KARR], [R12(BC,SUB)], [KARL], [R11], !
7!                             [PARLX(BC)] );                              !
7!                [R32(BC,SUB)] := TRANS([D(BC)]) * [R12(BC,SUB)] + [KARR]; !
7!                [R31(BC,SUB)] := TRANS([D(BC)]) * [R11] + [KARL];        !
7!$                                                                       $!
7!                CALL DECOMP ( [R11], [RL11(BC,SUB)], [RU11(BC,SUB)] );   !
7!$                                                                       $!
7!                CALL ROWPART ( [PAA], [PARBAR], [PAL], [PARLX(BC)] );    !
7!                CALL GFBS ( [RL11(BC,SUB)], [RU11(BC,SUB)], [PAL],       !
7!                            [R11PAL(BC,SUB)], -1);                       !
7!                [PRIGID] := [PARBAR] + TRANS([D(BC)]) * [PAL];           !
7!                [P1]     := [R21(BC,SUB)] * [R11PAL(BC,SUB)];            !
7!                [P2]     := [PRIGID] + [R31(BC,SUB)] * [R11PAL(BC,SUB)]; !
7!$                                                                       $!
7!                CALL GFBS ( [RL11(BC,SUB)], [RU11(BC,SUB)], [R12(BC,SUB)], !
7!                            [R1112(BC,SUB)], -1);                        !
7!                CALL GFBS ( [RL11(BC,SUB)], [RU11(BC,SUB)], [R13(BC,SUB)], !
7!                            [R1113(BC,SUB)], -1);                        !
7!                [K11]         := [R22] + [R21(BC,SUB)] * [R1112(BC,SUB)]; !
7!                [K12(BC,SUB)] := [R21(BC,SUB)] * [R1113(BC,SUB)];        !
7!                [K21(BC,SUB)] := [R32(BC,SUB)] +                         !
7!                                 [R31(BC,SUB)] * [R1112(BC,SUB)];        !
```

```
7!          [K22]          := [R33] + [R31(BC,SUB)] * [R1113(BC,SUB)];        !
7!$                                                                           $!
7!          CALL DECOMP ( [K11], [KL11(BC,SUB)], [KU11(BC,SUB)] );            !
7!          CALL GFBS ( [KL11(BC,SUB)], [KU11(BC,SUB)], [P1],                 !
7!                      [PAR(BC,SUB)] );                                      !
7!          CALL GFBS ( [KL11(BC,SUB)], [KU11(BC,SUB)], [K12(BC,SUB)],        !
7!                      [K1112(BC,SUB)],-1);                                  !
7!          [LHSA(BC,SUB)] := [K22] + [K21(BC,SUB)] * [K1112(BC,SUB)];        !
7!          [RHSA(BC,SUB)] := [P2] - [K21(BC,SUB)] * [PAR(BC,SUB)];           !
7!$                                                                           $!
7!$         FLEXIBLE STABILITY COEFFICIENTS COMPUTATION                       $!
7!$                                                                           $!
7!          CALL FLEXSTAB ( NITER, BCID, MINDEX, SUB, SYM, QDP, TRIMDATA,     !
7!                          STABCFA, STABCFS, BGPDT(BC), [LHSA(BC,SUB)],      !
7!                          [RHSA(BC,SUB)], [AAR], [DELTA(SUB)],              !
7!                          [PRIGID], [R33], CONST, AEFLG(SUB), [AARC],       !
7!                          [DELC], SYMTRN(BC), STDYGEOM, DOTRMCON );         !
7!$                                                                           $!
7!$         GENERATE FLEXIBLE STRUCTURAL TRIM PARAMETER LOAD VECTORS          $!
7!$         AND DEFLECTION VECTORS TO BE LOADED INTO GROUP FLEXLOAD           $!
7!$                                                                           $!
7!          CALL GFBS ( [RL11(BC,SUB)], [RU11(BC,SUB)], [R13(BC,SUB)],        !
7!                      [ULFLX2], -1);                                        !
7!          [ULFLX1] := - [R11PAL(BC,SUB)];                                   !
7!          CALL ROWMERGE ( [UAFLX1], , [ULFLX1], [PARLX(BC)] );              !
7!          CALL ROWMERGE ( [UAFLX2], , [ULFLX2], [PARLX(BC)] );              !
7!$                                                                           $!
7!          IF NOMIT <> 0 THEN                                                !
8!             CALL GFBS ( [KOOL(BC,SUB)], [KOOU(BC,SUB)],                    !
8!                         [POARO(BC,SUB)], [KOOPOA], 1);                     !
8!             CALL TRNSPOSE ( [KAO(BC,SUB)], [KAOT(BC,SUB)] );               !
8!             CALL GFBS ( [KOOL(BC,SUB)], [KOOU(BC,SUB)],                    !
8!                         [KAOT(BC,SUB)], [KOOKAO], -1);                     !
8!             [UOFLX1] := [KOOPOA] + [KOOKAO]*[UAFLX1];                      !
8!             [UOFLX2] := [KOOKAO]*[UAFLX2];                                 !
8!             CALL ROWMERGE ( [UFFLX1], [UOFLX1], [UAFLX1],                  !
8!                             [PFOAX(BC)] );                                 !
8!             CALL ROWMERGE ( [UFFLX2], [UOFLX2], [UAFLX2],                  !
8!                             [PFOAX(BC)] );                                 !
8!          ELSE                                                              !
8!             [UFFLX1] := [UAFLX1];                                          !
8!             [UFFLX2] := [UAFLX2];                                          !
8!          ENDIF;                                                            !
7!          IF SYMTRN(BC) THEN                                                !
8!             IF ACSMTR(BC) THEN                                             !
9!                [PAG] := (QDP)[GPTKG]*[KREALK]*[SAROLOAD];                  !
9!                [PAGI] := (QDP)[GPTKG]*[KIMAGK]*[SAROLOAD];                 !
9!             ELSE                                                           !
9!                [PAG] := (QDP)[GPTKG]*[SAROLOAD];                           !
9!             ENDIF;                                                         !
8!          ELSE                                                              !
8!                [PAG] := (QDP)[GPTKG]*[SAROLOAD];                           !
8!          ENDIF;                                                            !
7!$                                                                           $!
7!          IF SYMTRN(BC) THEN                                                !
8!                [UFX1HRT] := [HFREAL(BC)]*[UFFLX1];                         !
8!                [UFX1HIT] := [HFIMAG(BC)]*[UFFLX1];                         !
8!                [UFX2HRT] := [HFREAL(BC)]*[UFFLX2];                         !
8!                [UFX2HIT] := [HFIMAG(BC)]*[UFFLX2];                         !
8!$                                                                           !
8!                IF ACSMTR(BC) THEN                                          !
9!$                                                                           !
9!                [GTKGKR]   :=[GTKG]*[KREALK];                               !
9!                [GTKGKI]   :=[GTKG]*[KIMAGK];                               !
9!                [FLXTMP1]  :=[GSKF]*[HFREAL(BC)]+[GSKF]*[HFIMAG(BC)];       !
9!                [FLXTMP2]  :=[ [KREALT]+[KIMAGT] ]*[FLXTMP1];               !
9!                [FLXTMP3]  :=TRANS([ASAIC])*[FLXTMP2];                      !
9!                [FLXKX1]   :=[FLXTMP3]*[UFFLX1];                            !
9!                [FLXKX2]   :=[FLXTMP3]*[UFFLX2];                            !
9!$                                                                           !
9!                [FLXF1R] := (QDP)[GTKGKR]*[FLXKX1]  ;                       !
9!                [FLXF1I] := (QDP)[GTKGKI]*[FLXKX1]  ;                       !
9!                [FLXF2R] := (QDP)[GTKGKR]*[FLXKX2]  ;                       !
9!                [FLXF2I] := (QDP)[GTKGKI]*[FLXKX2]  ;                       !
9!             ELSE                                                           !
9!                [FLXTMPS]  :=(QDP)[GTKG]*[TRANS([AIC])*[GSKF]];             !
9!                [FLXTMPA]  :=(QDP)[GTKG]*[TRANS([AAIC])*[GSKF]];            !
9!$                                                                           !
9!                [UFX1SUM] := (0.5) [UFX1HRT] + (0.5) [UFX1HIT];             !
9!                [UFX1DIF] := (0.5) [UFX1HRT] - (0.5) [UFX1HIT];             !
9!                [UFX2SUM] := (0.5) [UFX2HRT] + (0.5) [UFX2HIT];             !
9!                [UFX2DIF] := (0.5) [UFX2HRT] - (0.5) [UFX2HIT];             !
9!$                                                                           !
9!                [FLXF1SUM]:= [FLXTMPS]*[UFX1SUM];                           !
9!                [FLXF1DIF]:= [FLXTMPA]*[UFX1DIF];                           !
9!                [FLXF2SUM]:= [FLXTMPS]*[UFX2SUM];                           !
9!                [FLXF2DIF]:= [FLXTMPA]*[UFX2DIF];                           !
9!$                                                                           !
9!                [FLXF1R] := [FLXF1SUM]+[FLXF1DIF] ;                         !
9!                [FLXF1I] := [FLXF1SUM]-[FLXF1DIF] ;                         !
```

```
9!                [FLXF2R] := [FLXF2SUM]+[FLXF2DIF] ;                      !
9!                [FLXF2I] := [FLXF2SUM]-[FLXF2DIF] ;                      !
9!             ENDIF;                                                      !
8!          ELSE                                                           !
8!             IF SYM =  1 THEN                                            !
9!                [FLXTMP]   :=(QDP)[GTKG]*[TRANS([AIC])*[GSKF]];          !
9!             ENDIF;                                                      !
8!             IF SYM = -1 THEN                                            !
9!                [FLXTMP]   :=(QDP)[GTKG]*[TRANS([AAIC])*[GSKF]];         !
9!             ENDIF;                                                      !
8!             IF SYM =  0 THEN                                            !
9!                [FLXTMP]   :=(QDP)[GTKG]*[TRANS([ASAIC])*[GSKF]];        !
9!             ENDIF;                                                      !
8!             [FLXFRC1] :=[FLXTMP]*[UPFLX1] ;                             !
8!             [FLXFRC2] :=[FLXTMP]*[UPFLX2] ;                             !
8!          ENDIF;                                                         !
7!$                                                                       $!
7!$         LOAD GROUP FLEXLOAD                                           $!
7!$                                                                       $!
7!          IF SYMTRN(BC) THEN                                             !
8!             CALL FLXLODLD ( BCID, SUB, TRIMTOC, TRIMDATA, [PAG],        !
8!                             [PAGI], [FLXF1R], [FLXF1I], [FLXF2R],       !
8!                             [FLXF2I], [MGG], [UFX1HRT], [UFX1HIT],      !
8!                             [UFX2HRT], [UFX2HIT], BGPDT(BC),            !
8!                             FLEXLOAD, SYMTRN(BC), ACSMTR(BC),           !
8!                             NEWITER );                                  !
8!          ELSE                                                           !
8!             CALL FLXLODLD ( BCID, SUB, TRIMTOC, TRIMDATA, [PAG], ,      !
8!                             [FLXFRC1], , [FLXFRC2], , [MGG],            !
8!                             [UPFLX1], , [UPFLX2], , BGPDT(BC),          !
8!                             FLEXLOAD, SYMTRN(BC), ACSMTR(BC),           !
8!                             NEWITER );                                  !
8!          ENDIF;                                                         !
7!$                                                                       $!
7!$         GENERATE TRIM PARAMETER BMST DATA                             $!
7!$                                                                       $!
7!          CALL PARMBMST ( TRIMTOC, FLEXLOAD, [PAG], [PAGI], [SAROLOAD], !
7!                          STDYGEOM, SYMTRN(BC), ACSMTR(BC), BMSTDATA ); !
7!$                                                                       $!
7!$         GENERALIZED TRIM AND TRIM OPTIMIZATION                        $!
7!$                                                                       $!
7!          SCHITER := 0;                                                  !
7!          SCHCNVG := FALSE;                                              !
7!          WHILE NOT SCHCNVG DO                                           !
8!             SCHITER := SCHITER + 1;                                     !
8!             CALL SCHDULER ( BCID, SUB, TRIMDATA, TRIMRSLT, TRIMTOC,     !
8!                             [AAR], [DELTA(SUB)], SCHITER,               !
8!                             SCHCNVG );                                  !
8!             CALL FLEXTRIM ( NITER, BCID, SUB, SYM, QDP,                 !
8!                             TRIMDATA, TRIMRSLT, TRIMTOC,                !
8!                             [LHSA(BC,SUB)], [RHSA(BC,SUB)],             !
8!                             [AAR], [DELTA(SUB)], [PRIGID], [R33] );     !
8!             CALL FTRIMOPT ( NITER, BCID, SUB, SYM, QDP,                 !
8!                             TRIMDATA, TRIMRSLT, TRIMTOC,                !
8!                             [LHSA(BC,SUB)], [RHSA(BC,SUB)],             !
8!                             [AAR], [DELTA(SUB)], [PRIGID], [R33],       !
8!                             BMSTDATA );                                 !
8!          ENDDO;                                                         !
7!$                                                                       $!
7!$         GENERATE TRIMMED BMST DATA                                    $!
7!$                                                                       $!
7!          TRMRIGD := FALSE;                                              !
7!          CALL OFPBMST ( NITER, BCID, SUB, QDP, [AAR], [DELTA(SUB)],     !
7!                         TRIMDATA, TRIMTOC, BMSTDATA, TRMRIGD,           !
7!                         OBMSTLOD );                                     !
7!$                                                                       $!
7!          [AAL] := [D(BC)] * [AAR];                                      !
7!          CALL ROWMERGE ( [AAA(SUB)], [AAR], [AAL], [PARLX(BC)] );       !
7!          [UAR] := [K1112(BC,SUB)] * [AAR] + [PAR(BC,SUB)] *             !
7!                   [DELTA(SUB)];                                         !
7!          [UAL] := [R1112(BC,SUB)] * [UAR] + [R1113(BC,SUB)] * [AAR]     !
7!                   - [R11PAL(BC,SUB)] * [DELTA(SUB)];                    !
7!          CALL ROWMERGE ( [UAA(SUB)], [UAR], [UAL], [PARLX(BC)] );       !
7!          IF NOMIT <> 0 [PAO(SUB)] := [POARO(BC,SUB)] * [DELTA(SUB)] ;   !
7!          IF AEFLG(SUB) THEN                                             !
8!             [AAL]          := [D(BC)] * [AARC];                         !
8!             CALL ROWMERGE ( [AAAC(SUB)], [AARC], [AAL], [PARLX(BC)] );  !
8!             [UAR]    := [K1112(BC,SUB)] * [AARC] + [PAR(BC,SUB)] *      !
8!                         [DELC];                                         !
8!             [UAL]    := [R1112(BC,SUB)] * [UAR] +                       !
8!                         [R1113(BC,SUB)] * [AARC] -                      !
8!                         [R11PAL(BC,SUB)] * [DELC];                      !
8!             CALL ROWMERGE ( [UAAC(SUB)], [UAR], [UAL], [PARLX(BC)] );   !
8!             IF NOMIT <> 0 [PAOC(SUB)] := [POARO(BC,SUB)]*[DELC] ;       !
8!          ENDIF;                                                         !
7!       ELSE                                                              !
7!$                                                                       $!
7!$         NO SUPPORT SET REDUCTION                                      $!
7!$         PROCESS STEADY AEROELASTIC DISCIPLINE                         $!
7!$                                                                       $!
```

```
7!              PRINT("LOG=('          >>>DISCIPLINE: STEADY AERO')");          !
7!$                                                                            $!
7!           ENDIF;                                                             !
6!        ENDDO;                                                                !
5!     ENDIF;                                                                   !
4!$                                                                            $!
4!$    PERFORM ANY DYNAMIC ANALYSES -- NOTE THAT THESE ARE INDEPENDENT         $!
4!$    OF THE SUPPORT SET                                                      $!
4!$                                                                            $!
4!     IF BDYN <> 0 THEN                                                        !
5!        IF BFLUTR <> 0 THEN                                                   !
6!           PRINT("LOG=('          >>>DISCIPLINE: FLUTTER')");                 !
6!           SUBF := 0;                                                         !
6!           LOOP := TRUE;                                                      !
6!           WHILE LOOP DO                                                      !
7!              SUBF := SUBF + 1;                                               !
7!              CALL FLUTDRV ( BCID, SUBF, LOOP );                              !
7!              CALL FLUTQHHL ( NITER, BCID, SUBF, ESIZE(BC), PSIZE(BC),        !
7!                              [QKKL], [UGTKA], [PHIA], USET(BC),              !
7!                              [TMN(BC)], [GSUBO(BC)], NGDR, AECOMPU, GEOMUA,  !
7!                              [PHIKH], [QHHLFL(BC,SUBF)], OAGRDDSP );         !
7!              CALL FLUTDMA ( NITER, BCID, SUBF, ESIZE(BC), PSIZE(BC),         !
7!                         BGPDT(BC), USET(BC), [MAA], [KAA], [TMN(BC)],        !
7!                         [GSUBO(BC)], NGDR, LAMBDA, [PHIA],                   !
7!                         [MHHFL(BC,SUBF)], [BHHFL(BC,SUBF)], [KHHFL(BC,SUBF)]);!
7!              CALL FLUTTRAN ( NITER, BCID, SUBF, [QHHLFL(BC,SUBF)], LAMBDA,   !
7!                              HSIZE(BC), ESIZE(BC), [MHHFL(BC,SUBF)],         !
7!                              [BHHFL(BC,SUBF)], [KHHFL(BC,SUBF)],             !
7!                              CLAMBDA, CONST );                               !
7!           ENDDO;                                                             !
6!        ENDIF;                                                                !
5!$                                                                            $!
5!        IF BDRSP <> 0 THEN                                                    !
6!           IF BMTR    <> 0 OR BDTR <> 0 THEN                                  !
7!               PRINT("LOG=('          >>>DISCIPLINE: TRANSIENT RESPONSE')");  !
7!           ENDIF;                                                             !
6!           IF BMFR <> 0 OR BDFR <> 0 THEN                                     !
7!               PRINT("LOG=('          >>>DISCIPLINE: FREQUENCY RESPONSE')");  !
7!           ENDIF;                                                             !
6!           CALL QHHLGEN (BCID, ESIZE(BC), [QKKL], [QKJL], [UGTKA], [PHIA],    !
6!                         [PHIKH], [QHHL], [QHJL]);                            !
6!           CALL DMA ( NITER, BC, ESIZE(BC), PSIZE(BC), BGPDT(BC), USET(BC),   !
6!                      [MAA], [KAA], [TMN(BC)], [GSUBO(BC)], NGDR,             !
6!                      LAMBDA, [PHIA], [MDD], [BDD], [KDDT], [KDDF],           !
6!                      [MHH], [BHH], [KHHT], [KHHF] );                         !
6!           CALL DYNLOAD ( NITER, BCID, GSIZE, ESIZE(BC), PSIZE(BC), SMPLOD,   !
6!                          BGPDT(BC), USET(BC), [TMN(BC)], [GSUBO(BC)],        !
6!                          NGDR, [PHIA], [QHJL], [PDT], [PDF],                 !
6!                          [PTGLOAD], [PTHLOAD], [PFGLOAD], [PFHLOAD] );       !
6!           CALL DYNRSP (BCID, ESIZE(BC), [MDD], [BDD], [KDDT], [KDDF],        !
6!                        [MHH], [BHH], [KHHT], [KHHF], [PDT], [PDF],           !
6!                        [QHHL], [UTRANA], [UFREQA], [UTRANI], [UFREQI],       !
6!                        [UTRANE], [UFREQE] );                                 !
6!           IF BMTR <> 0 [UTRANA] := [PHIA] * [UTRANI];                        !
6!           IF BMFR <> 0 [UFREQA] := [PHIA] * [UFREQI];                        !
6!        ENDIF;                                                                !
5!     ENDIF;                                                                   !
4!     IF BBLAST <> 0 THEN                                                      !
5!        PRINT("LOG=('          >>>DISCIPLINE: BLAST')");                      !
5!        CALL BLASTFIT ( BCID, [QJJL], [MATTR], [MATSS], BQDP, [BFRC],         !
5!                        [DWNWSH],  HSIZE(BC), [ID2], [MPART], [UGTKA],        !
5!                        [BLGTJA], [BLSTJA] );                                 !
5!        CALL COLPART ( [PHIA], , [PHIB], [MPART] );                           !
5!        CALL ROWMERGE ( [PHIR], [ID2], [D(BC)], [PARLX(BC)] );                !
5!        CALL COLMERGE ( [PHIB], [PHIR], [PHIE], [MPART] );                    !
5!        [GENM]    := TRANS( [PHIB] ) * [ [MAA] * [PHIB] ];                    !
5!        [GENK]    := TRANS( [PHIB] ) * [ [KAA] * [PHIB] ];                    !
5!        [DTSLP]   := TRANS ( [BLSTJA] ) * [PHIB];                             !
5!        [FTF]     := TRANS ( [PHIB] ) * [BLGTJA];                             !
5!        [GENF]    := (BQDP) [FTF] * [BFRC];                                   !
5!        [GENFA]   := (BQDP) [FTF] * [MATSS];                                  !
5!        [GENQ]    := [GENFA] * [DTSLP];                                       !
5!        [GENQL]   := (BQDP) [FTF] * [MATTR];                                  !
5!        CALL PARTN ( [GENQ],[QRR] , , [QRE], [QEE], [MPART] );                !
5!        CALL PARTN ( [GENK], , , , [KEE], [MPART] );                          !
5!        [KEQE]    := [QEE] + [KEE];                                           !
5!        CALL DECOMP ( [KEQE], [LKQ], [UKQ] );                                 !
5!        CALL ROWPART ( [GENF], [GFR], [GFE], [MPART] );                       !
5!        CALL GFBS ( [LKQ], [UKQ], [GFE], [BTEM] );                            !
5!        [DELM]    := -[QRE] * [BTEM] + [GFR];                                 !
5!        CALL BLASTRIM ( BCID, [DELM], [MRR(BC)], [URDB], [DELB] );            !
5!        [ELAS]    := [BTEM] * [DELB];                                         !
5!        [SLPMOD] := TRANS ( [BLSTJA] ) * [PHIE];                              !
5!        CALL BLASTDRV ( BCID, [GENM], [GENK], [GENFA], [GENQL], [DELB],       !
5!                        [URDB], [DWNWSH], [SLPMOD], [ELAS], [UBLASTI] );      !
5!     ENDIF;                                                                   !
4!$                                                                            $!
4!$    BEGIN THE DATA RECOVERY OPERATIONS                                      $!
4!$                                                                            $!
4!     PRINT("LOG=('          DATA RECOVERY')");                                !
```

```
4!          IF NUMOPTBC > 1 CALL NULLMAT ([UF], [AF], [PHIF], [UTRANF], [UFREQF],   !
5!                                    [UFX], [AFX]);                                 !
4!          IF NBNDCOND > 1 CALL NULLMAT ( [UAF], [UAFI], [AAF], [AAFI]);            !
4!          IF NGDR <> 0 THEN                                                        !
5!$                                                                                 $!
5!$            DATA RECOVERY WITH GDR                                               $!
5!$            APPEND THE GDR-GENERATED DOFS TO THE F-SET                           $!
5!$                                                                                 $!
5!             PRINT("LOG=('          DYNAMIC REDUCTION RECOVERY')");                !
5!             IF BLOAD <> 0 THEN                                                    !
6!                [UFGDR] := [GSUBO(BC)] * [UA];                                     !
6!                CALL ROWPART ( [UA], [UJK], , [PAJK] );                            !
6!                CALL ROWMERGE ( [UFX], [UJK], [UFGDR], [PFJK] );                   !
6!                IF NRSET <> 0 THEN                                                 !
7!                   [AFGDR] := [GSUBO(BC)] * [AA];                                  !
7!                   CALL ROWPART ( [AA], [UJK], , [PAJK] );                         !
7!                   CALL ROWMERGE ( [AFX], [UJK], [AFGDR], [PFJK] );                !
7!                ENDIF;                                                             !
6!             ENDIF;                                                                !
5!             IF BSAERO <> 0 THEN                                                   !
6!                FOR S = 1 TO SUB DO                                                !
7!                   [UFGDR] := [GSUBO(BC)] * [UAA(S)];                              !
7!                   CALL ROWPART ( [UAA(S)], [UJK], , [PAJK] );                     !
7!                   CALL ROWMERGE ( [UAFTMP], [UJK], [UFGDR], [PFJK] );             !
7!$                                                                                 $!
7!$                  MERGE THE CURRENT SUBCASE DEPENDENT RESULTS INTO A SINGLE      $!
7!$                  MATRIX OF RESPONSE QUANTITIES FOR FURTHER RECOVERY             $!
7!$                                                                                 $!
7!                   CALL SAEROMRG ( BCID, S, TRIMDATA, [UAFX], [UAFTMP] );          !
7!                   IF NRSET <> 0 THEN                                              !
8!                      [AFGDR] := [GSUBO(BC)] * [AAA(S)];                           !
8!                      CALL ROWPART ( [AAA(S)], [UJK], , [PAJK] );                  !
8!                      CALL ROWMERGE ( [AAFTMP], [UJK], [AFGDR], [PFJK] );          !
8!                      CALL SAEROMRG ( BCID, S, TRIMDATA, [AAFX], [AAFTMP] );       !
8!                   ENDIF;                                                          !
7!                   IF AEFLG(S) THEN                                                !
8!                      [UFGDR] := [GSUBO(BC)] * [UAAC(S)];                          !
8!                      CALL ROWPART ( [UAAC(S)], [UJK], , [PAJK] );                 !
8!                      CALL ROWMERGE ( [UAFCX(S)], [UJK], [UFGDR], [PFJK] );        !
8!                      [AFGDR] := [GSUBO(BC)] * [AAAC(S)];                          !
8!                      CALL ROWPART ( [AAAC(S)], [UJK], , [PAJK] );                 !
8!                      CALL ROWMERGE ( [AAFCX(S)], [UJK], [AFGDR], [PFJK] );        !
8!                   ENDIF;                                                          !
7!                ENDDO;                                                             !
6!             ENDIF;                                                                !
5!             IF BMODES <> 0 THEN                                                   !
6!                [UFGDR]    := [GSUBO(BC)] * [PHIA];                                !
6!                CALL ROWPART ( [PHIA], [UJK], , [PAJK] );                          !
6!                CALL ROWMERGE ( [PHIF], [UJK], [UFGDR], [PFJK] );                  !
6!             ENDIF;                                                                !
5!             IF BDTR   <> 0 OR BMTR <> 0 THEN                                      !
6!                [UFGDR] := [GSUBO(BC)] * [UTRANA];                                 !
6!                CALL ROWPART ( [UTRANA], [UJK], , [PAJK] );                        !
6!                CALL ROWMERGE ( [UTRANF], [UJK], [UFGDR], [PFJK] );                !
6!             ENDIF;                                                                !
5!             IF BDFR   <> 0 OR BMFR <> 0 THEN                                      !
6!                [UFGDR] := [GSUBO(BC)] * [UFREQA];                                 !
6!                CALL ROWPART ( [UFREQA], [UJK], , [PAJK] );                        !
6!                CALL ROWMERGE ( [UFREQF], [UJK], [UFGDR], [PFJK] );                !
6!             ENDIF;                                                                !
5!          ELSE                                                                     !
5!             IF NOMIT <> 0 THEN                                                    !
6!$                                                                                 $!
6!$               DATA RECOVERY WITH STATIC CONDENSATION                            $!
6!$                                                                                 $!
6!                PRINT("LOG=('          STATIC CONDENSATION RECOVERY')");             !
6!                IF BLOAD <> 0 THEN                                                 !
7!                   CALL RECOVA ( [UA], [PO], [GSUBO(BC)], NRSET, [AA],             !
7!                          [IFM(BC)],, [KOOINV(BC)],,[PFOAX(BC)], [UFX] );           !
7!                   IF NRSET <> 0 CALL RECOVA ( [AA], , [GSUBO(BC)],,,,,,,          !
8!                                          [PFOAX(BC)], [AFX] );                    !
7!                ENDIF;                                                             !
6!                IF BSAERO <> 0 THEN                                                !
7!                   FOR S = 1 TO SUB DO                                             !
8!                      CALL RECOVA ( [UAA(S)], [PAO(S)], [GASUBO(BC,S)],            !
8!                                    NRSET, [AAA(S)], [IFMA(BC,S)], BSAERO,         !
8!                                    [KOOL(BC,S)], [KOOU(BC,S)],                    !
8!                                    [PFOAX(BC)], [UAFTMP] );                       !
8!$                                                                                 $!
8!$                     MERGE THE CURRENT SUBCASE DEPENDENT RESULTS INTO A SINGLE   $!
8!$                     MATRIX OF RESPONSE QUANTITIES FOR FURTHER RECOVERY          $!
8!$                                                                                 $!
8!                      CALL SAEROMRG ( BCID, S, TRIMDATA, [UAFX], [UAFTMP] );       !
8!                      IF NRSET <> 0 THEN                                           !
9!                         CALL RECOVA ( [AAA(S)],,[GASUBO(BC,S)],,,,,,,             !
9!                                       [PFOAX(BC)], [AAFTMP]);                     !
9!                         CALL SAEROMRG ( BCID, S, TRIMDATA, [AAFX], [AAFTMP] );    !
9!                      ENDIF;                                                       !
8!                      IF AEFLG(S) THEN                                             !
9!                         CALL RECOVA ( [UAAC(S)], [PAOC(S)], [GASUBO(BC,S)],       !
```

```
9!                                  NRSET, [AAAC(S)], [IFMA(BC,S)], BSAERO,    !
9!                                  [KOOL(BC,S)], [KOOU(BC,S)],                !
9!                                  [PFOA(BC)], [UAFCX(S)] );                  !
9!                        CALL RECOVA ( [AAAC(S)],,[GASUBO(BC,S)],,,,,,,       !
9!                                  [PFOA(BC)], [AAFCX(S)]);                   !
9!                     ENDIF;                                                  !
8!                  ENDDO;                                                     !
7!               ENDIF;                                                        !
6!               IF BMODES <> 0 THEN                                           !
7!                  [PHIO] := [GSUBO(BC)] * [PHIA];                            !
7!                  CALL ROWMERGE ( [PHIF], [PHIO], [PHIA], [PFOAX(BC)] );     !
7!               ENDIF;                                                        !
6!               IF BDTR <> 0 OR BMTR <> 0 THEN                                !
7!                  CALL RECOVA ( [UTRANA], , [GSUBO(BC)],,,,,,,               !
7!                                        [PFOAX(BC)], [UTRANF] );             !
7!               ENDIF;                                                        !
6!               IF BDFR <> 0 OR BMFR <> 0 THEN                                !
7!                  CALL RECOVA ( [UFREQA], , [GSUBO(BC)],,,,,,,               !
7!                                        [PFOAX(BC)], [UFREQF] );             !
7!               ENDIF;                                                        !
6!            ELSE                                                             !
6!$                                                                           $!
6!$              DATA RECOVERY WITHOUT F-SET REDUCTION                        $!
6!$                                                                           $!
6!               IF BLOAD <> 0 THEN                                            !
7!                  [UFX] := [UA];                                             !
7!                  IF NRSET <> 0 [AFX]   := [AA];                             !
7!               ENDIF;                                                        !
6!               IF BSAERO <> 0 THEN                                           !
7!                  FOR S = 1 TO SUB DO                                        !
8!$                                                                           $!
8!$                    MERGE THE CURRENT SUBCASE DEPENDENT RESULTS INTO A SINGLE $!
8!$                    MATRIX OF RESPONSE QUANTITIES FOR FURTHER RECOVERY     $!
8!$                                                                           $!
8!                     CALL SAEROMRG ( BCID, S, TRIMDATA, [UAFX], [UAA(S)] );  !
8!                     IF NRSET <> 0                                           !
9!                        CALL SAEROMRG ( BCID, S, TRIMDATA, [AAFX], [AAA(S)] ); !
8!                     IF AEFLG(S) THEN                                        !
9!                        [UAFCX(S)] := [UAAC(S)];                             !
9!                        [AAFCX(S)] := [AAAC(S)];                             !
9!                     ENDIF;                                                  !
8!                  ENDDO;                                                     !
7!               ENDIF;                                                        !
6!               IF BMODES <> 0 [PHIF]    := [PHIA];                           !
6!               IF BDTR    <> 0 OR BMTR <> 0 [UTRANF] := [UTRANA];            !
6!               IF BDFR    <> 0 OR BMFR <> 0 [UFREQF] := [UFREQA];            !
6!            ENDIF;                                                           !
5!         ENDIF;                                                              !
4!$                                                                           $!
4!         IF SYMTRN(BC) THEN                                                  !
5!            IF BLOAD <> 0 THEN                                               !
6!               [UF] := [HFREAL(BC)] * [UFX];                                 !
6!               IF NRSET <> 0  [AF] := [HFREAL(BC)] * [AFX];                  !
6!            ENDIF;                                                           !
5!            IF BSAERO <> 0 THEN                                              !
6!               [UAF] := [HFREAL(BC)] * [UAFX];                               !
6!               [UAFI] := [HFIMAG(BC)] * [UAFX];                              !
6!               IF NRSET <> 0  [AAF] := [HFREAL(BC)] * [AAFX];                !
6!               IF NRSET <> 0  [AAFI] := [HFIMAG(BC)] * [AAFX];               !
6!               FOR S = 1 TO SUB DO                                           !
7!                  IF AEFLG(S) THEN                                           !
8!                     [UAFC(S)] := [HFREAL(BC)] * [UAFCX(S)];                 !
8!                     [UAFCI(S)] := [HFIMAG(BC)] * [UAFCX(S)];                !
8!                     [AAFC(S)] := [HFREAL(BC)] * [AAFCX(S)];                 !
8!                     [AAFCI(S)] := [HFIMAG(BC)] * [AAFCX(S)];                !
8!                  ENDIF;                                                     !
7!               ENDDO;                                                        !
6!            ENDIF;                                                           !
5!         ELSE                                                                !
5!            IF BLOAD <> 0 THEN                                               !
6!               [UF] := [UFX];                                                !
6!               IF NRSET <> 0  [AF] := [AFX];                                 !
6!            ENDIF;                                                           !
5!            IF BSAERO <> 0 THEN                                              !
6!               [UAF] := [UAFX];                                              !
6!               IF NRSET <> 0  [AAF] := [AAFX];                               !
6!               FOR S = 1 TO SUB DO                                           !
7!                  IF AEFLG(S) THEN                                           !
8!                     [UAFC(S)] := [UAFCX(S)];                                !
8!                     [AAFC(S)] := [AAFCX(S)];                                !
8!                  ENDIF;                                                     !
7!               ENDDO;                                                        !
6!            ENDIF;                                                           !
5!         ENDIF;                                                              !
4!$                                                                           $!
4!         IF NUMOPTBC > 1 CALL NULLMAT ( [UN], [AN], [PHIN] );                !
4!         IF NSPC <> 0 THEN                                                   !
5!$                                                                           $!
5!$           DATA RECOVERY WITH SPC-REDUCTION                                $!
5!$                                                                           $!
```

```
5!          PRINT("LOG=('          SPC RECOVERY')");
5!          IF BLOAD <> 0  THEN
6!             CALL YSMERGE ( [UN], [YS(BC)], [UF], [PNSF(BC)] );
6!             CALL OFPSPCF ( NITER, BCID, 1, 1, GSIZE, ESIZE(BC), NGDR,
6!                            [KFS], [KSS], [UF], [YS(BC)], [PS],
6!                            [PNSF(BC)], [PGMN(BC)], [PFJK], , , ,
6!                            BGPDT(BC), OGRIDLOD );
6!             IF NRSET <> 0 CALL YSMERGE ( [AN], , [AF], [PNSF(BC)] );
6!          ENDIF;
5!          IF BSAERO <> 0 THEN
6!             CALL YSMERGE ( [UAN], [YS(BC)], [UAF], [PNSF(BC)] );
6!             IF NRSET <> 0 CALL YSMERGE ( [AAN], , [AAF], [PNSF(BC)] );
6!             IF SYMTRN(BC) THEN
7!                CALL YSMERGE ( [UANI], [YS(BC)], [UAFI], [PNSF(BC)] );
7!                IF NRSET <> 0 CALL YSMERGE ( [AANI], , [AAFI], [PNSF(BC)] );
7!             ENDIF;
6!             FOR S = 1 TO SUB DO
7!                IF AEFLG(S) THEN
8!                   CALL YSMERGE ([UANC(S)], [YS(BC)], [UAFC(S)], [PNSF(BC)]);
8!                   CALL YSMERGE ([AANC(S)], , [AAFC(S)], [PNSF(BC)]);
8!                   IF SYMTRN(BC) THEN
9!                      CALL YSMERGE ([UANCI(S)], [YS(BC)], [UAFCI(S)],
9!                                    [PNSF(BC)]);
9!                      CALL YSMERGE ([AANCI(S)], , [AAFCI(S)], [PNSF(BC)]);
9!                   ENDIF;
8!                ENDIF;
7!             ENDDO;
6!          ENDIF;
5!          IF BMODES <> 0 THEN
6!             CALL YSMERGE ( [PHIN], [YS(BC)], [PHIF],
6!                                               [PNSF(BC)] );
6!             IF DMODES <> 0   CALL OFPSPCF ( NITER, BCID, 2, 1, GSIZE,
7!                                             ESIZE(BC), NGDR,
7!                                             [KFS], , [PHIF], , ,
7!                                             [PNSF(BC)], [PGMN(BC)], [PFJK],
7!                                             , , , BGPDT(BC), OGRIDLOD );
6!          ENDIF;
5!          IF BDTR   <> 0 OR BMTR <> 0
6!                          CALL YSMERGE ( [UTRANN], [YS(BC)], [UTRANF],
6!                                               [PNSF(BC)], BDTR );
5!          IF BDFR   <> 0 OR BMFR <> 0
6!                          CALL YSMERGE ( [UFREQN], [YS(BC)], [UFREQF],
6!                                               [PNSF(BC)], BDFR );
5!          IF BBLAST <> 0   THEN
6!             [UBLASTF]  := [PHIF]*[UBLASTI];
6!             CALL OFPSPCF ( NITER, BCID, 8, 1, GSIZE, ESIZE(BC), NGDR,
6!                            [KFS], , [UBLASTF], , , [PNSF(BC)], [PGMN(BC)],
6!                            [PFJK], , , , BGPDT(BC), OGRIDLOD );
6!          ENDIF;
5!       ELSE
5!$                                                                          $!
5!$         DATA RECOVERY WITHOUT SPC-REDUCTION                              $!
5!$                                                                          $!
5!          IF BLOAD <> 0 THEN
6!             [UN]  := [UF];
6!             IF NRSET <> 0 [AN] := [AF];
6!          ENDIF;
5!          IF BSAERO <> 0 THEN
6!             [UAN]  := [UAF];
6!             IF NRSET <> 0 [AAN]  := [AAF];
6!             IF SYMTRN(BC) THEN
7!                [UANI]  := [UAFI];
7!                IF NRSET <> 0 [AANI]  := [AAFI];
7!             ENDIF;
6!             FOR S = 1 TO SUB DO
7!                IF AEFLG(S) THEN
8!                   [UANC(S)]  := [UAFC(S)];
8!                   [AANC(S)]  := [AAFC(S)];
8!                   IF SYMTRN(BC) THEN
9!                      [UANCI(S)]  := [UAFCI(S)];
9!                      [AANCI(S)]  := [AAFCI(S)];
9!                   ENDIF;
8!                ENDIF;
7!             ENDDO;
6!          ENDIF;
5!          IF BMODES <> 0 [PHIN]    := [PHIF];
5!          IF BDTR    <> 0 OR BMTR <> 0 [UTRANN]  := [UTRANA];
5!          IF BDFR    <> 0 OR BMFR <> 0 [UFREQN]  := [UFREQA];
5!       ENDIF;
4!$                                                                          $!
4!       IF NUMOPTBC > 1 CALL NULLMAT ( [UG(BC)], [AG(BC)], [UAG(BC)],
5!                                      [AAG(BC)], [PHIG(BC)], [UAGI(BC)],
5!                                      [AAGI(BC)] );
4!$                                                                          $!
4!       IF NMPC <> 0 THEN
5!$                                                                          $!
5!$         DATA RECOVERY WITH MPC-REDUCTION                                 $!
5!$                                                                          $!
5!          PRINT("LOG=('          MPC RECOVERY')");
5!          IF BLOAD <> 0 THEN
```

```
6!          [UM] := [TMN(BC)] * [UN];                                          !
6!          CALL ROWMERGE ( [UG(BC)], [UM], [UN], [PGMN(BC)] );                !
6!          IF NRSET <> 0 THEN                                                 !
7!             [UM] := [TMN(BC)] * [AN];                                       !
7!             CALL ROWMERGE ( [AG(BC)], [UM], [AN], [PGMN(BC)] );             !
7!          ENDIF;                                                             !
6!       ENDIF;                                                                !
5!       IF BSAERO <> 0 THEN                                                   !
6!          [UM] := [TMN(BC)] * [UAN];                                         !
6!          CALL ROWMERGE ( [UAG(BC)], [UM], [UAN], [PGMN(BC)] );              !
6!          IF NRSET <> 0 THEN                                                 !
7!             [UM] := [TMN(BC)] * [AAN];                                      !
7!             CALL ROWMERGE ( [AAG(BC)], [UM], [AAN], [PGMN(BC)] );           !
7!          ENDIF;                                                             !
6!          IF SYMTRN(BC) THEN                                                 !
7!             [UM] := [TMN(BC)] * [UANI];                                     !
7!             CALL ROWMERGE ( [UAGI(BC)], [UM], [UANI], [PGMN(BC)] );         !
7!             IF NRSET <> 0 THEN                                              !
8!                [UM] := [TMN(BC)] * [AANI];                                  !
8!                CALL ROWMERGE ( [AAGI(BC)], [UM], [AANI], [PGMN(BC)] );      !
8!             ENDIF;                                                          !
7!          ENDIF;                                                             !
6!          FOR S = 1 TO SUB DO                                                !
7!             IF AEFLG(S) THEN                                                !
8!                [UM] := [TMN(BC)] * [UANC(S)];                               !
8!                CALL ROWMERGE ([UAGC(BC,S)], [UM], [UANC(S)], [PGMN(BC)]);   !
8!                [UM] := [TMN(BC)] * [AANC(S)];                               !
8!                CALL ROWMERGE ([AAGC(BC,S)], [UM], [AANC(S)], [PGMN(BC)]);   !
8!                IF SYMTRN(BC) THEN                                           !
9!                   [UM] := [TMN(BC)] * [UANCI(S)];                           !
9!                   CALL ROWMERGE ([UAGCI(BC,S)], [UM], [UANCI(S)],           !
9!                                  [PGMN(BC)]);                               !
9!                   [UM] := [TMN(BC)] * [AANCI(S)];                           !
9!                   CALL ROWMERGE ([AAGCI(BC,S)], [UM], [AANCI(S)],           !
9!                                  [PGMN(BC)]);                               !
9!                ENDIF;                                                       !
8!             ENDIF;                                                          !
7!          ENDDO;                                                             !
6!       ENDIF;                                                                !
5!       IF BMODES <> 0 THEN                                                   !
6!          [UM] := [TMN(BC)] * [PHIN];                                        !
6!          CALL ROWMERGE ( [PHIG(BC)], [UM], [PHIN], [PGMN(BC)] );            !
6!       ENDIF;                                                                !
5!       IF BDTR <> 0 OR BMTR <> 0 THEN                                        !
6!          [UM] := [TMN(BC)] * [UTRANN];                                      !
6!          CALL ROWMERGE ( [UTRANG], [UM], [UTRANN], [PGMN(BC)] );            !
6!       ENDIF;                                                                !
5!       IF BDFR <> 0 OR BMFR <> 0 THEN                                        !
6!          [UM] := [TMN(BC)] * [UFREQN];                                      !
6!          CALL ROWMERGE ( [UFREQG], [UM], [UFREQN], [PGMN(BC)] );            !
6!       ENDIF;                                                                !
5!    ELSE                                                                     !
5!$                                                                           $!
5!$      DATA RECOVERY WITHOUT MPC-REDUCTION                                  $!
5!$                                                                           $!
5!       IF BLOAD <> 0 THEN                                                    !
6!          [UG(BC)] := [UN];                                                  !
6!          IF NRSET <> 0 [AG(BC)] := [AN];                                    !
6!       ENDIF;                                                                !
5!       IF BSAERO <> 0 THEN                                                   !
6!          [UAG(BC)] := [UAN];                                                !
6!          IF NRSET <> 0 [AAG(BC)] := [AAN];                                  !
6!          IF SYMTRN(BC) THEN                                                 !
7!             [UAGI(BC)] := [UANI];                                           !
7!             IF NRSET <> 0 [AAGI(BC)] := [AANI];                             !
7!          ENDIF;                                                             !
6!          FOR S = 1 TO SUB DO                                                !
7!             IF AEFLG(S) THEN                                                !
8!                [UAGC(BC,S)] := [UANC(S)];                                   !
8!                [AAGC(BC,S)] := [AANC(S)];                                   !
8!                IF SYMTRN(BC) THEN                                           !
9!                   [UAGCI(BC,S)] := [UANCI(S)];                              !
9!                   [AAGCI(BC,S)] := [AANCI(S)];                              !
9!                ENDIF;                                                       !
8!             ENDIF;                                                          !
7!          ENDDO;                                                             !
6!       ENDIF;                                                                !
5!       IF BMODES <> 0 [PHIG(BC)] := [PHIN];                                  !
5!       IF BDTR   <> 0 OR BMTR <> 0 [UTRANG] := [UTRANN];                     !
5!       IF BDFR   <> 0 OR BMFR <> 0 [UFREQG] := [UFREQN];                     !
5!    ENDIF;                                                                   !
4!$                                                                           $!
4!$   RECOVER PHYSICAL BLAST DISCIPLINE DISPLACEMENTS                         $!
4!$                                                                           $!
4!    IF BBLAST <> 0 [UBLASTG] := [PHIG(BC)] * [UBLASTI];                      !
4!$                                                                           $!
4!$   PERFORM CONSTRAINT EVALUATION FOR STATIC DISCIPLINES                    $!
4!$                                                                           $!
4!    PRINT("LOG=('          CONSTRAINT EVALUATION')");                        !
4!    IF BLOAD <> 0 THEN                                                       !
```

```
5!          CALL DCEVAL ( NITER, BCID, [UG(BC)], BGPDT(BC), CONST );            !
5!          CALL SCEVAL ( NITER, BCID, [UG(BC)], [SMAT], [NLSMAT], SMATCOL,     !
5!                        NLSMTCOL, TREF, TREFD, [GLBSIG],                      !
5!                        [NLGLBSIG], CONST );                                  !
5!       ENDIF;                                                                 !
4!       IF BSAERO <> 0 THEN                                                    !
5!          CALL DCEVAL ( NITER, BCID, [UAG(BC)], BGPDT(BC), CONST, BSAERO,     !
5!                        TRIMDATA, [UAGI(BC)] );                               !
5!          CALL SCEVAL ( NITER, BCID, [UAG(BC)], [SMAT], [NLSMAT], SMATCOL,    !
5!                        NLSMTCOL, TREF, TREFD, [GLBSIG],                      !
5!                        [NLGLBSIG], CONST, BSAERO, TRIMDATA,                  !
5!                        [UAGI(BC)], [GLBSIGI], [NLGBSIGI] );                  !
5!       ENDIF;                                                                 !
4!$                                                                            $!
4!$      HANDLE OUTPUT REQUESTS                                                $!
4!$                                                                            $!
4!       PRINT("LOG=('        OUTPUT PROCESSING')");                            !
4!       IF BSAERO <> 0 THEN                                                    !
5!$                                                                            $!
5!$         RECOVER STATIC AEROELASTIC LOADS DATA                              $!
5!$                                                                            $!
5!          LOOP := TRUE;                                                       !
5!          SUB  := 0;                                                          !
5!          WHILE LOOP DO                                                       !
6!             SUB := SUB + 1;                                                  !
6!             CALL SAERODRV (BCID, SUB, LOOP, MINDEX, SYM, MACH, QDP,          !
6!                            TRIMDATA, TRIMRSLT, METHOD );                     !
6!$                                                                            $!
6!$            CALL THE TRIMMED LOADS COMPUTATION WITH PROPER MATRICES         $!
6!$                                                                            $!
6!             IF SYM = 1  THEN                                                 !
7!                CALL OFPALOAD ( NITER, BCID, MINDEX, SUB, GSIZE, TRIMDATA,    !
7!                                BGPDT(BC),                                    !
7!                                [GPTKG], [GTKG], [GSTKG], QDP, [SAROLOAD],    !
7!                                [DELTA(SUB)], [AIC],                          !
7!                                [UAG(BC)], [MGG], [AAG(BC)], [KFS],           !
7!                                [KSS], [UAF], [YS(BC)], [PNSF(BC)],           !
7!                                [PGMN(BC)], [PFJK], NGDR, USET(BC),           !
7!                                OGRIDLOD );                                   !
7!             ELSE                                                             !
7!                IF SYM = -1 THEN                                              !
8!                   CALL OFPALOAD ( NITER, BCID, MINDEX, SUB, GSIZE, TRIMDATA, !
8!                                BGPDT(BC),                                    !
8!                                [GPTKG], [GTKG], [GSTKG], QDP, [SAROLOAD],    !
8!                                [DELTA(SUB)], [AAIC],                         !
8!                                [UAG(BC)], [MGG], [AAG(BC)], [KFS],           !
8!                                [KSS], [UAF], [YS(BC)], [PNSF(BC)],           !
8!                                [PGMN(BC)], [PFJK], NGDR, USET(BC),           !
8!                                OGRIDLOD );                                   !
8!                ELSE                                                          !
8!                   IF SYM = 0 THEN                                            !
9!                      CALL OFPALOAD ( NITER, BCID, MINDEX, SUB, GSIZE,        !
9!                                TRIMDATA, BGPDT(BC),                          !
9!                                [GPTKG], [GTKG], [GSTKG], QDP, [SAROLOAD],    !
9!                                [DELTA(SUB)], [ASAIC],                        !
9!                                [UAG(BC)], [MGG], [AAG(BC)], [KFS],           !
9!                                [KSS], [UAF], [YS(BC)], [PNSF(BC)],           !
9!                                [PGMN(BC)], [PFJK], NGDR, USET(BC),           !
9!                                OGRIDLOD );                                   !
9!                   ENDIF;                                                     !
8!                ENDIF;                                                        !
7!             ENDIF;                                                           !
6!$                                                                            $!
6!$            CALL TO COMPUTE THE TRIMMED LOADS/DISPLACEMENTS ON THE          $!
6!$            AERODYNAMIC MODEL                                               $!
6!$                                                                            $!
6!             IF SYM = 1  THEN                                                 !
7!                CALL OFPAEROM ( NITER, BCID, MINDEX, SUB, GSIZE, SAGEOM,      !
7!                                  TRIMDATA,                                   !
7!                                  [GTKG], [GSTKG], QDP, [SAROLOAD],           !
7!                                  [DELTA(SUB)], [AIC],                        !
7!                                  [UAG(BC)], OAGRDLOD, OAGRDDSP );            !
7!             ELSE                                                             !
7!                IF SYM = -1 THEN                                              !
8!                   CALL OFPAEROM ( NITER, BCID, MINDEX, SUB, GSIZE, SAGEOM,   !
8!                                  TRIMDATA,                                   !
8!                                  [GTKG], [GSTKG], QDP, [SAROLOAD],           !
8!                                  [DELTA(SUB)], [AAIC],                       !
8!                                  [UAG(BC)], OAGRDLOD, OAGRDDSP );            !
8!                ELSE                                                          !
8!                   IF SYM = 0 THEN                                            !
9!                      CALL OFPAEROM ( NITER, BCID, MINDEX, SUB, GSIZE,        !
9!                                  SAGEOM, TRIMDATA,                           !
9!                                  [GTKG], [GSTKG], QDP, [SAROLOAD],           !
9!                                  [DELTA(SUB)], [ASAIC],                      !
9!                                  [UAG(BC)], OAGRDLOD, OAGRDDSP );            !
9!                   ENDIF;                                                     !
8!                ENDIF;                                                        !
7!             ENDIF;                                                           !
6!          ENDDO;                                                              !
```

```
5!           ENDIF;                                                          !
4!           IF BDRSP <> 0  THEN                                             !
5!              CALL OFPDLOAD ( NITER, BCID, BGPDT(BC), PSIZE(BC), ESIZE(BC),   !
5!                              [PHIG(BC)], [PTGLOAD], [PTHLOAD], [PFGLOAD],  !
5!                              [PFHLOAD], OGRIDLOD );                        !
5!              IF BDTR   <> 0 OR BMTR <> 0                                   !
6!                            CALL OFPSPCF ( NITER, BCID, 5, 1, GSIZE, ESIZE(BC), !
6!                                           NGDR, [KFS], , [UTRANF], , ,     !
6!                                           [PNSF(BC)], [PGMN(BC)], [PFJK],  !
6!                                           [PHIG(BC)], [PTGLOAD], [PTHLOAD], !
6!                                           BGPDT(BC), OGRIDLOD );           !
5!              IF BDFR   <> 0 OR BMFR <> 0                                   !
6!                            CALL OFPSPCF ( NITER, BCID, 6, 2, GSIZE, ESIZE(BC), !
6!                                           NGDR, [KFS], , [UFREQF], , ,     !
6!                                           [PNSF(BC)], [PGMN(BC)], [PFJK],  !
6!                                           [PHIG(BC)], [PFGLOAD], [PFHLOAD], !
6!                                           BGPDT(BC), OGRIDLOD );           !
5!           ENDIF;                                                          !
4!           CALL OFPLOAD ( BCID, NITER, GSIZE, BGPDT(BC), PSIZE(BC),        !
4!                          [PG] );                                          !
4!           CALL OFPDISP ( BCID, NITER, GSIZE, BGPDT(BC), ESIZE(BC),        !
4!                          PSIZE(BC), OGRIDDSP, [UG(BC)], [AG(BC)], [UAG(BC)], !
4!                          [AAG(BC)], [UBLASTG], , [UTRANG], [UTRANE], [UFREQG], !
4!                          [UFREQE], LAMBDA, [PHIG(BC)], , SYMTRN(BC), [UAGI(BC)], !
4!                          [AAGI(BC)] );                                    !
4!           CALL EDR ( BCID, NITER, NDV, GSIZE, EOSUMMRY, EODISC,           !
4!                      GLBDES, LOCLVAR, [PTRANS],                           !
4!                      [UG(BC)], [UAG(BC)], , [UTRANG], [UFREQG], [PHIG(BC)], !
4!                      , SYMTRN(BC), [UAGI(BC)] );                          !
4!           CALL PBKLEVAL ( BCID, NITER, NDV, GLBDES, LOCLVAR, [PTRANS], CONST, !
4!                           PDLIST, OPNLBUCK );                             !
4!           CALL EBKLEVAL ( BCID, NITER, NDV, GLBDES, LOCLVAR, [PTRANS], CONST, !
4!                           FDSTEP, OEULBUCK );                             !
4!           CALL OFPEDR ( BCID, HSIZE(BC), NITER, SYMTRN(BC) );             !
4!$                                                                         $!
4!$.........................................................................$!
4!$                                                                         $!
4!        ENDDO;                                                             !
3!$                                                                         $!
3!$                    SELECT ACTIVE CONSTRAINTS                            $!
3!$                                                                         $!
3!        CALL FNEVAL ( NITER, CONST );                                      !
3!        PRINT("LOG=('          CONSTRAINT SCREENING')");                   !
3!        CALL WOBJGRAD ( NITER, NDV, GLBDES, DWGH1, DDWGH2 );               !
3!        CALL ACTCON ( NITER, MAXITER,  NRFAC, NDV, EPS,   APPCNVRG, GLBCNVRG, !
3!                      CTL,   CTLMIN,   CONST, [AMAT], DESHIST, PFLAG );     !
3!        CALL DESPUNCH ( NITER, PFLAG, OLOCALDV );                          !
3!$                                                                         $!
3!        IF GLBCNVRG OR NITER > MAXITER  THEN                               !
4!$                                                                         $!
4!$          LAST ITERATION OUTPUT                                          $!
4!$                                                                         $!
4!           FOR BC = 1 TO NUMOPTBC DO                                       !
5!              CALL BCIDVAL ( BC, CASE, BCID );                             !
5!              CALL OFPMROOT ( NITER, BCID, LAMBDA, 1 );                    !
5!              CALL OFPDISP ( BCID, NITER, GSIZE, BGPDT(BC), ESIZE(BC),     !
5!                             PSIZE(BC), OGRIDDSP,,,,,,,,,,,LAMBDA,,,       !
5!                             , , , 1 );                                    !
5!              CALL OFPEDR ( BCID, HSIZE(BC), NITER, SYMTRN(BC), 1 );       !
5!           ENDDO;                                                          !
4!           LASTITER := 1;                                                  !
4!           CALL GDVPRINT ( NITER, NDV, GLBDES, MOVLIM, LASTITER, GPRINT ); !
4!           CALL GDVPUNCH ( NITER, NDV, GLBDES, GPUNCH, LASTITER );         !
4!           CALL LDVLOAD ( GLBDES, LOCLVAR, [PTRANS], OLOCALDV, NITER, NDV, !
4!                          LASTITER, LOADLDV );                             !
4!           CALL LDVPRINT ( OLOCALDV, NITER, LASTITER, LPRINT );            !
4!        ENDIF;                                                             !
3!$                                                                         $!
3!        IF NOT GLBCNVRG AND NITER <= MAXITER THEN                          !
4!$                                                                         $!
4!$          USE APPROPRIATE RESIZING METHOD                                $!
4!$                                                                         $!
4!           IF NITER >=  FSDS AND NITER <= FSDE THEN                        !
5!              PRINT("LOG=('         FULLY STRESSED DESIGN')");             !
5!              CALL FSD ( NDV, NITER, FSDS, FSDE, MPS, OCS, ALPHA,          !
5!                         CNVRGLIM, GLBDES, LOCLVAR, [PTRANS], CONST,       !
5!                         APPCNVRG, CTL, CTLMIN, DESHIST );                 !
5!           ENDIF;                                                          !
4!$                                                                         $!
4!           IF ( NITER >= MPS AND NITER <= MPE ) OR                         !
5!              ( NITER >= OCS AND NITER <= OCE ) THEN                       !
5!$                                                                         $!
5!$             USE MATHEMATICAL PROGRAMMING OR OC METHODS                  $!
5!$                                                                         $!
5!$             OBTAIN THE SENSITIVITIES OF THE CONSTRAINTS WRT THE         $!
5!$             DESIGN VARIABLES                                            $!
5!$                                                                         $!
5!              PRINT("LOG=('         SENSITIVITY ANALYSIS')");              !
5!              CALL GDVGRAD ( NITER, NDV, CONST, GLBDES );                  !
5!              CALL MSWGGRAD ( NITER, NDV, GLBDES, DESLINK, CONST );        !
```

```
5!              CALL MAKDFV ( NITER, NDV, [PMINT], [PMAXT], CONST, GLBDES,          !
5!                            DESLINK, FDSTEP, [AMAT] );                              !
5!              CALL MKDFDV ( NITER, NDV, CONST, DESLINK, GLBDES, [AMAT] );           !
5!              CALL LAMINSNS ( NITER, NDV, GLBDES, LOCLVAR, [PTRANS], CONST,         !
5!                              [AMAT] );                                             !
5!$                                                                                  $!
5!$********************************************************************************$!
5!$             SENSITIVITY EVALUATION FOR BOUNDARY CONDITION DEPENDENT CONSTRAINTS$!
5!$********************************************************************************$!
5!$                                                                                  $!
5!              FOR BC = 1 TO NUMOPTBC DO                                             !
6!                 CALL BCIDVAL ( BC, CASE, BCID );                                   !
6!                 CALL ABOUND ( NITER, BCID, CONST, NDV, ACTBOUND, NAUS, NACSD,      !
6!                               [PGAS], PCAS, PRAS, ACTAERO, ACTDYN, ACTFLUT,        !
6!                               ACTPNL, ACTBAR, NMPC, NSPC, NOMIT, NRSET, NGDR,      !
6!                               USET(BC) );                                          !
6!                 IF ACTBOUND THEN                                                   !
7!$                                                                                  $!
7!$                   REESTABLISH THE BASE USET AND PARTITIONING DATA FOR THE BC     $!
7!$                   IF GDR CHANGED IT                                              $!
7!$                   NOTE, THIS LEAVES AN INCOMPATIBILITY BETWEEN USET(BC) AND      $!
7!$                   BGPDT(BC) SINCE THE LATTER IS NOT REGENERATED.                 $!
7!$                   THIS INCOMPATIBILITY WILL NOT AFFECT THE SENSITIVITY ANALYSIS$!
7!$                   AND WILL BE CORRECTED IN THE SUBSEQUENT ANALYSIS               $!
7!$                                                                                  $!
7!                    IF NGDR <> 0 THEN                                               !
8!                       CALL MKUSET(BCID, GSIZEB, [YS(BC)], [TMN(BC)], [PGMN(BC)], !
8!                                   [PNSF(BC)], [PFOA(BC)], [PARL(BC)], USET(BC)); !
8!                    ENDIF;                                                          !
7!$                                                                                  $!
7!$                   EVALUATE FREQUENCY CONSTRAINT SENSITIVITIES                    $!
7!$                                                                                  $!
7!                    IF ACTDYN THEN                                                  !
8!                       IF NGDR <> 0 THEN                                            !
9!                          CALL ROWPART ( [PHIG(BC)], , [GTMP], [PGDRG(BC)] );       !
9!                          CALL FREQSENS ( NITER, BCID, NDV, GLBDES, CONST, LAMBDA, !
9!                                          GMKCT, DKVI, GMMCT, DMVI,                 !
9!                                          [GTMP], [AMAT] );                         !
9!                       ELSE                                                         !
9!                          CALL FREQSENS ( NITER, BCID, NDV, GLBDES, CONST, LAMBDA, !
9!                                          GMKCT, DKVI, GMMCT, DMVI,                 !
9!                                          [PHIG(BC)], [AMAT] );                     !
9!                       ENDIF;                                                       !
8!                    ENDIF;                                                          !
7!$                                                                                  $!
7!$                   EVALUATE FLUTTER CONSTRAINT SENSITIVITIES                      $!
7!$                                                                                  $!
7!                    IF ACTFLUT THEN                                                 !
8!                       SUBF  := 0;                                                  !
8!                       LOOP := TRUE;                                                !
8!                       IF NGDR <> 0 CALL ROWPART ([PHIG(BC)],,[GTMP],[PGDRG(BC)]);!
8!                       WHILE LOOP DO                                                !
9!                          SUBF := SUBF + 1;                                         !
9!                          IF NGDR <> 0 THEN                                         !
10!                            CALL FLUTSENS (NITER, BCID, SUBF, LOOP, GSIZEB, NDV, !
10!                                           GLBDES, CONST, GMKCT, DKVI, GMMCT,     !
10!                                           DMVI, CLAMBDA, LAMBDA,                 !
10!                                           [QHHLFL(BC,SUBF)],                     !
10!                                           [MHHFL(BC,SUBF)], [BHHFL(BC,SUBF)],    !
10!                                           [KHHFL(BC,SUBF)], [GTMP], [AMAT]);     !
10!                         ELSE                                                      !
10!                            CALL FLUTSENS (NITER, BCID, SUBF, LOOP, GSIZEB, NDV, !
10!                                           GLBDES, CONST, GMKCT, DKVI, GMMCT,     !
10!                                           DMVI, CLAMBDA, LAMBDA,                 !
10!                                           [QHHLFL(BC,SUBF)],                     !
10!                                           [MHHFL(BC,SUBF)], [BHHFL(BC,SUBF)],    !
10!                                           [KHHFL(BC,SUBF)], [PHIG(BC)], [AMAT]);!
10!                         ENDIF;                                                    !
9!                       ENDDO;                                                       !
8!                    ENDIF;                                                          !
7!$                                                                                  $!
7!$                   EVALUATE ACTIVE DISPLACEMENT DEPENDENT CONSTRAINTS FROM        $!
7!$                   THE STATICS DISCIPLINE                                         $!
7!$                                                                                  $!
7!                    IF NAUS > 0 THEN                                                !
8!$                                                                                  $!
8!$                      SENSITIVITIES OF CONSTRAINTS WRT DISPLACEMENTS FOR STATICS$!
8!$                                                                                  $!
8!                       CALL NULLMAT ( [DFDU], [DPGV] );                             !
8!                       IF NACSD > NAUS * NDV THEN                                   !
9!$                                                                                  $!
9!$                         USE GRADIENT METHOD                                      $!
9!$                                                                                  $!
9!                          CALL MAKDFU ( NITER, BCID, GSIZEB, [SMAT], [NLSMAT],      !
9!                                        SMATCOL, NLSMTCOL, [GLBSIG], [NLGLBSIG],    !
9!                                        CONST, BGPDT(BC), [DFDU] );                 !
9!                       ELSE                                                         !
9!$                                                                                  $!
9!$                         USE VIRTUAL LOAD METHOD                                  $!
9!$                                                                                  $!
```

```
 9!             CALL MAKDFU ( NITER, BCID, GSIZEB, [SMAT], [NLSMAT],      !
 9!                           SMATCOL, NLSMTCOL, [GLBSIG], [NLGLBSIG],    !
 9!                           CONST, BGPDT(BC), [DPGV] );                 !
 9!          ENDIF;                                                       !
 8!$                                                                     $!
 8!$         SOME RELATIVELY SIMPLE CALCULATIONS THAT PRECEDE THE        $!
 8!$         LOOP ON THE DESIGN VARIABLES                                $!
 8!$                                                                     $!
 8!          IF NGDR <> 0 THEN                                            !
 9!             CALL PARTN ( [UG(BC)],,,, [UGA], [PGAS], [PGDRG(BC)]);    !
 9!          ELSE                                                         !
 9!             CALL COLPART ( [UG(BC)], , [UGA], [PGAS] );               !
 9!          ENDIF;                                                       !
 8!$                                                                     $!
 8!          CALL MKDFSV ( NITER, BCID, GSIZEB, [NLGLBSIG], CONST,        !
 8!                        [NLSMAT], NLSMTCOL, [UGA], DESLINK, , NDV,     !
 8!                        GLBDES, LOCLVAR, [PTRANS], [DFSV], FDSTEP );   !
 8!$                                                                     $!
 8!$         OBTAIN THE SENSITIVITIES OF THE DESIGN                      $!
 8!$         DEPENDENT LOADS                                             $!
 8!$                                                                     $!
 8!          CALL DDLOAD(NDV, GSIZEB, BCID, SMPLOD, [DPTHVI], [DPGRVI],   !
 8!                      [DDPTHV], [DDPGRV], DDFLG,[PGAS],[DPVJ]);        !
 8!$                                                                     $!
 8!          CALL MAKDVU ( NITER, NDV, GLBDES, [UGA], [DKUG],             !
 8!                        GMKCT, DKVI );                                 !
 8!          CALL NULLMAT ( [DUG] );                                      !
 8!          IF NRSET <> 0 THEN                                           !
 9!             IF NGDR <> 0 THEN                                         !
10!                CALL PARTN ([AG(BC)],,,, [AGA], [PGAS], [PGDRG(BC)]);  !
10!             ELSE                                                      !
10!                CALL COLPART ( [AG(BC)], , [AGA], [PGAS] );            !
10!             ENDIF;                                                    !
 9!             CALL MAKDVU ( NITER, NDV, GLBDES, [AGA], [DMAG],          !
 9!                           GMMCT, DMVI );                              !
 9!             [DUG] := [DKUG] + [DMAG];                                 !
 9!          ELSE                                                         !
 9!             [DUG] := [DKUG];                                          !
 9!          ENDIF;                                                       !
 8!$                                                                     $!
 8!$         ACCOUNT FOR VIRTUAL LOAD METHOD                             $!
 8!$                                                                     $!
 8!          IF NACSD > NAUS * NDV THEN                                   !
 9!$                                                                     $!
 9!$            USE GRADIENT METHOD                                      $!
 9!$                                                                     $!
 9!             IF DDFLG > 0 THEN                                         !
10!                [DPGV] := [DPVJ] + [DUG];                              !
10!             ELSE                                                      !
10!                [DPGV] := [DUG];                                       !
10!             ENDIF;                                                    !
 9!          ELSE                                                         !
 9!$                                                                     $!
 9!$            USE VIRTUAL LOAD METHOD                                  $!
 9!$                                                                     $!
 9!             IF DDFLG > 0 THEN                                         !
10!                [DFDU] := [DPVJ] + [DUG];                              !
10!             ELSE                                                      !
10!                [DFDU] := [DUG];                                       !
10!             ENDIF;                                                    !
 9!          ENDIF;                                                       !
 8!$                                                                     $!
 8!$         REDUCE THE RIGHT HAND SIDES TO THE L SET                    $!
 8!$                                                                     $!
 8!          CALL NULLMAT ( [DPNV], [DMUN] );                             !
 8!          IF NMPC <> 0 THEN                                            !
 9!             CALL GREDUCE (,[DPGV], [PGMNS(BC)],[TMN(BC)],, [DPNV]);   !
 9!          ELSE                                                         !
 9!             [DPNV] := [DPGV];                                         !
 9!          ENDIF;                                                       !
 8!$                                                                     $!
 8!          CALL NULLMAT ( [DPFV], [DMUF], [DPFVX], [DMUFX] );           !
 8!          IF NSPC <> 0 THEN                                            !
 9!             CALL NREDUCE (, [DPNV], [PNSFS(BC)], , , , , [DPFV]);     !
 9!          ELSE                                                         !
 9!             [DPFV] := [DPNV];                                         !
 9!          ENDIF;                                                       !
 8!$                                                                     $!
 8!          CALL NULLMAT ( [DPAV], [DMUA] );                             !
 8!          IF NGDR <> 0 THEN                                            !
 9!             [DPAV] := TRANS( [GSUBO(BC)] ) * [DPFV];                  !
 9!          ELSE                                                         !
 9!             IF NOMIT <> 0 THEN                                        !
10!                CALL FREDUCE (, [DPFV], [PFOAS(BC)], ,                 !
10!                              [KOOINV(BC)], , , [GSUBO(BC)], ,         !
10!                              [DPAV], [DPOV], );                       !
10!             ELSE                                                      !
10!                [DPAV] := [DPFV];                                      !
10!             ENDIF;                                                    !
 9!          ENDIF;                                                       !
```

```
 8!$                                                                      $!
 8!         IF NRSET <> 0 THEN                                             !
 9!            CALL ROWPART ( [DPAV], [DPRV], [DPLV], [PARLS(BC)] );      !
 9!            [DRHS] := TRANS( [D(BC)] ) * [DPLV] + [DPRV];              !
 9!$                                                                      $!
 9!$           PROCESS ACTIVE CONSTRAINTS FOR STATICS DISCIPLINE          $!
 9!$                                                                      $!
 9!            CALL INERTIA ( [MRR(BC)], [DRHS], [DURD] );                 !
 9!            [DULD] := [D(BC)] * [DURD];                                 !
 9!            CALL ROWMERGE ( [DUAD], [DURD], [DULD], [PARLS(BC)] );      !
 9!            [DPLV] := [DPLV] + [IFR(BC)] * [DURD];                      !
 9!            CALL FBS ( [KLLINV(BC)], [DPLV], [DULV] );                  !
 9!            CALL YSMERGE ( [DUAV], , [DULV], [PARLS(BC)] );             !
 9!         ELSE                                                           !
 9!            CALL FBS ( [KLLINV(BC)], [DPAV], [DUAV] );                  !
 9!         ENDIF;                                                         !
 8!$                                                                      $!
 8!$        RECOVER TO THE F SET                                          $!
 8!$                                                                      $!
 8!         CALL NULLMAT ( [DUFV] );                                       !
 8!         IF NGDR <> 0 THEN                                              !
 9!            [DUFV] := [GSUBO(BC)] * [DUAV];                             !
 9!         ELSE                                                           !
 9!            IF NOMIT <> 0 THEN                                          !
10!               IF NRSET <> 0 THEN                                       !
11!                  [TMP1] := [DPOV] - [IFM(BC)] * [DUAD];                !
11!               ELSE                                                     !
11!                  [TMP1] := [DPOV];                                     !
11!               ENDIF;                                                   !
10!               CALL FBS ( [KOOINV(BC)], [TMP1], [UOO] );                !
10!               [UO] := [GSUBO(BC)] * [DUAV] + [UOO];                    !
10!               CALL ROWMERGE ([DUFV], [UO], [DUAV], [PFOAS(BC)]);       !
10!            ELSE                                                        !
10!               [DUFV] := [DUAV];                                        !
10!            ENDIF;                                                      !
 9!         ENDIF;                                                         !
 8!$                                                                      $!
 8!$        REDUCE THE LEFT HAND SIDE MATRIX                              $!
 8!$                                                                      $!
 8!         IF NMPC <> 0 THEN                                              !
 9!            CALL GREDUCE (,[DFDU],[PGMNS(BC)],[TMN(BC)],,[DFDUN]);      !
 9!         ELSE                                                           !
 9!            [DFDUN] := [DFDU];                                          !
 9!         ENDIF;                                                         !
 8!$                                                                      $!
 8!         IF NSPC <> 0 THEN                                              !
 9!            CALL ROWPART ( [DFDUN], , [DFDUF], [PNSFS(BC)] );           !
 9!         ELSE                                                           !
 9!            [DFDUF] := [DFDUN];                                         !
 9!         ENDIF;                                                         !
 8!$                                                                      $!
 8!$        ACCOUNT FOR VIRTUAL LOAD METHOD                               $!
 8!$                                                                      $!
 8!         IF NACSD > NAUS * NDV THEN                                     !
 9!$                                                                      $!
 9!$           USE GRADIENT METHOD                                        $!
 9!$                                                                      $!
 9!            CALL MKAMAT ([AMAT], [DFDUF], [DUFV], [DFSV], PCAS,         !
 9!                         PRAS, [PGAS] );                                !
 9!         ELSE                                                           !
 9!$                                                                      $!
 9!$           USE VIRTUAL LOAD METHOD                                    $!
 9!$                                                                      $!
 9!            CALL MKAMAT ([AMAT], [DUFV], [DFDUF], [DFSV], PCAS,         !
 9!                         PRAS, [PGAS] );                                !
 9!         ENDIF;                                                         !
 8!$                                                                      $!
 8!      ENDIF;   $   END IF ON ACTIVE APPLIED STATIC LOADS               $!
 7!$                                                                      $!
 7!$     EVALUATE ACTIVE CONSTRAINTS FROM                                 $!
 7!$     THE STATIC AEROELASTICITY DISCIPLINE                             $!
 7!$                                                                      $!
 7!      IF ACTAERO THEN                                                   !
 8!         LOOP     := TRUE;                                              !
 8!         ACTUAGG := FALSE;                                              !
 8!         ACTUAGGI:= FALSE;                                              !
 8!         SUB      := 0;                                                 !
 8!         CALL NULLMAT ( [DUFV] );                                       !
 8!         WHILE LOOP DO                                                  !
 9!            SUB := SUB + 1;                                             !
 9!            CALL AROSNSDR ( NITER, BCID, SUB, NDV, LOOP, MINDEX,        !
 9!                            CONST, SYM, TRIMDATA, NGDR,                 !
 9!                            [PGDRG(BC)], [UAG(BC)], [AAG(BC)],          !
 9!                            ACTUAG, [UGA], [AGA], [PGAA], [PGAU],       !
 9!                            PCAA, PRAA, [UAGC(BC,SUB)],                 !
 9!                            [AAGC(BC,SUB)], ACTAEFF, [AUAGC],           !
 9!                            [AAAGC], PCAE,                              !
 9!                            [UAGI(BC)], [AAGI(BC)], [UGAI], [AGAI],     !
 9!                            [UAGCI(BC,SUB)], [AAGCI(BC,SUB)],           !
 9!                            [AUAGCI],[AAAGCI] );                        !
```

```
9!       IF ACTABFF THEN                                                      !
10!$                                                                         $!
10!$        PROCESS PSEUDO DISPLACEMENTS FOR EFFECTIVENESS                   $!
10!$        CONSTRAINTS                                                      $!
10!$                                                                         $!
10!         CALL MAKDVU ( NITER, NDV, GLBDES, [AUAGC], [DKUG],                !
                          GMKCT, DKVI );                                      !
10!         IF NRSET <> 0 THEN                                                !
11!            CALL MAKDVU ( NITER, NDV, GLBDES, [AAAGC], [DMAG],!
                             GMMCT, DMVI);                                    !
11!            [DPGV] := [DKUG] + [DMAG];                                     !
11!            CALL MAKDVU ( NITER, NDV, GLBDES, [AUAGC], [DMUG],!
                             GMMCT, DMVI);                                    !
11!         ELSE                                                              !
11!            [DPGV] := [DKUG];                                              !
11!         ENDIF;                                                            !
10!$                                                                         $!
10!         IF SYMTRN(BC) THEN                                                !
11!            CALL MAKDVU ( NITER, NDV, GLBDES, [AUAGCI],                    !
                             [DKUGI], GMKCT, DKVI );                          !
11!            IF NRSET <> 0 THEN                                             !
12!               CALL MAKDVU ( NITER, NDV, GLBDES, [AAAGCI],                 !
                                [DMAGI], GMMCT, DMVI);                        !
12!               [DPGVI] := [DKUGI] + [DMAGI];                               !
12!               CALL MAKDVU ( NITER, NDV, GLBDES, [AUAGCI],                 !
                                [DMUGI], GMMCT, DMVI);                        !
12!            ELSE                                                           !
12!               [DPGVI] := [DKUGI];                                         !
12!            ENDIF;                                                         !
11!         ENDIF;                                                            !
10!$                                                                         $!
10!$        REDUCE THE RIGHT HAND SIDES TO THE L SET                         $!
10!$                                                                         $!
10!         CALL NULLMAT ( [DPNV], [DMUN], [DPNVI], [DMUNI] );                 !
10!         IF NMPC <> 0 THEN                                                 !
11!            CALL GREDUCE ( , [DPGV], [PGMNS(BC)], [TMN(BC)],, !
                                [DPNV]);                                      !
11!            IF NRSET <> 0 CALL GREDUCE ( , [DMUG],                         !
                             [PGMNS(BC)], [TMN(BC)],, [DMUN] );               !
11!            IF SYMTRN(BC) THEN                                             !
12!               CALL GREDUCE ( , [DPGVI], [PGMNS(BC)],                      !
                             [TMN(BC)],, [DPNVI]);                            !
12!               IF NRSET <> 0 CALL GREDUCE ( , [DMUGI],                     !
                             [PGMNS(BC)], [TMN(BC)],, [DMUNI] ); !
12!            ENDIF;                                                         !
11!         ELSE                                                              !
11!            [DPNV] := [DPGV];                                              !
11!            IF NRSET <> 0 [DMUN] := [DMUG];                                !
11!            IF SYMTRN(BC) THEN                                             !
12!               [DPNVI] := [DPGVI];                                         !
12!               IF NRSET <> 0 [DMUNI] := [DMUGI];                           !
12!            ENDIF;                                                         !
11!         ENDIF;                                                            !
10!$                                                                         $!
10!         CALL NULLMAT ( [DPFV], [DMUF], [DPFVX], [DMUFX],                  !
                           [DPFVI], [DMUFI] );                                !
10!         IF NSPC <> 0 THEN                                                 !
11!            CALL NREDUCE (,[DPNV],[PNSFS(BC)],,,,, [DPFV]);                !
11!            IF NRSET <> 0                                                  !
12!               CALL NREDUCE (,[DMUN],[PNSFS(BC)],,,,, [DMUF]);!
11!            IF SYMTRN(BC) THEN                                             !
12!               CALL NREDUCE (,[DPNVI],[PNSFS(BC)],,,,,                     !
                                [DPFVI]);                                     !
12!               IF NRSET <> 0                                               !
13!                  CALL NREDUCE (,[DMUNI],[PNSFS(BC)],,,,,                  !
                                   [DMUFI]);                                  !
12!            ENDIF;                                                         !
11!         ELSE                                                              !
11!            [DPFV] := [DPNV];                                              !
11!            IF NRSET <> 0 [DMUF] := [DMUN];                                !
11!            IF SYMTRN(BC) THEN                                             !
12!               [DPFVI] := [DPNVI];                                         !
12!               IF NRSET <> 0 [DMUFI] := [DMUNI];                           !
12!            ENDIF;                                                         !
11!         ENDIF;                                                            !
10!$                                                                         $!
10!         IF SYMTRN(BC) THEN                                                !
11!            IF SYM = 0 THEN                                                !
12!               [HRDPFV] := [HFREALT(BC)] * [DPFV];                         !
12!               [HIDPFV] := [HFIMAGT(BC)] * [DPFVI];                        !
12!               [DPFVX]  := [HRDPFV]  +  [HIDPFV];                          !
12!               IF NRSET <> 0  THEN                                         !
13!                  [HRDMUF] := [HFREALT(BC)] * [DMUF];                      !
13!                  [HIDMUF] := [HFIMAGT(BC)] * [DMUFI];                     !
13!                  [DMUFX]  := [HRDMUF]  +  [HIDMUF];                       !
13!               ENDIF;                                                      !
12!            ELSE                                                           !
12!               RSYM := SYM;                                                !
12!               [HRDPFV] := [HFREALT(BC)] * [DPFV];                         !
12!               [HIDPFV] := [HFIMAGT(BC)] * [DPFV];                         !
```

```
12!                 [DPFVX]   := [HRDPFV]   + (RSYM)  [HIDPFV];          !
12!                 IF NRSET <> 0  THEN                                   !
13!                    [HRDMUF]  := [HFREALT(BC)] * [DMUF];               !
13!                    [HIDMUF]  := [HFIMAGT(BC)] * [DMUF];               !
13!                    [DMUFX]   := [HRDMUF]   + (RSYM)  [HIDMUF];        !
13!                 ENDIF;                                                !
12!              ENDIF;                                                   !
11!           ELSE                                                        !
11!              [DPFVX]   := [DPFV];                                     !
11!              IF NRSET <> 0   [DMUFX]   := [DMUF];                     !
11!           ENDIF;                                                      !
10!$                                                                     $!
10!           CALL NULLMAT ( [DPAV], [DMUA] );                            !
10!           IF NGDR <> 0 THEN                                           !
11!              [DPAV] := TRANS( [GSUBO(BC)] ) * [DPFVX];                !
11!              IF NRSET <> 0  [DMUA]:=TRANS([GSUBO(BC)])*[DMUFX];       !
11!           ELSE                                                        !
11!              IF NOMIT <> 0 THEN                                       !
12!                 CALL FREDUCE ( , [DPFVX], [PFOAX(BC)], 1,             !
12!                                [KOOL(BC,SUB)], [KOOU(BC,SUB)],        !
12!                                [KAO(BC,SUB)], [GASUBO(BC,SUB)], ,     !
12!                                [DPAV], [DPOV], );                     !
12!                 IF NRSET <> 0                                         !
13!                    CALL FREDUCE ( , [DMUFX], [PFOAX(BC)], 1,          !
13!                                [KOOL(BC,SUB)], [KOOU(BC,SUB)],        !
13!                                [KAO(BC,SUB)], [GASUBO(BC,SUB)], ,     !
13!                                [DMUA], [DMUO], );                     !
12!              ELSE                                                     !
12!                 [DPAV] := [DPFVX];                                    !
12!                 IF NRSET <> 0   [DMUA] := [DMUFX];                    !
12!              ENDIF;                                                   !
11!           ENDIF;                                                      !
10!$                                                                     $!
10!           IF NRSET <> 0 THEN                                          !
11!              CALL ROWPART ([DPAV],[DPRV],[DPLV],[PARLX(BC)] );        !
11!              CALL ROWPART ([DMUA],[DMUR],[DMUL],[PARLX(BC)] );        !
11!              CALL GFBS ( [RL11(BC,SUB)], [RU11(BC,SUB)],              !
11!                          [DPLV], [R11DPL] );                          !
11!              [DP1]   := TRANS([D(BC)]) * [DMUL] + [DMUR] -            !
11!                         [R21(BC,SUB)] * [R11DPL];                     !
11!              [DRHS]  := TRANS( [D(BC)] ) * [DPLV] + [DPRV] -          !
11!                         [R31(BC,SUB)] * [R11DPL];                     !
11!$                                                                    $!
11!$             PROCESS ACTIVE CONSTRAINTS FOR SAERO DISCIPLINE        $!
11!$                                                                    $!
11!              CALL GFBS ( [KL11(BC,SUB)], [KU11(BC,SUB)],              !
11!                          [DP1],          [DK1V] );                    !
11!              [DRHS] := [DRHS] - [K21(BC,SUB)] * [DK1V];               !
11!$                                                                    $!
11!              CALL DECOMP ( [LHSA(BC,SUB)], [LHSL], [LHSU] );          !
11!              CALL GFBS ( [LHSL], [LHSU], [DRHS], [DU2] );             !
11!$                                                                    $!
11!              [DU1R]      := [DK1V] + [K1112(BC,SUB)] * [DU2];         !
11!              [DU1L]      := [R11DPL] + [R1112(BC,SUB)] * [DU1R] +     !
11!                                    [R1113(BC,SUB)] * [DU2];           !
11!              [EFFSENS] := - [R31(BC,SUB)] * [DU1L] -                  !
11!                             [R32(BC,SUB)] * [DU1R];                   !
11!$                                                                    $!
11!              CALL AEROEFFS ( NITER, BCID, SUB, SYM, TRIMDATA,         !
11!                              NDV, CONST, PCAE, [EFFSENS],             !
11!                              [AMAT] );                                !
11!           ELSE                                                        !
11!$                                                                    $!
11!$             NOTE THAT SAERO W/O SUPPORT IS NOT SUPPORTED           $!
11!$                                                                    $!
11!           ENDIF;                                                      !
10!        ENDIF; $ END IF ON ACTAEFF                                    $!
 9!$                                                                    $!
 9!        IF ACTUAG THEN                                                 !
10!$                                                                    $!
10!$          SENSITIVITIES OF CONSTRAINTS WRT DISPLACEMENTS            $!
10!$          FOR SAERO.  THE ACTUAGG FLAG WILL BE RETURNED             $!
10!$          FALSE IF ONLY TRIM PARAMETER CONSTRAINTS ARE ACTIVE       $!
10!$                                                                    $!
10!           CALL NULLMAT ( [DFDU], [DFDUI] );                           !
10!           CALL MAKDFU ( NITER, BCID, GSIZEB, [SMAT], [NLSMAT],        !
10!                         SMATCOL, NLSMTCOL, [GLBSIG],                  !
10!                         [NLGLBSIG], CONST, BGPDT(BC), [DFDU],         !
10!                         ACTUAGG, SUB,                                 !
10!                         [GLBSIGI], [NLGBSIGI], [DFDUI],               !
10!                         ACTUAGGI, SYMTRN(BC) );                       !
10!$                                                                    $!
10!           CALL MKDFSV ( NITER, BCID, GSIZEB, [NLGLBSIG], CONST,       !
10!                         [NLSMAT], NLSMTCOL, [UGA], DESLINK,           !
10!                         SUB, NDV, GLBDES, LOCLVAR, [PTRANS],          !
10!                         [DFSV], FDSTEP,                               !
10!                         [UGAI], [DFSVI] );                            !
10!$                                                                    $!
10!$          SOME RELATIVELY SIMPLE CALCULATIONS THAT PRECEDE          $!
10!$          THE LOOP ON THE DESIGN VARIABLES                          $!
```

```
10!$                                                                           $!
10!      CALL MAKDVU ( NITER, NDV, GLBDES, [UGA], [DKUG],                        !
10!                    GMKCT, DKVI );                                            !
10!      CALL NULLMAT ( [DPGV] );                                                !
10!      IF NRSET <> 0 THEN                                                      !
11!         CALL MAKDVU ( NITER, NDV, GLBDES, [AGA], [DMAG],                     !
11!                       GMMCT, DMVI );                                         !
11!         [DPGV] := [DKUG] + [DMAG];                                           !
11!         CALL MAKDVU ( NITER, NDV, GLBDES, [UGA], [DMUG],                     !
11!                       GMMCT, DMVI );                                         !
11!      ELSE                                                                    !
11!         [DPGV] := [DKUG];                                                    !
11!      ENDIF;                                                                  !
10!$                                                                           $!
10!      IF SYMTRN(BC) THEN                                                      !
11!         CALL MAKDVU ( NITER, NDV, GLBDES, [UGAI],                            !
11!                       [DKUGI], GMKCT, DKVI );                                !
11!         CALL NULLMAT ( [DPGVI] );                                            !
11!         IF NRSET <> 0 THEN                                                   !
12!            CALL MAKDVU ( NITER, NDV, GLBDES, [AGAI],                         !
12!                          [DMAGI], GMMCT, DMVI );                             !
12!            [DPGVI] := [DKUGI] + [DMAGI];                                     !
12!            CALL MAKDVU ( NITER, NDV, GLBDES, [UGAI],                         !
12!                          [DMUGI], GMMCT, DMVI );                             !
12!         ELSE                                                                 !
12!            [DPGVI] := [DKUGI];                                               !
12!         ENDIF;                                                               !
11!      ENDIF;                                                                  !
10!$                                                                           $!
10!$     REDUCE THE RIGHT HAND SIDES TO THE L SET                              $!
10!$                                                                           $!
10!      CALL NULLMAT ( [DPNV], [DMUN], [DPNVI], [DMUNI] );                      !
10!      IF NMPC <> 0 THEN                                                       !
11!         CALL GREDUCE ( , [DPGV], [PGMNS(BC)], [TMN(BC)],,                    !
11!                        [DPNV]);                                              !
11!         IF NRSET <> 0 CALL GREDUCE ( , [DMUG],                               !
12!                        [PGMNS(BC)], [TMN(BC)],, [DMUN] );                    !
11!         IF SYMTRN(BC) THEN                                                   !
12!            CALL GREDUCE ( , [DPGVI], [PGMNS(BC)],                            !
12!                           [TMN(BC)],, [DPNVI]);                              !
12!            IF NRSET <> 0 CALL GREDUCE ( , [DMUGI],                           !
13!                           [PGMNS(BC)], [TMN(BC)],,                           !
13!                           [DMUNI] );                                         !
12!         ENDIF;                                                               !
11!      ELSE                                                                    !
11!         [DPNV] := [DPGV];                                                    !
11!         IF NRSET <> 0 [DMUN] := [DMUG];                                      !
11!         IF SYMTRN(BC) THEN                                                   !
12!            [DPNVI] := [DPGVI];                                               !
12!            IF NRSET <> 0 [DMUNI] := [DMUGI];                                 !
12!         ENDIF;                                                               !
11!      ENDIF;                                                                  !
10!$                                                                           $!
10!      CALL NULLMAT ( [DPFV], [DMUF], [DPFVX], [DMUFX],                        !
10!                     [DPFVI], [DMUFI] );                                      !
10!      IF NSPC <> 0 THEN                                                       !
11!         CALL NREDUCE (,[DPNV],[PNSFS(BC)],,,,, [DPFV]);                      !
11!         IF NRSET <> 0                                                        !
12!            CALL NREDUCE (,[DMUN],[PNSFS(BC)],,,,, [DMUF]);                   !
11!         IF SYMTRN(BC) THEN                                                   !
12!            CALL NREDUCE (,[DPNVI],[PNSFS(BC)],,,,,                           !
12!                          [DPFVI]);                                           !
12!            IF NRSET <> 0                                                     !
13!               CALL NREDUCE (,[DMUNI],[PNSFS(BC)],,,,,                        !
13!                             [DMUFI]);                                        !
12!         ENDIF;                                                               !
11!      ELSE                                                                    !
11!         [DPFV] := [DPNV];                                                    !
11!         IF NRSET <> 0 [DMUF] := [DMUN];                                      !
11!         IF SYMTRN(BC) THEN                                                   !
12!            [DPFVI] := [DPNVI];                                               !
12!            IF NRSET <> 0 [DMUFI] := [DMUNI];                                 !
12!         ENDIF;                                                               !
11!      ENDIF;                                                                  !
10!$                                                                           $!
10!      IF SYMTRN(BC) THEN                                                      !
11!         [HRDPFV] := [HFREALT(BC)] * [DPFV];                                  !
11!         [HIDPFV] := [HFIMAGT(BC)] * [DPFVI];                                 !
11!         [DPFVX]  := [HRDPFV]  +  [HIDPFV];                                   !
11!         IF NRSET <> 0  THEN                                                  !
12!            [HRDMUF] := [HFREALT(BC)] * [DMUF];                               !
12!            [HIDMUF] := [HFIMAGT(BC)] * [DMUFI];                              !
12!            [DMUFX]  := [HRDMUF]  +  [HIDMUF];                                !
12!         ENDIF;                                                               !
11!      ELSE                                                                    !
11!         [DPFVX]  := [DPFV];                                                  !
11!         IF NRSET <> 0  [DMUFX]  := [DMUF];                                   !
11!      ENDIF;                                                                  !
10!$                                                                           $!
10!      CALL NULLMAT ( [DPAV], [DMUA] );                                        !
```

```
10!           IF NGDR <> 0 THEN                                           !
11!              [DPAV] := TRANS( [GSUBO(BC)] ) * [DPFVX];                !
11!              IF NRSET <> 0 [DMUA]:=TRANS([GSUBO(BC)])*[DMUFX];        !
11!           ELSE                                                        !
11!              IF NOMIT <> 0 THEN                                       !
12!                 CALL FREDUCE ( , [DPFVX], [PFOAX(BC)], 1,             !
12!                             [KOOL(BC,SUB)], [KOOU(BC,SUB)],           !
12!                             [KAO(BC,SUB)], [GASUBO(BC,SUB)], ,        !
12!                             [DPAV], [DPOV], );                        !
12!                 IF NRSET <> 0                                         !
13!                    CALL FREDUCE ( , [DMUFX], [PFOAX(BC)], 1,          !
13!                             [KOOL(BC,SUB)], [KOOU(BC,SUB)],           !
13!                             [KAO(BC,SUB)], [GASUBO(BC,SUB)], ,        !
13!                             [DMUA], [DMUO], );                        !
12!              ELSE                                                     !
12!                 [DPAV] := [DPFVX];                                    !
12!                 IF NRSET <> 0  [DMUA] := [DMUFX];                     !
12!              ENDIF;                                                   !
11!           ENDIF;                                                      !
10!$                                                                     $!
10!           IF NRSET <> 0 THEN                                          !
11!              CALL ROWPART ([DPAV],[DPRV],[DPLV],[PARLX(BC)] );        !
11!              CALL ROWPART ([DMUA],[DMUR],[DMUL],[PARLX(BC)] );        !
11!              CALL GFBS ( [RL11(BC,SUB)], [RU11(BC,SUB)],              !
11!                          [DPLV], [R11DPL] );                          !
11!              [DP1]  := TRANS([D(BC)]) * [DMUL] + [DMUR] -             !
11!                        [R21(BC,SUB)] * [R11DPL];                      !
11!              [DRHS] := TRANS( [D(BC)] ) * [DPLV] + [DPRV] -           !
11!                        [R31(BC,SUB)] * [R11DPL];                      !
11!$                                                                     $!
11!$             PROCESS ACTIVE CONSTRAINTS FOR SAERO DISCIPLINE         $!
11!$                                                                     $!
11!              CALL GFBS ( [KL11(BC,SUB)], [KU11(BC,SUB)],              !
11!                          [DP1],          [DK1V] );                    !
11!              [DRHS] := [DRHS] - [K21(BC,SUB)] * [DK1V];               !
11!$                                                                     $!
11!              CALL AEROSENS ( NITER, BCID, MINDEX, SUB, CONST,         !
11!                              SYM, NDV, BGPDT(BC),                     !
11!                              TRIMDATA, STABCFA, [PGAA],               !
11!                              [LHSA(BC,SUB)], [RHSA(BC,SUB)],          !
11!                              [DRHS], [AAR], [DDELDV], [AMAT] );       !
11!$                                                                     $!
11!              [DURV] := [K1112(BC,SUB)] * [AAR] +                      !
11!                        [PAR(BC,SUB)] * [DDELDV] + [DK1V];             !
11!              [DULV] := [R1112(BC,SUB)] * [DURV] +                     !
11!                        [R1113(BC,SUB)] * [AAR]  -                     !
11!                        [R11PAL(BC,SUB)] * [DDELDV] + [R11DPL];        !
11!              CALL ROWMERGE ([DUAV],[DURV],[DULV],[PARLX(BC)]);        !
11!           ELSE                                                        !
11!$                                                                     $!
11!$             NOTE THAT SAERO W/O SUPPORT IS NOT SUPPORTED            $!
11!$                                                                     $!
11!           ENDIF;                                                      !
10!$                                                                     $!
10!$          RECOVER SENSITIVITIES TO THE F SET                         $!
10!$                                                                     $!
10!           CALL NULLMAT ( [UAFTMP] );                                  !
10!           IF NGDR <> 0 THEN                                           !
11!              [UAFTMP] := [GASUBO(BC,SUB)] * [DUAV];                   !
11!           ELSE                                                        !
11!              IF NOMIT <> 0 THEN                                       !
12!                 IF NRSET <> 0 THEN                                    !
13!                    [TMP1] := [DPOV]+[POARO(BC,SUB)]*[DDELDV];         !
13!                 ELSE                                                  !
13!                    [TMP1] := [DPOV];                                  !
13!                 ENDIF;                                                !
12!                 CALL GFBS ( [KOOL(BC,SUB)], [KOOU(BC,SUB)],           !
12!                             [TMP1], [UOO]);                           !
12!                 [UO] := [GASUBO(BC,SUB)] * [DUAV] + [UOO];            !
12!                 CALL ROWMERGE ( [UAFTMP], [UO], [DUAV],               !
12!                                 [PFOAX(BC)] );                        !
12!              ELSE                                                     !
12!                 [UAFTMP] := [DUAV];                                   !
12!              ENDIF;                                                   !
11!           ENDIF;                                                      !
10!           CALL AROSNSMR ( BCID, SUB, TRIMDATA, NDV, [PGAA],           !
10!                           [PGAU], [DUFVX], [UAFTMP] );                !
10!$                                                                     $!
10!        ENDIF; $   END IF ON ACTUAG                                   $!
 9!     ENDDO; $   END DO ON SUBSCRIPT LOOP                              $!
 8!$                                                                     $!
 8!     IF ACTUAGG THEN                                                   !
 9!$                                                                     $!
 9!$       REDUCE THE LEFT HAND SIDE MATRIX                              $!
 9!$                                                                     $!
 9!        CALL NULLMAT ( [DFDUN] );                                      !
 9!        IF NMPC <> 0 THEN                                              !
10!           CALL GREDUCE ( , [DFDU], [PGMNS(BC)], [TMN(BC)],,           !
10!                          [DFDUN]);                                    !
10!        ELSE                                                           !
```

```
10!                     [DFDUN] := [DFDU];
10!                  ENDIF;
 9!$
 9!                  CALL NULLMAT ( [DFDUF] );
 9!                  IF NSPC <> 0 THEN
10!                     CALL ROWPART ( [DFDUN], , [DFDUF], [PNSFS(BC)] );
10!                  ELSE
10!                     [DFDUF] := [DFDUN];
10!                  ENDIF;
 9!$
 9!$                 TAKE MERGED SENSITIVITIES OF DISPLACEMENTS AND
 9!$                 COMPUTE THE AMAT MATRIX TERMS FOR THE SAERO
 9!$                 CONSTRAINTS
 9!$
 9!                  IF SYMTRN(BC) THEN
10!                     [DUFV]  := [HFREAL(BC)] * [DUFVX];
10!                  ELSE
10!                     [DUFV] := [DUFVX];
10!                  ENDIF;
 9!$
 9!                  CALL MKAMAT ([AMAT], [DFDUF], [DUFV], [DFSV], PCAA,
 9!                               PRAA, [PGAU] );
 9!$
 9!               ENDIF; $   END IF ON ANY ACTIVE DISPLACEMENTS
 8!$
 8!               IF ACTUAGGI THEN
 9!$
 9!$                 REDUCE THE LEFT HAND SIDE MATRIX FOR REFLECTED PORTION
 9!$
 9!                  CALL NULLMAT ( [DFDUNI] );
 9!                  IF NMPC <> 0 THEN
10!                     CALL GREDUCE ( , [DFDUI], [PGMNS(BC)], [TMN(BC)],,
10!                                    [DFDUNI]);
10!                  ELSE
10!                     [DFDUNI] := [DFDUI];
10!                  ENDIF;
 9!$
 9!                  CALL NULLMAT ( [DFDUFI] );
 9!                  IF NSPC <> 0 THEN
10!                     CALL ROWPART ( [DFDUNI], , [DFDUFI], [PNSFS(BC)] );
10!                  ELSE
10!                     [DFDUFI] := [DFDUNI];
10!                  ENDIF;
 9!$
 9!$                 TAKE MERGED SENSITIVITIES OF DISPLACEMENTS AND
 9!$                 COMPUTE THE AMAT MATRIX TERMS FOR THE SAERO
 9!$                 CONSTRAINTS FOR REFLECTED PORTION
 9!$
 9!                  [DUFVI]  := [HFIMAG(BC)] * [DUFVX];
 9!$
 9!                  CALL MKAMAT ([AMAT], [DFDUFI], [DUFVI], [DFSVI], PCAA,
 9!                               PRAAI, [PGAU] );
 9!$
 9!               ENDIF; $   END IF ON ANY ACTIVE DISPLACEMENTS (REFLECTED)
 8!$
 8!            ENDIF;       $   END IF ON ACTIVE AEROELASTIC CONSTRAINTS
 7!$
 7!$           EVALUATE PANEL BUCKLING CONSTRAINT SENSITIVITIES
 7!$
 7!            IF ACTPNL THEN
 8!               CALL PBKLSENS ( BCID, NITER, NDV, GLBDES, LOCLVAR,
 8!                               [PTRANS], PDLIST, [AMAT] );
 8!            ENDIF;
 7!            IF ACTBAR THEN
 8!               CALL EBKLSENS ( BCID, NITER, NDV, CONST, DESLINK, GLBDES,
 8!                               [AMAT] );
 8!            ENDIF;
 7!         ENDIF;          $   END IF ON ACTIVE BOUNDARY CONDITION
 6!      ENDDO;             $   END DO ON ACTIVE BOUNDARY CONDITIONS
 5!$
 5!      CALL CONORDER ( NITER, NUMOPTBC, CASE, CONST, CONSTORD );
 5!$
 5!      CALL OFPGRAD ( NITER, [AMAT], GLBDES, CONST, CONSTORD,
 5!                     GRADIENT );
 5!$
 5!      IF NITER >= OCS AND NITER <= OCE THEN
 6!         PRINT("LOG=('          VANGO MODULE')");
 6!         CALL VANGO ( NITER, NDV, APPCNVRG, MOVLIM, CNVRGLIM,
 6!                      CTL, CTLMIN, NUMOPTBC, CASE, GLBDES, CONST, [AMAT],
 6!                      DESHIST );
 6!      ELSE
 6!         IF NITER >= MPS AND NITER <= MPE THEN
 7!            PRINT("LOG=('          DESIGN MODULE')");
 7!            CALL DESIGN( NITER, NDV, APPCNVRG, CNVRGLIM,
 7!                         CTL, CTLMIN, GLBDES, CONST, CONSTORD,
 7!                         [AMAT], DESHIST );
 7!         ENDIF;
 6!      ENDIF;
 5!$
 5!   ENDIF; $   END IF ON FSD METHOD
```

```
4!       ENDIF;     $   END IF TEST AFTER ACTCON                                  $!
3!    ENDDO;        $   END WHILE LOOP FOR GLOBAL CONVERGENCE                     $!
2!ENDIF;            $   END IF ON OPTIMIZATION                                    $!
1!$                                                                               $!
1!$*******************************************************************************$!
1!$         BEGIN FINAL ANALYSIS LOOP                                             $!
1!$*******************************************************************************$!
1!$                                                                               $!
1!IF NBNDCOND > NUMOPTBC THEN                                                      !
2!$                                                                               $!
2!$  ASSEMBLE THE GLOBAL MATRICES                                                 $!
2!$                                                                               $!
2!   PRINT("LOG=('*****************************')");                               !
2!$                                                                               $!
2!$  ASSEMBLE THE GLOBAL MATRICES                                                 $!
2!$  BEGIN BOUNDARY CONDITION LOOP                                                $!
2!$                                                                               $!
2!   PRINT("LOG=('BEGIN FINAL ANALYSIS')");                                        !
2!   CALL ANALINIT;                                                                !
2!   CALL EMA2 ( , NDV, GSIZEB, GLBDES, GMKCTG, DKVIG, [K1GG],                     !
2!                                     GMMCTG, DMVIG, [M1GG] );                    !
2!   FOR BC = NUMOPTBC + 1 TO NBNDCOND DO                                          !
3!      CALL BCIDVAL ( BC, CASE, BCID );                                           !
3!      PRINT("LOG=('      BOUNDARY CONDITION ',I8)",BCID);                        !
3!$                                                                               $!
3!$     ESTABLISH THE BASE USET AND PARTITIONING DATA FOR THE BC                  $!
3!$                                                                               $!
3!      CALL MKUSET( BCID, GSIZEB, [YS(BC)], [TMN(BC)], [PGMN(BC)], [PNSF(BC)],    !
3!                   [PFOA(BC)], [PARL(BC)], USET(BC) );                           !
3!$                                                                               $!
3!$     MAKE B.C.-DEPENDENT BGPDT FROM BASE, ADDING THE EXTRA POINTS FOR          $!
3!$     THIS B.C.                                                                 $!
3!$                                                                               $!
3!      CALL BCBGPDT( BCID , GSIZEB , BGPDT(BC) , ESIZE(BC) );                     !
3!      GSIZE      := GSIZEB;                                                      !
3!      PSIZE(BC) := ESIZE(BC) + GSIZE;                                            !
3!$                                                                               $!
3!      CALL AROSYMCK (CASE, BGPDT(BC), USET(BC), RELES, BC, TOLVALUE,             !
3!                     STRSYM );                                                   !
3!$                                                                               $!
3!$     PROCESS MATRICES, TRANSFER FUNCTIONS, AND INITIAL CONDITIONS FOR          $!
3!$     THIS B.C.                                                                 $!
3!$                                                                               $!
3!      CALL BCBULK( BCID , PSIZE(BC) , BGPDT(BC) , USET(BC) );                    !
3!$                                                                               $!
3!      CALL BOUND  ( BCID, GSIZE, ESIZE(BC), USET(BC), BLOAD, BMASS, DMODES,      !
3!                    BMODES, BSAERO, BFLUTR, BDYN, BDRSP, BDTR, BMTR, BDFR,       !
3!                    BMFR, BGUST, BBLAST, NMPC, NSPC, NOMIT, NRSET, NGDR );       !
3!$                                                                               $!
3!$     DETERMINE IF ANY M2GG/K2GG INPUT DATA ARE TO BE ADDED                     $!
3!$                                                                               $!
3!      CALL NULLMAT ( [KGG], [MGG] );                                             !
3!      CALL MK2GG ( BCID, GSIZEB, [M2GG], M2GGFLAG, [K2GG], K2GGFLAG );           !
3!      IF M2GGFLAG THEN                                                           !
4!         [MGG] := [M1GG] + [M2GG];                                               !
4!      ELSE                                                                       !
4!         [MGG] := [M1GG];                                                        !
4!      ENDIF;                                                                     !
3!      IF K2GGFLAG THEN                                                           !
4!         [KGG] := [K1GG] + [K2GG];                                               !
4!      ELSE                                                                       !
4!         [KGG] := [K1GG];                                                        !
4!      ENDIF;                                                                     !
3!$                                                                               $!
3!$     CALL THE GRID POINT WEIGHT GENERATOR FOR THIS BOUNDARY CONDITON           $!
3!$                                                                               $!
3!      CALL GPWG ( , BCID, GPWGGRID, [MGG], OGPWG );                              !
3!$                                                                               $!
3!      IF BLOAD <> 0 CALL GTLOAD ( , BCID, GSIZE, BGPDT(BC), GLBDES,              !
4!                                  SMPLOD, [DPTHVI], [DPTHVD], [DPGRVI],          !
4!                                  [DPGRVD], [PG], OGRIDLOD);                     !
3!$                                                                               $!
3!$     PARTITION-REDUCTION OF GLOBAL MATRICES                                    $!
3!$                                                                               $!
3!      IF NBNDCOND > 1 CALL NULLMAT ( [KNN], [PN], [MNN], [GTKN], [GSTKN],        !
4!                                     [GPTKN], [UGTKN] );                         !
3!      IF NMPC <> 0 THEN                                                          !
4!$                                                                               $!
4!$        PERFORM MPC REDUCTION                                                  $!
4!$                                                                               $!
4!         PRINT("LOG=('         MPC REDUCTION')");                                !
4!         CALL GREDUCE ( [KGG], [PG], [PGMN(BC)], [TMN(BC)], [KNN], [PN] );       !
4!         IF BMASS <> 0 CALL GREDUCE ([MGG],,[PGMN(BC)], [TMN(BC)], [MNN]);       !
4!         IF BSAERO <> 0 THEN                                                     !
5!            CALL SPLINFND ( BCID, CASE, MODEL, SPLINE, [GTKG], [GSTKG],          !
5!                            [GPTKG] );                                           !
5!            CALL GREDUCE (, [GTKG], [PGMN(BC)],[TMN(BC)],, [GTKN]);              !
5!            CALL GREDUCE (, [GSTKG], [PGMN(BC)],[TMN(BC)],, [GSTKN]);            !
5!            CALL GREDUCE (, [GPTKG], [PGMN(BC)],[TMN(BC)],, [GPTKN]);            !
5!         ENDIF;                                                                  !
```

```
4!        IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0
5!           CALL GREDUCE (, [UGTKG], [PGMN(BC)],[TMN(BC)], , [UGTKN] );
4!     ELSE
4!$
4!$       NO MPC REDUCTION
4!$
4!        [KNN] := [KGG];
4!        IF BLOAD <> 0 [PN] := [PG];
4!        IF BMASS <> 0 [MNN] := [MGG];
4!        IF BSAERO <> 0 THEN
5!           [GTKN]  := [GTKG];
5!           [GSTKN] := [GSTKG];
5!           [GPTKN] := [GPTKG];
5!        ENDIF;
4!        IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0 [UGTKN]  := [UGTKG];
4!     ENDIF;
3!$
3!$    PERFORM AUTOSPC CALCULATIONS ON THE KNN MATRIX
3!$
3!     PRINT("LOG=('          AUTOSPC COMPUTATIONS')");
3!     CALL GPSP ( , BCID, NGDR, [KNN], BGPDT(BC), [YS(BC)], USET(BC),
3!                 GPST(BC) );
3!     CALL MKPVECT ( USET(BC), [PGMN(BC)], [PNSF(BC)], [PFOA(BC)], [PARL(BC)] );
3!     CALL BOUNDUPD ( BCID, GSIZE, ESIZE(BC), USET(BC), NSPC, NOMIT, NRSET );
3!$
3!     IF NBNDCOND > 1
4!        CALL NULLMAT ( [KFF], [PF], [MFF], [GTKF], [GSTKF],
4!                       [GPTKF], [UGTKF], [KFFX], [PFX], [MFFX] );
3!$
3!     IF NSPC <> 0 THEN
4!$
4!$       PERFORM SPC REDUCTION
4!$
4!        PRINT("LOG=('          SPC REDUCTION')");
4!        CALL NREDUCE ( [KNN], [PN], [PNSF(BC)], [YS(BC)], [KFF], [KFS],
4!                       [KSS], [PF], [PS] );
4!        IF BMASS <> 0 CALL NREDUCE ( [MNN], , [PNSF(BC)], , [MFF]);
4!        IF BSAERO <> 0 THEN
5!           CALL NREDUCE ( , [GTKN], [PNSF(BC)], , , , , [GTKF] );
5!           CALL NREDUCE ( , [GSTKN], [PNSF(BC)], , , , , [GSTKF] );
5!           CALL NREDUCE ( , [GPTKN], [PNSF(BC)], , , , , [GPTKF] );
5!        ENDIF;
4!        IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0
5!           CALL NREDUCE (, [UGTKN], [PNSF(BC)],,,,, [UGTKF]);
4!     ELSE
4!$
4!$       NO SPC REDUCTION
4!$
4!        [KFF] := [KNN];
4!        IF BLOAD <> 0 [PF] := [PN];
4!        IF BMASS <> 0 [MFF] := [MNN];
4!        IF BSAERO <> 0 THEN
5!           [GTKF]  := [GTKN];
5!           [GSTKF] := [GSTKN];
5!           [GPTKF] := [GPTKN];
5!        ENDIF;
4!        IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0 [UGTKF]  := [UGTKN];
4!     ENDIF;
3!$
3!$    ADD IN THE NEW MODULE TO GENERATE THE H MATRIX FOR SYM-TRAN
3!$
3!     CALL SAERODRV (BCID, 1, LOOP, MINDEX, SYM, MACH, QDP,
3!                    TRIMDATA, TRIMRSLT, METHOD );
3!     SYMTRN(BC) := FALSE;
3!     ACSMTR(BC) := FALSE;
3!     IF METHOD=USS THEN
4!        IF STRSYM THEN
5!           IF SYM=0 THEN
6!              SYMTRN(BC) := TRUE;
6!           ENDIF;
5!        ENDIF;
4!     ELSE
4!        IF METHOD=QP THEN
5!           IF STRSYM THEN
6!              IF SYM=0 OR SYM=-1 THEN
7!                 SYMTRN(BC) := TRUE;
7!                 ACSMTR(BC) := TRUE;
7!              ENDIF;
6!           ENDIF;
5!        ENDIF;
4!     ENDIF;
3!     IF SYMTRN(BC) THEN
4!        CALL RBMGEN ( BGPDT(BC), 20, [RGBDMG] );
4!        CALL AROHGEN (CASE, BGPDT(BC), USET(BC), RELES, BC, TOLVALUE,
4!                      [HFREALT(BC)], [HFIMAGT(BC)], USETX(BC) );
4!        CALL MKPVECT ( USETX(BC), [PGMNX(BC)], [PNSFX(BC)],
4!                       [PFOAX(BC)], [PARLX(BC)] );
4!        CALL TRNSPOSE ( [HFREALT(BC)], [HFREAL(BC)] );
4!        CALL TRNSPOSE ( [HFIMAGT(BC)], [HFIMAG(BC)] );
4!        IF ACSMTR(BC) THEN
```

```
5!             CALL TRNSPOSE ( [KREALK], [KREALT] );                              !
5!             CALL TRNSPOSE ( [KIMAGK], [KIMAGT] );                              !
5!          ENDIF;                                                                !
4!          [HTKFHR]  := [HFREALT(BC)] * [ [KFF] * [HFREAL(BC)] ];                 !
4!          [HTKFHI]  := [HFIMAGT(BC)] * [ [KFF] * [HFIMAG(BC)] ];                 !
4!          [KFFX]    := [HTKFHR] + [HTKFHI];                                     !
4!          IF BMASS <> 0 THEN                                                    !
5!             [HTMFHR] := [HFREALT(BC)] * [ [MFF] * [HFREAL(BC)] ];               !
5!             [HTMFHI] := [HFIMAGT(BC)] * [ [MFF] * [HFIMAG(BC)] ];               !
5!             [MFFX]   := [HTMFHR] + [HTMFHI];                                   !
5!          ENDIF;                                                                !
4!          IF BSAERO = 0 THEN                                                    !
5!             [HFRTPF]  := [HFREALT(BC)] * [PF];                                 !
5!             [HFITPF]  := [HFIMAGT(BC)] * [PF];                                 !
5!             [PFX]     := [HFRTPF] + [HFITPF];                                  !
5!          ELSE                                                                  !
5!             IF ACSMTR(BC) THEN                                                 !
6!                [HRGTKF]   := [HFREALT(BC)] * [ [GTKF] * [KREALK] ];            !
6!                [HIGTKF]   := [HFIMAGT(BC)] * [ [GTKF] * [KIMAGK] ];            !
6!                [HRGSTKF]  := [HFREALT(BC)] * [ [GSTKF] * [KREALK] ];           !
6!                [HIGSTKF]  := [HFIMAGT(BC)] * [ [GSTKF] * [KIMAGS] ];           !
6!                [HRGPTKF]  := [HFREALT(BC)] * [ [GPTKF] * [KREALK] ];           !
6!                [HIGPTKF]  := [HFIMAGT(BC)] * [ [GPTKF] * [KIMAGP] ];           !
6!             ELSE                                                               !
6!                [HRGTKF]   := [HFREALT(BC)] * [GTKF];                           !
6!                [HIGTKF]   := [HFIMAGT(BC)] * [GTKF] ;                          !
6!                [HRGSTKF]  := [HFREALT(BC)] * [GSTKF];                          !
6!                [HIGSTKF]  := [HFIMAGT(BC)] * [GSTKF] ;                         !
6!                [HRGPTKF]  := [HFREALT(BC)] * [GPTKF];                          !
6!                [HIGPTKF]  := [HFIMAGT(BC)] * [GPTKF] ;                         !
6!             ENDIF;                                                             !
5!          ENDIF;                                                                !
4!       ELSE                                                                     !
4!          [KFFX]      := [KFF];                                                 !
4!          [MFFX]      := [MFF];                                                 !
4!          [PFX]       := [PF];                                                  !
4!          [GPTKFX]    := [GPTKF];                                               !
4!          USETX(BC)   := USET(BC);                                              !
4!          [PGMNX(BC)] := [PGMN(BC)];                                            !
4!          [PNSFX(BC)] := [PNSF(BC)];                                            !
4!          [PFOAX(BC)] := [PFOA(BC)];                                            !
4!          [PARLX(BC)] := [PARL(BC)];                                            !
4!       ENDIF;                                                                   !
3!$                                                                              $!
3!       IF NBNDCOND > 1 CALL NULLMAT ([KAA], [PA], [MAA], [KAAA], [PAA], [UGTKA]);!
3!$                                                                              $!
3!       IF NGDR <> 0 THEN                                                        !
4!$                                                                              $!
4!$         PERFORM THE GENERAL DYNAMIC REDUCTION WHICH IS DISCIPLINE            $!
4!$         INDEPENDENT.  THE RESULTING [GSUBO] MATRIX WILL BE USED BY           $!
4!$         ALL DISCIPLINES                                                      $!
4!$                                                                              $!
4!          PRINT("LOG=('          DYNAMIC REDUCTION')");                         !
4!$                                                                              $!
4!$         OBTAIN THE OMITTED DOF PARTITION OF KFF AND MFF                      $!
4!$                                                                              $!
4!          CALL PARTN ( [KFFX], [KOO], , [KOA], , [PFOAX(BC)] );                 !
4!          CALL PARTN ( [MFFX], [MOO], , , , [PFOAX(BC)] );                      !
4!          ASIZE := GSIZE - NMPC - NSPC - NOMIT;                                 !
4!          LSIZE := ASIZE - NRSET;                                               !
4!          CALL GDR1 ( [KOO], [MOO], [KSOO], [GGO], LKSET, LJSET, NEIV,          !
4!                      FMAX, BCID, BGPDT(BC), USETX(BC), NOMIT, LSIZE );         !
4!$                                                                              $!
4!$         LKSET              MEANING                                           $!
4!$               <> 0         APPROX. MODE SHAPES SELECTED                      $!
4!$                = 0         NO APPROX. MODE SHAPES IN GDR                     $!
4!$                                                                              $!
4!          IF LKSET <> 0  THEN                                                   !
5!             CALL SDCOMP ( [KSOO], [LSOO], USETX(BC), SINGOSET );               !
5!             CALL GDR2   ( [LSOO], [MOO], [PHIOK], LKSET, LJSET,                !
5!                           NEIV,   FMAX,  BCID );                               !
5!          ENDIF;                                                                !
4!          CALL GDR3 ( [KOO], [KOA], [MGG], [PHIOK], [TMN(BC)], [GGO],           !
4!                      [PGMNX(BC)], [PNSFX(BC)], [PFOAX(BC)], [GSUBO(BC)],       !
4!                      BGPDT(BC), USETX(BC),                                     !
4!                      LKSET, LJSET, ASIZE, GNORM, BCID );                       !
4!          CALL GDR4 ( BCID, GSIZE, PSIZE(BC), LKSET, LJSET,                     !
4!                      [PGMNX(BC)], [TMN(BC)], [PNSFX(BC)], [PFOAX(BC)],         !
4!                      [PARLX(BC)], [PGDRG(BC)], [PAJK], [PFJK], BGPDT(BC),      !
4!                      USETX(BC) );                                              !
4!       ENDIF;                                                                   !
3!$                                                                              $!
3!       IF BLOAD <> 0 OR BMODES <> 0 OR BFLUTR <> 0 OR BDYN <> 0 THEN            !
4!$                                                                              $!
4!$         REDUCE THE MATRICES WITHOUT AEROELASTIC CORRECTIONS                  $!
4!$                                                                              $!
4!          IF NGDR <> 0 THEN                                                     !
5!$                                                                              $!
5!$            PERFORM THE GENERAL DYNAMIC REDUCTION                             $!
5!$                                                                              $!
```

```
5!          PRINT("LOG=('          SYMMETRIC DYNAMIC REDUCTION')");          !
5!$                                                                           $!
5!          [MAA] := TRANS ( [GSUBO(BC)] ) * [ [MFFX] * [GSUBO(BC)] ];       !
5!          [KAA] := TRANS ( [GSUBO(BC)] ) * [ [KFFX] * [GSUBO(BC)] ];       !
5!          IF BLOAD <> 0  [PA]     := TRANS ( [GSUBO(BC)] ) * [PFX];        !
5!          IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0 THEN                 !
6!             [TMP1]   := TRANS ( [UGTKF] ) * [GSUBO(BC)];                  !
6!             CALL TRNSPOSE ( [TMP1], [UGTKA] );                            !
6!          ENDIF;                                                           !
5!       ELSE                                                                !
5!          IF NOMIT <> 0 THEN                                               !
6!$                                                                           $!
6!$            PERFORM THE STATIC REDUCTION                                   $!
6!$                                                                           $!
6!             PRINT("LOG=('          STATIC CONDENSATION')");               !
6!$                                                                           $!
6!             CALL FREDUCE ( [KFFX], [PFX], [PFOAX(BC)], , [KOOINV(BC)], , , !
6!                            [GSUBO(BC)], [KAA], [PA], [PO], USETX(BC) );   !
6!$                                                                           $!
6!             IF BMASS <> 0 THEN                                            !
7!$                                                                           $!
7!$               PERFORM GUYAN REDUCTION OF THE MASS MATRIX                  $!
7!$                                                                           $!
7!                CALL PARTN ( [MFFX], [MOO], , [MOA], [MAABAR], [PFOAX(BC)] ); !
7!                [MAA] := [MAABAR] + TRANS([MOA]) * [GSUBO(BC)] +           !
7!                         TRANS([GSUBO(BC)]) * [MOA] +                      !
7!                         TRANS([GSUBO(BC)]) * [ [MOO] * [GSUBO(BC)] ];     !
7!                IF NRSET <> 0 [IFM(BC)] := [MOO] * [GSUBO(BC)] + [MOA];    !
7!             ENDIF;                                                        !
6!             IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0 THEN              !
7!                CALL ROWPART ( [UGTKF], [UGTKO], [UGTKAB], [PFOAX(BC)] );  !
7!                [TMP1]   := TRANS( [UGTKO] ) * [GSUBO(BC)];                !
7!                CALL TRNSPOSE ( [TMP1], [TMP2] );                          !
7!                [UGTKA] := [UGTKAB] + [TMP2];                              !
7!             ENDIF;                                                        !
6!          ELSE                                                             !
6!$                                                                           $!
6!$            NO F-SET REDUCTION                                             $!
6!$                                                                           $!
6!             [KAA] := [KFFX];                                              !
6!             IF BLOAD <> 0 [PA] := [PFX];                                  !
6!             IF BFLUTR <> 0 OR BGUST <> 0 OR BBLAST <> 0 [UGTKA]:=[UGTKF]; !
6!             IF BMASS <> 0 [MAA] := [MFFX];                                !
6!          ENDIF;                                                           !
5!       ENDIF;                                                              !
4!$                                                                           $!
4!       IF NRSET <> 0 THEN                                                  !
5!$                                                                           $!
5!$         PERFORM THE SUPPORT SET REDUCTION                                 $!
5!$                                                                           $!
5!          PRINT("LOG=('          SUPPORT REDUCTION')");                    !
5!          CALL PARTN  ( [KAA], [KRR], [KLR], , [KLL], [PARLX(BC)] );       !
5!          CALL SDCOMP ( [KLL], [KLLINV(BC)], USETX(BC), SINGLSET );        !
5!          CALL FBS ( [KLLINV(BC)], [KLR], [D(BC)], -1 );                   !
5!          CALL RBCHECK ( BCID, USETX(BC), BGPDT(BC), [D(BC)], [KLL],       !
5!                         [KRR], [KLR] );                                   !
5!$                                                                           $!
5!$         CALCULATE THE REDUCED MASS MATRIX                                 $!
5!$                                                                           $!
5!          CALL PARTN ([MAA], [MRRBAR], [MLR], , [MLL], [PARLX(BC)]);       !
5!          [IFR(BC)] := [MLL] * [D(BC)] + [MLR];                            !
5!          [MRR(BC)] := [MRRBAR] + TRANS ( [MLR] )    * [D(BC)] +           !
5!                                  TRANS ( [D(BC)] ) * [IFR(BC)];           !
5!          [R22]     := TRANS ( [D(BC)] ) * [MLR] + [MRRBAR];               !
5!$                                                                           $!
5!          IF BLOAD <> 0 THEN                                               !
6!$                                                                           $!
6!$            PROCESS STATICS WITH INERTIA RELIEF                            $!
6!$                                                                           $!
6!             PRINT("LOG=('          >>>DISCIPLINE: STATICS(INERTIA RELIEF)')");!
6!             CALL ROWPART ( [PA], [PR], [PLBAR], [PARLX(BC)] );            !
6!             [LHS(BC)] := [MRR(BC)];                                       !
6!             [RHS(BC)] := TRANS([D(BC)]) * [PLBAR] + [PR];                 !
6!             CALL INERTIA ( [LHS(BC)], [RHS(BC)], [AR] );                  !
6!             [AL]       := [D(BC)] * [AR];                                 !
6!             CALL ROWMERGE ( [AA], [AR], [AL], [PARLX(BC)] );              !
6!             [RHS(BC)] := [PLBAR] - [IFR(BC)] * [AR];                      !
6!             CALL FBS ( [KLLINV(BC)], [RHS(BC)], [UL] );                   !
6!             CALL YSMERGE ( [UA], , [UL], [PARLX(BC)] );                   !
6!          ENDIF;                                                           !
5!          IF BMODES <> 0 THEN                                              !
6!             PRINT("LOG=('          >>>DISCIPLINE: NORMAL MODES')");       !
6!             CALL REIG ( , BCID, USETX(BC), [KAA], [MAA], [MRR(BC)],       !
6!                         [D(BC)], LAMBDA, [PHIA], [MII], HSIZE(BC) );      !
6!             CALL OFPMROOT ( , BCID, LAMBDA );                             !
6!          ENDIF;                                                           !
5!       ELSE                                                                !
5!$                                                                           $!
5!$         NO SUPPORT SET REDUCTION                                          $!
5!$                                                                           $!
```

```
5!            IF BLOAD <> 0 THEN
6!               PRINT("LOG=('        >>>DISCIPLINE: STATICS')");
6!               CALL SDCOMP ( [KAA], [KLLINV(BC)], USETX(BC), SINGASET );
6!               CALL FBS ( [KLLINV(BC)], [PA], [UA] );
6!            ENDIF;
5!            IF BMODES <> 0 THEN
6!               PRINT("LOG=('        >>>DISCIPLINE: NORMAL MODES')");
6!               CALL REIG ( , BCID, USETX(BC), [KAA], [MAA], , , LAMBDA,
6!                           [PHIA], [MII], HSIZE(BC) );
6!               CALL OFPMROOT ( , BCID, LAMBDA );
6!            ENDIF;
5!         ENDIF;
4!      ENDIF;
3!$
3!      IF BSAERO <> 0 THEN
4!$
4!$        PERFORM STATIC AEROELASTIC ANALYSES
4!$
4!         PRINT("LOG=('         SAERO INITIALIZATION')");
4!         CALL TRNSPOSE ( [GSTKF], [GSKF] );
4!         IF SYMTRN(BC) THEN
5!            CALL TRNSPOSE ( [HRGSTKF], [GSKFHRT] );
5!            CALL TRNSPOSE ( [HIGSTKF], [GSKFHIT] );
5!         ENDIF;
4!         LOOP := TRUE;
4!         SUB  := 0;
4!         WHILE LOOP DO
5!            SUB := SUB + 1;
5!            CALL FTRIMDRV ( BCID, SUB, TRIMDATA, METHOD, MODEL, MACH,
5!                            SYM, SAMODEL, SAEMODEL, STDYGEOM, RIGDALOD,
5!                            RIGDSLOD, FLEXLOAD, AICMAT, AEROGRID, CAEROBOX,
5!                            SACOMPS, SAGEOM, [AIC], [AAIC], [ASAIC] );
5!            CALL SAERODRV (BCID, SUB, LOOP, MINDEX, SYM, MACH, QDP,
5!                           TRIMDATA, TRIMRSLT, METHOD, 1);
5!$
5!$           ADJUST THE KFF MATRIX AND DETERMINE THE RIGID AIR LOADS
5!$
5!            IF SYMTRN(BC) THEN
6!              IF ACSMTR(BC) THEN
7!                 [AICS1] := [HRGTKF]*[TRANS([ASAIC])*[GSKFHRT]];
7!                 [AICS2] := [HRGTKF]*[TRANS([ASAIC])*[GSKFHIT]] + [AICS1];
7!                 [AICS3] := [HIGTKF]*[TRANS([ASAIC])*[GSKFHRT]] + [AICS2];
7!                 [AICS]  := [HIGTKF]*[TRANS([ASAIC])*[GSKFHIT]] + [AICS3];
7!              ELSE
7!                 [AICSUM] := (0.5) [AIC] + (0.5)  [AAIC];
7!                 [AICDIF] := (0.5) [AIC] - (0.5)  [AAIC];
7!                 [AICS1] := [HRGTKF]*[TRANS([AICSUM])*[GSKFHRT]];
7!                 [AICS2] := [HRGTKF]*[TRANS([AICDIF])*[GSKFHIT]] + [AICS1];
7!                 [AICS3] := [HIGTKF]*[TRANS([AICDIF])*[GSKFHRT]] + [AICS2];
7!                 [AICS]  := [HIGTKF]*[TRANS([AICSUM])*[GSKFHIT]] + [AICS3];
7!              ENDIF;
6!            ELSE
6!              IF SYM =  1 [AICS] := [GTKF]*[TRANS([AIC])*[GSKF]];
6!              IF SYM = -1 [AICS] := [GTKF]*[TRANS([AAIC])*[GSKF]];
6!$
6!$       ADD IN OPTION FOR ASYMMETRIC AIC
6!$
6!              IF SYM =  0 [AICS] := [GTKF]*[TRANS([ASAIC])*[GSKF]];
6!            ENDIF;
5!$
5!$
5!$       DEFINE ZERO LOAD VECTOR FOR ACCELERATION PARAMETERS
5!$
5!            CALL ACCFGEN (, BCID, SUB, SYMTRN(BC), RIGDALOD, RIGDSLOD,
5!                          STDYGEOM, TRIMDATA, CONLINK, TRIMTOC, [KFFX],
5!                          TLABEL, [ACCFORCE], [ACCELOAD], MACH);
5!$
5!$
5!$       USER DEFINED LOADS FROM STATIC LOAD PARAMETER DEFINITION
5!$
5!            CALL UDEFGEN (, BCID, SUB, SYMTRN(BC), RIGDSLOD, [UDGFORCE],
5!                          [UDFALOAD], GSIZE, TLABEL, TRIMDATA, STDYGEOM,
5!                          TRIMTOC, MACH, YESUDEF);
5!$
5!            IF NMPC <> 0 THEN
6!               CALL GREDUCE (, [UDGFORCE], [PGMN(BC)],[TMN(BC)],, [UDNFORCE]);
6!            ELSE
6!               [UDNFORCE] := [UDGFORCE];
6!            ENDIF;
5!$
5!            CALL NREDUCE ( , [UDNFORCE], [PNSF(BC)], , , , , [UDFFORCE] );
5!$
5!            IF SYMTRN(BC) THEN
6!               CALL UDEFTRAN (, BCID, SUB, [UDFFORCE], TRIMTOC, [HFREALT(BC)],
6!                              [HFIMAGT(BC)], [UDFFORCX] );
6!            ELSE
6!               [UDFFORCX] := [UDFFORCE];
6!            ENDIF;
5!$
5!$
```

```
5!$     NEW AIR FORCE MERGE ROUTINE                                              $!
5!$                                                                              $!
5!         IF SYMTRN(BC) THEN                                                     !
6!            CALL ARFMRG (, BCID, SUB, SYMTRN(BC), TRIMDATA, TLABEL,             !
6!                      RIGDALOD, STDYGEOM, [HRGPTKF], [HIGPTKF], CASE,           !
6!                      [AEROLOAD], [AIRFORCE], TRIMTOC, MACH, YESAERO);          !
6!         ELSE                                                                   !
6!            CALL ARFMRG (, BCID, SUB, SYMTRN(BC), TRIMDATA, TLABEL,             !
6!                      RIGDALOD, STDYGEOM, [GPTKF], [HIGPTKF], CASE,             !
6!                      [AEROLOAD], [AIRFORCE], TRIMTOC, MACH, YESAERO);          !
6!         ENDIF;                                                                 !
5!$                                                                              $!
5!$        MERGE LOADS IN THE AERODYNAMIC DOMAIN FOR STABILITY DERIVATIVES       $!
5!$                                                                              $!
5!         [SAROLOAD] := [ACCELOAD];                                              !
5!         IF YESUDEF THEN                                                        !
6!            CALL APPEND ( [UDFALOAD], [SAROLOAD] );                             !
6!         ENDIF;                                                                 !
5!         IF YESAERO THEN                                                        !
6!            CALL APPEND ( [AEROLOAD], [SAROLOAD] );                             !
6!         ENDIF;                                                                 !
5!         CALL RIGDSTAB ( BCID, SUB, TRIMTOC, [SAROLOAD], STDYGEOM,              !
5!                         BGPDT(BC), QDP, STABCFA );                             !
5!$                                                                              $!
5!$        MERGE LOADS IN THE STRUCTURAL DOMAIN FOR AEROLELATIC SOLUTION         $!
5!$                                                                              $!
5!         [PAF]    := [ACCFORCE];                                                !
5!         IF YESUDEF THEN                                                        !
6!            CALL APPEND ( [UDFFORCX], [PAF] );                                  !
6!         ENDIF;                                                                 !
5!         IF YESAERO THEN                                                        !
6!            [PAFX]   := (QDP) [AIRFORCE];                                       !
6!            CALL APPEND ( [PAFX], [PAF] );                                      !
6!         ENDIF;                                                                 !
5!         CALL UTMPRG ( [PAFX] );                                                !
5!         [KAFF] := [KFFX] - (QDP) [AICS];                                       !
5!$        CALL UTMPRG ( [AICS] );                                               $!
5!$                                                                              $!
5!$        REDUCE THE MATRICES WITH AEROELASTIC CORRECTIONS                      $!
5!$        SAVE THE SUBCASE/BC DEPENDENT DATA FOR SENSITIVITY ANALYSIS           $!
5!$                                                                              $!
5!         IF NGDR <> 0 THEN                                                      !
6!$                                                                              $!
6!$           PERFORM THE GENERAL DYNAMIC REDUCTION                              $!
6!$                                                                              $!
6!            PRINT("LOG=('          SAERO DYNAMIC REDUCTION')");                 !
6!            [MAAA] := TRANS ( [GSUBO(BC)] ) * [ [MFFX] * [GSUBO(BC)] ];         !
6!            [KAAA] := TRANS ( [GSUBO(BC)] ) * [ [KAFF] * [GSUBO(BC)] ];         !
6!            [PAA]  := TRANS ( [GSUBO(BC)] ) * [PAF];                            !
6!         ELSE                                                                   !
6!            IF NOMIT <> 0 THEN                                                  !
7!$                                                                              $!
7!$              PERFORM THE STATIC REDUCTION                                    $!
7!$                                                                              $!
7!               PRINT("LOG=('          SAERO STATIC CONDENSATION')");            !
7!$                                                                              $!
7!               IF NRSET <> 0 AND SUB = 1 AND BLOAD = 0 AND BMODES = 0 AND       !
8!                  BFLUTR = 0 AND BDYN = 0 THEN                                  !
8!$                                                                              $!
8!$                 FORM [KAA] ON SO [D] CAN BE FORMED                           $!
8!$                                                                              $!
8!                  CALL FREDUCE ([KFFX], , [PFOAX(BC)], , [KOOINV(BC)], , ,      !
8!                                [GSUBO(BC)], [KAA], , , USETX(BC) );            !
8!               ENDIF;                                                           !
7!$                                                                              $!
7!               CALL FREDUCE ( [KAFF], [PAF], [PFOAX(BC)], BSAERO,               !
7!                              [KOOL(BC,SUB)], [KOOU(BC,SUB)],                   !
7!                              [KAO(BC,SUB)], [GASUBO(BC,SUB)], [KAAA],          !
7!                              [PAA], [POARO(BC,SUB)], USETX(BC));               !
7!$                                                                              $!
7!               IF BMASS <> 0 THEN                                               !
8!$                                                                              $!
8!$                 PERFORM GUYAN REDUCTION OF THE MASS MATRIX                   $!
8!$                                                                              $!
8!                  CALL PARTN ( [MFFX], [MOO], , [MOA], [MAABAR],                !
8!                               [PFOAX(BC)] );                                   !
8!                  [MAAA] := [MAABAR] + TRANS([MOA]) * [GASUBO(BC,SUB)] +        !
8!                            TRANS([GASUBO(BC,SUB)]) * [MOA] +                   !
8!                            TRANS([GASUBO(BC,SUB)]) * [[MOO] *                  !
8!                            [GASUBO(BC,SUB)]];                                  !
8!                  IF NRSET <> 0                                                 !
9!                            [IFMA(BC,SUB)] := [MOO]*[GASUBO(BC,SUB)]+[MOA];     !
8!               ENDIF;                                                           !
7!            ELSE                                                                !
7!$                                                                              $!
7!$              NO F-SET REDUCTION                                              $!
7!$                                                                              $!
7!               IF NRSET <> 0 AND SUB    = 1 AND BLOAD = 0  AND                  !
8!                  BMODES = 0 AND BFLUTR = 0 AND BDYN = 0 THEN                   !
8!$                                                                              $!
```

```
8!$                 FORM [KAA] ON FIRST PASS SO [D] CAN BE FORMED            $!
8!$                                                                          $!
8!                [KAA] := [KFFX];                                            !
8!              ENDIF;                                                        !
7!              [KAAA] := [KAFF];                                             !
7!              [MAAA] := [MFFX];                                             !
7!              [PAA]  := [PAF];                                              !
7!            ENDIF;                                                          !
6!          ENDIF;                                                            !
5!$                                                                          $!
5!          IF NRSET <> 0 THEN                                                !
6!$                                                                          $!
6!              PERFORM THE SUPPORT SET REDUCTION                            $!
6!$                                                                          $!
6!              PRINT("LOG=('        SAERO SUPPORT REDUCTION')");             !
6!$                                                                          $!
6!              IF SUB = 1 AND BLOAD = 0 AND BMODES = 0 AND BFLUTR = 0        !
7!                         AND BDYN  = 0 THEN                                 !
7!$                                                                          $!
7!                [D] WAS NOT COMPUTED FOR NON-SAERO DISCIPLINES SO          $!
7!                NEED TO COMPUTE IT NOW                                     $!
7!$                                                                          $!
7!                CALL PARTN  ( [KAA], [KRR], [KLR], , [KLL], [PARLX(BC)] );  !
7!                CALL SDCOMP ( [KLL], [KLLINV(BC)], USETX(BC), SINGLSET );   !
7!                CALL FBS ( [KLLINV(BC)], [KLR], [D(BC)], -1 );              !
7!                CALL RBCHECK ( BCID, USETX(BC), BGPDT(BC), [D(BC)], [KLL],  !
7!                               [KRR], [KLR] );                              !
7!              ENDIF;                                                        !
6!$                                                                          $!
6!              CALCULATE THE REDUCED MASS MATRIX                            $!
6!$                                                                          $!
6!              CALL PARTN ([MAAA], [MRRBAR], [MLR], , [MLL], [PARLX(BC)]);   !
6!              [R13(BC,SUB)] := [MLL] * [D(BC)] + [MLR];                     !
6!              [R33]         := [MRRBAR] + TRANS ( [MLR] ) * [D(BC)] +       !
6!                               TRANS ( [D(BC)] ) * [R13(BC,SUB)];           !
6!              [R22]         := TRANS ( [D(BC)] ) * [MLR] + [MRRBAR];        !
6!              CALL TRNSPOSE ( [R13(BC,SUB)], [R21(BC,SUB)] );               !
6!$                                                                          $!
6!              PROCESS STEADY AEROELASTIC DISCIPLINE                        $!
6!$                                                                          $!
6!              PRINT("LOG=('        >>>DISCIPLINE: STEADY AERO')");          !
6!              CALL PARTN ( [KAAA], [KARR], [R12(BC,SUB)], [KARL], [R11],    !
6!                           [PARLX(BC)] );                                   !
6!              [R32(BC,SUB)] := TRANS([D(BC)]) * [R12(BC,SUB)] + [KARR];     !
6!              [R31(BC,SUB)] := TRANS([D(BC)]) * [R11] + [KARL];             !
6!$                                                                          $!
6!              CALL DECOMP ( [R11], [RL11(BC,SUB)], [RU11(BC,SUB)] );        !
6!$                                                                          $!
6!              CALL ROWPART ( [PAA], [PARBAR], [PAL], [PARLX(BC)] );         !
6!              CALL GFBS ( [RL11(BC,SUB)], [RU11(BC,SUB)], [PAL],            !
6!                          [R11PAL(BC,SUB)], -1);                            !
6!              [PRIGID] := [PARBAR] + TRANS([D(BC)]) * [PAL];                !
6!              [P1]     := [R21(BC,SUB)] * [R11PAL(BC,SUB)];                 !
6!              [P2]     := [PRIGID] + [R31(BC,SUB)] * [R11PAL(BC,SUB)];      !
6!$                                                                          $!
6!              CALL GFBS ( [RL11(BC,SUB)], [RU11(BC,SUB)], [R12(BC,SUB)],    !
6!                          [R1112(BC,SUB)], -1);                             !
6!              CALL GFBS ( [RL11(BC,SUB)], [RU11(BC,SUB)], [R13(BC,SUB)],    !
6!                          [R1113(BC,SUB)], -1);                             !
6!              [K11]         := [R22] + [R21(BC,SUB)] * [R1112(BC,SUB)];     !
6!              [K12(BC,SUB)] := [R21(BC,SUB)] * [R1113(BC,SUB)];             !
6!              [K21(BC,SUB)] := [R32(BC,SUB)] + [R31(BC,SUB)] * [R1112(BC,SUB)];!
6!              [K22]         := [R33] + [R31(BC,SUB)] * [R1113(BC,SUB)];     !
6!$                                                                          $!
6!              CALL DECOMP ( [K11], [KL11(BC,SUB)], [KU11(BC,SUB)] );        !
6!              CALL GFBS ( [KL11(BC,SUB)], [KU11(BC,SUB)], [P1],             !
6!                          [PAR(BC,SUB)] );                                  !
6!              CALL GFBS ( [KL11(BC,SUB)], [KU11(BC,SUB)], [K12(BC,SUB)],    !
6!                          [K1112(BC,SUB)],-1 );                             !
6!              [LHSA(BC,SUB)] := [K22] + [K21(BC,SUB)] * [K1112(BC,SUB)];    !
6!              [RHSA(BC,SUB)] := [P2] - [K21(BC,SUB)] * [PAR(BC,SUB)];       !
6!$                                                                          $!
6!              FLEXIBLE STABILITY COEFFICIENTS COMPUTATION                  $!
6!$                                                                          $!
6!              CALL FLEXSTAB ( , BCID, MINDEX, SUB, SYM, QDP, TRIMDATA,      !
6!                              STABCFA, STABCFS, BGPDT(BC), [LHSA(BC,SUB)],  !
6!                              [RHSA(BC,SUB)], [AAR], [DELTA(SUB)], [PRIGID], !
6!                              [R33], CONST, AEFLG(SUB), [AARC], [DELC],     !
6!                              SYMTRN(BC), STDYGEOM, DOTRMCON );             !
6!$                                                                          $!
6!              GENERATE FLEXIBLE STRUCTURAL TRIM PARAMETER LOAD VECTORS     $!
6!              AND DEFLECTION VECTORS TO BE LOADED INTO GROUP FLEXLOAD      $!
6!$                                                                          $!
6!              CALL GFBS ( [RL11(BC,SUB)], [RU11(BC,SUB)], [R13(BC,SUB)],    !
6!                          [ULFLX2], -1);                                    !
6!              [ULFLX1] := - [R11PAL(BC,SUB)];                               !
6!              CALL ROWMERGE ( [UAFLX1], , [ULFLX1], [PARLX(BC)] );          !
6!              CALL ROWMERGE ( [UAFLX2], , [ULFLX2], [PARLX(BC)] );          !
6!$                                                                          $!
6!              IF NOMIT <> 0 THEN                                            !
```

```
7!              CALL GFBS ( [KOOL(BC,SUB)], [KOOU(BC,SUB)], [POARO(BC,SUB)],
7!                          [KOOPOA], 1);
7!              CALL TRNSPOSE ( [KAO(BC,SUB)], [KAOT(BC,SUB)] );
7!              CALL GFBS ( [KOOL(BC,SUB)], [KOOU(BC,SUB)], [KAOT(BC,SUB)],
7!                          [KOOKAO], -1);
7!              [UOFLX1] := [KOOPOA] + [KOOKAO]*[UAFLX1];
7!              [UOFLX2] := [KOOKAO]*[UAFLX2];
7!              CALL ROWMERGE ( [UFFLX1], [UOFLX1], [UAFLX1], [PFOAX(BC)] );
7!              CALL ROWMERGE ( [UFFLX2], [UOFLX2], [UAFLX2], [PFOAX(BC)] );
7!           ELSE
7!              [UFFLX1] := [UAFLX1];
7!              [UFFLX2] := [UAFLX2];
7!           ENDIF;
6!$
6!           IF SYMTRN(BC) THEN
7!              IF ACSMTR(BC) THEN
8!                 [PAG] := (QDP)[GPTKG]*[KREALK]*[SAROLOAD];
8!                 [PAGI] := (QDP)[GPTKG]*[KIMAGK]*[SAROLOAD];
8!              ELSE
8!                 [PAG] := (QDP)[GPTKG]*[SAROLOAD];
8!              ENDIF;
7!           ELSE
7!                 [PAG] := (QDP)[GPTKG]*[SAROLOAD];
7!           ENDIF;                                                        $
6!$
6!           IF SYMTRN(BC) THEN
7!              [UFX1HRT] := [HFREAL(BC)]*[UFFLX1];
7!              [UFX1HIT] := [HFIMAG(BC)]*[UFFLX1];
7!              [UFX2HRT] := [HFREAL(BC)]*[UFFLX2];
7!              [UFX2HIT] := [HFIMAG(BC)]*[UFFLX2];
7!              IF ACSMTR(BC) THEN
8!                 [GTKGKR]   :=[GTKG]*[KREALK];
8!                 [GTKGKI]   :=[GTKG]*[KIMAGK];
8!                 [FLXTMP1]  :=[GSKF]*[HFREAL(BC)]+[GSKF]*[HFIMAG(BC)];
8!                 [FLXTMP2]  :=[[KREALT]+[KIMAGT]]*[FLXTMP1];
8!                 [FLXTMP3]  :=TRANS([ASAIC])*[FLXTMP2];
8!                 [FLXKX1]   :=[FLXTMP3]*[UFFLX1];
8!                 [FLXKX2]   :=[FLXTMP3]*[UFFLX2];
8!$
8!                 [FLXF1R] := (QDP)[GTKGKR]*[FLXKX1]  ;
8!                 [FLXF1I] := (QDP)[GTKGKI]*[FLXKX1]  ;
8!                 [FLXF2R] := (QDP)[GTKGKR]*[FLXKX2]  ;
8!                 [FLXF2I] := (QDP)[GTKGKI]*[FLXKX2]  ;
8!              ELSE
8!                 [FLXTMPS] :=(QDP)[GTKG]*[TRANS([AIC])*[GSKF]];
8!                 [FLXTMPA] :=(QDP)[GTKG]*[TRANS([AAIC])*[GSKF]];
8!                 [UFX1SUM] := (0.5) [UFX1HRT] + (0.5) [UFX1HIT];
8!                 [UFX1DIF] := (0.5) [UFX1HRT] - (0.5) [UFX1HIT];
8!                 [UFX2SUM] := (0.5) [UFX2HRT] + (0.5) [UFX2HIT];
8!                 [UFX2DIF] := (0.5) [UFX2HRT] - (0.5) [UFX2HIT];
8!                 [FLXF1SUM]:= [FLXTMPS]*[UFX1SUM];
8!                 [FLXF1DIF]:= [FLXTMPA]*[UFX1DIF];
8!                 [FLXF2SUM]:= [FLXTMPS]*[UFX2SUM];
8!                 [FLXF2DIF]:= [FLXTMPA]*[UFX2DIF];
8!$
8!                 [FLXF1R] := [FLXF1SUM]+[FLXF1DIF]  ;
8!                 [FLXF1I] := [FLXF1SUM]-[FLXF1DIF]  ;
8!                 [FLXF2R] := [FLXF2SUM]+[FLXF2DIF]  ;
8!                 [FLXF2I] := [FLXF2SUM]-[FLXF2DIF]  ;
8!              ENDIF;
7!           ELSE
7!              IF SYM =  1 THEN
8!                 [FLXTMP]   :=(QDP)[GTKG]*[TRANS([AIC])*[GSKF]];
8!              ENDIF;
7!              IF SYM = -1 THEN
8!                 [FLXTMP]   :=(QDP)[GTKG]*[TRANS([AAIC])*[GSKF]];
8!              ENDIF;
7!              IF SYM =  0 THEN
8!                 [FLXTMP]   :=(QDP)[GTKG]*[TRANS([ASAIC])*[GSKF]];
8!              ENDIF;
7!              [FLXFRC1] :=[FLXTMP]*[UFFLX1]  ;
7!              [FLXFRC2] :=[FLXTMP]*[UFFLX2]  ;
7!           ENDIF;
6!$
6!$          LOAD GROUP FLEXLOAD                                          $!
6!$                                                                       $!
6!           IF SYMTRN(BC) THEN
7!              CALL FLXLODLD ( BCID, SUB, TRIMTOC, TRIMDATA, [PAG],
7!                              [PAGI], [FLXF1R], [FLXF1I], [FLXF2R],
7!                              [FLXF2I], [MGG], [UFX1HRT], [UFX1HIT],
7!                              [UFX2HRT], [UFX2HIT], BGPDT(BC),
7!                              FLEXLOAD, SYMTRN(BC), ACSMTR(BC),
7!                              NEWITER );
7!           ELSE
7!              CALL FLXLODLD ( BCID, SUB, TRIMTOC, TRIMDATA, [PAG], ,
7!                              [FLXFRC1], , [FLXFRC2], , [MGG],
7!                              [UFFLX1], , [UFFLX2], , BGPDT(BC),
7!                              FLEXLOAD, SYMTRN(BC), ACSMTR(BC),
7!                              NEWITER );
7!           ENDIF;
```

```
6!$                                                                          $!
6!$              GENERATE TRIM PARAMETER BMST DATA                           $!
6!$                                                                          $!
6!               CALL PARMBMST ( TRIMTOC, FLEXLOAD, [PAG], [PAGI], [SAROLOAD],  !
6!                               STDYGEOM, SYMTRN(BC), ACSMTR(BC), BMSTDATA );  !
6!$                                                                          $!
6!$              GENERALIZED TRIM AND TRIM OPTIMIZATION                      $!
6!$                                                                          $!
6!               SCHITER := 0;                                                 !
6!               SCHCNVG := FALSE;                                             !
6!               WHILE NOT SCHCNVG DO                                          !
7!                  SCHITER := SCHITER + 1;                                    !
7!                  CALL SCHDULER ( BCID,SUB,TRIMDATA,TRIMRSLT,TRIMTOC,        !
7!                                  [AAR],[DELTA(SUB)],SCHITER,SCHCNVG );      !
7!                  CALL FLEXTRIM ( , BCID, SUB, SYM, QDP,                     !
7!                                  TRIMDATA, TRIMRSLT, TRIMTOC,               !
7!                                  [LHSA(BC,SUB)], [RHSA(BC,SUB)],            !
7!                                  [AAR], [DELTA(SUB)], [PRIGID], [R33] );    !
7!                  CALL FTRIMOPT ( , BCID, SUB, SYM, QDP,                     !
7!                                  TRIMDATA, TRIMRSLT, TRIMTOC,               !
7!                                  [LHSA(BC,SUB)], [RHSA(BC,SUB)],            !
7!                                  [AAR], [DELTA(SUB)], [PRIGID], [R33],      !
7!                                  BMSTDATA );                                !
7!               ENDDO;                                                        !
6!$                                                                          $!
6!$                 GENERATE TRIMMED BMST DATA                               $!
6!$                                                                          $!
6!                  TRMRIGD := FALSE;                                          !
6!                  CALL OFPBMST ( , BCID, SUB, QDP, [AAR], [DELTA(SUB)],      !
6!                                 TRIMDATA, TRIMTOC, BMSTDATA, TRMRIGD,       !
6!                                 OBMSTLOD );                                 !
6!$                                                                          $!
6!               [AAL]          := [D(BC)] * [AAR];                            !
6!               CALL ROWMERGE ( [AAA(SUB)], [AAR], [AAL], [PARLX(BC)] );      !
6!               [UAR]    := [K1112(BC,SUB)] * [AAR] + [PAR(BC,SUB)] *         !
6!                           [DELTA(SUB)];                                     !
6!               [UAL]    := [R1112(BC,SUB)] * [UAR] + [R1113(BC,SUB)] * [AAR] !
6!                           - [R11PAL(BC,SUB)] * [DELTA(SUB)];                !
6!               CALL ROWMERGE ( [UAA(SUB)], [UAR], [UAL], [PARLX(BC)] );      !
6!               IF NOMIT <> 0 [PAO(SUB)] := [POARO(BC,SUB)] * [DELTA(SUB)] ;  !
6!            ELSE                                                             !
6!$                                                                          $!
6!$              NO SUPPORT SET REDUCTION                                    $!
6!$                                                                          $!
6!$                                                                          $!
6!$              PROCESS STEADY AEROELASTIC DISCIPLINE                       $!
6!$                                                                          $!
6!               PRINT("LOG=('        >>>DISCIPLINE: STEADY AERO')");          !
6!            ENDIF;                                                           !
5!         ENDDO;                                                              !
4!      ENDIF;                                                                 !
3!$                                                                          $!
3!$     PERFORM ANY DYNAMIC ANALYSES -- NOTE THAT THESE ARE INDEPENDENT      $!
3!$     OF THE SUPPORT SET                                                   $!
3!$                                                                          $!
3!      IF BDYN <> 0 THEN                                                      !
4!         IF BFLUTR <> 0 THEN                                                 !
5!            PRINT("LOG=('        >>>DISCIPLINE: FLUTTER')");                 !
5!            SUBF := 0;                                                       !
5!            LOOP := TRUE;                                                    !
5!            WHILE LOOP DO                                                    !
6!               SUBF := SUBF + 1;                                             !
6!               CALL FLUTDRV ( BCID, SUBF, LOOP );                            !
6!               CALL FLUTQHHL ( , BCID, SUBF, ESIZE(BC), PSIZE(BC), [QKKL],   !
6!                               [UGTKA], [PHIA], USET(BC),                    !
6!                               [TMN(BC)], [GSUBO(BC)], NGDR, AECOMPU, GEOMUA, !
6!                               [PHIKH], [QHHLFL(BC,SUBF)], OAGRDDSP );       !
6!               CALL FLUTDMA ( , BCID, SUBF, ESIZE(BC), PSIZE(BC), BGPDT(BC), !
6!                              USET(BC), [MAA], [KAA], [TMN(BC)],[GSUBO(BC)], !
6!                              NGDR, LAMBDA, [PHIA], [MHHFL(BC,SUBF)],        !
6!                              [BHHFL(BC,SUBF)], [KHHFL(BC,SUBF)] );          !
6!               CALL FLUTTRAN (,BCID, SUBF, [QHHLFL(BC,SUBF)], LAMBDA, HSIZE(BC),!
6!                               ESIZE(BC), [MHHFL(BC,SUBF)], [BHHFL(BC,SUBF)], !
6!                               [KHHFL(BC,SUBF)], CLAMBDA );                  !
6!            ENDDO;                                                           !
5!         ENDIF;                                                              !
4!         IF BDRSP <> 0 THEN                                                  !
5!            IF BMTR <> 0 OR BDTR <> 0 THEN                                   !
6!               PRINT("LOG=('        >>>DISCIPLINE: TRANSIENT RESPONSE')");   !
6!            ENDIF;                                                           !
5!            IF BMFR <> 0 OR BDFR <> 0 THEN                                   !
6!               PRINT("LOG=('        >>>DISCIPLINE: FREQUENCY RESPONSE')");   !
6!            ENDIF;                                                           !
5!            CALL QHHLGEN (BCID, ESIZE(BC), [QKKL], [QKJL], [UGTKA], [PHIA],  !
5!                          [PHIKH], [QHHL], [QHJL]);                          !
5!            CALL DMA ( , BCID, ESIZE(BC), PSIZE(BC), BGPDT(BC), USET(BC), [MAA],!
5!                       [KAA], [TMN(BC)], [GSUBO(BC)], NGDR,                  !
5!                       LAMBDA, [PHIA], [MDD], [BDD], [KDDT], [KDDF],         !
5!                       [MHH], [BHH], [KHHT], [KHHF] );                       !
5!            CALL DYNLOAD ( , BCID, GSIZE, ESIZE(BC), PSIZE(BC), SMPLOD,      !
```

```
5!                         BGPDT(BC), USET(BC), [TMN(BC)], [GSUBO(BC)],
5!                         NGDR, [PHIA], [QHJL], [PDT], [PDF],
5!                         [PTGLOAD], [PTHLOAD], [PFGLOAD], [PFHLOAD] );
5!          CALL DYNRSP (BCID, ESIZE(BC), [MDD], [BDD], [KDDT], [KDDF],
5!                       [MHH], [BHH], [KHHT], [KHHF], [PDT], [PDF],
5!                       [QHHL], [UTRANA], [UFREQA], [UTRANI], [UFREQI],
5!                       [UTRANE], [UFREQE] );
5!          IF BMTR <> 0 [UTRANA] := [PHIA] * [UTRANI];
5!          IF BMFR <> 0 [UFREQA] := [PHIA] * [UFREQI];
5!       ENDIF;
4!    ENDIF;
3!    IF BBLAST <> 0 THEN
4!       PRINT("LOG=('         >>>DISCIPLINE: BLAST')");
4!       CALL BLASTFIT ( BCID, [QJJL], [MATTR], [MATSS], BQDP, [BFRC],
4!                       [DWNWSH],  HSIZE(BC), [ID2], [MPART], [UGTKA],
4!                       [BLGTJA], [BLSTJA] );
4!       CALL COLPART ( [PHIA], , [PHIB], [MPART] );
4!       CALL ROWMERGE ( [PHIR], [ID2], [D(BC)], [PARLX(BC)] );
4!       CALL COLMERGE ( [PHIB], [PHIR], [PHIB], [MPART] );
4!       [GENM]    := TRANS( [PHIB] ) * [ [MAA] * [PHIB] ];
4!       [GENK]    := TRANS( [PHIB] ) * [ [KAA] * [PHIB] ];
4!       [DTSLP]   := TRANS ( [BLSTJA] ) * [PHIB];
4!       [FTF]     := TRANS ( [PHIB] ) * [BLGTJA];
4!       [GENF]    := (BQDP) [FTF] * [BFRC];
4!       [GENFA]   := (BQDP) [FTF] * [MATSS];
4!       [GENQ]    := [GENFA] * [DTSLP];
4!       [GENQL]   := (BQDP) [FTF] * [MATTR];
4!       CALL PARTN ( [GENQ],[QRR] , , [QRE], [QEE], [MPART] );
4!       CALL PARTN ( [GENK], , , , [KEE], [MPART] );
4!       [KEQE]    := [QEE] + [KEE];
4!       CALL DECOMP ( [KEQE], [LKQ], [UKQ] );
4!       CALL ROWPART ( [GENF], [GFR], [GFE], [MPART] );
4!       CALL GFBS ( [LKQ], [UKQ], [GFE], [BTEM] );
4!       [DELM]    := -[QRE] * [BTEM] + [GFR];
4!       CALL BLASTRIM ( BCID, [DELM], [MRR(BC)], [URDB], [DELB] );
4!       [ELAS]    := [BTEM] * [DELB];
4!       [SLPMOD] := TRANS ( [BLSTJA] ) * [PHIB];
4!       CALL BLASTDRV ( BCID, [GENM], [GENK], [GENFA], [GENQL], [DELB],
4!                       [URDB], [DWNWSH], [SLPMOD], [ELAS], [UBLASTI] );
4!    ENDIF;
3!$
3!$   BEGIN THE DATA RECOVERY OPERATIONS
3!$
3!    IF NBNDCOND > 1 CALL NULLMAT ( [UF], [AF], [PHIF], [UFX], [AFX] );
3!    IF NBNDCOND > 1 CALL NULLMAT ( [UAF], [UAFI], [AAF], [AAFI]);
3!    IF NGDR <> 0 THEN
4!$
4!$      DATA RECOVERY WITH GDR
4!$      APPEND THE GDR-GENERATED DOFS TO THE F-SET
4!$
4!       PRINT("LOG=('         DYNAMIC REDUCTION RECOVERY')");
4!       IF BLOAD <> 0 THEN
5!          [UFGDR] := [GSUBO(BC)] * [UA];
5!          CALL ROWPART ( [UA], [UJK], , [PAJK] );
5!          CALL ROWMERGE ( [UFX], [UJK], [UFGDR], [PFJK] );
5!          IF NRSET <> 0 THEN
6!             [AFGDR] := [GSUBO(BC)] * [AA];
6!             CALL ROWPART ( [AA], [UJK], , [PAJK] );
6!             CALL ROWMERGE ( [AFX], [UJK], [AFGDR], [PFJK] );
6!          ENDIF;
5!       ENDIF;
4!       IF BSAERO <> 0 THEN
5!          FOR S = 1 TO SUB DO
6!             [UFGDR] := [GSUBO(BC)] * [UAA(S)];
6!             CALL ROWPART ( [UAA(S)], [UJK], , [PAJK] );
6!             CALL ROWMERGE ( [UAFTMP], [UJK], [UFGDR], [PFJK] );
6!$
6!$            MERGE THE CURRENT SUBCASE DEPENDENT RESULTS INTO A SINGLE
6!$            MATRIX OF RESPONSE QUANTITIES FOR FURTHER RECOVERY
6!$
6!             CALL SAEROMRG ( BCID, S, TRIMDATA, [UAFX], [UAFTMP] );
6!             IF NRSET <> 0 THEN
7!                [AFGDR] := [GSUBO(BC)] * [AAA(S)];
7!                CALL ROWPART ( [AAA(S)], [UJK], , [PAJK] );
7!                CALL ROWMERGE ( [AAFTMP], [UJK], [AFGDR], [PFJK] );
7!                CALL SAEROMRG ( BCID, S, TRIMDATA, [AAFX], [AAFTMP] );
7!             ENDIF;
6!          ENDDO;
5!       ENDIF;
4!       IF BMODES <> 0 THEN
5!          [UFGDR]   := [GSUBO(BC)] * [PHIA];
5!          CALL ROWPART ( [PHIA], [UJK], , [PAJK] );
5!          CALL ROWMERGE ( [PHIF], [UJK], [UFGDR], [PFJK] );
5!       ENDIF;
4!       IF BDTR   <> 0 OR BMTR <> 0 THEN
5!          [UFGDR] := [GSUBO(BC)] * [UTRANA];
5!          CALL ROWPART ( [UTRANA], [UJK], , [PAJK] );
5!          CALL ROWMERGE ( [UTRANF], [UJK], [UFGDR], [PFJK] );
5!       ENDIF;
4!       IF BDFR   <> 0 OR BMFR <> 0 THEN
```

```
5!              [UFGDR] := [GSUBO(BC)] * [UFREQA];                                   !
5!              CALL ROWPART ( [UFREQA], [UJK], , [PAJK] );                          !
5!              CALL ROWMERGE ( [UFREQF], [UJK], [UFGDR], [PFJK] );                  !
5!           ENDIF;                                                                  !
4!        ELSE                                                                       !
4!           IF NOMIT <> 0 THEN                                                      !
5!$                                                                                 $!
5!$             DATA RECOVERY WITH STATIC CONDENSATION                              $!
5!$                                                                                 $!
5!              PRINT("LOG=('          STATIC CONDENSATION RECOVERY')");             !
5!              IF BLOAD <> 0 THEN                                                   !
6!                 CALL RECOVA ( [UA], [PO], [GSUBO(BC)], NRSET, [AA],               !
6!                               [IFM(BC)], , [KOOINV(BC)],,[PFOAX(BC)], [UFX] );    !
6!                 IF NRSET <> 0 CALL RECOVA ( [AA], , [GSUBO(BC)],,,,,,,            !
7!                                             [PFOAX(BC)], [AFX] );                 !
6!              ENDIF;                                                               !
5!              IF BSAERO <> 0 THEN                                                  !
6!                 FOR S = 1 TO SUB DO                                               !
7!                    CALL RECOVA ( [UAA(S)], [PAO(S)], [GASUBO(BC,S)],              !
7!                                  NRSET, [AAA(S)], [IFMA(BC,S)], BSAERO,           !
7!                                  [KOOL(BC,S)], [KOOU(BC,S)],                      !
7!                                  [PFOAX(BC)], [UAFTMP] );                         !
7!$                                                                                 $!
7!$                   MERGE THE CURRENT SUBCASE DEPENDENT RESULTS INTO A SINGLE     $!
7!$                   MATRIX OF RESPONSE QUANTITIES FOR FURTHER RECOVERY            $!
7!$                                                                                 $!
7!                    CALL SAEROMRG ( BCID, S, TRIMDATA, [UAFX], [UAFTMP] );         !
7!                    IF NRSET <> 0 THEN                                             !
8!                       CALL RECOVA ( [AAA(S)],,[GASUBO(BC,S)],,,,,,,               !
8!                                     [PFOAX(BC)], [AAFTMP]);                       !
8!                       CALL SAEROMRG ( BCID, S, TRIMDATA, [AAFX], [AAFTMP] );      !
8!                    ENDIF;                                                         !
7!                 ENDDO;                                                            !
6!              ENDIF;                                                               !
5!              IF BMODES <> 0 THEN                                                  !
6!                 [PHIO] := [GSUBO(BC)] * [PHIA];                                   !
6!                 CALL ROWMERGE ( [PHIF], [PHIO], [PHIA], [PFOAX(BC)] );            !
6!              ENDIF;                                                               !
5!              IF BDTR <> 0 OR BMTR <> 0 THEN                                       !
6!                 CALL RECOVA ( [UTRANA], , [GSUBO(BC)],,,,,,,                      !
6!                                             [PFOAX(BC)], [UTRANF] );              !
6!              ENDIF;                                                               !
5!              IF BDFR <> 0 OR BMFR <> 0 THEN                                       !
6!                 CALL RECOVA ( [UFREQA], , [GSUBO(BC)],,,,,,,                      !
6!                                             [PFOAX(BC)], [UFREQF] );              !
6!              ENDIF;                                                               !
5!           ELSE                                                                    !
5!$                                                                                 $!
5!$             DATA RECOVERY WITHOUT F-SET REDUCTION                               $!
5!$                                                                                 $!
5!              IF BLOAD <> 0 THEN                                                   !
6!                 [UFX] := [UA];                                                    !
6!                 IF NRSET <> 0 [AFX] := [AA];                                      !
6!              ENDIF;                                                               !
5!              IF BSAERO <> 0 THEN                                                  !
6!                 FOR S = 1 TO SUB DO                                               !
7!$                                                                                 $!
7!$                   MERGE THE CURRENT SUBCASE DEPENDENT RESULTS INTO A SINGLE     $!
7!$                   MATRIX OF RESPONSE QUANTITIES FOR FURTHER RECOVERY            $!
7!$                                                                                 $!
7!                    CALL SAEROMRG ( BCID, S, TRIMDATA, [UAFX], [UAA(S)] );         !
7!                    IF NRSET <> 0 CALL SAEROMRG(BCID,S,TRIMDATA,[AAFX],[AAA(S)]);  !
7!                 ENDDO;                                                            !
6!              ENDIF;                                                               !
5!              IF BMODES <> 0 [PHIF]    := [PHIA];                                  !
5!              IF BDTR    <> 0 OR BMTR <> 0 [UTRANF] := [UTRANA];                   !
5!              IF BDFR    <> 0 OR BMFR <> 0 [UFREQF] := [UFREQA];                   !
5!           ENDIF;                                                                  !
4!        ENDIF;                                                                     !
3!$                                                                                 $!
3!        IF SYMTRN(BC) THEN                                                         !
4!           IF BLOAD <> 0 THEN                                                      !
5!              [UF] := [HFREAL(BC)] * [UFX];                                        !
5!              IF NRSET <> 0  [AF] := [HFREAL(BC)] * [AFX];                         !
5!           ENDIF;                                                                  !
4!           IF BSAERO <> 0 THEN                                                     !
5!              [UAF] := [HFREAL(BC)] * [UAFX];                                      !
5!              [UAFI] := [HFIMAG(BC)] * [UAFX];                                     !
5!              IF NRSET <> 0  [AAF] := [HFREAL(BC)] * [AAFX];                       !
5!              IF NRSET <> 0  [AAFI] := [HFIMAG(BC)] * [AAFX];                      !
5!           ENDIF;                                                                  !
4!        ELSE                                                                       !
4!           IF BLOAD <> 0 THEN                                                      !
5!              [UF] := [UFX];                                                       !
5!              IF NRSET <> 0  [AF] := [AFX];                                        !
5!           ENDIF;                                                                  !
4!           IF BSAERO <> 0 THEN                                                     !
5!              [UAF] := [UAFX];                                                     !
5!              IF NRSET <> 0  [AAF] := [AAFX];                                      !
5!           ENDIF;                                                                  !
```

```
4!      ENDIF;                                                              !
3!$                                                                        $!
3!      IF NBNDCOND > 1 CALL NULLMAT ( [UN], [AN], [PHIN] );                !
3!      IF NSPC <> 0 THEN                                                   !
4!$                                                                        $!
4!$        DATA RECOVERY WITH SPC-REDUCTION                                $!
4!$                                                                        $!
4!         PRINT("LOG=('          SPC RECOVERY')");                         !
4!         IF BLOAD <> 0 THEN                                               !
5!            CALL YSMERGE ( [UN], [YS(BC)], [UF], [PNSF(BC)] );            !
5!            CALL OFPSPCF ( 0, BCID, 1, 1, GSIZE, ESIZE(BC), NGDR,         !
5!                           [KFS], [KSS], [UF], [YS(BC)], [PS],            !
5!                           [PNSF(BC)], [PGMN(BC)], [PFJK], , , ,          !
5!                           BGPDT(BC), OGRIDLOD );                         !
5!            IF NRSET <> 0 CALL YSMERGE ( [AN], , [AF], [PNSF(BC)] );      !
5!         ENDIF;                                                           !
4!         IF BSAERO <> 0 THEN                                              !
5!            CALL YSMERGE ( [UAN], [YS(BC)], [UAF], [PNSF(BC)] );          !
5!            IF NRSET <> 0 CALL YSMERGE ( [AAN], , [AAF], [PNSF(BC)] );    !
5!            IF SYMTRN(BC) THEN                                            !
6!               CALL YSMERGE ( [UANI], [YS(BC)], [UAFI], [PNSF(BC)] );     !
6!               IF NRSET <> 0 CALL YSMERGE ( [AANI], , [AAFI], [PNSF(BC)] ); !
6!            ENDIF;                                                        !
5!         ENDIF;                                                           !
4!         IF BMODES <> 0 THEN                                              !
5!            CALL YSMERGE ( [PHIN], [YS(BC)], [PHIF],                      !
5!                                      [PNSF(BC)] );                       !
5!            IF DMODES <> 0   CALL OFPSPCF ( 0, BCID, 2, 1, GSIZE,         !
6!                                      ESIZE(BC), NGDR,                    !
6!                                      [KFS], , [PHIF], , ,                !
6!                                      [PNSF(BC)], [PGMN(BC)], [PFJK],     !
6!                                      , , , BGPDT(BC), OGRIDLOD );        !
5!         ENDIF;                                                           !
4!         IF BDTR    <> 0 OR BMTR <> 0                                     !
5!                     CALL YSMERGE ( [UTRANN], [YS(BC)], [UTRANF],         !
5!                                    [PNSF(BC)], BDTR );                   !
4!         IF BDFR    <> 0 OR BMFR <> 0                                     !
5!                     CALL YSMERGE ( [UFREQN], [YS(BC)], [UFREQF],         !
5!                                    [PNSF(BC)], BDFR );                   !
4!         IF BFLUTR <> 0                                                   !
5!            CALL OFPSPCF ( 0, BCID, 4, 2, GSIZE, ESIZE(BC), NGDR, [KFS], , !
5!                           [PHIF], , , [PNSF(BC)], [PGMN(BC)], [PFJK],    !
5!                           , , , BGPDT(BC), OGRIDLOD );                   !
4!         IF BBLAST <> 0    THEN                                           !
5!            [UBLASTF] := [PHIF]*[UBLASTI];                                !
5!            CALL OFPSPCF ( 0, BCID, 8, 1, GSIZE, ESIZE(BC), NGDR,         !
5!                           [KFS], , [UBLASTF], , , [PNSF(BC)], [PGMN(BC)], !
5!                           [PFJK], , , , BGPDT(BC), OGRIDLOD );           !
5!         ENDIF;                                                           !
4!      ELSE                                                                !
4!$                                                                        $!
4!$        DATA RECOVERY WITHOUT SPC-REDUCTION                             $!
4!$                                                                        $!
4!         IF BLOAD <> 0 THEN                                               !
5!            [UN] := [UF];                                                 !
5!            IF NRSET <> 0 [AN] := [AF];                                   !
5!         ENDIF;                                                           !
4!         IF BSAERO <> 0 THEN                                              !
5!            [UAN] := [UAF];                                               !
5!            IF NRSET <> 0 [AAN] := [AAF];                                 !
5!            IF SYMTRN(BC) THEN                                            !
6!               [UANI] := [UAFI];                                          !
6!               IF NRSET <> 0 [AANI] := [AAFI];                            !
6!            ENDIF;                                                        !
5!         ENDIF;                                                           !
4!         IF BMODES <> 0 [PHIN] := [PHIF];                                 !
4!         IF BDTR    <> 0 OR BMTR <> 0 [UTRANN] := [UTRANA];               !
4!         IF BDFR    <> 0 OR BMFR <> 0 [UFREQN] := [UFREQA];               !
4!      ENDIF;                                                              !
3!$                                                                        $!
3!      IF NBNDCOND > 1 CALL NULLMAT ( [UG(BC)], [AG(BC)], [UAG(BC)], [AAG(BC)], !
4!                               [PHIG(BC)], [UAGI(BC)], [AAGI(BC)] );      !
3!      IF NMPC <> 0 THEN                                                   !
4!$                                                                        $!
4!$        DATA RECOVERY WITH MPC-REDUCTION                                $!
4!$                                                                        $!
4!         PRINT("LOG=('          MPC RECOVERY')");                         !
4!         IF BLOAD <> 0 THEN                                               !
5!            [UM] := [TMN(BC)] * [UN];                                     !
5!            CALL ROWMERGE ( [UG(BC)], [UM], [UN], [PGMN(BC)] );           !
5!            IF NRSET <> 0 THEN                                            !
6!               [UM] := [TMN(BC)] * [AN];                                  !
6!               CALL ROWMERGE ( [AG(BC)], [UM], [AN], [PGMN(BC)] );        !
6!            ENDIF;                                                        !
5!         ENDIF;                                                           !
4!         IF BSAERO <> 0 THEN                                              !
5!            [UM] := [TMN(BC)] * [UAN];                                    !
5!            CALL ROWMERGE ( [UAG(BC)], [UM], [UAN], [PGMN(BC)] );         !
5!            IF NRSET <> 0 THEN                                            !
6!               [UM] := [TMN(BC)] * [AAN];                                 !
```

```
6!               CALL ROWMERGE ( [AAG(BC)], [UM], [AAN], [PGMN(BC)] );          !
6!            ENDIF;                                                            !
5!            IF SYMTRN(BC) THEN                                                !
6!               [UM] := [TMN(BC)] * [UANI];                                    !
6!               CALL ROWMERGE ( [UAGI(BC)], [UM], [UANI], [PGMN(BC)] );        !
6!               IF NRSET <> 0 THEN                                             !
7!                  [UM] := [TMN(BC)] * [AANI];                                 !
7!                  CALL ROWMERGE ( [AAGI(BC)], [UM], [AANI], [PGMN(BC)] );     !
7!               ENDIF;                                                         !
6!            ENDIF;                                                            !
5!         ENDIF;                                                               !
4!         IF BMODES <> 0 THEN                                                  !
5!            [UM] := [TMN(BC)] * [PHIN];                                       !
5!            CALL ROWMERGE ( [PHIG(BC)], [UM], [PHIN], [PGMN(BC)] );           !
5!         ENDIF;                                                               !
4!         IF BDTR <> 0 OR BMTR <> 0 THEN                                       !
5!            [UM] := [TMN(BC)] * [UTRANN];                                     !
5!            CALL ROWMERGE ( [UTRANG], [UM], [UTRANN], [PGMN(BC)] );           !
5!         ENDIF;                                                               !
4!         IF BDFR <> 0 OR BMFR <> 0 THEN                                       !
5!            [UM] := [TMN(BC)] * [UFREQN];                                     !
5!            CALL ROWMERGE ( [UFREQG], [UM], [UFREQN], [PGMN(BC)] );           !
5!         ENDIF;                                                               !
4!      ELSE                                                                    !
4!$                                                                            $!
4!$        DATA RECOVERY WITHOUT MPC-REDUCTION                                 $!
4!$                                                                            $!
4!         IF BLOAD <> 0 THEN                                                   !
5!            [UG(BC)] := [UN];                                                 !
5!            IF NRSET <> 0 [AG(BC)] := [AN];                                   !
5!         ENDIF;                                                               !
4!         IF BSAERO <> 0 THEN                                                  !
5!            [UAG(BC)] := [UAN];                                               !
5!            IF NRSET <> 0 [AAG(BC)] := [AAN];                                 !
5!            IF SYMTRN(BC) THEN                                                !
6!               [UAGI(BC)] := [UANI];                                          !
6!               IF NRSET <> 0 [AAGI(BC)] := [AANI];                            !
6!            ENDIF;                                                            !
5!         ENDIF;                                                               !
4!         IF BMODES <> 0 [PHIG(BC)] := [PHIN];                                 !
4!         IF BDTR   <> 0 OR BMTR <> 0 [UTRANG] := [UTRANN];                    !
4!         IF BDFR   <> 0 OR BMFR <> 0 [UFREQG] := [UFREQN];                    !
4!      ENDIF;                                                                  !
3!$                                                                            $!
3!$     RECOVER PHYSICAL BLAST DISCIPLINE DISPLACEMENTS                        $!
3!$                                                                            $!
3!      IF BBLAST <> 0 [UBLASTG] := [PHIG(BC)] * [UBLASTI];                     !
3!$                                                                            $!
3!$     HANDLE OUTPUT REQUESTS                                                 $!
3!$                                                                            $!
3!      PRINT("LOG=('        OUTPUT PROCESSING')");                             !
3!      IF BSAERO <> 0 THEN                                                     !
4!$                                                                            $!
4!$        RECOVER STATIC AEROELASTIC LOADS DATA                               $!
4!$                                                                            $!
4!         LOOP := TRUE;                                                        !
4!         SUB  := 0;                                                           !
4!         WHILE LOOP DO                                                        !
5!            SUB := SUB + 1;                                                   !
5!            CALL SAERODRV (BCID, SUB, LOOP, MINDEX, SYM, MACH, QDP, TRIMDATA, !
5!                           TRIMRSLT, METHOD );                                !
5!$                                                                            $!
5!$           CALL THE TRIMMED LOADS COMPUTATION WITH PROPER MATRICES          $!
5!$                                                                            $!
5!            IF SYM = 1  THEN                                                  !
6!               CALL OFPALOAD ( , BCID, MINDEX, SUB, GSIZE, TRIMDATA, BGPDT(BC), !
6!                               [GPTKG], [GTKG], [GSTKG], QDP, [SAROLOAD],     !
6!                               [DELTA(SUB)], [AIC],                           !
6!                               [UAG(BC)], [MGG], [AAG(BC)], [KFS],            !
6!                               [KSS], [UAF], [YS(BC)], [PNSF(BC)],            !
6!                               [PGMN(BC)], [PFJK], NGDR, USET(BC),            !
6!                               OGRIDLOD );                                    !
6!            ELSE                                                              !
6!               IF SYM = -1 THEN                                               !
7!                  CALL OFPALOAD(, BCID, MINDEX, SUB, GSIZE, TRIMDATA, BGPDT(BC),!
7!                               [GPTKG], [GTKG], [GSTKG], QDP, [SAROLOAD],     !
7!                               [DELTA(SUB)], [AAIC],                          !
7!                               [UAG(BC)], [MGG], [AAG(BC)], [KFS],            !
7!                               [KSS], [UAF], [YS(BC)], [PNSF(BC)],            !
7!                               [PGMN(BC)], [PFJK], NGDR, USET(BC),            !
7!                               OGRIDLOD );                                    !
7!               ELSE                                                           !
7!                  IF SYM = 0 THEN                                             !
8!                     CALL OFPALOAD ( , BCID, MINDEX, SUB, GSIZE,              !
8!                               TRIMDATA, BGPDT(BC),                           !
8!                               [GPTKG], [GTKG], [GSTKG], QDP, [SAROLOAD],     !
8!                               [DELTA(SUB)], [ASAIC],                         !
8!                               [UAG(BC)], [MGG], [AAG(BC)], [KFS],            !
8!                               [KSS], [UAF], [YS(BC)], [PNSF(BC)],            !
8!                               [PGMN(BC)], [PFJK], NGDR, USET(BC),            !
```

```
8!                             OGRIDLOD );
8!                   ENDIF;
7!                ENDIF;
6!             ENDIF;
5!$
5!$            CALL TO COMPUTE THE TRIMMED LOADS/DISPLACEMENTS ON THE
5!$            AERODYNAMIC MODEL
5!$
5!             IF SYM = 1  THEN
6!                CALL OFPAEROM ( , BCID, MINDEX, SUB, GSIZE, SAGEOM,
6!                                TRIMDATA,
6!                                [GTKG], [GSTKG], QDP, [SAROLOAD],
6!                                [DELTA(SUB)], [AIC],
6!                                [UAG(BC)], OAGRDLOD, OAGRDDSP );
6!             ELSE
6!                IF SYM = -1 THEN
7!                   CALL OFPAEROM ( , BCID, MINDEX, SUB, GSIZE, SAGEOM,
7!                                   TRIMDATA,
7!                                   [GTKG], [GSTKG], QDP, [SAROLOAD],
7!                                   [DELTA(SUB)], [AAIC],
7!                                   [UAG(BC)], OAGRDLOD, OAGRDDSP );
7!                ELSE
7!                   IF SYM = 0 THEN
8!                      CALL OFPAEROM ( , BCID, MINDEX, SUB, GSIZE, SAGEOM,
8!                                   TRIMDATA,
8!                                   [GTKG], [GSTKG], QDP, [SAROLOAD],
8!                                   [DELTA(SUB)], [ASAIC],
8!                                   [UAG(BC)], OAGRDLOD, OAGRDDSP );
8!                   ENDIF;
7!                ENDIF;
6!             ENDIF;
5!          ENDDO;
4!       ENDIF;
3!       IF BDRSP <> 0  THEN
4!          CALL OFPDLOAD ( , BCID, BGPDT(BC), PSIZE(BC), ESIZE(BC), [PHIG(BC)],
4!                          [PTGLOAD], [PTHLOAD], [PFGLOAD], [PFHLOAD], OGRIDLOD );
4!          IF BDTR   <> 0 OR BMTR <> 0
5!                         CALL OFPSPCF ( 0, BCID, 5, 1, GSIZE, ESIZE(BC),
5!                                        NGDR, [KFS], , [UTRANF], , ,
5!                                        [PNSF(BC)], [PGMN(BC)], [PFJK],
5!                                        [PHIG(BC)], [PTGLOAD], [PTHLOAD],
5!                                        BGPDT(BC), OGRIDLOD );
4!          IF BDFR   <> 0 OR BMFR <> 0
5!                         CALL OFPSPCF ( 0, BCID, 6, 2, GSIZE, ESIZE(BC),
5!                                        NGDR, [KFS], , [UFREQF], , ,
5!                                        [PNSF(BC)], [PGMN(BC)], [PFJK],
5!                                        [PHIG(BC)], [PFGLOAD], [PFHLOAD],
5!                                        BGPDT(BC), OGRIDLOD );
4!       ENDIF;
3!       CALL OFPLOAD ( BCID, , GSIZE, BGPDT(BC), PSIZE(BC),
3!                      [PG] );
3!       CALL OFPDISP( BCID, , GSIZE, BGPDT(BC), ESIZE(BC), PSIZE(BC),
3!                     OGRIDDSP, [UG(BC)], [AG(BC)], [UAG(BC)], [AAG(BC)],
3!                     [UBLASTG], , [UTRANG], [UTRANE], [UFREQG], [UFREQE],
3!                     LAMBDA, [PHIG(BC)], , SYMTRN(BC), [UAGI(BC)], [AAGI(BC)] );
3!       CALL EDR ( BCID, , NDV, GSIZE, EOSUMMRY, EODISC,
3!                  GLBDES, LOCLVAR, [PTRANS],
3!                  [UG(BC)], [UAG(BC)], , [UTRANG], [UFREQG], [PHIG(BC)],
3!                  , SYMTRN(BC), [UAGI(BC)] );
3!       CALL OFPEDR ( BCID, HSIZE(BC), , SYMTRN(BC) );
3!$
3!$..................................................................
3!$
3!    ENDDO;
2!ENDIF;
1!END;
```

THIS PAGE INTENTIONALLY LEFT BLANK.

3. THE SOLUTION CONTROL PACKET

The following pages describe the new and modified Solution Control commands developed under this effort. These are summarized in the following table:

Command	Description of Change/Modification
ARCHIVE	New command
ASSEMBLE	New command
FTRIM	New command
IMPORT	New command
OVERLAY	New command
SAERO	Heavily modified command.

The new commands enable a new ASTROS capability called MODEL ASSEMBLY. This feature permits the use multiple aerodynamic databases perhaps created from aerodynamic methods alternative from the native ASTROS methods. The data from a variety of sources can be combined to create aerodynamic or aeroelastic models for ASTROS use. Aerodynamic and Aeroelastic models are defined at the subcase level, therefore, a new hierarchical level was inserted in solution control between the BOUNDARY and DISCIPLINE levels.

```
SOLUTION
    OPTIMIZE/ANALYZE
        BOUNDARY
            MODEL_ASSEMBLY_COMMANDS
                DISCIPLINE MODEL=model_name
    END
```

Notice that each discipline level command requires the specification of only a single model_name rather than a collection of model components. This removes redundant specification of model assembly commands for disciplines which use the same model. However, redundant model assembly commands are still required for the existing ASTROS paradigm of one model and multiple symmetric and antisymmetric boundary conditions (and even the case of both OPTIMIZE and ANALYZE sub-packets).

Four basic commands provide sufficient generality to assemble aeroelastic models from a combination of existing and archived data groups.

1. IMPORT
2. ARCHIVE
3. OVERLAY
4. ASSEMBLE

Aerodynamic and aeroelastic data are segregated into logical groupings as shown in Table 3-1. The first two group types (QUADPAN and USSAERO input packets) are inputs to the two available ASTROS aerodynamic methods of the SAERO discipline. The next three group types (STDYGEOM, RIGDALOD, and AIC) are either outputs of the available ASTROS aerodynamic methods (run-time or archived from previous runs) or *alternate* data created from some external method. In either case, the aerodynamic model is assembled from groups of aerodynamic data which exist on either the run-time database and/or alternate database(s) of archived data. The RIGDSLOD group is defined by the user through the new SLPARM and current static load bulk data cards. It may be archived from previous runs in similar function as the STDYGEOM, RIGDALOD, and AIC groups. The subsequent aeroelastic discipline, FTRIM, requires a combination of the assembled aerodynamic model and the SPLINES group and computes as output the FLEXLOAD group.

Table 3-1 ASTROS Aeroelastic Solution is Assembled From Aerodynamic and Aeroelastic Data Groups.

Group Type	Contents
QUADPAN Input Packet	• geometry input • control surface definition
USSAERO Input Packet,	• geometry input • control surface definition
STDYGEOM	• aerodynamic model geometry
RIGDALOD	• trim parameter rigid actual aerodynamic pressure vectors (not increment)
AIC	• symmetric AIC matrix • antisymmetric AIC matrix • asymmetric AIC matrix
SPLINE	• rigid load spline • slope spline • aeroelastic load increment spline
RIGDSLOD	• user defined rigid structural load (e.g. thrust load, distributed actuator load, ..)
FLEXLOAD	• trim parameter flexible load and deflection increment vectors

For ASTROS discipline purposes, these groups are collected into a master group called a MODEL. For instance, ASTROS runtime FTRIM discipline subcases have associated with it a unique MODEL group which specifies the set of entities associated with the MODEL. However, members of a particular MODEL may also be members of other MODELs (i.e., the same STDYGEOM group may be used by more than one group). Two formalized MODEL groups have been established as depicted in Figure 3-1. A unique SAMODEL will be established for each SAERO subcase while a unique SAEMODEL will be established for each FTRIM subcase.

The steady aerodynamic model (SAMODEL) includes attributes traditionally thought of as aerodynamic with the exception of the RIGDSLOD group. The STDYGEOM, RIGDALOD, and RIGDSLOD groups will be discussed in the following paragraphs. The AIC group contains information on the aerodynamic influence coefficient matrices. A model may contain symmetric, antisymmetric and asymmetric AIC matrices, and it may contain matrices for multiple Mach numbers. Therefore in use, a model may appear in multiple boundary conditions and subcases once it is either imported or created at runtime.

The STDYGEOM group (illustrated in Figure 3-2) consists of the traditional ASTROS relations that define aerodynamic models. The addresses of these relations are stored in this group. By grouping the data, a single model geometry may be used with any RIGDALOD group to integrate rigid pressure data from many sources and at many Mach numbers. Future ASTROS enhancements could include using this geometry for both steady and unsteady aerodynamic analysis. Current restrictions however require separate definitions for steady versus unsteady discretizations.

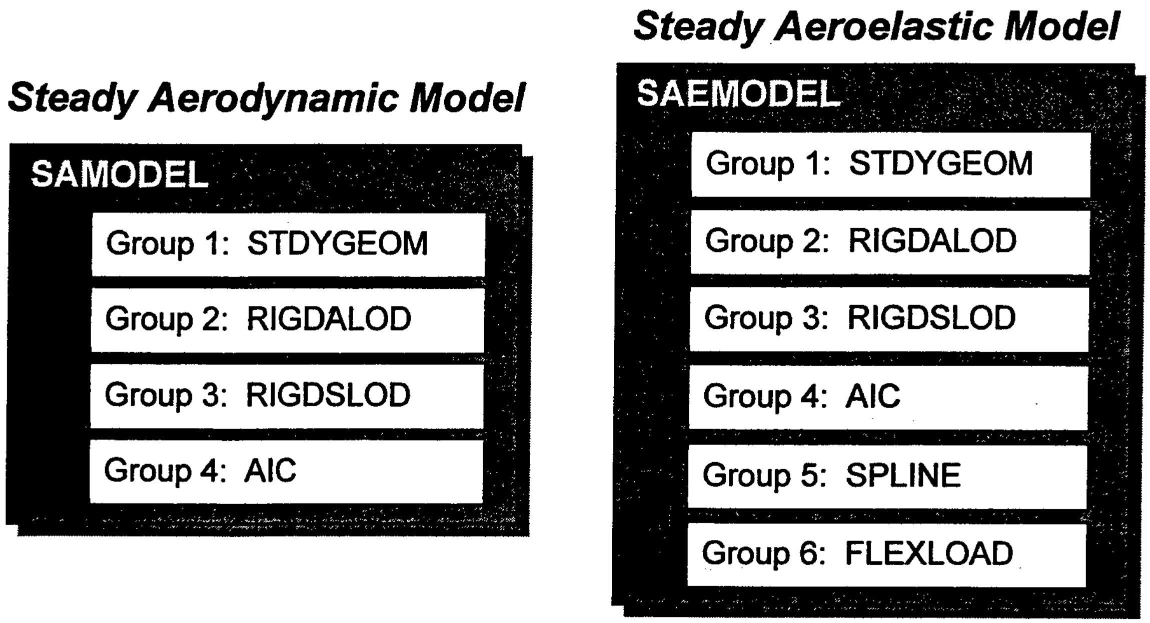


Figure 3-1 Steady Aerodynamic and Steady Aeroelastic Model Groups

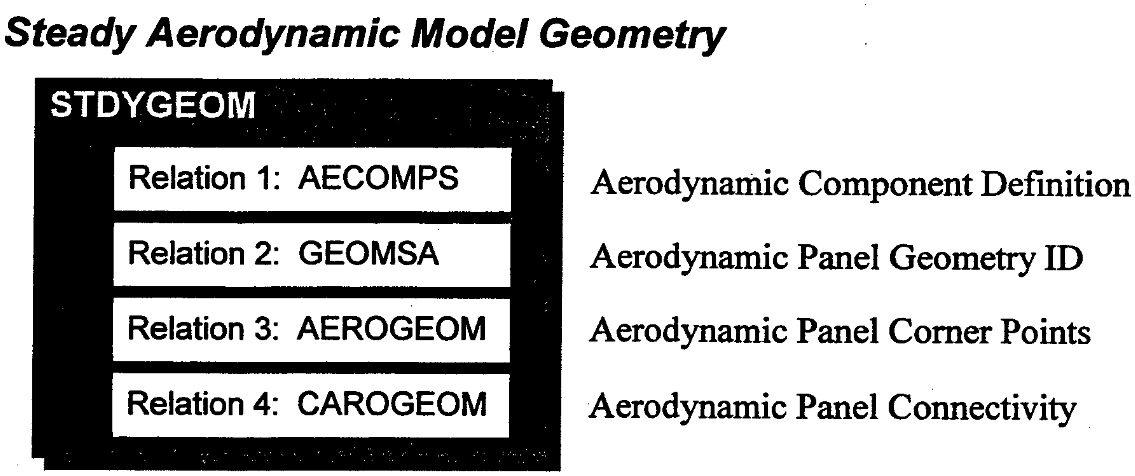


Figure 3-2 STDYGEOM - Model Geometry & Connectivity

The new SAERO and FTRIM solution control commands enable the user to execute aerodynamic and aeroelastic analyses that can take advantage of this formalized set of data groups (refer to Table 3-1). Aerodynamic and aeroelastic models are either assembled or modified prior to these basic aerodynamic/aeroelastic discipline level commands. Models can be comprised of entirely run-time data, archived data, or some combination thereof. For instance, a trim analysis scenario might include development of aerodynamic geometry and AIC matrices using the available ASTROS aerodynamic method, QUADPAN, and inclusion of archived CFD derived rigid pressure vectors and perhaps even wind tunnel derived pressure vectors. Figure 3-3 illustrates the final pressure vector that would be created for aeroelastic analysis. In this case, the user specifies 1) execution of the SAERO discipline to generate STDYGEOM, RIGDALOD, and AIC output groups from the QUADPAN input packet, 2) assembly of the modified aerodynamic model from run-time QUADPAN groups (STDYGEOM and AIC) and archived CFD group (RIGDALOD) and Wind Tunnel group (RIGDALOD), and 3) execution of subsequent aeroelastic disciplines.

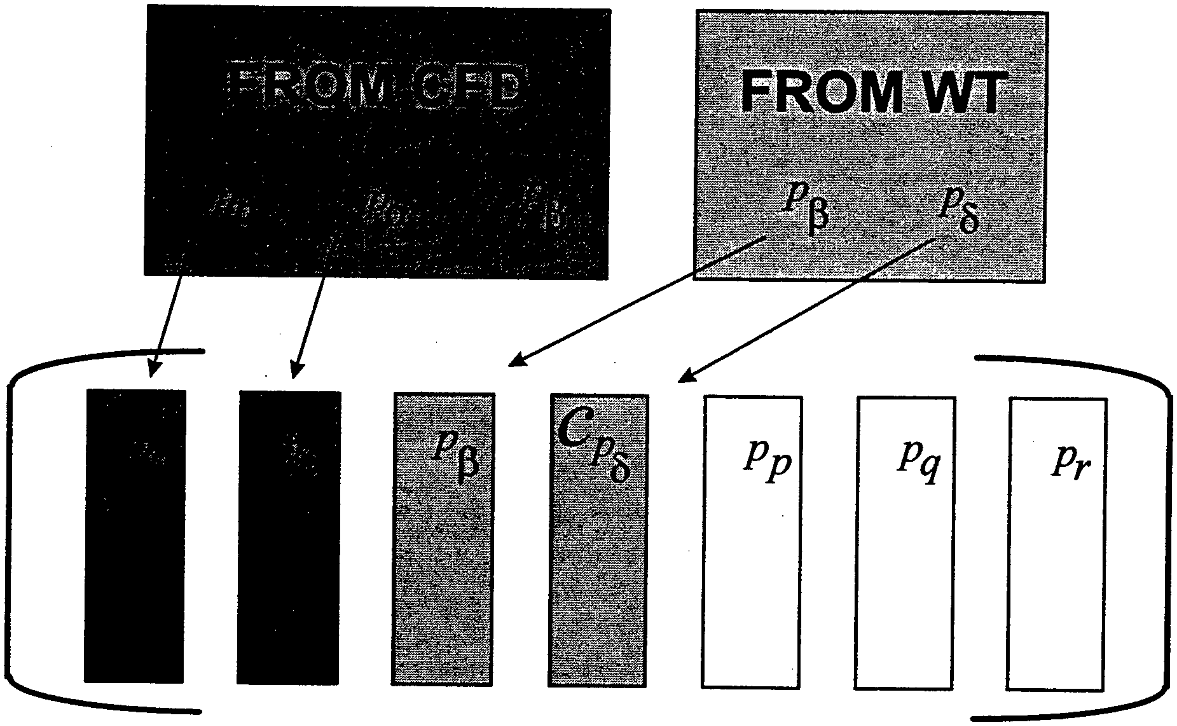


Figure 3-3 Overlay Base Aerodynamics, CFD,Wind Tunnel As Best Aerodynamics Using Type RIGDALOD

The RIGDALOD group is depicted in Figure 3-4. This group contains the table of contents of relations and matrices associated with actual full rigid aerodynamic pressure data. As such, this data is not restricted to linear theory (See Ref. 4). An important distinction in the AANDE paradigm from the original ASTROS paradigm is the database storage and usage of rigid aerodynamic pressure data. In the original ASTROS paradigm, pressure data was created and stored as increment (or unit) data for each control parameter (e.g. α, P, δ_a). In the AANDE paradigm, rigid aerodynamic pressure data is stored as whole or *actual* data (e.g. pressure at α = 12.5 degrees, pressure at P = 200 deg./sec, ..). A basis pressure vector is defined in each RIGDALOD group. The vector defines the pressure state at a specified set of angles and rates. This new paradigm allows the ASTROS user generality in defining pressure states from various sources. Ensuing logic in the model manipulation creates the necessary incremental pressure vectors ASTROS needs to perform linear aeroelastic analysis. This paradigm also allows for future growth in ASTROS including nonlinear iterative maneuver trim analysis.

The ASTROS user may modify individual data groups such as in Table 3-1 by creating combinations from two or more existing run-time and/or archived data groups. Extending the analysis scenario of the previous paragraph, the analyst may also have an aerodynamic influence coefficient matrix created from a high order computation fluid dynamics basis. The AIC group from the CFD basis may be assembled with the new RIGDALOD group and the original STDYGEOM group. ***The only requirement for group manipulations such as described is that the geometry of the combined groups must be compliant.***

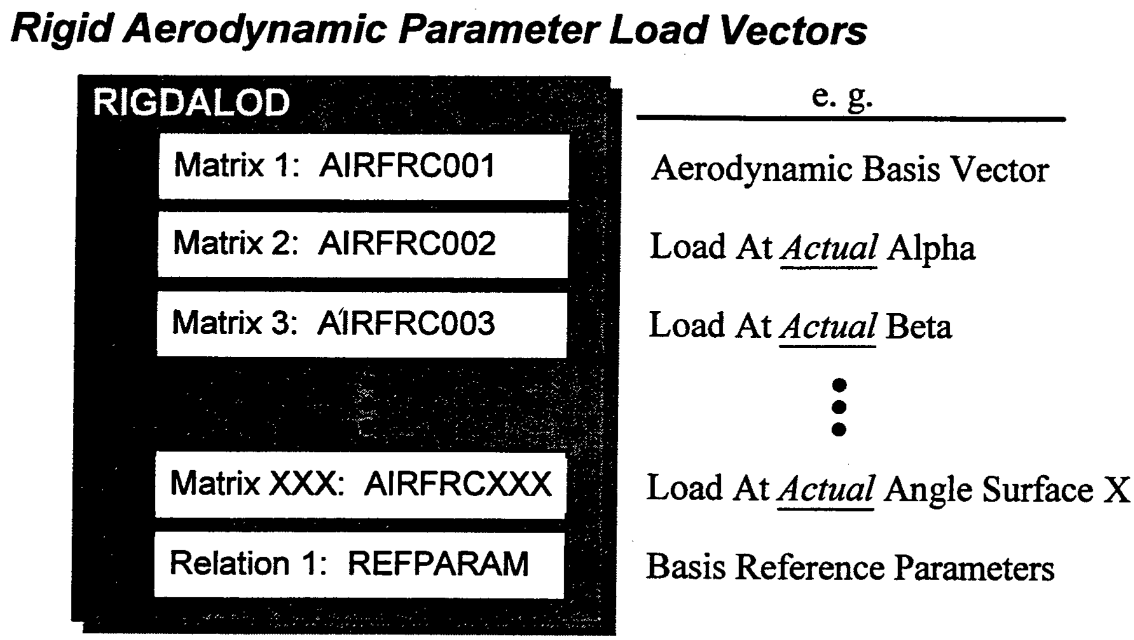


Figure 3-4 RIGDALOD - Rigid Aerodynamic Loads

The last group mentioned in the steady aerodynamic model is a new feature developed under the AANDE program. The RIGDSLOD group contains the addresses and table of contents of user defined loads (Ref. 4). These loads are created from ASTROS' STATICs load parameters such as FORCE, MOMENT, GRAV, and TEMP. The loads are created in the structural domain, and they can be used to add augment the aerodynamic simulation. For instance, force increments from a wind tunnel model may be used to simulate aerodynamic store loads. Another example of this capability is the development of force actuation simulations typifying adaptive materials in smart structures. As shown in Figure 3-5, a new ASTROS Bulk data entry has been created called SLPARM. In like manner to the RIGDALOD group, the load vectors in RIGDSLOD are stored as actual loads referenced to a load parameter magnitude.

A steady aeroelastic model (SAEMODEL) is created by the user through model assembly commands or automatically from specification of steady aerodynamic model in an FTRIM discipline. Note that the steady aerodynamic model is a subset of the steady aeroelastic model. The two groups, SPLINE and FLEXLOAD are added to the aerodynamic model in the creation of the aeroelastic model. The SPLINE group is created during the processing of splines defined in the traditional fashion of ASTROS' bulk data entries. A SPLINE group is a permanent ASTROS entity. That is, once it is created, it is never purged from the ASTROS database and may be reused. The FLEXLOAD group, however, is recreated within each aeroelastic solution.

Presented in Figure 3-6 is a case where three databases are manipulated to assemble the aerodynamic model desired for a symmetrical flexible maneuver trim simulation. From the commands in this solution sequence, addresses of physical data are made known to the functional modules in ASTROS so that the physical data of a complete aerodynamic model may be assembled on the runtime database.

Rigid Structural Parameter Load Vectors

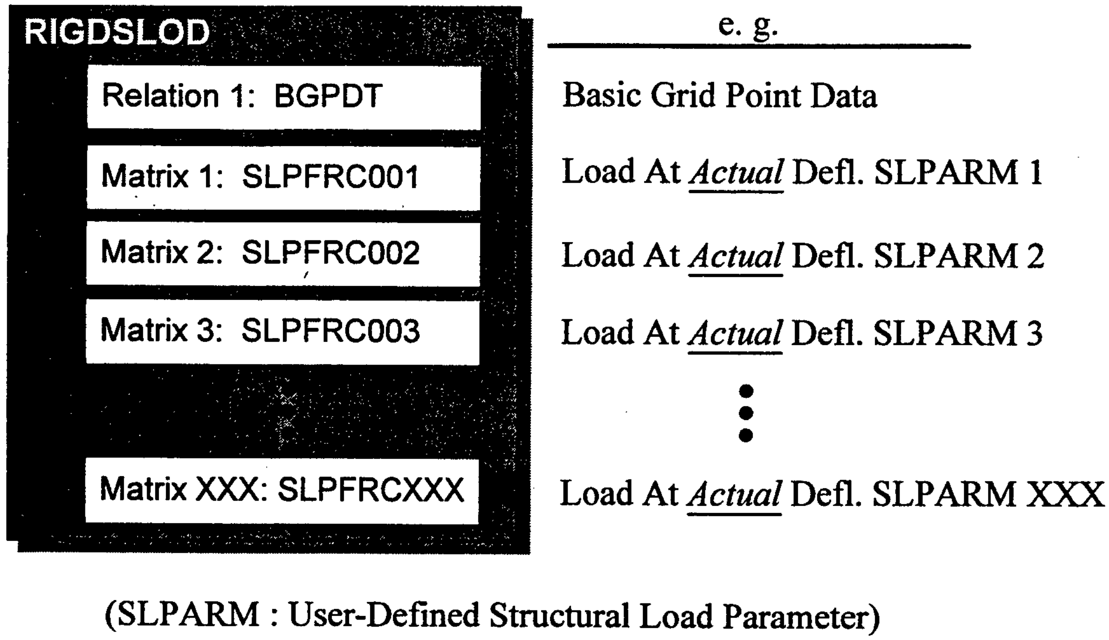


Figure 3-5 RIGDSLOD - Rigid Structural Loads

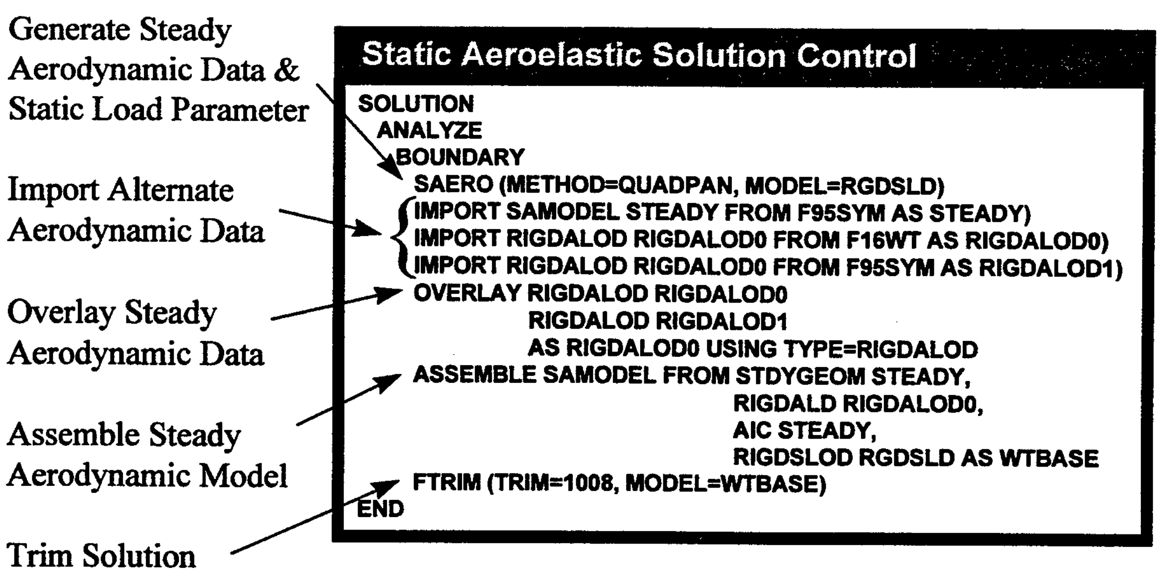


Figure 3-6 Assembly And Trim Solutions of Aeroelastic Equations

The SAERO command allows the generation of a user defined load from the static load parameter capability. The data is stored in the ASTROS runtime database in the MODEL name RGDSLD. A steady aerodynamic MODEL named STEADY is imported from the database F95SYM. From that model, the RIGDALOD group - table of contents for rigid aerodynamic pressure data which is named RIGDALOD0 is imported and named RIGDALOD1. The physical aerodynamic data still lies out on the F95SYM database, but its address and table of contents are placed on the runtime database in RIGDALOD1. A third database RIGDALOD is also imported from F16WT and named RIGDALOD0. Similarly, the address and table of contents of the physical aerodynamic data associated with the model in F16WT is stored on the ASTROS runtime database.

The OVERLAY command is used in this case to acquire data from the F16WT database. This data is wind tunnel pressures, and by virtue of the ordering of the RIGDALOD groups and the intersection of certain aerodynamic parameters in both RIGDALOD groups (e.g. α), certain addresses in the table of contents of RIGDALOD0 will be replaced with addresses from RIGDALOD1. Also, by virtue of a union, the new RIGDALOD0 group will contain items from both original groups.

Once these data are known to the runtime database, an aerodynamic model may be assembled. In this case, the geometry (identified by the group STDYGEOM) comes from the MODEL STEADY. The physical data is stored on the F95SYM database, and via the import of STEADY, the functional modules now have the address of all the geometry. Similarly, the AIC group is designated from STEADY. The new aerodynamic model is named WTBASE.

The FTRIM discipline identifies the aerodynamic model for use, and an AEROELASTIC MODEL is formed with this name. In the database, the AEROELASTIC MODEL name will be WTBASESAE. The AEROELASTIC MODEL will look for a SPLINE group named WTBASE since no group is identified in the FTRIM callout. However, a SPLINE group could be identified if the user chose to.

The new solution control commands are defined in the following pages.

Solution Control Command: **ARCHIVE**

Description: Requests that aerodynamic group data be archived to an eBase database.

Format and Example:

```
ARCHIVE            group_type group_name TO logical_db AS new_group_name

ARCHIVE SAMODEL STEADYSAE TO FSWARC AS ALTAERO
```

Option	Meaning
group_type	The GROUP type (e.g. SAMODEL, STDYGEOM, AIC, etc.) to be archived.
group_name	Name of GROUP that will be archived.
logical_db	The logical name of an eBase database that has been ASSIGNed during the current job.
new_group_name	A new group name that will be created for the archived data.

Remarks:

1. The ARCHIVE command physically copies all the entities in the group onto the specified database.

Solution Control Command: **ASSEMBLE**

Description: Creates an aerodynamic model from the GROUP entity.

Format and Example:

```
ASSEMBLE          model_type FROM <group_type_name_list> AS model_name

                  <group_type_name_list> ⇒ group_type  group_name,...

ASSEMBLE SAMODEL FROM STDYGEOM STDYGEOM000, RIGDALOD RIGDALOD001,
                    'AIC AIC003 AS ALTAERO

ASSEMBLE SAMODEL FROM STDYGEOM STEADY1, RIGDALOD STEADY1,
                     AIC STEADY2 AS ALTAERO
```

Option	Meaning
group_type	The GROUP type (e.g. SAMODEL, STDYGEOM, AIC, etc.) to be assembled from.
group_name	Name of GROUP that will be assembled from.
model_name	A new model name that will be assembled.
model_type	The type of the new model that will be assembled.

Remarks:

1. The ASSEMBLE command will only create the table of contents for the new group. All of the entities in the groups will remain on their original databases, i.e. no data are copied.
2. The Group Names specified in the ASSEMBLE command may be the names of the group entities that will be assembled, or the names of group entities which themselves contain the groups to be assembled.

Solution Control Command: **FTRIM**

Description: Creates an aerodynamic model from the GROUP entity.

Format and Example:

```
FTRIM        sym ( TRIM = a, TRIMOPT = b, MODEL = name, DCON=c)

FTRIM
```

Option	Meaning
sym	The symmetry option from SYMMETRIC, ANTISYMMETRIC or ASYMMETRIC.
a	TRIM condition identification number.
b	TRIM optimization identification number..
name	The name of the aerodynamic model..
c	Identification number for design constraints.

Remarks:

1. The aerodynamic model to be used by the **FTRIM** discipline may be the result of a previous **SAERO** discipline, or it may be assembled from various sources, including databases (not including the run-time database), results of other **SAERO** or **FTRIM** disciplines, or through the aerodynamic model assembly Solution Control commands

Solution Control Command: **IMPORT**

Description: To import an aerodynamic GROUP from an existing eBase database.

Format and Example:

```
IMPORT            group_type group_name FROM logical_db AS
new_group_name

IMPORT SAMODEL STEADY FROM FSWARC AS ALTAERO
```

Option	Meaning
`group_type`	The GROUP type (e.g. SAMODEL, STDYGEOM, AIC, etc.) to be imported..
`group_name`	Name of GROUP that will be imported.
`logical_db`	The logical name of an eBase database that has been ASSIGNed during the current job.
`new_group_name`	A new group name that will be used for the imported data.

Remarks:

1. The IMPORT command will only create the table of contents for the new group. All of the entities in the groups will remain on their original databases, i.e. no data are copied.

Solution Control Command: **OVERLAY**

Description: Creates a new aerodynamic model by overlaying two or more GROUPs.

Format and Example:

```
OVERLAY              <group_type_list> AS <new_group_name> USING TYPE
group_type

                     <group_type_name_list> ⇒ group_type group_name,...

      OVERLAY RIGDALOD ALTER1, RIGDALOD ALTER2, RIGDALOD ALTER3 AS
      ALTNEW
              USING TYPE RIGDALOD
```

Option	Meaning
`group_type_list`	Specifies the types and names of one or more groups, separated by commas, that will be overlayed. The overlay is performed in the order listed. This means that the common data from the last `group_name` will be the data used.
`new_group_name`	Name of thew new GROUP created by the overlay.
`group_type`	The GROUP type (e.g. SAMODEL, STDYGEOM, AIC, etc.) of the data to be overlayed.

Remarks:

1. The OVERLAY command will only create the table of contents for the new group. All of the entities in the groups will remain on their original databases, i.e. no data are copied.

Solution Control Command: **SAERO**

Description: Invokes the generation of aerodynamic matrices.

Hierarchy Level: Discipline

Format and Examples:

```
SAERO       sym ( METHOD = meth, MACH = mach, MODEL = name,
                  AIC = option  )

SAERO
```

Option	Meaning
sym	The symmetry option from SYMMETRIC, ANTISYMMETRIC or ASYMMETRIC.
meth	The method for aerodynamic matrix generation from USSAERO or QUADPAN.
mach	Mach number at which matrices will be generated.
name	The name of aerodynamic model.
Option	NONE for no AIC computation
	BOTH for computation of both symmetric and antisymmetric AIC
	ANTI for antisymmetric AIC
	SYMM for symmetric AIC
	ASYM for asymmetric AIC

Remarks:

1. **SAERO** will generate the specified aerodynamic model to be used by downstream steady aerodynamic disciplines, or to be archived on the specified database
2. The default for USSAERO is to create both the symmetric and antisymmetric AIC matrix
3. QUADPAN will check the flow computation requests and may override the AIC request in select cases. If combined symmetric and antisymmetric boundary condition flow request are made, and an antisymmetric or symmetric AIC is requested, the AIC request will be overridden, and an asymmetric AIC matrix will be computed.

THIS PAGE INTENTIONALLY LEFT BLANK

4. THE INPUT DATA STREAM

The following modification to Section 3.1 of the version 20 CRDA release of the ASTROS User's Guide has been made. The underlined passages indicate new text. The figure numbers called out in this section must be found in the version 20 ASTROS' User's Guide. The underlined changes apply to the Air Force version 12.0 as was as the CRDA version 20.

The ASTROS user directs the system through an input data stream composed of a ***Resource Section***, which allocates ASTROS databases and specifies memory utilization, which is followed by multiple ***Data Packets.*** Each packet contains a set of related data providing the information needed to execute ASTROS. The packets begin with a keyword indicating the nature of the data within the packet and terminate with an ending keyword or with the start of the next data packet. All the packets in the input data stream are optional, and, with only two exceptions, they may appear in any order in the input data stream. The purpose of this section is to document the structure of the input data stream. Detailed documentation of the data within each data packet in then presented in separate chapters.

Figure 3-1 shows the general form of the input data stream, and Figure 3-2 illustrates the actual input stream features with a sample stream for a ten bar truss model. The first non-blank record of the input file must be either the **ASSIGN** command or a Resource Section. If an **ASSIGN** command is used, then this command will enable you to attach the run-time database(s) that are used during the execution of the ASTROS system. These runtime databases may include several aerodynamic models targeted for OVERLAY and ASSEMBLE commands in the SOLUTION packet. There are five optional data packets following the **ASSIGN** command. The first is the DEBUG packet, which contains debug commands to control or select specific actions within the executive and database management systems. The remaining four packets may appear in any order in the input stream. The second packet is the MAPOL packet containing the executive system control directives consisting of either a standalone MAPOL program or EDIT commands to modify the standard MAPOL program. If the MAPOL packet is absent, the unmodified standard MAPOL sequence directs the execution. The Solution Control commands appear in the third optional packet denoted by the keyword SOLUTION. These commands select the engineering data to be used in each subcase from the set of data provided in the Bulk Data packet. The fourth packet is the FUNCTION packet. It contains the definition of functions which allow the user to define new design constraints or an objective function. These functions may combine nodal and element response quantities for various boundary conditions and disciplines. The fifth data packet is the BULK DATA packet. The BULK DATA packet contains the engineering data describing the finite element structural model, the aerodynamic model(s), and the design model, as well as all the data needed to perform the specific analysis and/or optimization tasks. The MAPOL, SOLUTION, and BULK DATA packets are analogous to the NASTRAN executive control, case control, and bulk data decks, respectively. The final packet, called the QUADPAN packet, contains specific data used by the QUADPAN module. This packet must start with BEGIN QUADPAN, and end with END QUADPAN. The QUADPAN packet can be input in any order with or without the BULK DATA packet.

THIS PAGE IS INTENTIONALLY LEFT BLANK

5. THE BULK DATA PACKET

This section includes changes made to the Bulk Data packet.

The following pages describe the new and modified Bulk Data entries. These are summarized in the following table:

ATTACH	Completely changed to support new features
BMST, BMST2	New entries
DCONBMST	New entry.
DCONSCF	Extended to support new capability
PANLST1, 2	New entries.
RELES	New entry.
RMASS	New entry.
SCHEDULE	New entry.
SLPARM	New entry.
SPLINE1, 2, 3	Completely changed to support new features
TCONBMST	New entry.
TCONFUNC	New entry.
TCONTRM	New entry.
TFUNC	New entry.
TODVPRM	New entry.
TOMPPARM	New entry.
TRIM	Extended to support new capability
TRIMOPT	New entry.
TRIMR	New entry.

Input Data Entry: **ATTACH** Aerodynamic Panel Attachment

Description: Defines the aerodynamic control points to be attached to a reference grid for load transfer and aeroelastic feedback.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
ATTACH	EID	MODEL	SETK	RGRID	FEEDBK				

ATTACH	100	QDPAN1	111	1					

Field	Contents
EID	Element identification number (Integer > 0)
MODEL	Name of steady aerodynamic input packet (text or blank)
SETK	Refers to a PANLSTi entry which lists the aerodynamic boxes whose motions are interpolated using this spline (Integer > 0)
RGRID	Grid point identification of reference grid point (Integer > 0)
FEEDBK	Aeroelastic feedback selection. One of the strings RIGID, FLEX or blank (Default = FLEX)

Remarks:

1. The ATTACH EID must be unique with respect to all other SPLINEi and ATTACH bulk data entries, it is used only for error messages.
2. This entry applies to both the steady and unsteady aerodynamic models.

Input Data Entry: **BMST** Load Components Definition

Description: Defines load components (BMST) for the FTRIM discipline.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
BMST	BMSTLAB	CID	GID	GLSTID	PLSTID				

BMST	WROOT		100	101	102				

Field	Contents
BMSTLAB	Label of component load (Text)
CID	Reference coordinate system identification number (Integer > 0 or blank)
GID	Reference grid point identification number (Integer > 0)
GLSTID	Grid list set identification (Integer > 0)
PLSTID	Aerodynamic panel list set identification (Integer > 0)

Remarks:

1. The component load will be integrated with respect to the reference coordinate system at the reference grid point location. If a reference coordinate system is not specified, the basic coordinate system will be used.
2. The grid list set identification references a GRIDLIST entry and defines the structural grid points to be integrated to produce the rigid-splined and flexible component load. This list is applicable to the flexible trim, FTRIM, discipline only.
3. The panel list set identification references a PANLST1 or PANLST2 entry and defines the aerodynamic panels to be integrated to produce the rigid component load.

Input Data Entry: **BMST2** Load Components Definition

Description: Defines load components (BMST) for the RTRIM discipline.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
BMST	BMSTLAB	PLSTID	X1	Y1	Z1	X2	Y2	Z2	CONT
CONT	X3	Y3	Z3	XREF	YREF	ZREF			

BMST	WROOT	101	0.	0.	0.	0.	0.	100.	+ABC
+ABC	100.	0.	100.	300.	0.	0.			

Field	Contents
BMSTLAB	Label of component load (Text)
PLSTID	Aerodynamic panel list set identification (Integer > 0)
(X1, Y1, Z1)	Coordinates of the origin of the coordinate system in which the component load will be defined.
(X2, Y2, Z2)	Coordinates on the z-axis of the coordinate system in which the component load will be defined.
(X3, Y3, Z3)	Coordinates of a point in the x-z plane of the coordinate system in which the component load will be defined.
(XREF, YREF, ZREF)	Coordinates of the reference point at which the component load is defined.

Remarks:

1. The panel list set identification references a PANLST1 or PANLST2 entry and defines the aerodynamic panels to be integrated to produce the rigid component load.
2. The three points (X1, Y1, Z1) , (X2, Y2, Z2) , and (X3, Y3, Z3) must be non-collinear.
3. The component loads will be integrated at the reference location specified by (XREF, YREF, ZREF) with respect to the reference coordinate system defined by (X1, Y1, Z1) , (X2, Y2, Z2) , and (X3, Y3, Z3).
4. The reference coordinates must be defined in the same coordinate system as the aerodynamic model.

THIS FEATURE IS NOT IMPLEMENTED.

Input Data Entry: **DCONBMST** Structural Optimization Component Load (BMST) Constraint

Description: Defines a component load constraint to be used in a design optimization.

$$BMST(i) \leq BMST(i)_{Limit}^{Upper\ Bound} \quad or \quad BMST(i) \geq BMST(i)_{Limit}^{Lower\ Bound}$$

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
DCONBMST	SETID	BMSTLAB	CMPNT	CTYPE	BMSTLIM				

DCONBMST	101	WROOT	4	UPPER	1.0+7				

Field	Contents
SETID	Set identification number (Integer > 0)
BMSTLAB	Label of a component load (Text)
CMPNT	Component number of the component load (Integer: ± 1, 2, 3, 4, 5 or 6)
CTYPE	Constraint type; either **UPPER**, for upper bound, or **LOWER** for lower bound. (Text, Default = **UPPER**)
BMSTLIM	Bound for the component load (Real) (Consistent units)

Remarks:

1. BMST constraints are selected in Solution Control with the discipline option : **DCON=SETID** .
2. The **BMSTLAB** must be defined by a **BMST** entry.
3. In the case of a structural optimization using a BMST constraint and an asymmetric trim with a half model, a negative BMST component number, **CMPNT**, refers to the image (left) side of the model.
4. A **LOWER** bound constraint excludes all values to the left of **BMSTLIM** on a real number line, while an **UPPER** bound excludes all values to the right, irrespective of the sign of **BMSTLIM**.

THIS FEATURE IS NOT IMPLEMENTED.

Input Data Entry: **DCONSCF** Stability Derivative Constraint

Description: Defines a constraint on the flexible stability derivative at the reference grid point associated with the force or moment due to a trim parameter or control surface deflection of the form:

$$\left[\frac{\partial C_F}{\partial \delta_{trim}}\right]_{lower} \leq \frac{\partial C_F}{\partial \delta_{trim}} \leq \left[\frac{\partial C_F}{\partial \delta_{trim}}\right]_{upper}$$

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
DCONSCF	SETID	ACCLAB	PRMLAB	CTYPE	PRMREQ	UNITS	FSCFLAG		

DCONSCF	1001	PACCEL	AILERON	LOWER	1.0	RADIANS	0		

Field	Contents
SETID	Set identification number referenced by the **DCONSTRAINT** Solution Control option of the **FTRIM** or **FLEXSTAB** command. (Integer > 0)
ACCLAB	Alphanumeric string identifying the aerodynamic force or moment by naming the corresponding structural acceleration in a manner consistent with the **TRIM** entry. See Remarks 2 and 4.
PRMLAB	Alphanumeric string identifying a constrained control surface or aeroelastic trim parameter (e.g. **ALPHA** or **PRATE**). See Remarks 3 and 4.
CTYPE	Constraint type; either **UPPER**, for upper bound, or **LOWER** for lower bound. (Character, Default = **UPPER**)
PRMREQ	Bound for the stability coefficient. For units, see Remarks 5 an 6. (Real)
UNITS	Units for the stability coefficient. Either **RADIANS** or **DEGREES.** See Remark 6. (Real, Default = **DEGREES**)
FSCFLAG	Flag signifying flexible stability coefficient definition. Either **0** for unrestrained definition (includes inertia relief) or **1** for restrained definition (does not include inertia relief). (Integer, Default = 0)

Remarks:

1. The **DCONSCF** entry is selected in Solution Control with the **DCONSTRAINT=SETID** option of the **FTRIM** or **FLEXSTAB** command.
2. The **ACCLAB** may refer to any of the **TRIM** Bulk Data entry trim parameters that are **structural accelerations.** Valid trim parameters are **NX, NY, NZ, PACCEL, QACCEL,** and **RACCEL.**
3. The **PRMLAB** may refer to **AESURF** or **CONLINK** control surfaces or to any of the **TRIM** entry parameters except the structural accelerations. Valid selections are: **PRATE, QRATE, RRATE, ALPHA, BETA, THKCAM** and any control surface label. Invalid trim parameters are: **NX, NY, NZ, PACCEL, QACCEL** and **RACCEL.**
4. Any combination of forces or moments and trim parameters/control surfaces may be used on this entry **provided** they have the same symmetry as the associated **TRIM** entry.

Furthermore, to apply the constraint to the flexible derivative, the degree of freedom corresponding to the force or moment must be supported in the boundary condition. For example, to constrain the pitching moment, **QACCEL,** due to angle of attack, **ALPHA**, the y-rotation of the support point must be on the **SUPPORT** entry for the boundary condition in which the **TRIM** is analyzed.

5. The stability derivatives are nondimensional quantities derived from the flexible forces and moments due to "unit" parameters. The constraint is applied to the nondimensional derivative at the user-defined reference point. To assist the defining **PRMREQ,** the following normalizations are used in ASTROS:

SYMMETRIC DERVIATIVES	FORCES	CONTROL SURFACES	stability coefficient = F/ (QDP*S)
		RATES	stability coefficient = F*2*V0 / (QDP*S*C) "unit" rate = unit dimensional rate * C/2*V0
	MOMENTS	CONTROL SURFACES	stability coefficient = M/ (QDP*S*C)
		RATES	stability coefficient = M*4*V0/ (QDP*S*C**2) "unit" rate = unit dimensional rate * C/2*V0
ANTISYMMETRIC DERVIATIVES	FORCES	CONTROL SURFACES	stability coefficient = F/ (QDP*S)
		RATES	stability coefficient = F*2*V0/ (QDP*S*B) "unit" rate = unit dimensional rate * B/2*V0
	MOMENTS	CONTROL SURFACES	stability coefficient = M/ (QDP*S*B)
		RATES	stability coefficient = M*4*V0/ (QDP*S*B**2) "unit" rate = dimensional rate * B/2*V0

F and **M** are the dimensional flexible forces and moments for the full vehicle; **S, C**, and **B** are the non-dimensional factors from the **AEROS** Bulk Data entry (the inputs are assumed to be for the full vehicle); and **QDP** and **V0** are defined on the **TRIM** Bulk Data entry.

6. **RADIANS** or DEGREES refer to the units of the unit control surface deflection or unit rate. **RADIANS** imply the value due to a unit RAD or RAD/S while **DEGREES** imply the value due to a unit DEG or DEG/S. **THKCAM** has no valid angular unit, hence the **UNITS** field is ignored.

7. A **LOWER** bound constraint excludes all values to the left of **PRMREQ** on a real number line, while an **UPPER** bound excludes all values to the right, irrespective of the sign of **PRMREQ**.

Input Data Entry: **PANLST1** Aerodynamic Panel Set Definition.

Description: Defines a set of aerodynamic panels by identifying opposite corner panels.

Format and Example(s):

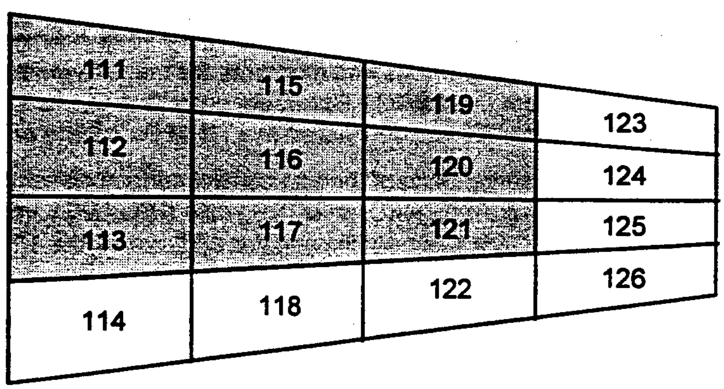
1	2	3	4	5	6	7	8	9	10
PANLST1	SETID	MACROID	BOX1	BOX2					

PANLST1	101	111	111	121					

Field	Contents
SETID	Set identification number (Integer > 0)
MACROID	Identification number of aerodynamic macroelement (Integer > 0)
BOX1, BOX2	The identification number of the first and last boxes on the aerodynamic macroelement (Integer > 0)

Remarks:

1. The set of aerodynamic boxes (k-set) will be defined by the aero-cells. The sketch shows how BOX1 and BOX2 define the set of aerodynamic boxes.



Input Data Entry: **PANLST2** Aerodynamic Panel Set Definition.

Description: Defines a set of aerodynamic panels by a list.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
PANLST2	SETID	MACROID	BOX1	BOX2	BOX3	BOX4	BOX5	BOX6	CONT
CONT	BOX7	BOX8	-etc-						

PANLST2	101	51	51	52	53	54	55	56	CONT
CONT	57	58	59						

Alternate Form:

PANLST2	SETID	MACROID	BOX1	THRU	BOX2				

Field	Contents
SETID	Set identification number (Integer > 0)
MACROID	Identification number of aerodynamic macroelement (Integer > 0)
BOXi	The identification number of the aerodynamic box associated with macroelement MACROID (Integer > 0)

Input Data Entry: **RELES**

Description: Defines sets of component degrees of freedom at substructure GRID points which are not to be connected during a substructure reflection operation.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
RELES	SID	SNAME	GID1	DOF1	GID2	DOF2	GID3	DOF3	-CONT-
-CONT-	GID4	DOF4							

RELES	61	WINGR	110	45	119	124	137	456	

Field	Contents
SID	Connection set identification number. [1] (Integer > 0)
SNAME	Basic Substructure name. (Name)
GIDi	GRID or SCALAR point identification number. (Integer > 0)
DOFi	List of degrees of freedom to be released. (Integer > 0)

Remarks:

1. The release connectivity set must be selected in the solution control packet with the command: **RELES = SID**
2. The **RELES** data will override connections automatically generated.
3. The SNAME is only used as label.

Input Data Entry: **RMASS** Rigid Mass Definition

Description: Defines the rigid mass for the rigid trim discipline, **RTRIM**.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
RMASS	RMID	M	I11	I21	I22	I31	I32	I33	

RMASS	1001	49.7	16.2		16.2			7.8	

Field	Contents
RMID	Rigid mass identification number (Integer > 0)
M	Mass value (Real)
Iij	Mass moments of inertia measured at the mass center-of-gravity in the coordinate system defined by the aerodynamic model (Real)

Remarks:

1. The form of the inertia matrix about its center-of-gravity is taken as:

$$\mathbf{M} = \begin{bmatrix} M & & & & & \\ & M & & & SYM & \\ & & M & & & \\ & & & I_{11} & & \\ & & & -I_{21} & I_{22} & \\ & & & -I_{31} & -I_{32} & I_{33} \end{bmatrix}$$

where:

$$M = \int \rho \, dv$$

$$I_{11} = \int \rho \left(y^2 + z^2\right) dv$$

$$I_{22} = \int \rho \left(x^2 + z^2\right) dv$$

$$I_{33} = \int \rho \left(x^2 + y^2\right) dv$$

$$I_{21} = \int \rho \, x \, y \, dv$$

$$I_{31} = \int \rho \, x \, z \, dv$$

$$I_{32} = \int \rho \, y \, z \, dv$$

and x, y, z are components of distance from the center-of-gravity in the coordinate system defined by the aerodynamic model. The negative signs for the off-diagonal terms are supplied by the program.

THIS FEATURE IS NOT IMPLEMENTED.

Input Data Entry: **SCHEDULE** Control Surface Schedule Definition

Description: Defines a schedule of control surface position values as a function of other trim parameter values.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
SCHEDULE	TRIMID	SURFLAB	CTOL	MAXITER	SVID				CONT
CONT	PRMLAB1	PVID1	PRMLAB2	PVID2	-etc-				

SCHEDULE	101	LEF			103				+ABC
+ABC	ALPHA	105	QDP	107					

Field	Contents
TRIMID	Trim set identification number (Integer > 0)
SURFLAB	Label of the control surface to be scheduled. (Text)
CTOL	Convergence tolerance for the change in schedule value between iterations (Real) (Default = 0.1)
MAXITER	Maximum number of schedule iterations allowed (Integer > 0) (Default = 20)
SVID	Identification number of an **AEFACT** entry which contains the schedule values. (Integer > 0)
PRMLABi	Label of a trim parameter which the schedule is a function of. (Text)
PVIDi	Identification of an **AEFACT** entry which contains the parameter values at which the schedule values are defined. (Integer > 0)

Remarks:

1. The **TRIMID** and **SURFLAB** are referenced to the **TRIMID** and **LABELi** fields, respectively, on a **TRIM** entry.
2. The **SURFLAB** must appear on the **TRIM** card with the character string **SCHD** in the **VALi** field.
3. The **SURFLAB** may be any valid trim parameter.
4. If the change in schedule value between iterations is less than or equal to **CTOL**, the schedule will be considered converged.
5. The first **PRMLABi** specified in the **SCHEDULE** entry will vary the fastest in the **SVID** **AEFACT** entry, the second will vary the second fastest, and so on.
6. The **PRMLABi** may be any trim parameter which can be specified on the **TRIM** entry, as well as MACH and QDP.

Example:

A leading edge flap control surface, LEF, is to be scheduled as a function of ALPHA and QDP as follows:

	QDP = 500 psf	QDP = 1000 psf	QDP = 1500 psf
ALPHA = 0°	40°	25°	10°
ALPHA = 10°	35°	20°	5°
ALPHA = 20°	30°	15°	0°

The bulk data entries defining this schedule would appear as :

SCHEDULE	101	LEF			103				+ABC
+ABC	ALPHA	105	QDP	107					

AEFACT	103	40.	35.	30.	25.	20.	15.	10.	SCH1
+CH1	5.	0.							

AEFACT	105	0.	10.	20.					

AEFACT	107	500.	1000.	1500.					

Input Data Entry: **SPLINE1** Two Dimensional Surface Spline

Description: Defines a two dimensional surface spline for interpolating out-of-plane motion for aeroelastic problems.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
SPLINE1	EID	MODEL	CP	SETK	SETG	DZ			

SPLINE1	3	QDPAN1		21	101				

Field	Contents
EID	Element identification number (Integer > 0)
MODEL	Name of steady aerodynamic input packet (text or blank)
CP	Coordinate system defining the spline plane (Integer ≥ 0, or blank)
SETK	Refers to a PANLSTi entry which lists the aerodynamic boxes whose motions are interpolated using this spline (Integer > 0)
SETG	Refers to a SETi entry which lists the structural grid points to which the spline is attached (Integer > 0)
DZ	Linear attachment flexibility (Real ≥ 0.0)

Remarks:

1. The SPLINE1 EID must be unique with respect to all other SPLINEi and ATTACH bulk data entries, it is used only for error messages.
2. If no CP is specified, the spline plane is assumed to be the CAERO macro element mean plane.
3. Only the component of aerodynamic box force perpendicular to the spline plane is splined to the structural model.
4. Aerodynamic panel slope is computed using structural displacements perpendicular to the spline plane rotation axis. The spline plane rotation axis is equal to the cross product of the spline plane normal and the direction of free stream velocity.
5. The attachment flexibility (units of area) is used for smoothing the interpolation. If DZ = 0.0, the spline will pass through all deflected grid points. If DZ >> (area of spline), a least squares plane fit will occur. Intermediate values will provide smoothing.

Input Data Entry: **SPLINE2** Linear Spline

Description: Defines a beam spline for interpolating panels and bodies for aeroelastic analyses.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
SPLINE2	EID	MODEL	SETK	SETG	DZ	DTOR	CID	DTHX	CONT
CONT	DTHY								

SPLINE2	1000		22	102	0.	1.0	4	-1.	

Field	Contents
EID	Element identification number (Integer > 0)
MODEL	Name of steady aerodynamic input packet (text or blank)
SETK	Refers to a PANLSTi entry which lists the aerodynamic boxes whose motions are interpolated using this spline (Integer > 0)
SETG	Refers to a SETi entry which lists the structural grid points to which the spline is attached (Integer > 0)
DZ	Linear attachment flexibility (Real ≥ 0.0)
DTOR	Torsional flexibility, $\frac{EI}{GJ}$ (Real ≥ 0.0; use 1.0 for bodies)
CID	Rectangular coordinate system which defines the y-axis of the spline (Integer ≥ 0 or blank; not used for bodies)
DTHX, DTHY	Rotational attachment flexibility. DTHX is for rotation about the x-axis; not used for bodies. DTHY is for rotation about the y-axis; used for slope of bodies. (Real)

Remarks:

1. For panels, the spline axis is the projection of the y-axis of coordinate system CID, projected onto the plane of the panel. For bodies, the spline axis is parallel to the x-axis of the aerodynamic coordinate system.
2. The flexibilities are used for smoothing. Zero attachment flexibilities will imply rigid attachment, i.e., no smoothing. Negative values of DTHX and/or DTHY will imply no attachment.
3. The SPLINE2 EID must be unique with respect to all other SPLINEi and ATTACH bulk data entries, it is used only for error messages.

Input Data Entry: **SPLINE3** Three Dimensional Surface Spline

Description: Defines a three dimensional surface spline for interpolating out-of-plane motion for aeroelastic problems.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
SPLINE3	EID	MODEL	CP	SETK	SETG	DZ			

SPLINE3	3	QDPAN1		21	101				

Field	Contents
EID	Element identification number (Integer > 0)
MODEL	Name of steady aerodynamic input packet (text or blank)
CP	Coordinate system defining the spline plane (Integer ≥ 0, or blank)
SETK	Refers to a PANLSTi entry which lists the aerodynamic boxes whose motions are interpolated using this spline (Integer > 0)
SETG	Refers to a SETi entry which lists the structural grid points to which the spline is attached (Integer > 0)
DZ	Linear attachment flexibility (Real ≥ 0.0)

Remarks:

1. The SPLINE3 EID must be unique with respect to all other SPLINEi and ATTACH bulk data entries, it is used only for error messages.
2. If no CP is specified, the spline plane is assumed to be the CAERO macro element mean plane.
3. The spline plane (either default of user defined) is automatically corrected such that it includes the free stream velocity vector. First, the corrected spline plane rotation axis is computed from the cross product of the defined normal and the free stream velocity ($\langle \hat{\mathbf{n}}_{R'} \rangle = \langle \hat{\mathbf{n}}_N \rangle \times \langle \hat{\mathbf{n}}_{V0} \rangle$). Second, the corrected spline plane normal is computed from the cross product of the free stream velocity and the corrected spline plane rotation axis ($\langle \hat{\mathbf{n}}_{N'} \rangle = \langle \hat{\mathbf{n}}_{V0} \rangle \times \langle \hat{\mathbf{n}}_{R'} \rangle$).
4. Each xyz-component of aerodynamic box force is splined to the structural model.
5. Each aerodynamic panel slope is computed using structural displacements perpendicular to the local panel rotation axis.
6. The attachment flexibility (units of area) is used for smoothing the interpolation. If DZ = 0.0, the spline will pass through all deflected grid points. If DZ >> (area of spline), a least squares plane fit will occur. Intermediate values will provide smoothing.

Input Data Entry: **TCONBMST** Trim Optimization Component Load (BMST) Constraint

Description: Defines a component load constraint to be used in a trim optimization.

$$BMST(i) \le BMST(i)_{\substack{Upper\ Bound \\ Limit}} \quad or \quad BMST(i) \ge BMST(i)_{\substack{Lower\ Bound \\ Limit}}$$

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
TCONBMST	SETID	BMSTLAB	CMPNT	CTYPE	BMSTLIM				

TCONBMST	101	WROOT	4	UPPER	1.0+7				

Field	Contents
SETID	Set identification number (Integer > 0)
BMSTLAB	Label of a component (Text)
CMPNT	Component number of the component load (+/- Integer; 1, 2, 3, 4, 5, or 6)
CTYPE	Constraint type; either **UPPER**, for upper bound, or **LOWER** for lower bound. (Text, Default = **UPPER**)
BMSTLIM	Bound for the component load. (Real) (Consistent units)

Remarks:

1. The **SETID** is selected by the **TCONID** field of the **TRIMOPT** entry.
2. The **BMSTLAB** must be defined by a **BMST** entry.
3. In the case of an asymmetric trim optimization including a BMST constraint with a half model, a negative BMST component number, **CMPNT**, refers to the image (left) side of the model.
4. A **LOWER** bound constraint excludes all values to the left of **BMSTLIM** on a real number line, while an **UPPER** bound excludes all values to the right, irrespective of the sign of **BMSTLIM**.

Input Data Entry: **TCONFUNC** Trim Optimization Functional Design Constraint

Description: Define one or more functional constraints to be used in a trim optimization.

$$F(v) \le F(v)_{\substack{\text{Upper Bound} \\ \text{Limit}}} \quad or \quad F(v) \ge F(v)_{\substack{\text{Lower Bound} \\ \text{Limit}}}$$

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
TCONFUNC	SETID	TFID	CTYPE	FUNCLIM					

TCONFUNC	101	103	UPPER	1.0+5					

Field	Contents
SETID	Set identification number selected by the **TCONID** field in the **TRIMOPT** entry (Integer > 0)
TFID	Trim optimization function identification which references a **TFUNC** entry (Integer > 0)
CTYPE	Constraint type; either **UPPER**, for upper bound, or **LOWER**, for lower bound (Text, Default = **UPPER**)
FUNCLIM	Bound for the function value (Real)

Remarks:

1. The **TCONFUNC** set identification is selected by the **TCONID** field in the **TRIMOPT** entry.
2. The **TFID** is referenced to a **TFUNC** Bulk Data entry.
3. A **LOWER** bound constraint excludes all values to the left of **FUNCLIM** on a real number line, while an **UPPER** bound excludes all values to the right, irrespective of the sign of **FUNCLIM**.

Input Data Entry: **TCONTRM** Trim Optimization Constraint Definition for a Trim Parameter

Description: Defines a trim parameter constraint to be used in the trim optimization.

$$\delta_{Trim} \leq \delta_{Trim\ Limit}^{Upper\ Bound} \quad or \quad \delta_{Trim} \geq \delta_{Trim\ Limit}^{Lower\ Bound}$$

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
TCONTRM	SETID	PRMLAB	CTYPE	PRMLIM					

TCONTRM	100	AILERON	UPPER	30.0					

Field	Contents
SETID	Set identification number referenced by the **TCONID** in the **TRIMOPT** bulk data entry. (Integer > 0)
PRMLAB	Alphanumeric string identifying a constrained trim parameter (Text) (See Remark 2.)
CTYPE	Constraint type; either **UPPER**, for upper bound, or **LOWER** for lower bound. (Text, Default = **UPPER**)
PRMLIM	Bound for the trim parameter. For units, see Remark 3. (Real)

Remarks:

1. The **TCONTRM SETID** is selected by the **TCONID** option in the **TRIMOPT** bulk data entry.
2. The **PRMLAB** may refer to **AESURF** or **CONLINK** control surfaces or to any of the **TRIM** entry parameters, **NX**, **NY**, **NZ**, **PACCEL**, **QACCEL**, **RACCEL**, **PRATE**, **QRATE**, **RRATE**, **ALPHA**, or **BETA**. The only requirement is that the constrained trim parameter must be declared as a trim optimization design variable using the **TODVPRM** entry. The user will be warned if trim parameters not declared as trim optimization design variables are constrained (since these parameters are fixed, they are design invariant).
3. The units for control surface deflections are degrees. For rates, the units should be radians/sec. For linear accelerations **NX**, **NY**, **NZ**, the units should be consistent, (length/sec/sec) or, if a **CONVERT**, **MASS** entry was used, should be dimensionless. Angular accelerations should be in radians/sec/sec.
4. A **LOWER** bound constraint excludes all values to the left of **PRMLIM** on a real number line, while an **UPPER** bound excludes all values to the right, irrespective of the sign of **PRMLIM**.

Input Data Entry: **TFUNC** Trim Optimization Function Definition

Description: Define a weighted sum function to be used in a trim optimization.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
TFUNC	TFID	FTYPE1	FNAME1	FID1	WFACT1				CONT
CONT		FTYPE2	FNAME2	FID2	WFACT2	-etc-			

TFUNC	101	PARM	NZ		1.0				+ABC
+ABC		PARM	PRATE		1.0				

Field	Contents
TFID	Trim optimization function identification number (Integer > 0)
FTYPEi	Function argument type selected from the following : **PARM**, **BMST**, or **DRAG** (Text)
FNAMEi	Function argument name (Text)
FIDi	Function argument identification number (Integer)
WFACTi	Weighting factor for this function argument (Real) (Default = 1.0)

Remarks:

1. The **TFID** field is referenced to the **TFID** field in a **TCONFUNC** entry or the **OBJID** field in a **TRIMOPT** entry.
2. The function argument name, **FNAMEi**, field is referenced to a text label for the **PARM** and **BMST** function argument types.
3. The function argument identification number, **FIDi**, field is referenced to an identification number for the **DRAG** function argument type or a component number for the **BMST** function argument type. In the case of an asymmetric trim optimization of a BMST with a half model, a negative BMST component number, **FIDi**, refers to the image (left) side of the model.
4. The use of the **FNAMEi** and **FIDi** fields vary with the **FTYPEi**. This usage is shown in the table below.

FTYPEi	FNAMEi	FIDi
PARM	**PRMLAB**	
BMST	**BMSTLAB**	component #
DRAG		**DRAGID**

5. The value of each function argument will be multiplied by the value in the weighting factor, **WFACTi**, field.
6. The function value will be the weighted sum of the function argument values.

Input Data Entry: **TODVPRM** Trim Optimization Design Variable Definition for Trim Parameters

Description: Defines a design variable for the trim optimization problem when the variable is a trim parameter.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
TODVPRM	SID	DVID	PRMLAB	DVMIN	DVMAX	DVINIT			

TODVPRM	101	1001	NZ	-3.0	9.0	0.0			

Field	Contents
SID	Set identification number (Integer > 0)
DVID	Design variable identification number (Integer > 0)
PRMLAB	Trim parameter label (Text)
DVMIN	Minimum allowable value of the design variable (Real) (Default = -1000.)
DVMAX	Maximum allowable value of the design variable (Real) (Default = 1000.)
DVINIT	Initial value of the design variable (Real, **DVMIN ≤ DVINIT ≤ DVMAX**) (Default = 0.0) or the character string, **TRIM**

Remarks:

1. **DVID** must be unique within the set identification.
2. The character string **TRIM** in the **DVINIT** field indicates that the initial value of the design variable is to be determined by a TRIM solution where **PRMLAB** must appear as a trim variable.

Input Data Entry: **TOMPPARM** Trim Optimization Mathematical Programming Parameters

Description: Identify values of user defined optimizer parameters that override the default values for use in the trim optimization.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
TOMPPARM	TRIMID	PARAM	VALUE	PARAM	VALUE	PARAM	VALUE		CONT
CONT	PARAM	VALUE	-etc-						

TOMPPARM	101	CTMIN	0.0005	ITMAX	30				

Field	Contents
TRIMID	Trim identification number (Integer > 0)
PARAM	Name of parameter to be overridden (Text)
VALUE	Integer or real value to be used for the parameter (Integer or real)

Remarks:

1. Any number of **PARAM - VALUE** combinations can be specified on a **TOMPPARM** entry.
2. See μ-**DOT** software manual for a definition of parameters, but the most useful are shown below:

REAL PARAMETER	DEFINITION	DEFAULT
CT	Constraint tolerance. A constraint is active if its numerical value is more positive than CT.	-0.003
CTMIN	Minimum constraint tolerance for nonlinear constraints. If a constraint is more positive than CTMIN, it is considered to be violated.	-0.003
DABOBJ	Maximum absolute change in the objective between two consecutive iterations to indicate convergence in optimization.	$\max(0.001\,\lvert F_0\rvert, 0.0001)$
DELOBJ	Maximum relative change in the objective between two consecutive iterations to indicate convergence in optimization.	0.001
DOBJ1	Relative change in the objective function attempted on the first optimization iteration. Used to estimate initial move in the one-directional search. Updated as the optimization progresses.	0.1
DOBJ2	Absolute change in the objective function attempted on the first optimization iteration. Used to estimate initial move in the one-directional search. Updated as the	0.2 max(X_i)

	optimization progresses.	

REAL PARAMETER	DEFINITION	DEFAULT
DX1	Maximum relative change in a design variable attempted on the first optimization iteration. Used to estimate initial move in the one-dimensional search. Updated as the optimization progresses.	0.01
DX2	Maximum absolute change in a design variable attempted on the first optimization iteration. Used to estimate initial move in the one-dimensional search. Updated as the optimization progresses.	0.02

INTEGER PARAMETER	DEFINITION	DEFAULT
ISCAL	Scaling parameter. By default, scaling is done every NDV iterations, otherwise scaling is performed every ISCAL iterations. (-1 turns off scaling)	NDV
ITMAX	Maximum number of iterations allowed at the optimizer level.	40
ITRMOP	The number of consecutive iterations for which the absolute or relative convergence criteria must be met to indicate convergence at the optimizer level.	2

Input Data Entry: **TRIM** Trim Variable Specification

Description: Specifies conditions for steady aeroelastic trim

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
TRIM	SETID	MACH	QDP	TRMTYP	EFFID	V0			CONT
CONT	LABEL1	VAL1	LABEL2	VAL2	LABEL3	VAL3	LABEL4	VAL4	-etc-

TRIM	1001	0.90	1200.	PITCH	100	926.3			+ABC
+ABC	NZ	8.0	QRATE	0.243	ELEV	FREE	ALPHA	FREE	

Field	Contents
SETID	Trim set identification number (Integer > 0)
MACH	Mach number (Real ≥ 0.0)
QDP	Dynamic pressure (Real ≥ 0.0)
TRMTYP	Type of trim required (Text or blank) (See Remark 3) blank **SUPORT** controlled trim **ROLL** Antisymmetric roll trim (1 DOF) **LIFT** Symmetric trim of lift forces (1 DOF) **PITCH** Symmetric trim of lift and pitching moment (2 DOF)
EFFID	Identification number of **CONEFFS** Bulk Data entries which modify control surface effectiveness values (Integer > 0 or blank) (Remark 2)
V0	True velocity (Real ≥ 0.0 or blank) (See Remark 14)
LABELi	Label defining aerodynamic trim parameters
VALi	Magnitude of the specified trim parameter (Real), the character string **FREE** , or the character string **SCHD**

Remarks:

1. The **TRIM** entry is selected in Solution Control in the **FTRIM** or **NPSAERO** disciplines with the **TRIM** option.
2. All aerodynamic forces created by the control surface will be reduced to the referenced amount. For example, an **EFF1** of 0.70 indicated a 30% reduction in the forces.
3. The **TRMTYP** filed has the following interpretation:

LIFT Implies that the vertical acceleration will be trimmed by one **FREE** symmetric control parameter of surface — OR — the acceleration computed for some set of symmetric parameters/surfaces.

ROLL Implies that the roll acceleration, **PACCEL**, will be trimmed by one **FREE**

antisymmetric
control parameter of surface — OR — the acceleration computed for some set of
antisymmetric parameters/surfaces. Any number of antisymmetric parameters may be
fixed, but the **FREE** parameters are limited to **PACCEL** — OR — any one antisymmetric
parameter or surface. For example, **PACCEL=0.0; AILERON=1.0; PRATE=FREE**

PITCH Implies that the vertical acceleration, **NZ**, and the pitch acceleration, **QACCEL**, will be
trimmed by no more than two **FREE** symmetric control parameters or surfaces — OR —
the accelerations computed for some set of symmetric parameters/surfaces. Any number
of symmetric surfaces may be fixed, but the **FREE** parameters are limited to **QACCEL**
and **NZ** — OR — up to two symmetric parameters or surfaces — OR — some
combination. For example, **NZ=8.0g's; QACCEL=0.0; ALPHA=FREE; ELEV=FREE**

blank Implies that the support DOFs are equal to the number of free parameters. Appropriate
trim equations are assembled and solved.

4. Units for **QDP** are force per unit area.
5. Units for **V0** are length per second.
6. Allowable options for **LABELi** for symmetric trim (the symmetry option is selected in Solution Control) are:

Structural Accelerations

NX	Longitudinal load factor (acceleration) (Remark 12)
NZ	Vertical load factor (acceleration) (Remark 12)
QACCEL	Pitch acceleration (Remark 13)

Aerodynamic Parameters

ALPHA	Angle of attack in degrees
QRATE	Pitch rate (Remark 14)
THKCAM	Thickness and camber (Remark 15)
and	Control surface position in degrees

7. Allowable options for **LABELi** for antisymmetric trim (the symmetry option is selected in Solution Control) are:

Structural Accelerations

NY	Lateral load factor (acceleration) (Remark 12)
PACCEL	Roll acceleration (Remark 13)
RACCEL	Yaw acceleration (Remark 13)

Aerodynamic Parameters

BETA	Side-slip angle in degrees
PRATE	Roll rate (Remark 14)
RRATE	Yaw rate (Remark 14)
and	Antisymmetric control surfaces in degrees

8. Allowable options for **LABELi** for asymmetric trim (the symmetry option is selected in Solution Control) are any of the options listed above for symmetric and antisymmetric trim. It is up to the user to specify a solvable trim problem.

9. If **VALi** is a real number, the associated aerodynamic parameter of structural acceleration is set to that value. If **VALi** is the character string **FREE**, then the associated parameter will be determined as part of the trim analysis. If **VALi** contains the character string **SCHD**, then a **SCHEDULE** entry with the same **TRIMID** and **SURFLAB** = **LABELi** will be used to determine the value of the associated parameter.

10. The number of **FREE** values of **VALi** must correspond exactly to the number of unknowns in the trim analysis. If **TRMTYP** is blank, the number of **SUPORT** DOFs.

11. If **TRIMID** is referenced by an **NPSAERO** discipline, **TRMTYP** must be blank and **FREE** is not allowed for **VALi**.

12. For **NX**, **NY**, and **NZ**, units are length per second per second in consistent units unless a **CONVERT/MASS** Bulk Data entry is provided. In this case, the values are dimensionless.

13. The angular accelerations, **PACCEL**, **QACCEL**, and **RACCEL**, are entered in units of radians per second per second.

14. **PRATE**, **QRATE**, and **RRATE** are entered in units of radians per second. The true velocity, **V0**, must be input if any of the "rate" parameters are given since its value is needed to dimensionalize the forces computed for a unit rate per velocity in the aerodynamic preface.

15. The **THKCAM** label refers to thickness and camber effects and its corresponding value is usually set to 1.0. Non-unit values of the **THKCAM** parameter are available only to provide added generality.

16. Any control surfaces, trim parameters, or structural accelerations not specified on the **TRIM** entry will not participate in the analysis; they will be given fixed values of 0.0. This includes **THKCAM**.

17. Refer to the **STATIC AEROELASTIC TRIM** Application Note for more information.

Input Data Entry: **TRIMOPT** Trim Optimization Definition

Description: Defines the trim optimization problem.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
TRIMOPT	TRIMID	OPTFLG	OBJTYP	OBJNAM	OBJID	TCONID	TDVSID		CONT
CONT	MAXITER	MOVLIM	CNVGLIM	PRINT					

TRIMOPT	3	MIN	BMST	WROOT	4	101	103		+ABC
+ABC	5	2.0	1.0	1					

Field	Contents
TRIMID	Trim set identification number (Integer > 0) of the associated TRIM or RTRIM entry.
OPTFLG	Optimization flag (Text) selected from the following: **MIN** **MAX**
OBJTYP	Objective function type (Text) selected from the following: **FUNC** **PARM** **BMST** **DRAG**
OBJNAM	Objective function name (Text or blank)
OBJID	Objective function identification number or BMST load component number (+/- Integer or blank) (See Remark 5.)
TCONID	Trim optimization constraint set identification (Integer > 0)
TDVSID	Trim optimization design variable set identification (Integer > 0)
MAXITER	Maximum number of optimization iterations to be performed (Integer > 0) (Default = 5)
MOVLIM	The move limit applied to trim optimization design variables. The trim variable after each iteration will lie between **t / MOVLIM** and **t * MOVLIM** where **t** is the initial value. (Real ≥ 0.0) (Default = 2.0) **(NOT IMPLEMENTED)**
CNVGLIM	Convergence limit specifying the maximum allowable imbalance in a supported degree of freedom that can be considered converged. (Real ≥ 0.0, consistent units) (Default = 1.0)
PRINT	Trim Optimization μ-**DOT** print flag (Integer 0 - 7) (Default = 0) (See Remark 8.)

Remarks:

1. The trim optimization identification number, **TOPTID**, is referenced in the **FTRIM** or **RTRIM** entry in solution control packet.

2. In the **OPTFLG** field, **MIN** signifies that the objective function will be minimized and **MAX** signifies that the objective function will be maximized.

3. In the **OBJTYP** field, **FUNC** indicates that the objective function is a function defined by a **TFUNC** entry, **PARM** indicates that the objective function is a trim parameter, **BMST** indicates that the objective function is a component load defined by a **BMST** entry, and **DRAG** indicates that the objective function is the drag defined in a **DRAG** entry.

4. The objective function name, **OBJNAM**, field is referenced to a text label for the **PARM** and **BMST** objective function types. If **OBJTYP=PARM**, **OBJNAM** may be any trim parameter label which can be specified on a **TRIM** entry for the given symmetry. If **OBJTYP=BMST**, **OBJNAM** must be defined by a **BMST** entry.

5. The objective function identification number, **OBJID**, field is referenced to an identification number for the **FUNC** and **DRAG** objective function types or a BMST load component number for the **BMST** objective function type. In the case of an asymmetric trim optimization of a BMST with a half model, a negative BMST component number, **OBJID**, refers to the image (left) side of the model.

6. The use of the **OBJNAM** and **OBJID** fields vary with the **OBJTYP**. This usage is shown in the table below.

OBJTYP	OBJNAM	OBJID
FUNC		**TFID**
PARM	**PRMLAB**	
BMST	**BMSTLAB**	component #
DRAG		**DRAGID**

5. The trim optimization constraint set identification, **TCONID**, is referenced to a **TCONTRM**, **TCONFUNC**, **TCONBMST**, or **TCONDRAG** bulk data entry.

6. The trim optimization design variable set identification, **TDVID**, is referenced to a **TODVPRM** bulk data entry.

7. The **MAXITER**, **MOVLIM**, and **CNVGLIM** values are applicable to the trim optimization process external to μ-DOT. μ-DOT default parameters may be changed via the **TOMPPARM** bulk data entry.

8. The Trim Optimization μ-DOT print flag may take the following values:

PRINT = 0	No output
PRINT = 1	Initial information and results
PRINT = 2	Same as above plus function values at each iteration
PRINT = 3	Same as above plus internal parameters
PRINT = 4	Same as above plus search directions
PRINT = 5	Same as above plus gradients
PRINT = 6	Same as above plus scaling information
PRINT = 7	Same as above plus 1-D search information

Input Data Entry: **TRIMR** Rigid Trim Variable Specification

Description: Specifies conditions for rigid trim.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
TRIMR	SETID	MACH	QDP	TRMTYP	EFFID	V0			CONT
CONT	XREF	YREF	ZREF	MASSID	CMPNTS				CONT
CONT	LABEL1	VAL1	LABEL2	VAL2	LABEL3	VAL3	LABEL4	VAL4	-etc-

TRIMR	1001	0.90	1200.	PITCH	100	926.3			+ABC
+ABC	300.	0.	0.	1001					+DEF
+DEF	NZ	8.0	QRATE	0.243	ELEV	FREE	ALPHA	FREE	

Field	Contents
SETID	Trim set identification number (Integer > 0)
MACH	Mach number (Real ≥ 0.0)
QDP	Dynamic pressure (Real ≥ 0.0)
TRMTYP	Type of trim required (Text or blank) (See Remark 3) blank — **CMPNTS** controlled trim **ROLL** — Antisymmetric roll trim (1 DOF) **LIFT** — Symmetric trim of lift forces (1 DOF) **PITCH** — Symmetric trim of lift and pitching moment (2 DOF)
EFFID	Identification number of **CONEFFS** Bulk Data entries which modify control surface effectiveness values (Integer > 0 or blank) (Remark 2)
V0	True velocity (Real ≥ 0.0 or blank) (See Remark 16)
XREF	Reference x-axis location for sum of moments (Real)
YREF	Reference y-axis location for sum of moments (Real)
ZREF	Reference z-axis location for sum of moments (Real)
MASSID	Identification number of **RMASS** Bulk Data entry specifying the inertia data to be used for trim (Integer > 0)
CMPNTS	Component number(s) to be used for trim (blank or any unique combination of the digits 1 through 6)
LABELi	Label defining aerodynamic trim parameters
VALi	Magnitude of the specified trim parameter (Real), the character string **FREE**, or the character string **SCHD**

Remarks:

1. The **TRIMR** entry is selected in Solution Control in the **RTRIM** discipline with the **TRIM** option.

2. All aerodynamic forces created by the control surface will be reduced to the referenced amount. For example, an **EFF1** of 0.70 indicated a 30% reduction in the forces.

3. The **TRMTYP** filed has the following interpretation:

   LIFT Implies that the vertical acceleration will be trimmed by one FREE symmetric control parameter of surface — OR — the acceleration computed for some set of symmetric parameters/surfaces.

   ROLL Implies that the roll acceleration, **PACCEL**, will be trimmed by one **FREE** antisymmetric control parameter of surface — OR — the acceleration computed for some set of antisymmetric parameters/surfaces. Any number of antisymmetric parameters may be fixed, but the **FREE** parameters are limited to **PACCEL** — OR — any one antisymmetric parameter or surface. For example, **PACCEL=0.0; AILERON=1.0; PRATE=FREE**

   PITCH Implies that the vertical acceleration, **NZ**, and the pitch acceleration, **QACCEL**, will be trimmed by no more than two **FREE** symmetric control parameters or surfaces — OR — the accelerations computed for some set of symmetric parameters/surfaces. Any number of symmetric surfaces may be fixed, but the **FREE** parameters are limited to **QACCEL** and **NZ** — OR — up to two symmetric parameters or surfaces — OR — some combination. For example, **NZ=8.0g's; QACCEL=0.0; ALPHA=FREE; ELEV=FREE**

   blank Implies that the specified DOFs are equal to the number of free parameters. Appropriate trim equations are assembled and solved.

4. Units for **QDP** are force per unit area.

5. Units for **V0** are length per second.

6. **XREF**, **YREF**, and **ZREF** define the reference location about which the moments will be summed to trim the aircraft. These values must reference the same coordinate system that the aerodynamic model was defined in. This is also the location where the specified **RMASS** inertia is assumed to be concentrated. In other words, these values define the center-of-gravity of the aircraft.

7. The **MASSID** should specify a **RMASS** Bulk Data entry that defines the inertia of the full aircraft to be trimmed.

8. The **CMPNTS** field specifies the DOFs in which the aircraft will be trimmed, i.e. zero net load. It is up to the user to specify a reasonable combination of DOFs and free trim parameters that will define a solvable trim problem. The **CMPNTS** field is required if the **TYMTYP** field is blank. Any entry in the **CMPNTS** field will be ignored if a **TRMTYP** is specified.

9. Allowable options for **LABELi** for symmetric trim (the symmetry option is selected in Solution Control) are:

   Structural Accelerations

NX	Longitudinal load factor (acceleration) (Remark 14)
NZ	Vertical load factor (acceleration) (Remark 14)
QACCEL	Pitch acceleration (Remark 15)

   Aerodynamic Parameters

ALPHA	Angle of attack in degrees
QRATE	Pitch rate (Remark 16)
THKCAM	Thickness and camber (Remark 17)
and	Control surface position in degrees

10. Allowable options for **LABELi** for antisymmetric trim (the symmetry option is selected in Solution Control) are:

    Structural Accelerations

NY	Lateral load factor (acceleration) (Remark 14)
PACCEL	Roll acceleration (Remark 15)
RACCEL	Yaw acceleration (Remark 15)

    Aerodynamic Parameters

BETA	Side-slip angle in degrees
PRATE	Roll rate (Remark 16)
RRATE	Yaw rate (Remark 16)
and	Antisymmetric control surfaces in degrees

11. Allowable options for **LABELi** for asymmetric trim (the symmetry option is selected in Solution Control) are any of the options listed above for symmetric and antisymmetric trim. It is up to the user to specify a solvable trim problem.

12. If **VALi** is a real number, the associated aerodynamic parameter of structural acceleration is set to that value. If **VALi** is the character string **FREE**, then the associated parameter will be determined by the trim analysis. If **VALi** contains the character string **SCHD**, then a **SCHEDULE** entry with the same **TRIMID** and **SURFLAB** = **LABELi** will be used to determine the value of the associated parameter.

13. The number of **FREE** values of **VALi** must correspond exactly to the number of unknowns in the trim analysis. If **TRMTYP** is blank, the number of specified DOFs in the **CMPNTS** field.

14. For **NX**, **NY**, and **NZ**, units are length per second per second in consistent units unless a **CONVERT/MASS** Bulk Data entry is provided. In this case, the values are dimensionless.

15. The angular accelerations, **PACCEL**, **QACCEL**, and **RACCEL**, are entered in units of radians per second per second.

16. **PRATE**, **QRATE**, and **RRATE** are entered in units of radians per second. The true velocity, **V0**, must be input if any of the "rate" parameters are given since its value is needed to dimensionalize the forces computed for a unit rate per velocity in the aerodynamic preface.

17. The **THKCAM** label refers to thickness and camber effects and its corresponding value is usually set to 1.0. Non-unit values of the **THKCAM** parameter are available only to provide added generality.

18. Any control surfaces trim parameters, or structural accelerations not specified on the **TRIM** entry will not participate in the analysis; they will be given fixed values of 0.0. This includes **THKCAM**.

19. Refer to the **STATIC AEROELASTIC TRIM** Application Note for more information.

THIS FEATURE IS NOT IMPLEMENTED.

Input Data Entry: **SLPARM** User Defined Static Load Control Parameter

Description: Defines linear load combination of Static Load components as a control parameter for aircraft trim.

Format and Example(s):

1	2	3	4	5	6	7	8	9	10
SLPARM	PARAM1	VAL1	PARAM2	VAL2	PARAM3	VAL3	SYM	PARMTYP	CONT
CONT	SCALE	LOAD							CONT

SLPARM	SURF1	1.5					ASYM	MECH	+123
+123	1.0	101	-1.5	102					

Field	Contents
PARAM1	Label of user defined controller for first level trim parameter (Text)
VAL1	Value of first level trim parameter setting (Real)
PARAM2	Label of user defined controller for second level trim parameter (Text)
VAL2	Value of second level trim parameter setting (Real)
PARAM3	Label of user defined controller for third level trim parameter (Text)
VAL3	Value of third level trim parameter setting (Real)
SYM	Trim parameter symmetry (SYM, ANTI, or ASYM) (Text)
PARMTYP	Load type (MECH, GRAV, THRM) (Text)
SCALE	Scale on load set LOAD (Real)
LOAD	Load set ID of FORCE, MOMENT, FORCE1, MOMENT1, PLOAD, GRAV, TEMP, and TEMPD load type (Integer > 0)

Remarks:

1. The user defined control parameter load will be integrated as a linear combination of the component loads.
2. Scalar combination of component loads are only valid on common load types (LODTYP).
3. The user defined control parameter load is assembled into the RIGDSLOD group in preparation for the assembly of the AIRFORCE matrix during the boundary condition loop for aeroelastic analysis.
4. The routine UDEFMRG combines the RIGDSLOD parameters into the AIRFORCE matrix through direction from Model assembly commands. Also, a blank column is designated in the AEROLOAD matrix for each SLPARM. Finally, the TRIMTOC relation is updated with the SLPARM controller name and values.

THIS PAGE INTENTIONALLY LEFT BLANK

6. QUADPAN MODEL GEOMETRY

6.1 INTRODUCTION

This chapter describes the basic concepts required for modeling configurations with QUADPAN. The geometric requirements, definitions, and conventions pertaining to configurations, panels, and elements are discussed first. This introductory material ignores the many options available for geometry generation. The initial sections deal only with the "developed" lattice, which may be input directly or may be defined using the options available in the input package.

Finally, the capabilities available as input options to enhance the geometry definition process are described, including transformations that may be applied to input data, alternate input coordinate systems and panel respacing options. The actual contents of QUADPAN data sets are detailed in Chapter 7.

6.2 OVERVIEW OF MODEL DEFINITION

QUADPAN calculates potential flow by representing the surface of the configuration with an essentially continuous lattice of quadrilateral elements. The task of developing a geometric model begins with a full surface definition of the configuration and consists of generating a lattice of elements which represents this surface and meets the requirements imposed by the numerical formulation (no gaps, no highly skewed or twisted elements, etc.). In principle, the user could manually construct a grid on the surface and input the lattice corner points to the program. When none of the input options are employed, the input to the program defaults to exactly this form. In practice, this method of producing a geometric model is unnecessarily time consuming because the number of elements in a typical model is large and describing the surface with the element lattice directly is not very efficient.

The input options exist to partially automate the task of producing a lattice from the typical engineering definitions of the surface of a configuration. The user inputs two kinds of information; a representation of the true surface of the configuration, and instructions on how this surface is to be subdivided to produce an acceptable lattice. Figure 6-1 shows the surface defined by the user and the lattice which the program generates from the lattice spacing instructions. Only when the actual element lattice is input directly is the second kind of input information not required.

The basic building block of the geometric model is the PANEL. Panels and the element lattice which the program generates from them are illustrated in Figures 6-1 and 6-2. A panel is an arbitrary curved surface with four, generally curved, edges. The geometric shape of the panel is specified by the user with several curves called SECTIONS which are defined by points on the configuration surface. The program develops each section by interpolating curves through the user-specified points. The program then develops the surface of the configuration by interpolating between the section curves. Aside from the points which define the end points of the panel edges, the user-specified points need not be related to the location of the lattice points which will ultimately be produced, as shown in Figure 6-1. Generally, the points used to define the panel are points which are convenient in terms of the geometrical data at the user's disposal. Each panel is then automatically subdivided into a lattice of quadrilateral elements whose corner points lie on the surface of the panel, and whose number and distribution is specified by the user.

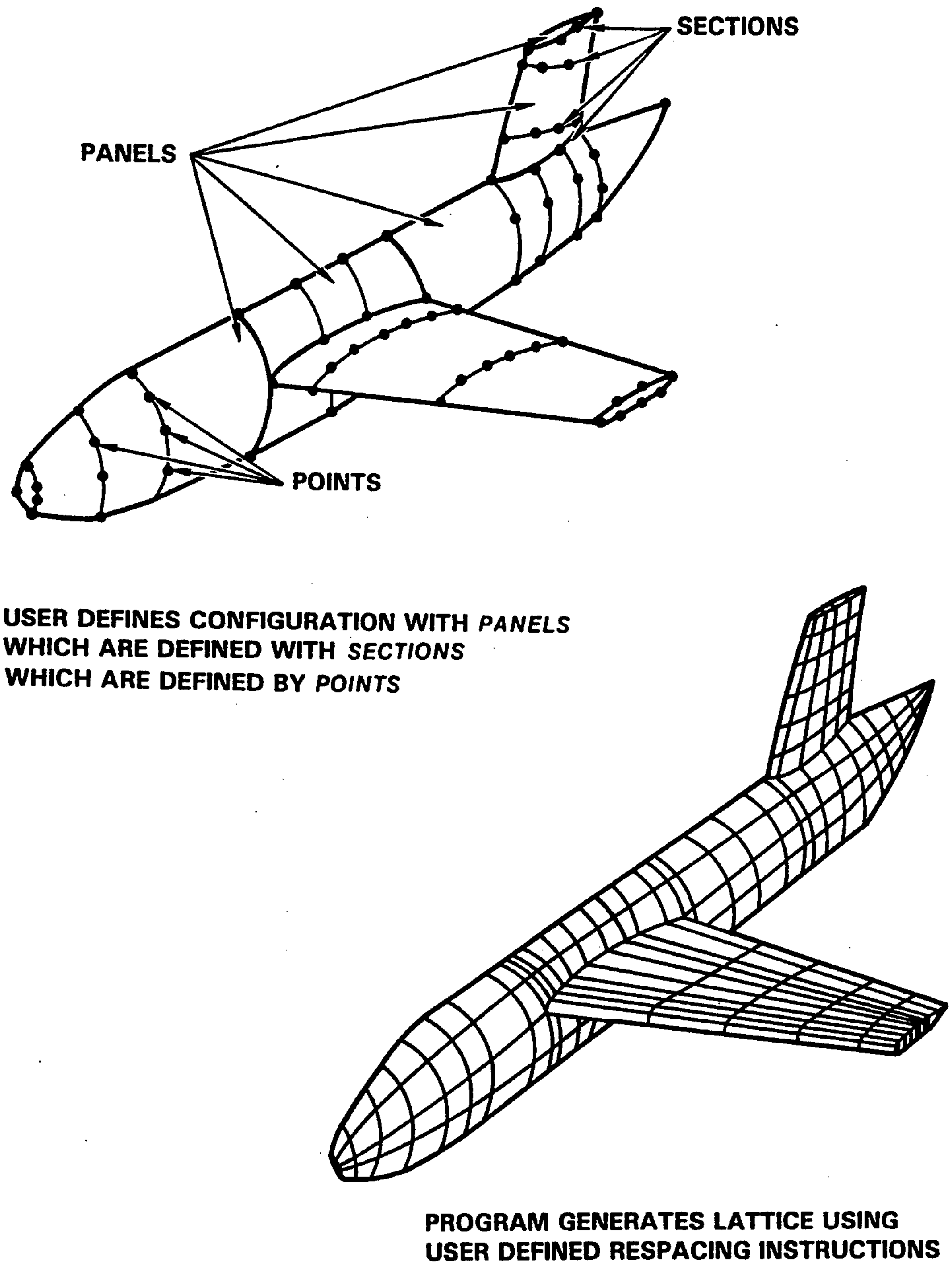


Figure 6-1 Generation of Mesh From User Defined Geometry

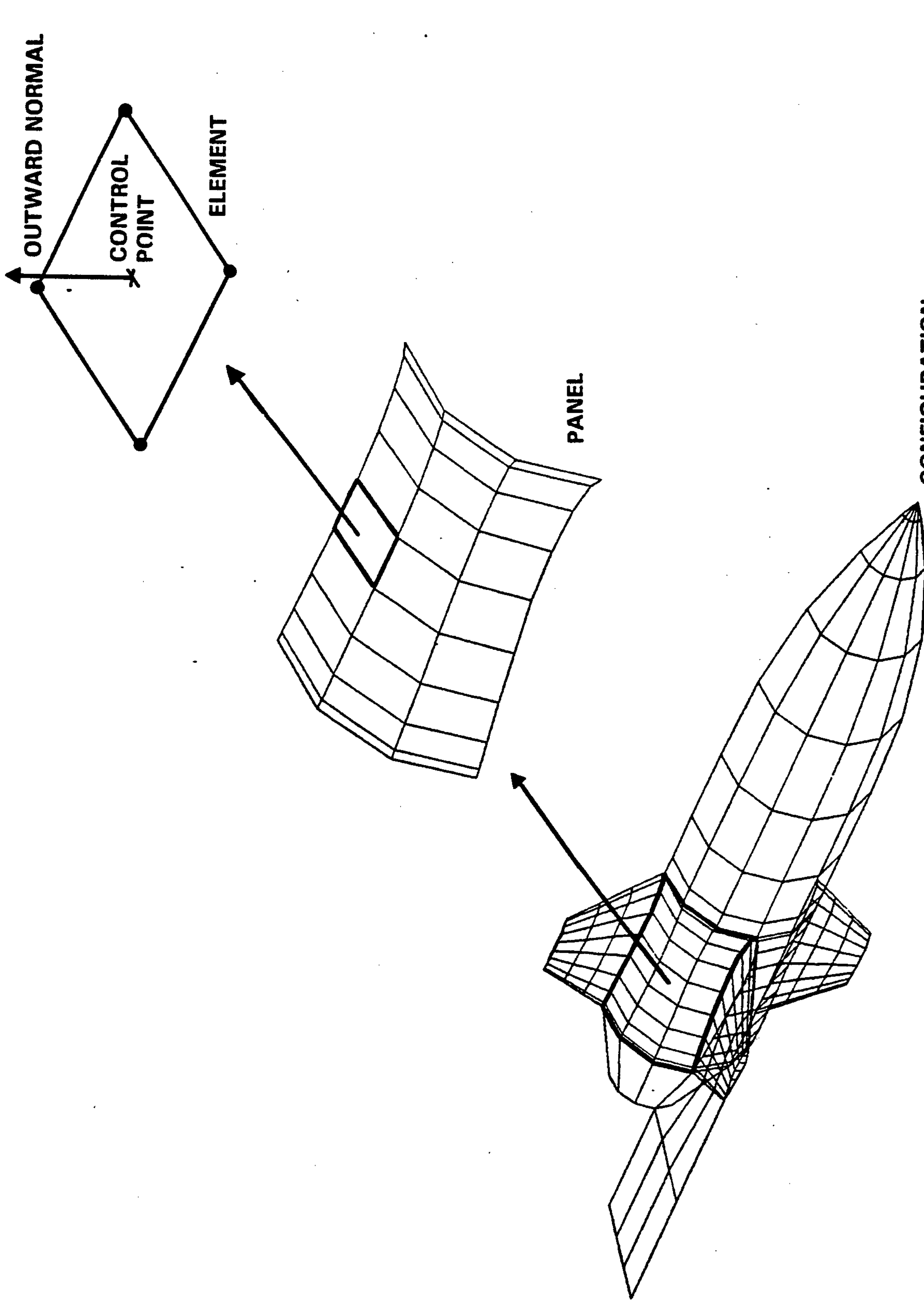


Figure 6-2 Configuration, Panel and Element Hierarchy

The problem of generating a lattice composed of many quadrilateral elements is simplified, by using panels and respacing options, to the problem of specifying the shape of a few arbitrarily curved panels, and specifying how each panel is to be divided into a lattice of elements. Furthermore, the geometry definition points and lattice points need not be identical, so the user can define the geometry at points where geometrical data is available, while establishing an element lattice which meets the computational requirements.

6.3 QUADPAN LATTICE

The characteristics of QUADPAN which must be considered in the generation of a suitable computational mesh are summarized in the following list. Additional details on each characteristic can be found in the Ref. 4.

- The mesh must represent one or more closed bodies in which only one side is exposed to the flow.
- If the configuration is symmetric about the X-Z plane the mesh should be defined by the user for only one side.
- It is desirable that the element mesh be continuous; that is, each element should touch only one neighbor on each side. Under special circumstances this can be relaxed.
- Elements in the mesh should not be highly kinked or twisted.
- The best accuracy for a given number of elements (cost) is obtained when elements are concentrated in areas of high curvature or any other area where large velocity gradients are expected.
- The user must define lattice elements which represent a vortex wake for each surface which develops lift.
- The upper and lower surfaces which shed a wake must have similar element shape and size near the shedding edge.
- The side edges of the wake mesh should be continuous with the mesh on the body (in fact it must join the body at a panel boundary) for correct calculation of the pressure on the body.

6.4 REPRESENTATION OF SURFACES WITH PANELS OF ELEMENTS

The method used to define input geometries for QUADPAN preserves most of the flexibility of element by element input and adds a number of useful options that can dramatically reduce the amount of data required to describe the geometry. The configuration surface is represented with one or more PANELS of elements, where each panel is subdivided into a "rectangular" lattice of elements. Figure 6-2 graphically illustrates the relationship between configurations, panels and elements.

6.4.1 Division of Configuration Into Panels

Simple shapes such as a straight tapered wing or a cylindrical fuselage forebody may be well defined by one panel, while more complex shapes such as the finned body in Figure 6-3 may require a number of panels to represent its surface details adequately. QUADPAN places no specific geometric restrictions on panels as far as being wing type panels or fuselage type panels, instead all panels are treated uniformly as pieces of surface geometry whose particular boundary condition (body or wake) is specified separately. This boundary condition is applied to all of the elements that make up the panel, necessitating the division of the configuration into panels. A wing and its wake, for example, must be separate panels due to the different boundary conditions required for each. The division of the configuration surface into panels is done primarily as a means of organizing the input to and output from the program.

6.4.2 Continuity Of Panel Mesh

Panels normally abut (touch) one another at their edges to form a continuous surface mesh. This is possible when the user has specified the geometry in such a way that the elements on both edges of an abutment are contiguous. It is not necessary that panels containing the elements be contiguous to have contiguous elements. See Figure 6-4 for example of contiguous elements and panels. The program uses an automatic Abutment Search procedure (8.4) to establish the panel abutments and element neighbors. This procedure will find all contiguous abutments in the configuration, relieving the user of the burden of specifying connecting panel edges and elements in all but a few extreme cases.

A geometric model with the panels connected together at their edges to form a surface mesh of contiguous elements will generally give the most accurate results. In part, this is due to the representation of the edge curves of the panel with straight line segments. Gaps between such curved edges may only be avoided when the edges have matching elements. Continuity of the surface mesh is not strictly necessary, and in some cases may not be desirable. Mesh continuity is necessary, however, at the intersections of vortex wakes and bodies. Examples are discussed in Ref. 4 on Modeling Techniques which give additional insight into where contiguous and noncontiguous elements should not be used.

6.5 PANEL AND LATTICE NOMENCLATURE

A configuration to be analyzed with QUADPAN is represented by one or more panels of elements, as required to describe the surface shape and represent the various boundary conditions to be used (body), wake, etc.). Each panel defined is logically distinct, i.e., its geometric definition is independent of all other panels. This section will discuss the panel in its "developed" sense, as defined by its lattice of grid points after geometry generation is finished.

6.5.1 Panel And Image Panel Identification

Every panel has a user-assigned panel identification number and a panel title associated with it. This number and title are used to refer to that panel in the input or output. The image of that panel,

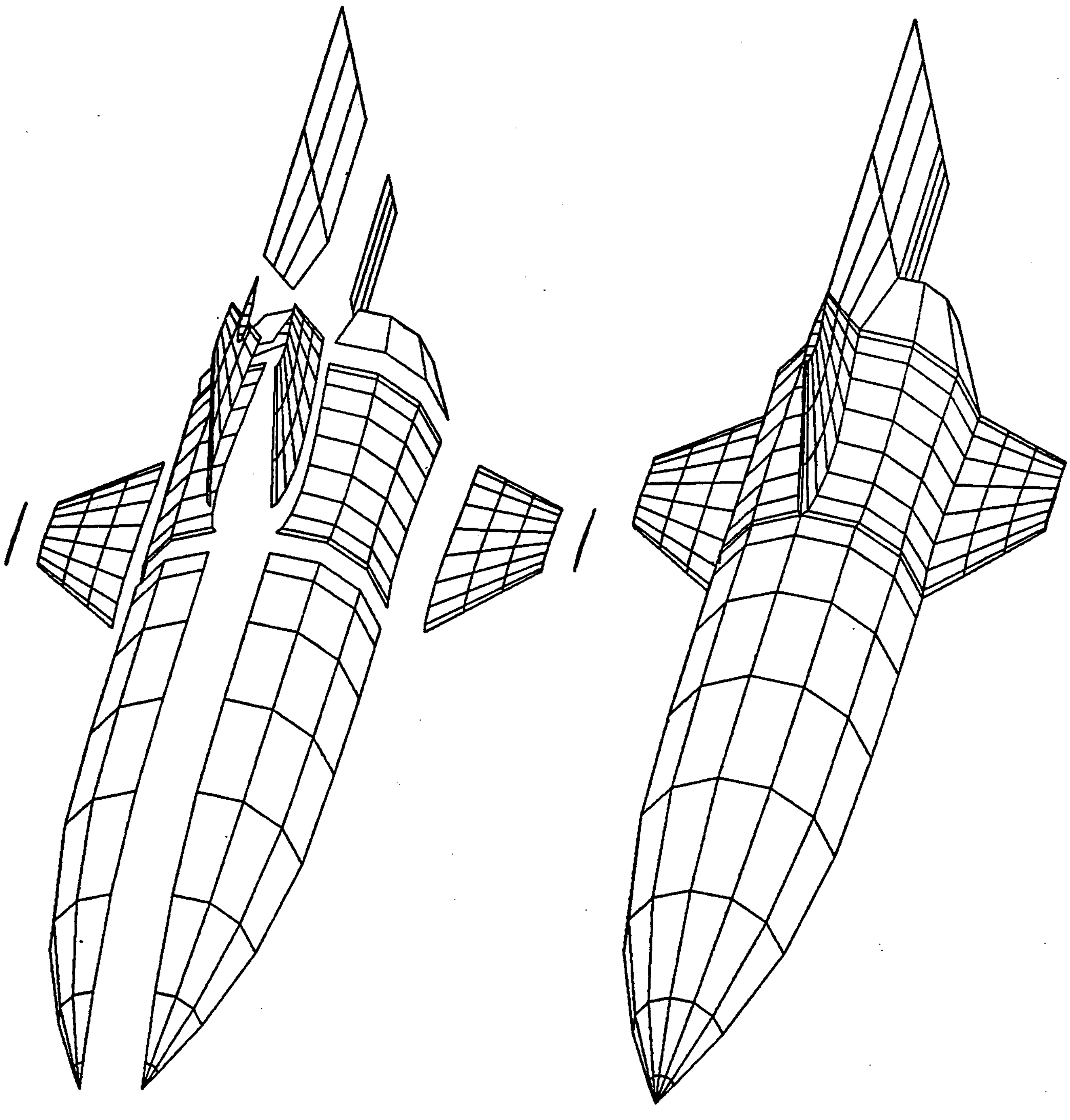


Figure 6-3 Division of a Configuration Into Panels

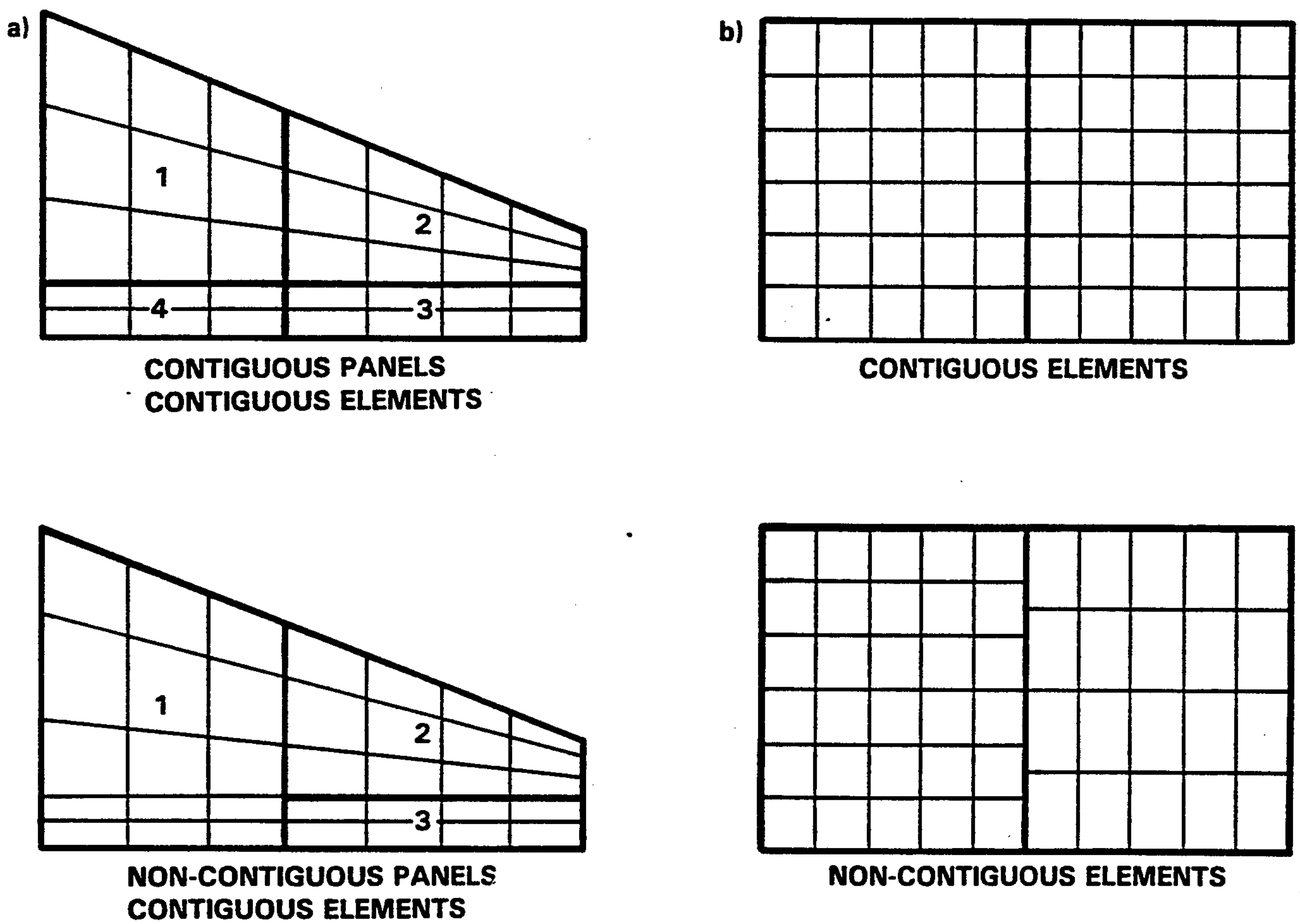


Figure 6-4 Contiguous Panels and Elements

if it possesses an image, lies across the X-Z plane of symmetry and is referred to using the same title and the negative of the identification number. This user defined ID number allows an input specification referring to a given panel (to connect panel edges, for example) to remain independent of the number and order of the input panel definitions.

6.5.2 Panel Structure

The program treats each panel as a rectangular lattice with four distinct sides (or edges) regardless of the actual shape. The "rectangular" array of points, with a constant number of points in each row and each column, defines a rectangular grid of quadrilateral elements that constitute the panel. Although one panel edge or two non-adjoining edges may be of zero length, the degenerate edge(s) are still defined in the same manner as any other edge.

6.5.3 Panel Lattice

The "rectangular" panel lattice is defined by a row and column array of corner points. This is illustrated in Figure 6-5. Two or more columns of points, called SECTIONS (from cross section), are required to define a lattice, with each section being a line on the panel surface defined by two or more points (i.e., there must be at least two rows of points). The rows and columns of the lattice define two indicial directions within the panel, the K direction corresponding to the columns, and the J direction corresponding to the rows.

In order that a proper lattice exists, three conditions must be met:

1. The sections must not cross one another.

2. The points defining each section must be entered in a consistent direction.

3. Every section within a "developed" panel lattice must consist of an equal number of points.

6.5.4 Panel Edges And Corner Points

The panel's four edges are referred to as edge 1 through edge 4 and are determined by the order in which the panel information is entered. Edge 1 is always the first section defined within a panel, while edge 3 is always the last section. Edge 4 is the line formed by the first points of each section, and edge 2 is the line formed by the last points of each section. Panel corner points are defined such that corner points 1 and 2 lie at the ends of the first section, and corner points 4 and 3 lie at the ends of the last section (see Figure 6-6).

The points that form the panel lattice are numbered sequentially in column order, from the first point in the first column of the first panel defined, to the last point in the last column of the last panel (see Figure 6-5). The user does not have to deal directly with lattice points, however, except in the context of the corner points associated with elements.

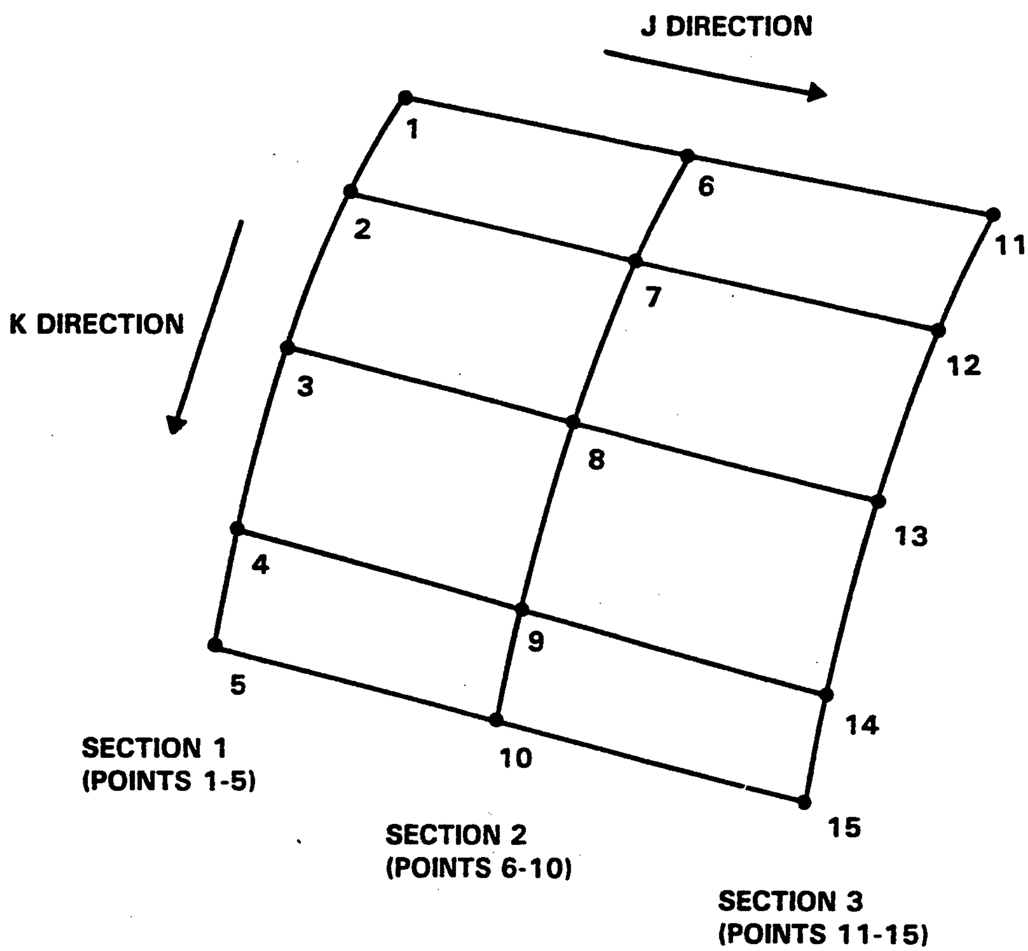


Figure 6-5 Panel Layout with Section and Point Numbering

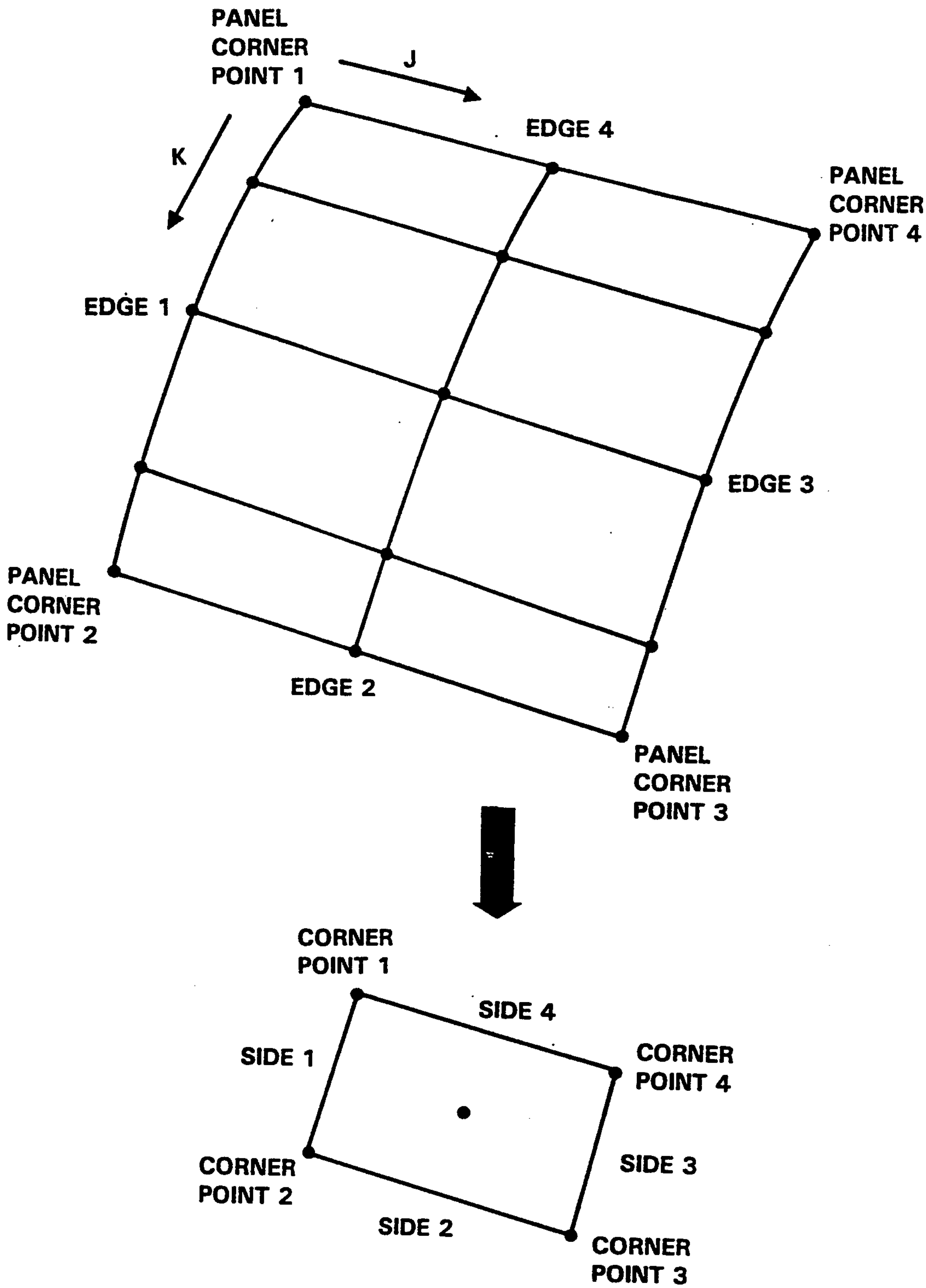


Figure 6-6 Panel and Element Edge Numbering

6.5.5 Panel Element Array

The panel's rectangular lattice of points form the corner points of a rectangular array of quadrilateral ELEMENTS. Elements take the same side and corner point conventions as their parent panel (see Figure 6-6). The two indicial directions, K and J, are local to the panel and serve to locate an element within the regular row and column structure of the panel lattice. The order in which the section points are input defines the K indicial direction, and the order in which the sections are entered defines the J indicial direction. In this indicial notation edge 1 is the locus of J=1, edge 2 is the locus of K=kmax, edge 3 is the locus of J=jmax, and edge 4 is the locus of K=1. In addition to the local K,J indexing within the panel, QUADPAN numbers the elements sequentially from the first element in the first panel entered, to the last element in the last panel. Sequential element numbering, like point numbering, always proceeds in the K direction starting from J=1 (edge 1) to J=jmax (edge 3. See Figure 6-7.

6.5.6 "Positive and Negative" Panel Surfaces

The panel's "positive" surface is determined by the order of point and section input in the same way as the panel's edges and the K and J directions. The convention adopted in QUADPAN is that the "positive" surface is given by the cross product of a vector in the "K direction" with a vector in the "J direction." This is illustrated in Figure 6-7. This may be alternatively stated as the direction given by the right-hand rule as the edges of the panel are taken in cyclic numerical (12341234...) order.

The exterior surface of a panel is the side of the panel that the program treats as wetted by the flow. This will be the direction of the outward normal vectors from the elements. The program can use either the panel "positive" or "negative" surface as the exterior, depending on the setting of the WET flag. If the panel "positive" surface (as defined above) is to be the exterior surface, the WET flag should be set to +1. If the panel "negative" surface is to be the exterior surface, the WET flag should be set to -1. Note that the action of the WET flag is to reverse the direction of the outward normal vector without reordering the input points. This is particularly handy in the event of a mistake in the order of a panel definition.

6.6 LIMITATIONS AND RULES ON PANELS

As discussed above, each panel is regarded by the program as topologically rectangular, regardless of the actual geometrical details. The lattice of points, with a constant number of points in each row and each column, defines a rectangular grid of quadrilateral elements that make up the panel. The panel lattice defines four panel edges. The program places no major restrictions on what orientations the panel" four edges may take, even in panel abutments.

The following rules and limitations apply to panels:

1. One edge or two nonadjoining edges may be reduced to zero length.

2. Triangular elements are only permitted at panel edges. The number of elements in each row or column within the lattice must be constant. In addition, triangular elements are restricted to occur only at degenerate (zero length) panel edges. See Figure 6-8.

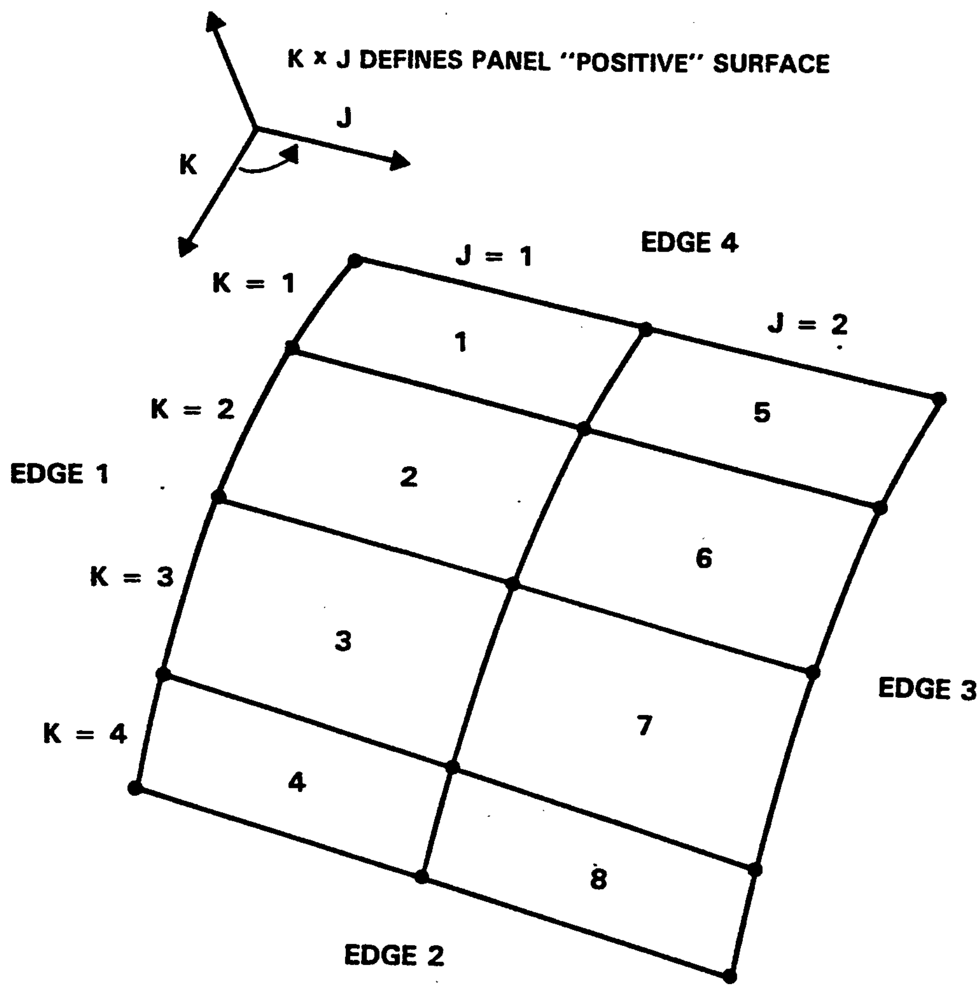


Figure 6-7 Panel Layout with Element Numbering

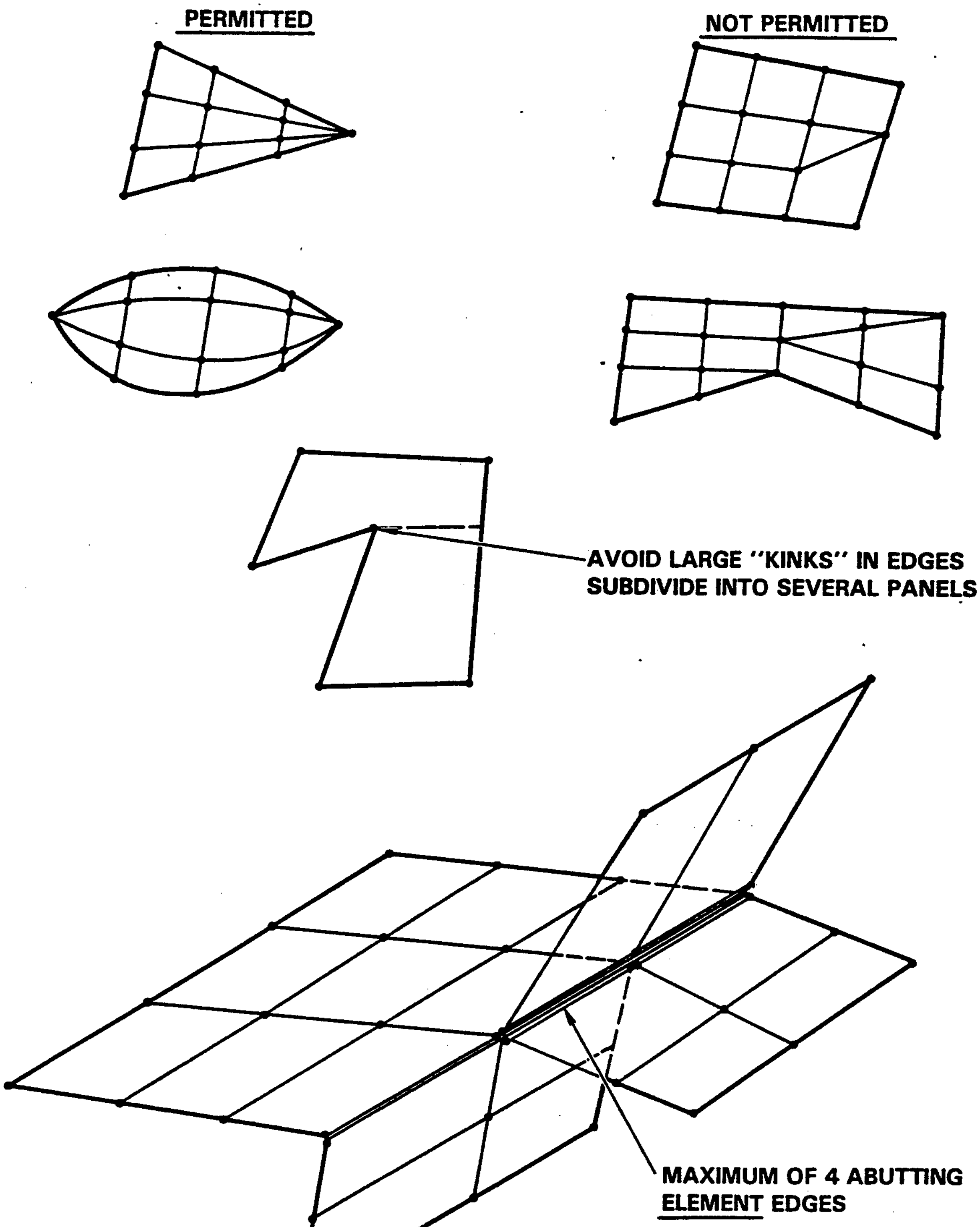


Figure 6-8 Limitations on Panels

3. The mesh of elements should be as orthogonal as possible. This does not mean that orthogonality of the panel mesh is required, but highly distorted meshes should be redefined. In addition, the edges of the panel should not include large "kink" angles that distort the element lattice structure. An edge kink angle of 90 degrees or more should be avoided by subdividing the panel into several panels. See Figure 6-8.

4. Panels must themselves be simply connected, that is, they may not contain any holes. The elements within a panel are assumed to form an unbroken mesh.

5. Panels may only (only!!) abut (touch) one another at their edges.

Any number of panels may abut a panel edge, however, no more than four elements may abut (touch) one another along a common element edge. See Figure 6-8.

6.7 OVERVIEW OF OPTIONS FOR GEOMETRY GENERATION

The discussion of panels thus far has dealt with the "developed" panel lattice, defined by several sections (columns) of points, with each section consisting of the same number of points (rows). The points defining this lattice may be input directly, or the user may take advantage of several options that are available to simplify geometry generation or modification. These include:

1. **Panel transformation** - This allows the user to translate, rotate, and scale the entire array of panel corner points. The panel may be described in its own coordinate system and then transformed into the global system.

2. **Options to define, repeat, or transform sections** - These simplify the panel definition process by allowing similar sections to be defined once in their own coordinate system and then transformed in to the panel system.

3. **Panel respacing** - This option permits the element density or distribution on the panel to be altered. This can be done on a panel basis, or individually for each section of points. This is a very powerful capability that simplifies the geometry definition and modification process.

This capability is available through the use of optional KEYWORDS in the input data set. If the keywords are not present in the input data set, they are simply defaulted out and nothing is scaled, transformed, respaced, etc. See Chap. 7 for more information on keywords.

6.8 PANEL TRANSFORMATIONS

The user may specify a transformation to be applied to the panel so that it may be described in its own coordinate system, the panel coordinate system (PCS). This transformation is defined by three points whose location is given in both the panel coordinate system and the global coordinate system to define the transformation (see Figure 6-9). Combined translation, isomorphic scaling and rotation can be specified with these three points. This option is available through the use of the keyword LOCATE (0).

The locater points may be any three points which are not colinear, i.e., the three points form a triangle. These points need not lie on the panel, or correspond to any of the lattice points. The action

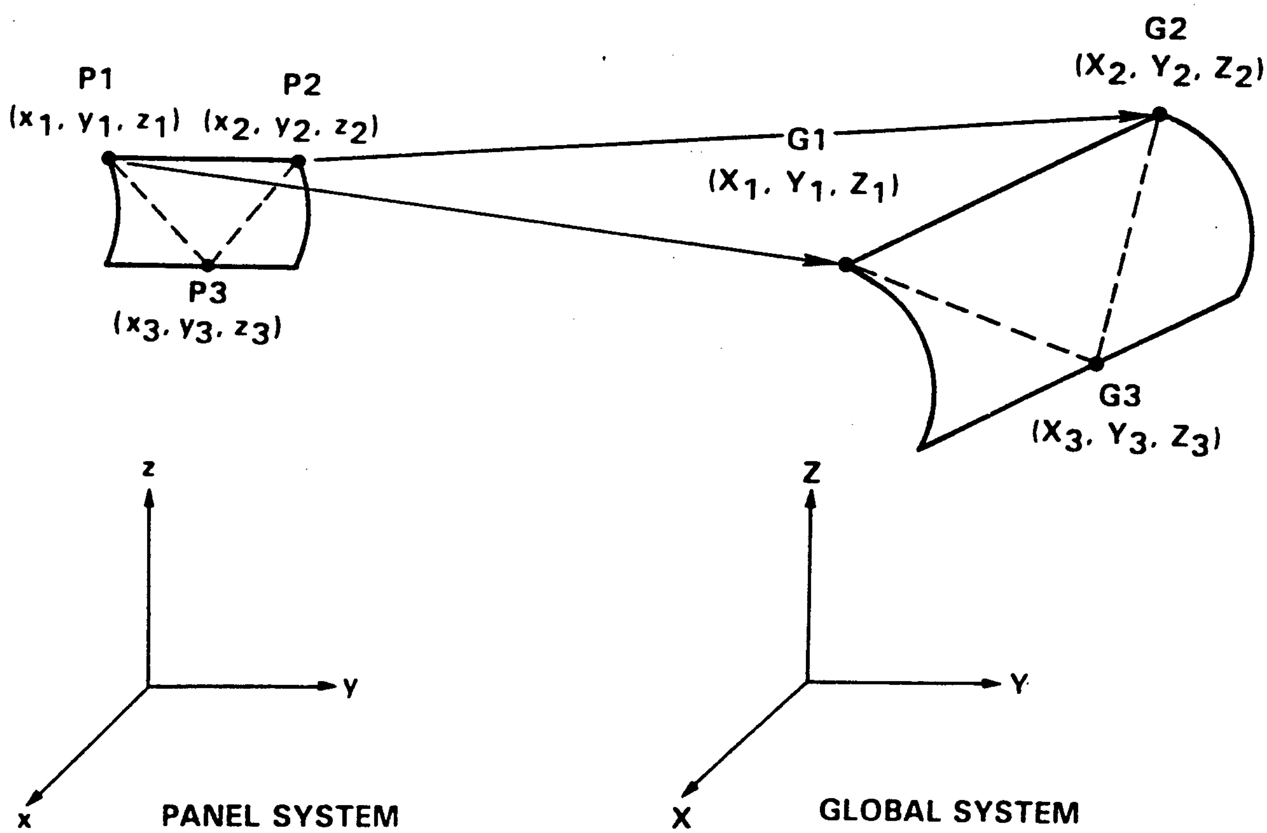


Figure 6-9 Action of Locater Points

of the locater points is most easily understood if the triangle of locater points in panel coordinates is similar to the triangle of locater points in global coordinates, but this is not a requirement. The panel size is scaled (without changing its shape) by the distance between the first and second locater points in the two coordinate systems. The panel is placed so that the first and second panel locater points coincide with the first and second global locater points, and the panel triangle is coplanar with the global triangle.

6.9 DEFINING PANELS WITH SECTIONS

As discussed previously in this chapter, each panel is defined by two or more SECTIONs (columns) of points. These sections can often be treated as cross sections, such as on a wing, where the sections could correspond to the airfoil coordinate definitions, or the cross sections defining a fuselage.

In addition to the basic coordinates of the section points, the user may specify transformation data to translate, scale, and rotate the input points. This permits the sections to be defined in their own coordinate systems, the Section Coordinate System (SCS), and transformed into the panel system. Respacing data may also be specified to alter the distribution of lattice points, both along the section curves defined by the input points and the J direction between sections. The use of the transformation or respacing capability is entirely optional and, if not used, the input points define a column of lattice points in the K direction of the panel.

There are two coordinate systems available for defining sections - a rectangular system and a cylindrical system. Both rectangular and cylindrical sections may be mixed within a panel definition.

6.9.1 Rectangular Sections

The most commonly used method of entering panel data is with rectangular sections. A rectangular section is defined with input points specified in an orthogonal X, Y, Z coordinate system. See 7.8 for further information on rectangular sections.

6.9.2 Cylindrical Sections

A section may be defined using cylindrical coordinates (polar angle, radius, and polar axis coordinate). Either the X, Y, or axis may be used as the pair in the cylindrical section. Cylindrical sections have special properties if respacing is specified, as will be discussed in 0. See 7.8 for further information on cylindrical sections.

6.9.3 Repeated Sections

The previous section defined may be reused. This repeated section may be translated, scaled, and rotated to change the coordinates of the points.

6.9.4 Section Transformations

Each section definition consists of two parts, a geometric part made up of a list of points and an optional "thread point," specified by the THREAD keyword, and a transformation part in which translation, scaling, and rotation may be applied to the section points. The thread point may be thought of as a way to reorigin the section coordinates to the location of the thread point. The transformations that may be applied to the section points are illustrated in Figure 6-10 and consist of:

> **Translation of the section,** using the AT keyword, is done by moving the thread point to a specified point in the panel coordinate system (PCS).
>
> **Rotation of the section,** using the TWIST keyword, is done about an axis that passes through the thread point.
>
> **Scaling of the section,** using the SCALE keyword, is done about the thread point.

No transformation is done to the section points if the transformation information is omitted or defaulted. The thread point is treated as part of the section definition and is unchanged if the section is repeated. The thread point may also be omitted, in which case it defaults to the origin (0.,0.,0.) of the section coordinates.

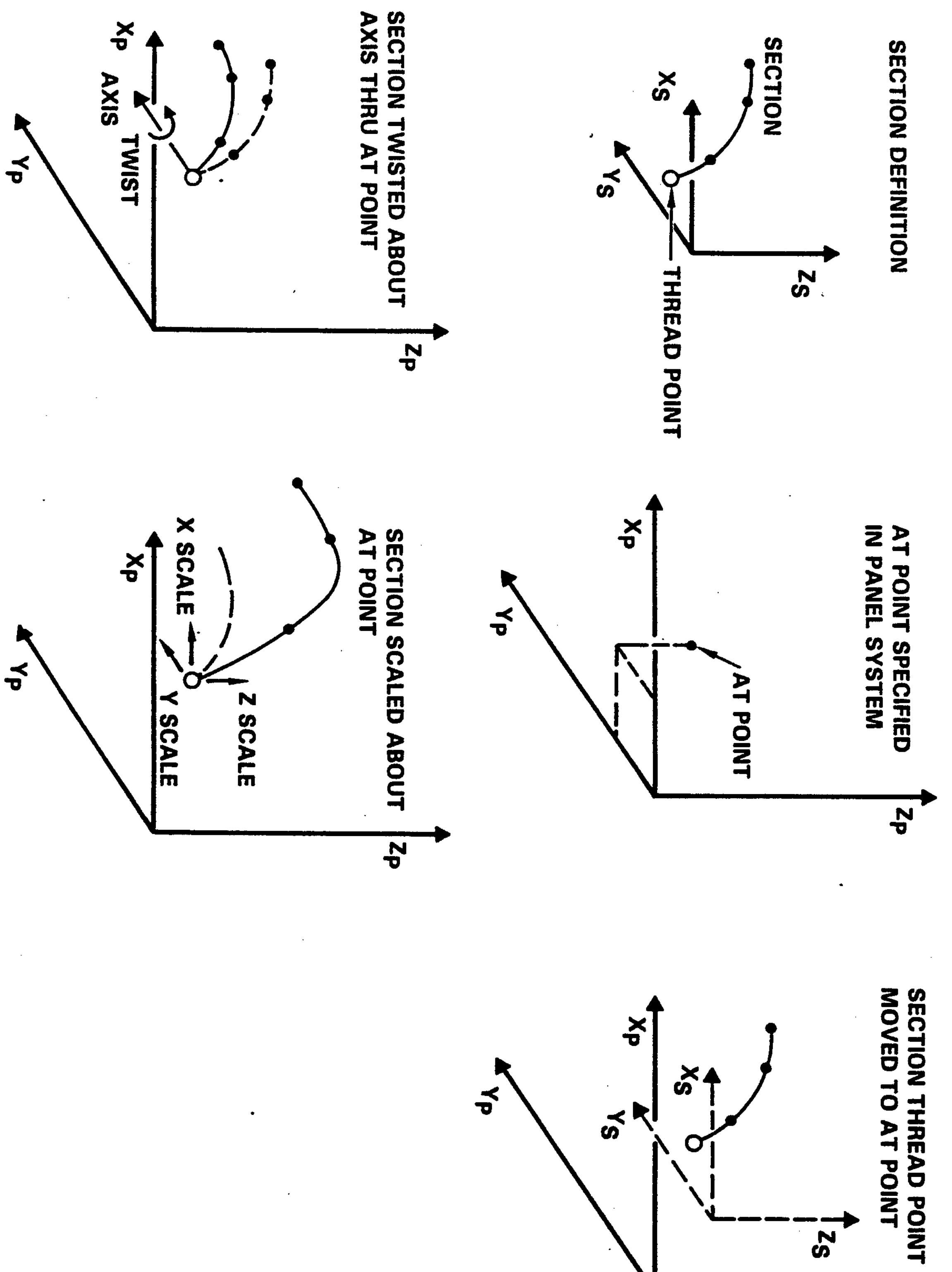


Figure 6-10 Section Transformations

6.10 PANEL RESPACING

The panel respacing options are powerful tools for geometry generation. They allow the number and distribution of the lattice lines (and elements) within a panel to be changed without changing any of the points defining the panel in the input data set. This redistribution of the panel lattice can be specified independently in either or both of the J and K panel directions.

The capability to respace a sparse set of input points to form a denser lattice can reduce the amount of input data required to describe a configuration. Only the number of points required to adequately describe the geometric shape need be input. The benefits of panel respacing are illustrated in Figure 6-1, where the input points and sections have been respaced into a dense, regular lattice. The example data sets in 7.11.2 and 7.11.3 also demonstrate the respacing capability. The first example is a simple rectangular wing which is described only by a root and tip airfoil sections. The intervening mesh is created by respacing between the sections. The second example is a fuselage type configuration, described using cylindrical cross sections, and respaced to create the panel lattice.

6.10.1 Representation Of Curves With Cubic Splines

The basis of the panel respacing options is the representation of the shape of a space curve defined by a set of input points. This is necessary because an additional (or different) set of points must be found (respaced) that lie on that curve.

The method used in QUADPAN to represent space curves is a locally fitted cubic spline which is parameterized by arc length along the curve. This method differs from the classical cubic spline by using only a few points in the vicinity of the desired point to fit the curve, and virtually eliminates the oscillatory behavior normally associated with cubic splines. The coordinates of the input curve are parameterized by arc length to make the representation of space curves independent of any specific axis.

Any number of points may be used to define the input curve, as required to characterize its shape. The more points used to describe a complex shape, the more accurately it will be represented. If only one point is used the spline simply degenerates to that point. If two points are used the spline degenerates to a straight line passing through the points. If three or more points are used the spline will produce a curved line passing through the points.

6.10.1.1 Break Points -

The splines used to represent space curves have the property that their derivatives are continuous everywhere on the curve. In some situations a discontinuous curve may be required. This can be done by designating a point (for respacing the K direction) or a section (for respacing in the J direction) to be a slope break. Matching of slopes will not be done across a slope break. See Figure 6-11. Note that specifying a break does not force a lattice point to be located at the break - this can be done with spacing intervals.

See 7.8 for more details on slope breaks (KBREAK or JBREAK) specified for the input points, or for the input section.

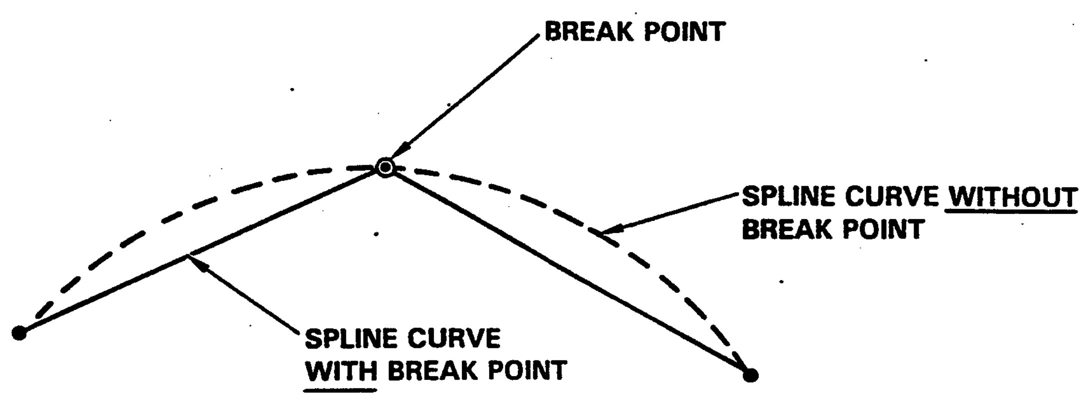


Figure 6-11 Effects of Break Points On Spline Curve

6.10.2 Respacing Along Spline Curve

After the input points have been represented with cubic splines, new points may be obtained along that curve by interpolation. The respacing of points along the curve is controlled by the number of elements (intervals between lattice points) and the distribution, or spacing, of elements that have been specified to lie on the respaced curve. Since the curves are parameterized by arc length along the curve, from the first to last point defined, the number of elements and spacing distribution are also applied to the arc length along the curve in a direction from first to last point.

See 7.7 and 7.7 for more information on the number of elements (NJ or NK) that have been specified to lie on the respaced curve.

6.10.2.1 Spacing Intervals -

The respacing of points along the curve is done within "spacing intervals." A spacing interval may be the entire curve between the first and last input points, as discussed above or may be any subset of the input curve. The same basic respacing operations are done in either case. Figure 6-12 illustrates a curve with two spacing intervals specified.

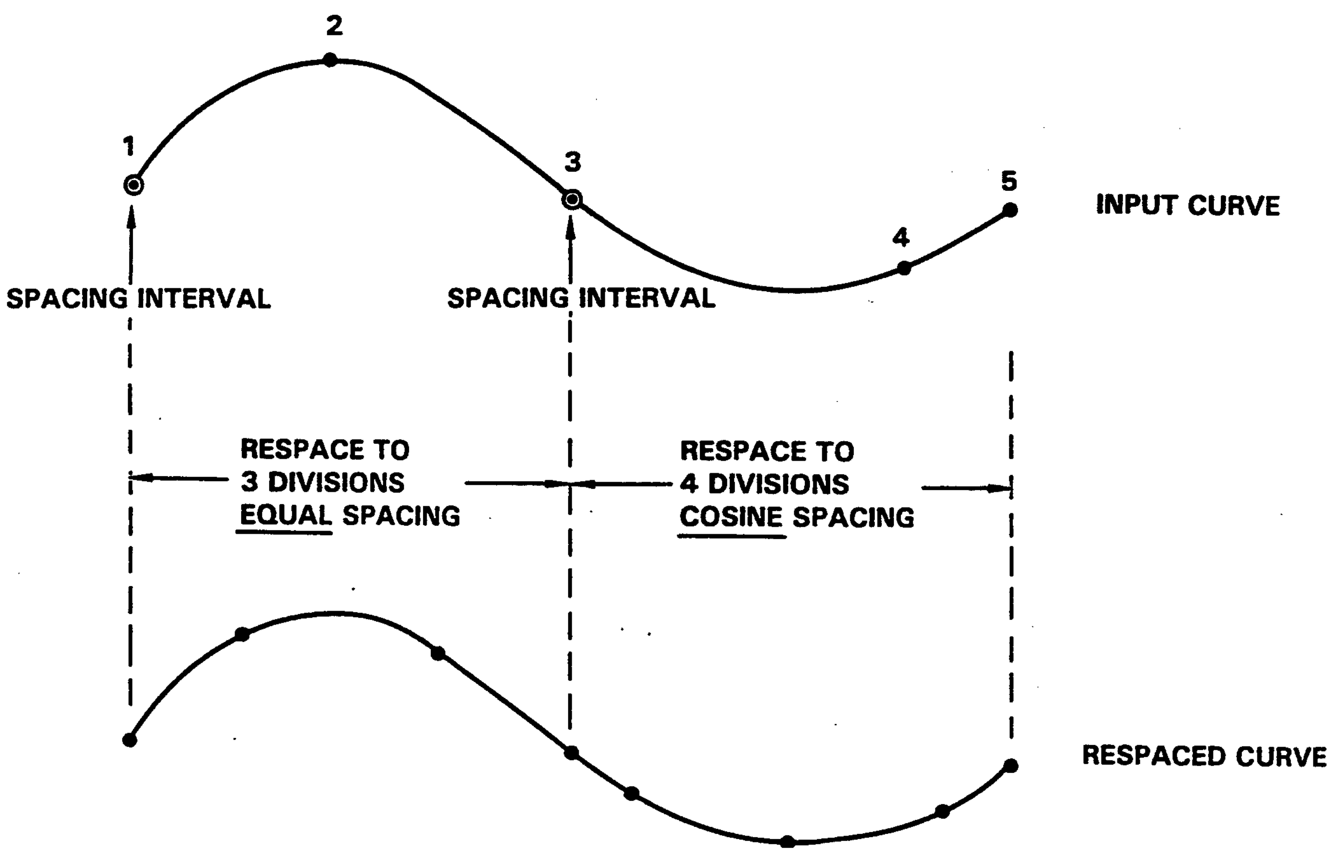


Figure 6-12 Use of Spacing Intervals and Respace Curve

The advantage of the spacing interval is that the ends of the interval are guaranteed to be lattice points. This allows a portion of a curve to be spaced in such a way that it matches a curve on another panel, to guarantee contiguous elements at the panel edges. Spacing intervals can be defined between input points in a section (K direction), or between sections (J direction). Each point (or section) except the last may define the beginning of a spacing interval. Slope continuity of the curve is maintained across spacing intervals so long as slope breaks are not used.

6.10.2.2 Spacing Distributions -

The distribution of intervals along the respaced curve is controlled by the desired spacing distribution. Three basic types of spacing distributions are used, as illustrated in Figure 6-13 and listed below:

Equal spacing - The input curve is divided into equal arc length intervals.

Cosine spacing - The input curve is divided using a spacing distribution that concentrates points at the ends of the input curve, as given by a cosine law.

Sine spacing - The input curve is divided using a spacing distribution that concentrates points a one end of the input curve, as given by a sine law. Either end of the input curve may be selected as the concentrated end.

In addition to the spacing distributions listed above, a "blended" spacing may be specified that combines the basic spacing distributions using linear blending. This is illustrated in Figure 6-14 where a ten interval spacing has been generated for the possible spacing distributions (SPACE=-3.0 to 3.0). A blended spacing distribution is given a line parallel to the spacing distribution lines (i.e., 1.2 spacing is on a line 1/5 of the way from the SPACE=1.0 line to the SPACE=2.0 line).

See 7.7 and 7.8 for more information on the spacing distribution of elements (JSPACE or KSPACE) that has been specified to lie on the respaced curve.

6.10.3 Panel Respacing Process

Panel respacing may be specified independently in either or both of the panel's J and K directions. The following sections detail the operations done on the panel lattice for each of these cases.

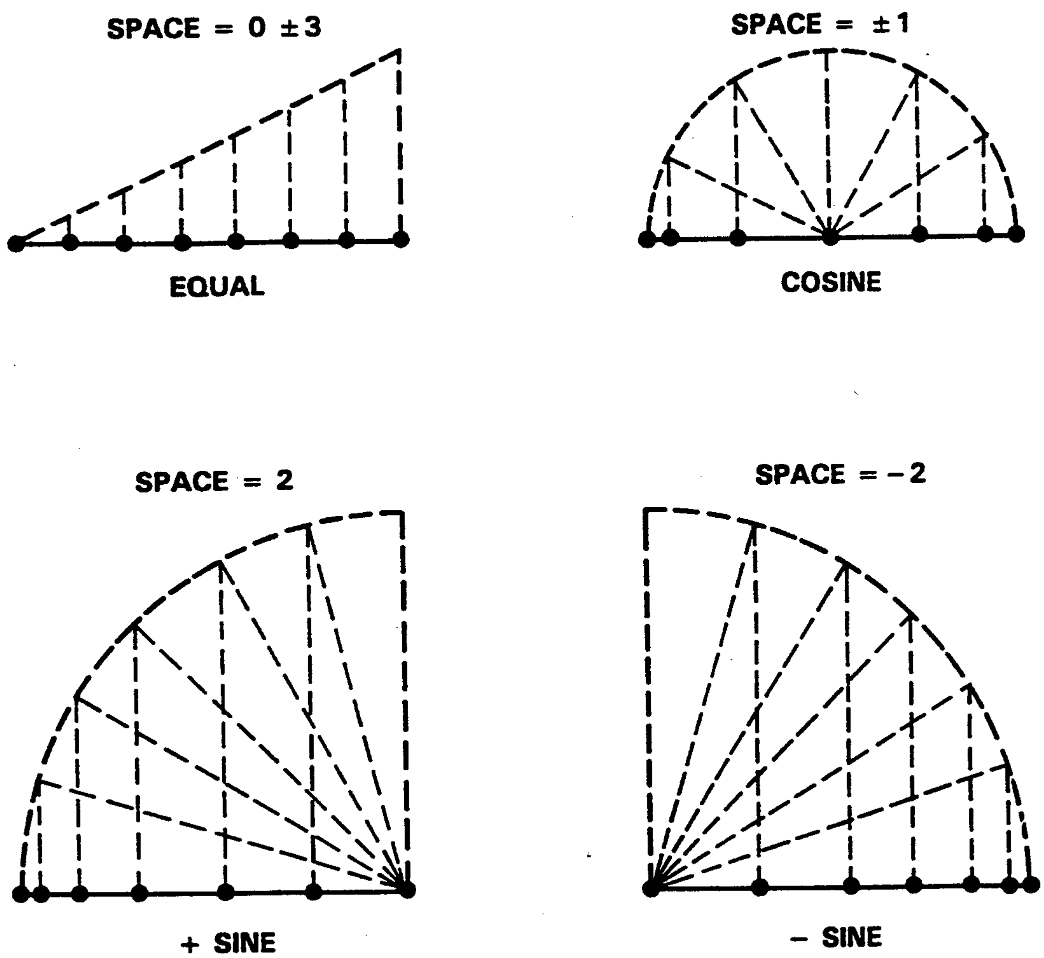


Figure 6-13 Basic Spacing Distributions

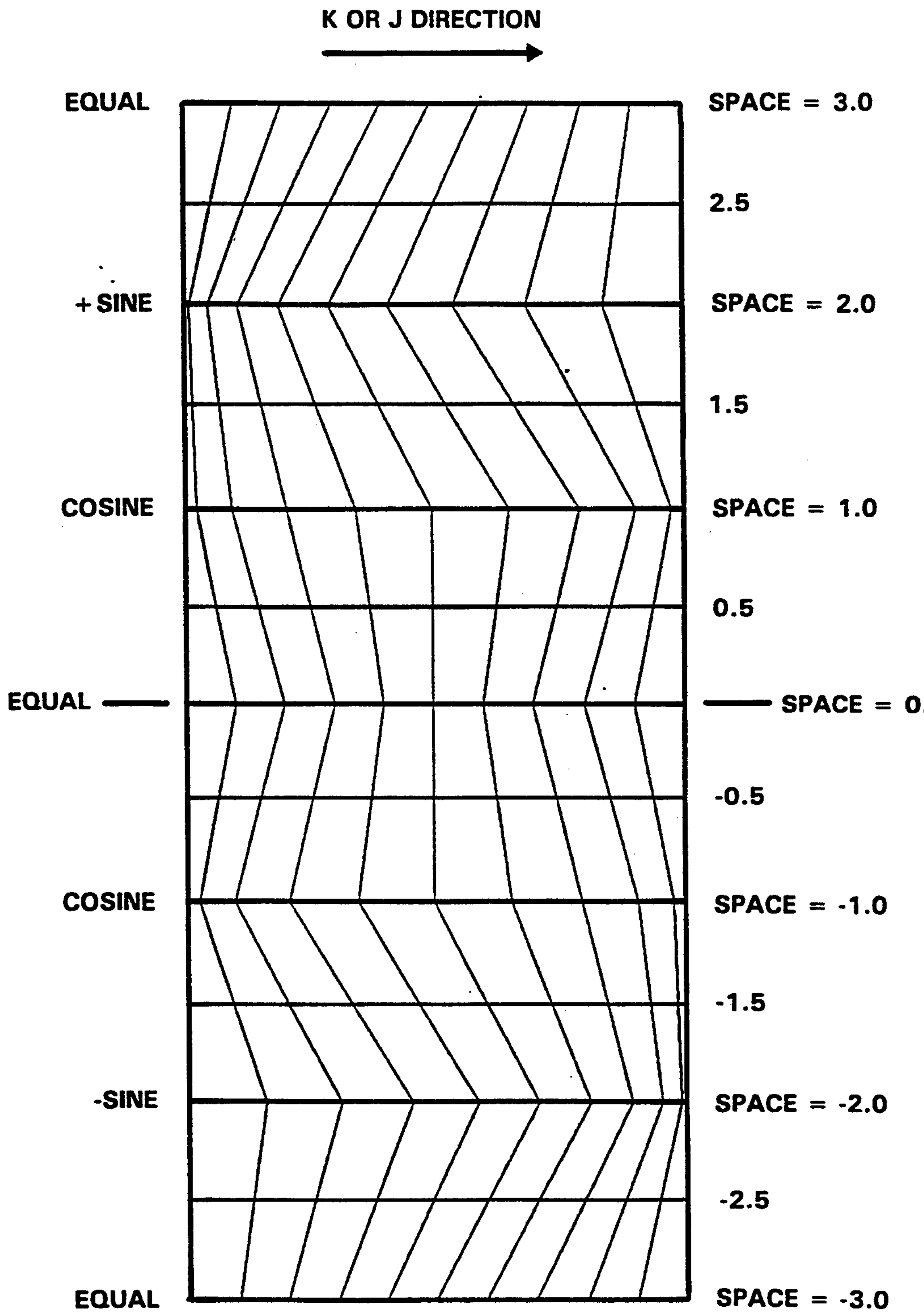


Figure 6-14 Blended Spacing Distributions

6.10.3.1 Respacing In The K Direction -

The panel lattice respacing in the K direction is done section by section, corresponding to columns in the panel lattice. The respacing information in the K direction may be specified for the panel as a whole (all the sections) or specified individually for each section.

The points defining each section are fitted with a spline curve that passes through the input points. The number of elements (NK) and the element spacing distribution (KSPACE) are used to interpolate new points on the spline curves. If input points in the section are designated as spacing intervals, the sum of the number of elements specified (for all the spacing intervals) within each section must be the same for all the sections in the panel.

Figure 6-15 illustrates these operations on a panel defined with three sections. The first section is a curve defined with four points, the second section is a line defined by two points, and the third section is a curve defined by six points. Note that a lattice could not be constructed from these sections without respacing, owing to different numbers of points, within the sections. For this example respacing is specified over the whole K extent of the panel with four elements (five points) in a cosine type distribution. Once the respaced points have been generated the lattice is well defined, with the required number and distribution of elements.

6.10.3.2 Respacing In The J Direction -

The panel lattice respacing in the J direction is done using spline curves connecting across the sections, corresponding to the rows in the panel lattice. The respacing information in the J direction may be specified only for the panel as a whole (across all the rows). This differs from panel respacing in the K direction, where each section (column) may be individually respaced.

A spline curve is fitted across the points of equal K index from each section, i.e., all the first points in the sections, all the second points in the sections, etc.. The number of elements (NJ) and the element spacing distribution (JSPACE) are used to interpolate new points on the spline curves.

Figure 6-16 illustrates these operations on a panel defined with three sections. Note that each section consists of the same number of points. This is a requirement for respacing in the J direction. For this example respacing is specified over the whole J extent of the panel with four elements (five points) in a cosine type distribution. Since the interior section was not designated a spacing interval, there is no longer a straight line column of points in the panel.

6.10.3.3 Respacing In Both Directions -

If respacing in both directions is specified the operations used to respace the lattice are done by respacing in the K direction first, followed by respacing in the J direction. The figures for the previous two subsections, taken in sequence, illustrate this.

This method of respacing the panel lattice leads to a difference in the capability to control the panel lattice for the two respacing directions. Respacing information in the J direction can only be specified for all the rows of points, not individually for each row. Panel respacing in the K direction,

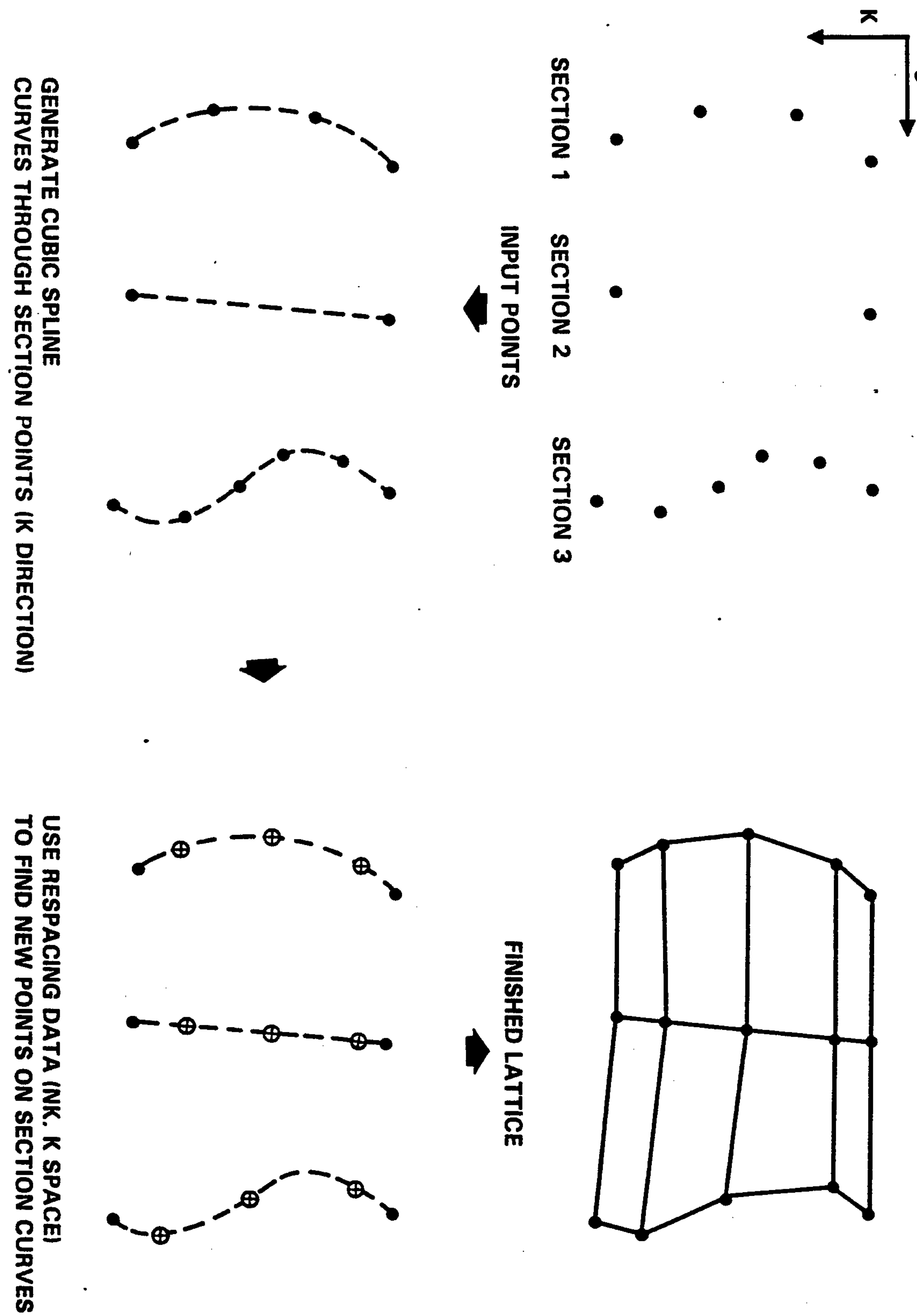


Figure 6-15 Panel Respacing in the K Direction

on the other hand, can be individually specified for each column. This loss in capability is due to the input of the panel lattice information in columns (sections) rather than rows.

6.10.4 Cylindrical Sections And Panel Respacing

As discussed above, a cylindrical coordinate system may be used as an alternative to the rectangular coordinate system for defining section points. These cylindrical sections posses special properties if respacing is specified in the K (section) direction.

When respacing is specified for a cylindrical section, the points defining the section are first interpolated in the cylindrical coordinate space by arc length to generate an enriched cylindrical section. This enriched section definition is converted to rectangular coordinates and is then treated as if it were a rectangular section for further respacing operations. The enrichment of the section definition before the final respacing in rectangular coordinates places points near the ends of the curve to ensure the correct slope behavior there. This can be a problem with standard rectangular sections, requiring extra points to be input.

The real benefit of the first interpolation in polar coordinates is that only two points need be input to specify a circular arc section. If a section curve is defined with constant radius values, then any radius value interpolated between the ends of the curve will also be constant. In addition, near-circular sections can be defined using fewer points than would be required for purely rectangular definitions. The difference between rectangular and cylindrical sections is illustrated in Figure 6-17 for sections defined with two and three points.

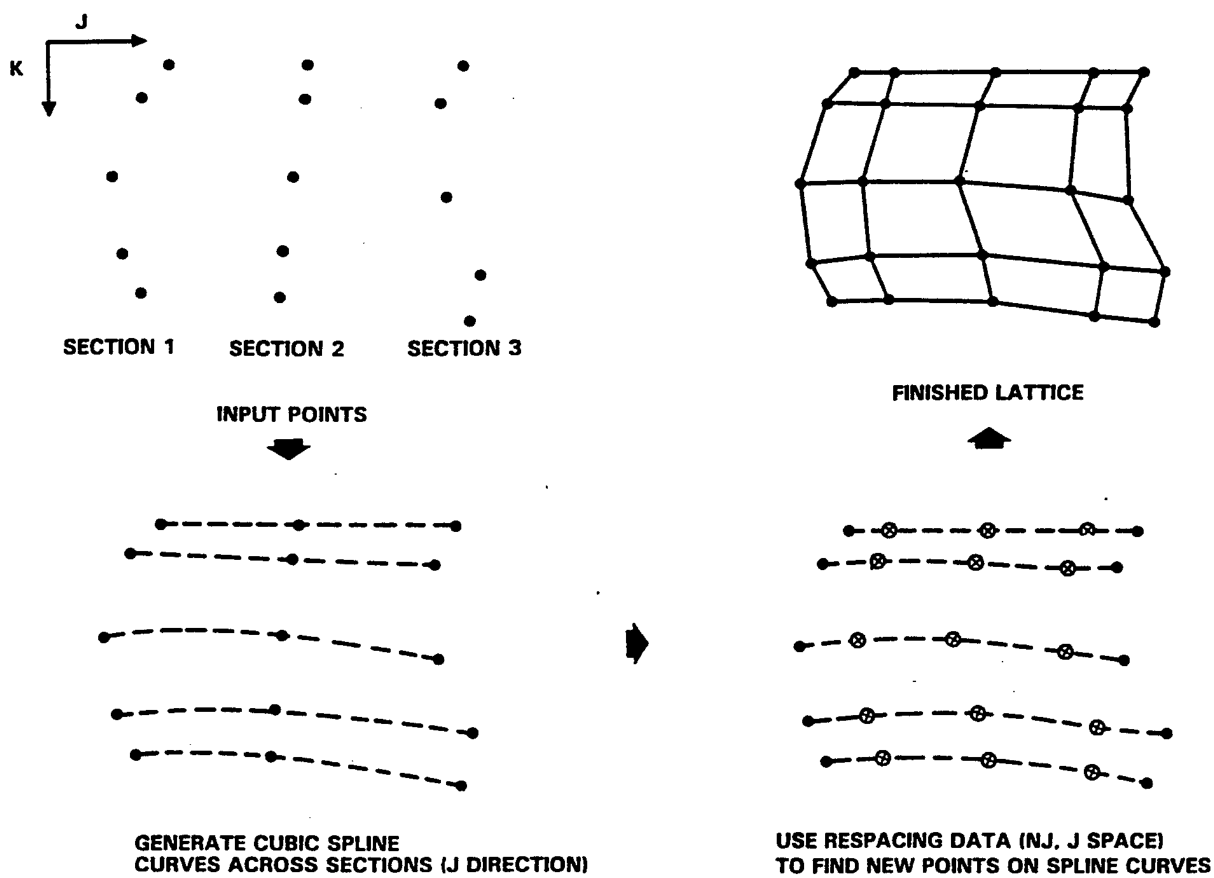


Figure 6-16 Panel Respacing in the J Direction

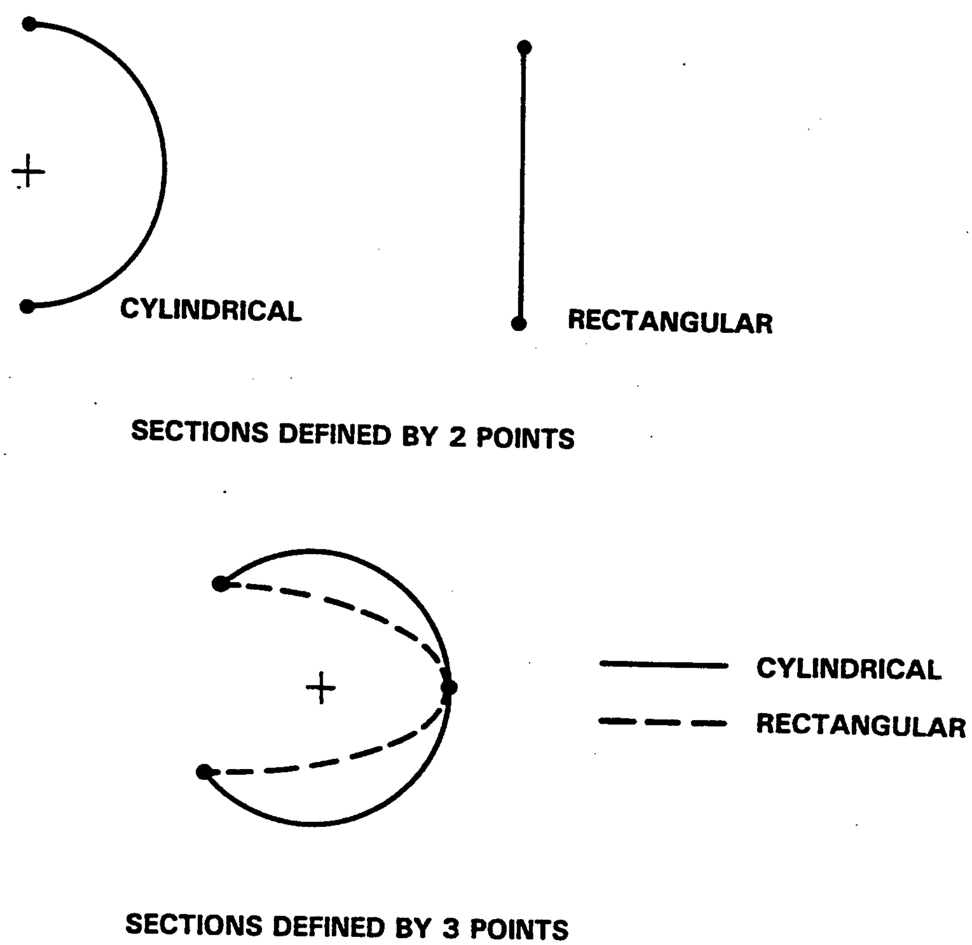


Figure 6-17 Comparison of Rectangular and Cylindrical Sections

7. QUADPAN DATASET

7.1 INTRODUCTION

This chapter describes the input dataset for a QUADPAN run. The basic dataset structure is introduced, followed by the ground rules for the data format. The chapter provides a detailed definition of the input dataset and describes each parameter in each record.

Several examples are provided at the end of the chapter in order of increasing complexity. These illustrate both the syntax of the dataset and the relationship between the geometry of a configuration and the corresponding input dataset. The reader is encouraged to refer to these examples frequently.

7.2 INPUT DATASET STRUCTURE

An input dataset for QUADPAN is divided into four basic parts, separated by keywords (to be discussed below). The parts are listed here in the order they are to appear in the dataset.

- **Global data** – This part consists of information required to run the case, the reference quantities, and flow conditions desired.

- **Panel data** – The geometric definition of the configuration is input by panels in this part. The panel definitions consist of panel information and several section definitions.

  Sections are used to actually define the panel geometry. For this reason section data is treated as a separate part of the input, although this is only in the context of defining the panel.

- **Propeller data (optional)** – The parameters defining the propeller slipstream model are specified in this part.

- **Survey data (optional)** – This part specifies additional locations at which the program is to calculate flow field quantities.

The structure of the input dataset is presented in Figures 7-1 and 7-2. Figure 7-1 is a flow chart which shows the order and possible combinations of the basic blocks of data in the input dataset. Each block on the flow chart consists of one or more records. Figure 7-2 shows the structure of the dataset and the variables which comprise each record. The record identification number in each block in the flow chart (e.g., P4) refers to one or more records in Figure 7-2 (e.g., P4, P4a, and P4b). These identification numbers are used throughout the remainder of the chapter, including the examples at the end of the chapter.

Many of the records in the dataset are optional and, if not required, may be left out of the input data. This is done by omitting the keyword for the option and any records associated with it. The parts of the dataset and their input records are discussed later in this chapter. Not shown in this Figures 7-1 and 7-2 is the new integrated capability for ASTROS for creating control surfaces. A protocol of AESURF unique to QUADPAN was integrated into the QUADPAN dataset. This protocol for user input will be described in the following appropriate subsections.

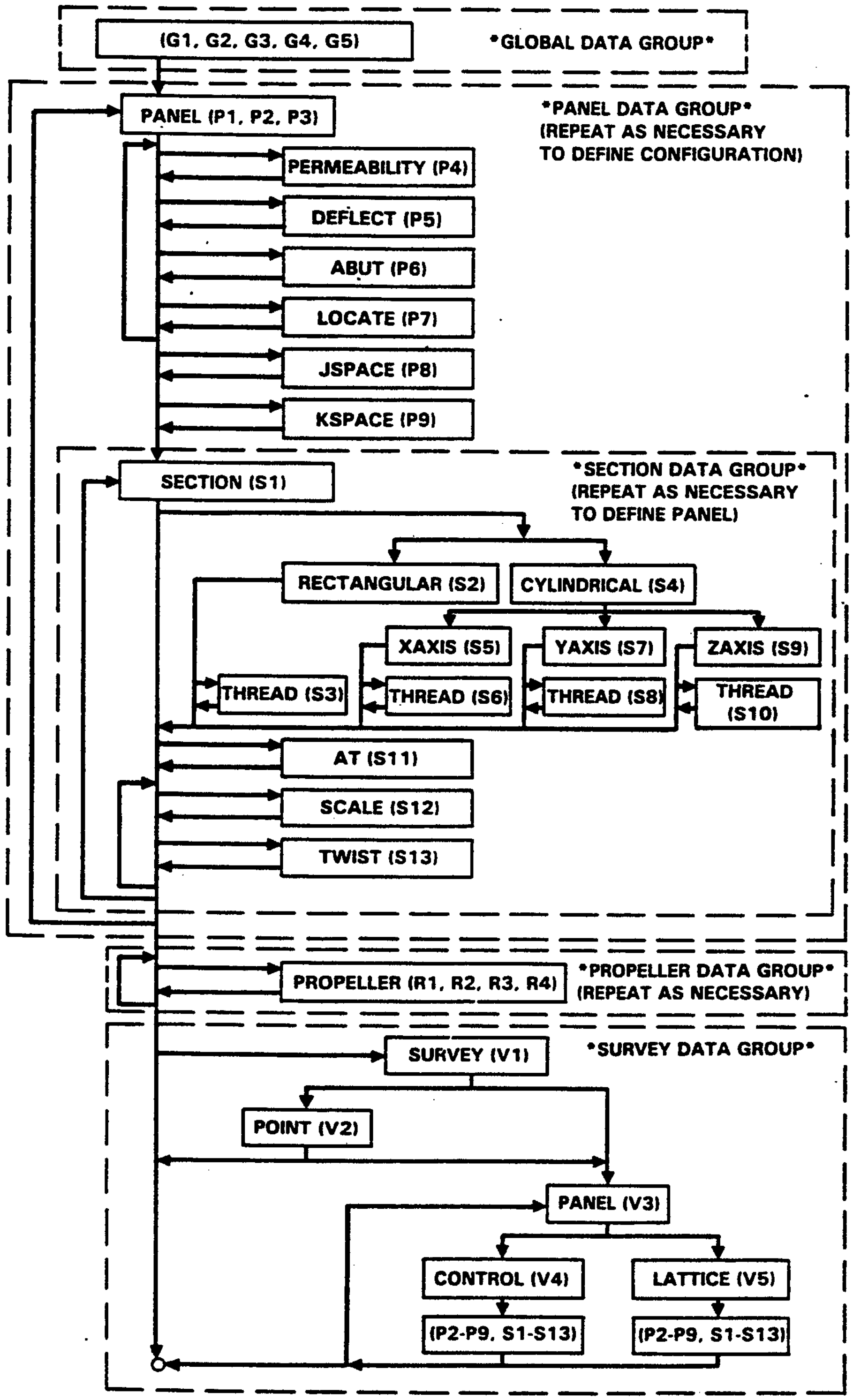


Figure 7-1 Flow Chart of Dataset Structure

```
Record         Global Data

G1         Case Id
G2         Run  Id
G3         Mach        Run         Print       Dump        Abstol      Reltol
G4         Sref        Cbar        Wspan       Xbar        Ybar        Zbar
G5     *   Alpha       Beta        Omegax      OmegaY      OmegaZ      Vinf

Record         Panel Data

P1     *   PANEL
P2            PIdnum      Ptitle
P3            Type        Wet         Force       Image
P4            PERMEABILITY
P4a              Vnorm
P4b    +         Vnorm (image)
P5            DEFLECT
P5a              Xtail      Ytail      Ztail       Xhead      Yhead      Zhead      Deflect
P5b    +         Xtail      Ytail      Ztail       Xhead      Yhead      Zhead      Deflect (image)
P6            ABUT
P6a    *         EdgeN           Panel          Edge
P7            LOCATE
P7a              Xp1        Yp1        Zp1         Xg1        Yg1        Zg1
P7b              Xp2        Yp2        Zp2         Xg2        Yg2        Zg2
P7c              Xp3        Yp3        Zp3         Xg3        Yg3        Zg3
P8            JSPACE
P8a              Nj         Jspace
P9            KSPACE
P9a              Nk         Kspace

Record         Section Data

S1     *      SECTION
S2               RECTANGULAR
S2a    *                X          Y          Z                Kbreak    Nk    Kspace
S3                  THREAD
S3a                     X0         Y0         Z0
S4               CYLINDRICAL
S5                  XAXIS
S5a    *                Theta      R          X                Kbreak    Nk    Kspace
S6                  THREAD
S6a                     Theta0     R0         X0
S7                  YAXIS
S7a    *                Theta      R          Y                Kbreak    Nk    Kspace
S8                  THREAD
S8a                     Theta0     R0         Y0
S9                  ZAXIS
S9a    *                Theta      R          Z                Kbreak    Nk    Kspace
S10                 THREAD
S10a                    Theta0     R0         Z0
S11              AT
S11a                Xthread     Ythread     Zthread        Jbreak    Nj    Jspace
S12    *         SCALE
S12a                Xscale      Yscale      Zscale
S13    *         TWIST
S13a                Xaxis       Yaxis       Zaxis          Twist

Record         Propeller Data

R1     *   PROPELLER
R2            Xprop       Yprop       Zprop          Alphaprop       Betaprop
R3            Rprop       Dir         Propimage
R4            Cthrust     Ctorque     Ccrossforce

Record         Survey Data

V1         SURVEY
V2            POINT
V2a    *         Xsurv         Ysurv        Zsurv
V3     *      PANEL
V4               CONTROL
                    (panel and section records P2-P9, S1-S13)
V5               LATTICE
                    (panel and section records P2-P9, S1-S13)

All capital letters indicate a KEYWORD which is to be literally entered
Small letters indicate a parameter whose value is to be entered
* indicates that the card (or KEYWORD and cards it heads) may be repeated
+ indicates that the item refers to the image panel and is only required
  if the parameter must be non-symmetric
```

Figure 7-2 Input Dataset Schematic

7.3 GENERAL RULES FOR INPUT DATA

The input to QUADPAN is record oriented, one record per line of input with no extensions allowed onto additional cards. The program reads only the first 72 columns of any input line, allowing datasets to be line numbered (for IBM systems) or otherwise identified as the user desires. There are three types of input lines recognized by the program; comments, keywords, and numeric data.

7.3.1 Comments

Any line beginning with an asterisk '*' is considered a comment and is ignored by QUADPAN. An input record is considered to be any noncomment line. Comments may also be put on noncomment records by preceding the comment text with an asterisk. The input record will be processed from left to right until the comment is encountered. Input fields skipped due to the presence of a comment on the record are assumed to be blank.

Examples:

```
* THIS IS A COMMENT LINE
* 234567890 1234567890123456789012345678901234567890RULER LINE
 102.733       33.845       1201.993    *REST OF LINE IS A COMMENT
```

7.3.2 Keywords

Keywords are used in the input for several functions. They serve to organize the input. They allow the user to specify options that may safely be defaulted when not required, thus reducing the complexity of the input dataset. Keywords are also used to terminate lists of data (such as a list of X, Y, Z points) so that the user need not redundantly specify the length of the list.

Keywords must be in capital letters, starting in the first column of the record. Only the first four letters of each keyword are significant. Some keywords also require additional data to be entered (numeric or keyword) that is associated with that option.

7.3.3 Numeric Data

The input of numeric data is organized into fields of ten columns each, in much the same manner as the Fortran 'F10.xx' format. The value of a parameter may be placed anywhere in its assigned field. Decimal points or right justification are optional as numbers can be delimited by blanks. It is recommended that decimal points be used for all floating point number inputs for clarity. A maximum of 7 fields may be present on a record. Blank fields or fields not read due to a comment on the record are assigned default values (the default value of any parameter is zero, unless otherwise specified).

Examples:

```
* 234567890 1234567890123456789012345678901234567890(RULER COMMENT)
 102.733     33         1201.993    -59.432
   -8        .67E-05   +125.669
```

7.4 ADDITIONAL CONSIDERATIONS FOR INPUT DATA

As mentioned above, open ended lists, where any number of records may be input, are delimited by keywords, or the end of file. A blank line in the dataset will be interpreted by the program as an end of file in most situations.

The need for comments in datasets cannot be overemphasized. It is difficult to work with an old dataset that was not properly documented. An additional benefit of comments is that they may be used to eliminate alignment errors by marking the beginning of each field. The facility for comments has been provided. . . use it! Comments will be used to clarify the material in all examples in this chapter.

7.5 SUMMARY OF KEYWORDS RECOGNIZED BY QUADPAN

The keywords recognized by the program, ordered by the groups in which they appear, are listed below:

Global Data Group

AESURF	Initiates control surface definition

Panel Data Group

PANEL	Initiates panel definition
PERMEABILITY	Allows specification of panel normal velocity
DEFLECT	Simulates panel deflection about an axis
ABUT	Allows user to specify panel abutment data
LOCATE	Specifies panel transformation (translation, rotation, scaling)
JSPACE	Allows respacing in J direction
KSPACE	Allows respacing in K direction

Section Data Group

SECTION	Initiates section definition
RECTANGULAR	Precedes new section in rectangular coordinates
CYLINDRICAL	Precedes new section in cylindrical coordinates
XAXIS	Specifies X-axis for polar axis
YAXIS	Specifies Y-axis for polar axis
ZAXIS	Specifies Z-axis for polar axis
THREAD	Allows user to specify section thread point
AT	Specifies translation of section
SCALE	Specifies section scaling
TWIST	Specifies section rotation about an axis

Propeller Data Group

PROPELLER	Initiates propeller definition

Survey Data Group

SURVEY	Initiates survey definition
POINT	Precedes definition of survey points

PANEL	Initiates survey panel definition
CONTROL	Specifies panel control points as survey points
LATTICE	Specifies panel lattice points as survey points

7.6 GLOBAL DATA

The global data (records G1-G5) contains information to identify and control the run, the reference quantities to be used to normalize the forces and moments, and the flow conditions to be analyzed. The program expects this section to be present in every dataset in the order given here.

The global data is terminated by the keyword PANEL of the first panel definition.

No keywords are used in the global data group.

Record G1: Case ID (col. 1-72)

The first noncomment card in the dataset. To be used for an identifying description of the configuration.

This title is associated with the geometry of the configuration and must be maintained unchanged for all future restarts using the matrices generated on the initial run. The title will appear in the output print, the output dump, and the matrices from this run. It is recommended that meaningful information be used to eliminate ambiguity on what configuration was actually run.

Record G2: Run ID (col. 1-72)

A noncomment card, to be used for an identifying description of the particular run (flow condition or boundary conditions) the dataset corresponds to.
The run identifier is an additional label that may be used to identify a run, but which differs from the title (record G1) because it has no effect on restarts. This permits the title to be permanently associated with the configuration geometry. The run title will appear in the output print and the output dump from this run. Again, it is recommended that meaningful information be used to eliminate ambiguity on what flow condition was actually run.

Record G3: MACH, RUN, PRINT, DUMP, ABSTOL, RELTOL

This record specifies the Mach number to be used for the run, the control flags, and the tolerance parameters to be used for the Automatic Abutment check.

MACH – Mach number (col. 1-10)

The freestream Mach number (MACH < 1.0). Since the matrices generated are peculiar to a given Mach number the Mach number must be unchanged if the program is restarted using matrices from a previous run.

RUN – Run control flag (col. 11-20)

RUN = -1: Program generates geometry only.
RUN = 0: Program generates geometry and performs abutment checks, wake checks, and interior surface checks.
RUN = 1: Normal program operation.
RUN = 2: Restart, program uses existing influence matrices from units 1 and 2.
RUN = 3: Restart, program uses existing influence matrices from units 1 and 2 and

survey influence matrices from unit 3.

RUN = -1 is used when only geometry data needs to be computed. This is normally used in conjunction with DUMP = 1 to produce a dump file for plotting the configuration geometry.

RUN = 0 is used for geometry data and abutment checking, but not a full run. This is normally used for debugging, usually with DUMP = 1 to produce a dump file for plotting. This allows the configuration panel edge abutments to be checked.

RUN = 1 is used when complete program execution is needed.

RUN = 2 or 3 is used when the program is to be restarted using influence matrices from a previous run. If a run merely involves a change in flow conditions or propeller effects with no change in the geometry or Mach number, then the program may be restarted using the potential influence matrices from units 1 and 2 from the initial run. If there is a flow survey and the survey grid has not changed, the flow survey may also be restarted by using RUN = 3. In this case the velocity influence matrices are needed from unit 3 of the initial run. See 2.5.4 and Appendix C for further information on restarts.

PRINT – Print control (col. 21-30)

PRINT = 0: all output data printed
PRINT = 1: all output data printed except geometric data
PRINT = 2: only force data and element pressure data printed
PRINT = 3: only force data printed

Flag controlling the output data which is printed by the program on unit 6. Some of the output data print may be suppressed if it is not needed. Chapter 8 fully defines the output generated for each value of this flag.

The program prints four sets of output data for the run in addition to a literal input echo, an interpreted input echo, a list of defined panels and the panel abutment list, all of which are printed for all cases. The four optional sets are:

- A full listing of the geometry of the elements in the configuration including indices, corner points, control point, normal vector, and area.
- The forces and moments on the whole configuration and on each panel in three coordinate systems: body axes, stability axes, and wind axes.
- A listing of the pressure and velocity on each element in the configuration, grouped by panels.
- The singularity strengths (source and doublet) on each element in the configuration, grouped by panels.

DUMP – Dump file control flag (col. 31-40)

DUMP = 0: dump file is not created
DUMP = 1: dump file is created

Flag controlling the creation of an output dump file on unit 7. The dump file may be used by other programs which post-process geometry or pressure data, for example. See Appendix B for information on the Dump File.

ABSTOL – Absolute geometric tolerance distance (col. 41-50)

The absolute tolerance distance to be used in determining element edge abutments. Element edges will be considered abutted if the distance between the midpoints of the edges is less than the smaller of ABSTOL and RELTOL *(element edge length).

The default for ABSTOL (also used if ABSTOL is input as 0) is 10 times the minimum resolvable distance (using the machine precision) at the maximum X, Y or Z coordinate in the configuration. Additional information is available in Chapter 5.

RELTOL – Relative tolerance (col. 51-60)

The fraction of the element edge length to be used in determining element edge abutments (defaults to 0.1). Element edges will be considered abutted if the distance between the midpoints of the edges is less than the smaller of ABSTOL and RELTOL *(element edge length). See Chapter 5 for more information.

Record G4: SREF, CBAR, WSPAN, XBAR, YBAR, ZBAR

This record specifies the reference quantities to be used for normalizing forces and moments for the configuration. These quantities should be in the same units as the input variables defining the geometry.

SREF – Configuration reference area (col. 1-10)

The reference area for all aerodynamic coefficients (defaults to 1.0). This is the reference area for the entire configuration (both sides of the plane of symmetry) when the configuration is laterally symmetric. The program will include forces from image panels in the total forces and moments.

CBAR – Mean chord (col. 11-20)

The reference length for all body axis moments and the pitching moments in the wind and stability axis systems (defaults to 1.0).

WSPAN – Wingspan (col. 21-30)

The reference length for the rolling and yawing moments in wind and stability axis systems (defaults to 1.0). This is the span across both sides of the plane of symmetry when the configuration is laterally symmetric.

XBAR – Coordinates of moment reference and rotation center
YBAR (col. 31-40, 41-50, 51-60)
ZBAR

This point fixes the location of the moment reference center (defaults to 0.,0.,0.). See Fig. 4.3. It is also the center of rotation for the configuration (if the aircraft has no rotation, this point has meaning only as the moment center).

Record G5: ALPHA, BETA, OMEGAX, OMEGAY, OMEGAZ, VINF

Flow conditions to be analyzed in the run. This record may be repeated for up to 30 flow conditions. The free stream is specified by angles of attack and sideslip, and the angular velocity vector is specified along with a free stream velocity to nondimensionalize the time derivatives implied in the rotation rates. The angular velocity is the body axis rotation rate of the configuration about the rotation center. See Appendix

D for more information on using the program for nonzero angular velocities. The keyword PANEL terminates the global data.

ALPHA – Angle of attack (degrees) (col. 1-10)

The angle of attack is the angle made by the projection of the free stream vector into the X-Z plane with the X-axis. It is considered positive when the wind has a component along the positive Z-axis (wind from below) as shown in Figure 7-3.

BETA – Angle of sideslip (degrees) (col. 11-20)

The angle of sideslip is the angle between the free stream vector and the X-Z plane. It is considered positive when the wind has a component along the positive Y-axis (wind from the port side) as shown in Figure 7-3. Note that this is not the same as the angle between the X-axis and the projection of the velocity vector into the X-Y plane!

OMEGAX – X-component of angular velocity (deg/sec) (col. 21-30)

The component of the aircraft's angular velocity about the X-axis (positive in the sense defined by the right-hand rule).

OMEGAY – Y-component of angular velocity (deg/sec) (col. 31-40)

The component of the aircraft's angular velocity about the Y-axis (positive in the sense defined by the right-hand rule).

OMEGAZ – Z-component of angular velocity (deg/sec) (col. 41-50)

The component of the aircraft's angular velocity about the Z-axis (positive in the sense defined by the right-hand rule).

VINF – free stream velocity (col. 51-60)

The translational velocity used to normalize the angular rates (defaults to 1.0). It is necessary to input VINF only when there are nonzero angular rates. If the aircraft is nonrotating, all computed output is independent of free stream velocity.

The final set of data in the QUADPAN Global Data Group is the control surface specifications. The keyword **AESURF** was added to the Global Data Group to integrate QUADPAN with ASTROS. The control surface functionality operates in conjunction with the **DEFLECT** option which is defined within the Panel Data Group. QUADPAN control surfaces are constructed from a collection of panels. Surface geometry, symmetry, and pressure magnitude are determined through **AESURF** input. Table 7-1 displays the sub-keywords that are used in this data flow to define the control surfaces. The hingeline and sign convention for surface rotation is defined within the **DEFLECT** options for each unique panel. *The user must be careful to define consistent rotation axes in the **DEFLECT** option with intended surface deflection symmetries in the **AESURF** option*!

QUADPAN flow solutions are obtained in the sequence of an outer loop on the specified flow conditions and an inner loop on each surface. Consequently, the number of flow solutions to be acquired within a QUADPAN analysis is determined by the product of the number of flow conditions and the number of control surfaces. Any number of control surfaces may be defined. Any number of Panels may belong to an **AESURF** declaration. Reference 4 provides further discussion and an example of the **AESURF** option.

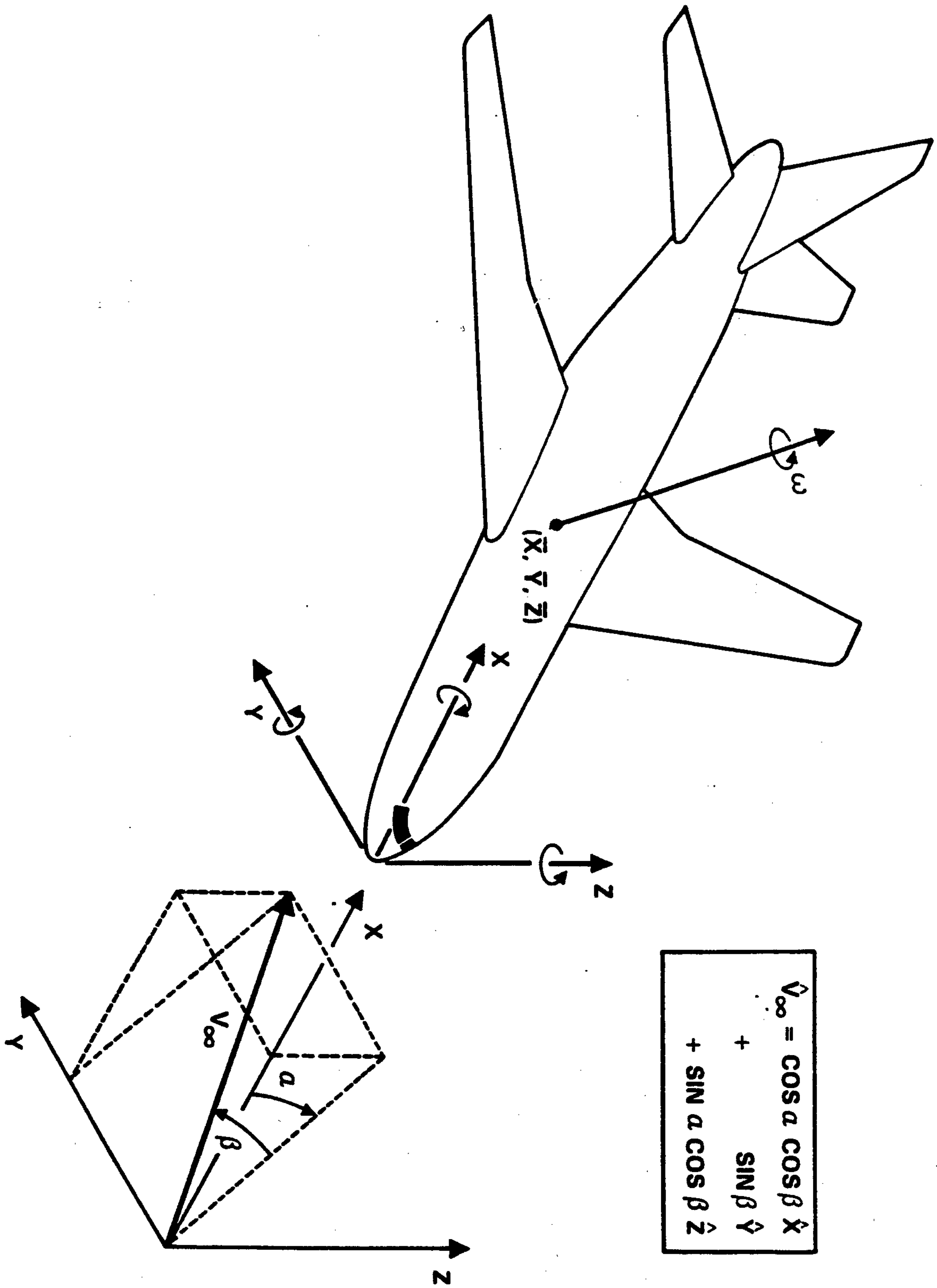


Figure 7-3 Global Coordinate System

Table 7-1 Definition of Control Surfaces in QUADPAN

<option>	Description
"AESURF"	(keyword)
User Name	an 8 character (truncated from 10) surface name
Symmetry	the surface symmetry (-1, 1 or blank)
"PANELS"	(keyword) panel ids' one or more records of 1 to 8 panel ids that comprise the surface
"ANGLES"	(keyword) angular deflection, one or more records of 1 to 0 angles (in deg.) that comprise the surface sweeps.

7.7 PANEL DEFINITION DATA

The panel definition data (records P1-P9) begins with the first keyword PANEL, ending the global data. Each panel definition must begin with the keyword PANEL and is terminated by the keyword PANEL (starting the next panel definition), the keyword PROPELLER, the keyword SURVEY, the end of file, or a blank line. The program expects at least one panel definition to be present in every input dataset, with a limit of 200 panels.

The panel is the basis of the QUADPAN dataset. The concepts and nomenclature to use panels to create geometric models is presented in Chapter 6.

The panel definition data contains several header records for identification, specifying boundary conditions and some required flags, plus additional keywords to exercise various useful options. These may be selected using records P4-P9 and are entirely optional. They include the specification of quantities that alter the boundary conditions to be applied to the panel, a transformation to be applied to the defined panel so that it may be described in its own coordinate system (the panel coordinate system or PCS) and information that will respace the element mesh in either or both of the J or K lattice directions.

Following this is a set of section definitions, each beginning with the keyword SECTION. Sections are the building blocks of the panel in much the same way as the panels are the building blocks of the configuration. After each section is defined, the program checks to see if the panel is finished or another section begins. Sections are discussed under Section Data.

Panel Data Keywords

PANEL	Initiates panel definition
PERMEABILITY	Allows specification of panel normal velocity
DEFLECT	Simulates panel deflection about an axis
ABUT	Allows user to specify panel abutment data
LOCATE	Specifies panel transformation (translation, rotation, scaling)
JSPACE	Allows respacing in J direction
KSPACE	Allows respacing in K direction

Record P1: PANEL – <KEYWORD>

The keyword PANEL indicates the beginning of the definition of each panel. Any pending lists are terminated by this keyword.

Record P2: PIDNUM, PTITLE

A number and a title to be associated with the panel for identification purposes.

PIDNUM – Panel identification number (col. 1-10)

A panel identification number, assigned by the user, that is associated with this panel. The number may be any positive whole number between 1 and 9999. The panel ID number must be unique to this panel (no repeated identification numbers within the dataset). The image of this panel, if it has one, will be referred to by the negative of this number.

PTITLE – Panel title (col. 11-58)

An identifying description for the panel, up to 48 characters long, starting in column 11. This title should be as informative as possible as to the identity of the panel.

Record P3: TYPE, WET, FORCE, IMAGE

This record contains the panel flags that control the boundary conditions to be applied to the panel, the side of the panel to be used as the exterior surface, the disposition of panel forces, and the generation of an image panel.

TYPE – panel type (col. 1-10)

TYPE =-1: body panel with exterior potential specified
TYPE = 0: body panel with exterior normal velocity specified
TYPE = 1: wake panel where edge 1 is the Kutta edge
TYPE = 2: wake panel where edge 2 is the Kutta edge
TYPE = 3: wake panel where edge 3 is the Kutta edge
TYPE = 4: wake panel where edge 4 is the Kutta edge

Flag specifying the type of boundary condition to be applied to a panel.

TYPE = 0 – Body panel with specified normal velocity

This type of panel is used for nearly all body (nonwake) panels. It is used for body panels that will have hydrodynamic boundary conditions (specified normal velocity at the exterior surface) applied to them. This includes impermeable panels, where the exterior normal velocity is specified to be zero. On TYPE = 0 panels, the source strength is specified by the onset flow velocity at the control point, plus a normal flow (given by the panel permeability). The doublet strength is determined by the solution to the linear system.

TYPE = 1-4 – Wake panel

Wake panels are used to represent the vortex wake from a lifting surface and enforce the Kutta condition at one of their edges. A wake panel is specified with TYPE = 1-4, the TYPE indicating which of the four panel edges is to have the Kutta condition applied (see Figure 7-4) for an illustration of the four edges of a panel). This edge should normally be the one coincident with the trailing edge of a wing, where the Kutta condition is applied. The "Kutta" edge can also be placed coincident with an edge of another wake panel to further extend the vortex wake. In this way, a short wake panel can be continued into an entire wake system to transmit the vortex downstream.

The Kutta condition basically enforces the condition that the sum of the doublet strengths at the Kutta edge be zero.

TYPE = -1 – Body panel with specified exterior potential

These are special purpose boundary conditions, used on body panels for problems where a specified potential (and tangential velocity) must be applied on the exterior surface. This is normally done to make an ill-posed potential problem well-posed. Its use includes internal flow problems and situations involving closed wake volumes such as the closure panel at a thick trailing edge with wakes shed from the upper and from the lower surface. This is discussed further in Ref. 4. On TYPE = -1 panels, the doublet strength is specified such that the total tangential velocity on the exterior surface at the control point is zero. The source strength is determined by the solution to the linear system.

WET – Panel surface wetted flag (col. 11-20)

WET = +1: Panel wetted only on positive surface
WET = -1: Panel wetted only on negative surface

Flag specifying which of the panel's surfaces (positive or negative) is to be used as the "exterior" surface, wetted by the flow. Figure 7-4 illustrates the panel positive and negative surfaces. The positive surface can be determined by the use of the right hand rule on the panel edge sequence edge 1, edge 2, edge 3, edge 4 (the first section entered is edge 1, the last point in each section defines edge 2, the last section entered is edge 3, and the first point in each section defines edge 4). The thumb of the right hand will point in the direction of the positive side.

Either the positive or negative surface of the panel can be specified as the "exterior" surface (wetted by the flow) with the WET flag. The "exterior" surface (defining the outward normals) will be the positive panel surface for WET = 1, or the negative panel surface for WET = -1. Note that the value of WET for a wake panel may be either +1 or –1 since it is wetted on both sides (the only difference will be the sign of the wake element doublet strengths).

FORCE – Panel force flag (col. 21-30)

FORCE = 0: Panel which does not contribute to total forces and moments
FORCE = 1: Panel which does contribute to total forces and moments (default)

Flag specifying whether or not the forces and moments on a panel are to be included in the total forces and moments on the configuration. As an example, FORCE = 0 might be used on panel(s) representing an external store which has just been released, or on a panel used to model the exhaust plume behind a nacelle. The FORCE flag of a wake panel (TYPE = 1-4) is automatically set to zero (no forces), no matter what is input.

IMAGE – Panel image flag (col. 31-40)

IMAGE = 0: panel has no image
IMAGE = 1: panel has an image (default)

Flag specifying whether a panel is to be reflected in the X-Z plane to create an image panel. Panels which appear symmetrically in the geometry need be input only once with an image specified. If the geometry is completely symmetric about the X-Z plane, then only half the configuration need be input, with all panels having IMAGE = 1. The program will then construct the other half by reflection.

Record P4: PERMEABILITY – <KEYWORD>

The keyword PERMEABILITY precedes the specification of the normal velocity (permeability) on the panel exterior surface in records P4a and P4b. If the keyword is omitted, the panel is assumed to be impermeable and records P4a and P4b are not read. The value of the normal velocity must be entered on the next record, and (optionally) the image normal velocity on the record following that.

An example of where this might be used is on a panel representing the compressor face in a nacelle inlet, where the permeability serves to set the mass flow at the compressor face. This is discussed further in Ref. 4.

Records P4a: VNORM – panel normal velocity
Records P4b: VNORM – panel normal velocity (image)

This is the ratio of the normal velocity prescribed on the surface of the panel to the free-stream velocity (for a TYPE = 0 panel). A positive value indicates outflow (a velocity vector pointing out along the normal vector into the flow which wets the panel). A negative value indicates inflow. It should be noted that this is an explicit specification of the normal component of velocity only when the Mach number is zero.

If the panel has an image, the image panel normal velocity may be entered on a following card (record P4b). If omitted, it is assumed that the image panel normal velocity is identical to that of the original panel, making the boundary conditions symmetrical.

The specification of panel permeability controls the source strength for TYPE = 0 panels. The actual relation used for SIGMA (the source strength) is:

SIGMA = VNORM – (component of onset flow along outward normal)

For VNORM = 0 (default) this gives zero total normal velocity on the exterior surface (impermeable surface).

The panel permeability also controls the doublet strength for TYPE = -1 panels. A VNORM = 0 (default) gives zero total tangential velocity on the exterior surface. If VNORM is specified as 1.0 the tangential velocity on the exterior surface is the tangential component of the free stream.

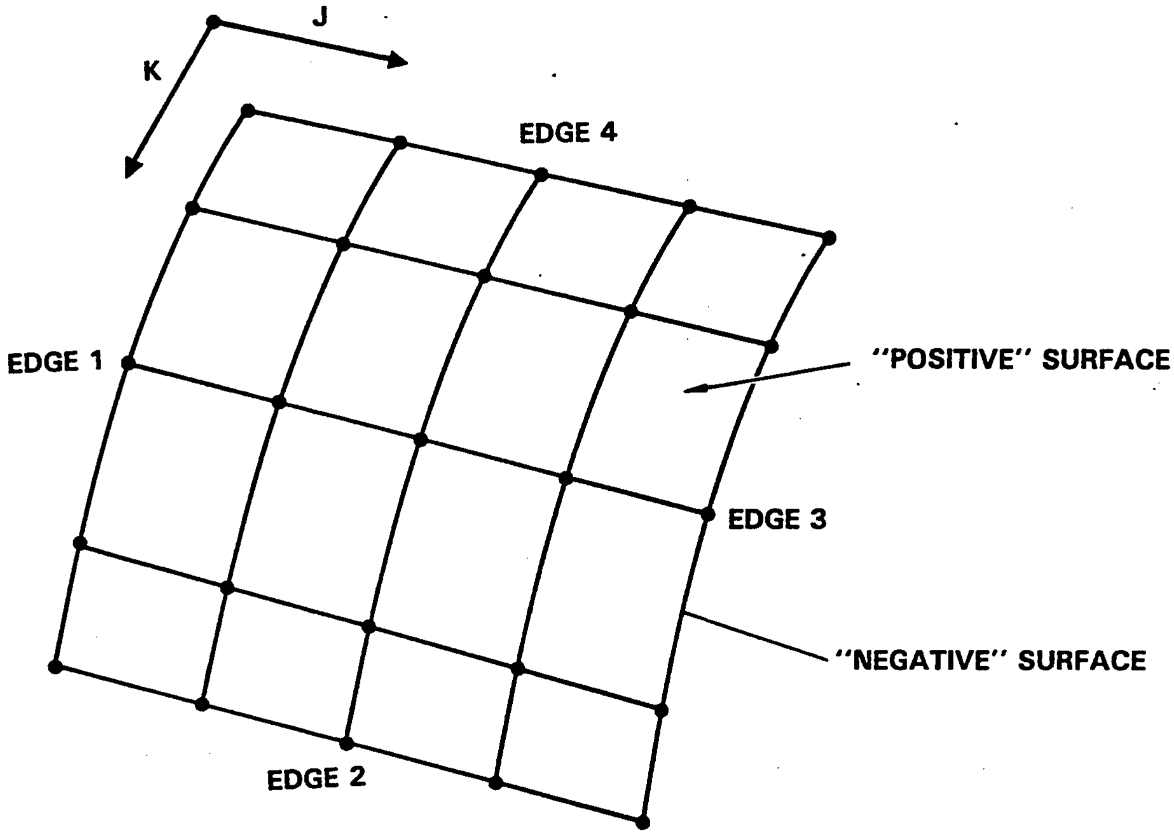


Figure 7-4 Positive and Negative Panel Surfaces

Record P5: DEFLECT – <KEYWORD>

The keyword DEFLECT precedes a specification of simulated panel deflection in records P5a and P5b. If the deflection specification and keyword are omitted no panel deflection occurs and records P5a and P5b are not read. The axis specification and deflection angle must be entered on the next record, and (optionally) the image panel axis and deflection on the record following that if the panel has an image.

This option allows the user to alter the boundary conditions to be used on a panel to simulate a geometric deflection (rotation about an axis) without actually having to construct a deflected panel geometric definition. Since the simulated deflection affects only the boundary conditions, this option may be used in restarts with matrices generated without deflections. The simulated panel deflection will not be as accurate as a physically deflected panel for large deflections, but for small deflection angles the results are quite acceptable.

An example of where this option may be used is to simulate the deflection of the panel(s) modeling an elevator on an aircraft. The simulated deflection has no effect on restarts, so the use of this option is extremely cost effective in restart mode. This is discussed further in Ref. 4.

Records P5a: XTAIL,YTAIL,ZTAIL,XHEAD,YHEAD,ZHEAD,DEFLECT
Records P5b: XTAIL,YTAIL,ZTAIL,XHEAD,YHEAD,ZHEAD,DEFLECT (image)

This record specifies a panel deflection axis and deflection angle for simulated deflection of the panel. This is given by two points on the deflection axis and a deflection angle. If the panel has an image, the image panel deflection may also be specified on the following card (record P5b). If omitted, it is assumed that the image panel deflection is identical (symmetrical across the X-Z plane) to that of the original panel.

> **XTAIL,YTAIL,ZTAIL** – tail of axis vector (col. 1-10, 11-20, 21-30)
> **XHEAD,YHEAD,ZHEAD** – head of axis vector (col. 31-40, 41-50, 51-60)
>
> The (global) coordinates of the head and tail of a vector defining the axis of rotation (the vector points from tail to head). The magnitude of the vector is immaterial, but the direction of the vector defines the positive sense of rotation via the right-hand rule. For example, the head and tail may be any two points on the hinge line of a control surface.
>
> **DEFLECT** – deflection angle (degrees) (col. 61-70)
>
> Angle through which the panel is deflected in degrees. The positive sense of rotation is determined by the right-hand rule about the rotation axis vector. Since the deflection is merely simulated by rotating the free stream vector that the panel "sees," the deflection angle should be restricted to small values.

Example:

```
*234567890 1234567890123456789012345678901234567890123456789ORULER
*
DEFLECT
*XTAIL    YTAIL     ZTAIL     XHEAD     YHEAD     ZHEAD     ANGLE
  0.        0.        0.        0.        2.        0.        5.
```

This specifies a simulated deflection of 5 degrees about an axis parallel to the Y global axis.

Record P6: ABUT – <KEYWORD>

The keyword ABUT precedes a specification of the panel edge abutments.

The program contains an automatic procedure to search all panel edges to establish contiguous neighboring elements across panel boundaries. The element neighbors are used for surface differentiation of the potential to establish velocity and to set up Kutta conditions at wake shedding lines.

In the great majority of cases where the input geometry is relatively gapless, the automatic procedure will establish the element connectivity without further user intervention. In some cases, however, it may be desirable to restrict the search space for the automatic abutment search to eliminate possible abutment ambiguities.

The ABUT keyword allows the user to specify the search space to be used on the panel's edges. See Chapter 8 for a description of the abutment search procedure and 8.5 for details on user specified abutment searches.

Record P6a: EDGEN, PANEL, EDGE

The abutment specification follows the keyword ABUT in record P6. This record allows the user to specify the search space to be used in the automatic abutment procedure for any or all of the four edges in the current panel to user selected panels and edges. The specification of an abutting panel and edge in the abutment search space does not guarantee that the program will consider them connected, it merely restricts the edges to be checked in the geometric search. There is currently no method provided for the direct specification of edge abutments.

The user may define up to 8 panel/edge combinations per panel edge to be searched by the abutment search in a user-specified abutment. Record P6a may be repeated as necessary to specify abutments for the panel.

EDGEN – edge number of current panel (col. 1-10)

A panel edge may be specified independently of the other edges in the panel. This must be in the range from 1 to 4. See Figure 7-4 for a definition of the panel edges.

PANEL – panel ID number of abutting panel (col. 11-20)

If the abutting panel ID number is not specified (or is specified as zero), the program will consider the edge not connected to any other edges. Image panels in abutment specifications use ID numbers which are the negative of their real counterparts.

EDGE – edge number of abutting panel (col. 21-30)

The edge number must in the range from 0 to 4. If the abutting edge number is not specified (or is specified as zero), the program will search all 4 edges of that abutting panel.

Example:

```
*234567890 1234567890123456789012345678901234567890123456789 0RULER
*
ABUT
*EDGEN    PANEL    EDGE
   1.       0.
   3.      30.      2.
   3.      31.      4.
   2.       8.      0.
```

This specifies that edge 1 of this panel has no neighbors, the program should look for neighbors for edge 3 from panel 30 edge 2 and from panel 31 edge 4, and finally that the program should look for neighbors for edge 2 from all four edges of panel 8.

Record P7: LOCATE – <KEYWORD>

The keyword LOCATE must precede the specification of the locater points. If the keyword LOCATE is omitted records P7 a, b, c are not read and the global coordinates of the panel will be identical to the panel coordinates.

The locater points specify a transformation to be applied to the defined panel so that it may be described in its own coordinate system, the panel coordinate system (PCS). This transformation uses three points whose location is given in both the panel coordinate system and the global coordinate system. Combined translation, isomorphic scaling, and rotation may be specified by this option (Figure 7-5).

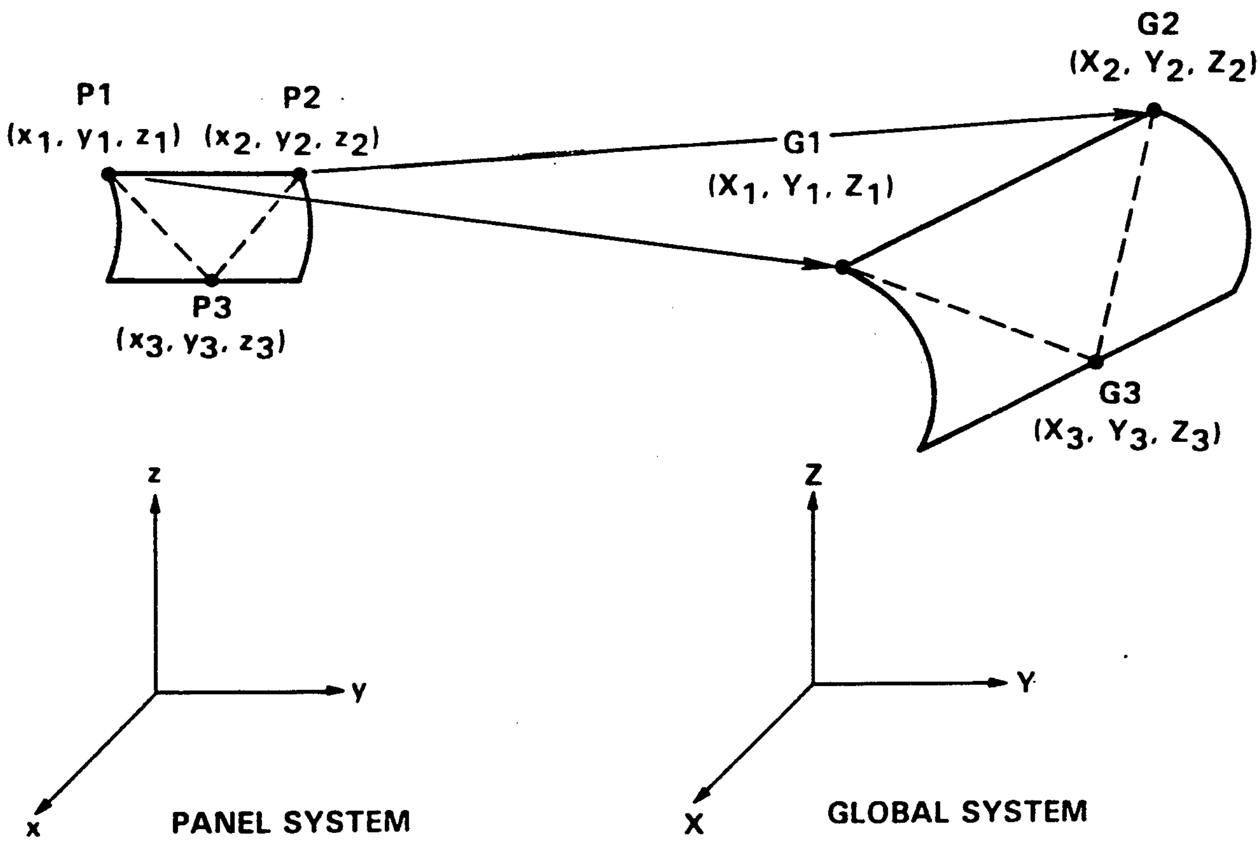


Figure 7-5 Action of Locater Points

Multiple locater point transformations can be specified using additional LOCATE keywords and locater point records. The transformations are done in the order in which they are entered, i.e., the first locater transformation is the first applied to the panel while the last locater transformation will be the last applied to the panel. Note that locater transformations are not, in general, commutative.

Records P7a: XP1,YP1,ZP1,XG1,YG1,ZG1 (col. 1-10, 11-20, 21-30, 31-40, 41-50, 51-60)
Records P7b: XP2,YP2,ZP2,XG2,YG2,ZG2 (col. 1-10, 11-20, 21-30, 31-40, 41-50, 51-60)
Records P7c: XP3,YP3,ZP3,XG3,YG3,ZG3 (col. 1-10, 11-20, 21-30, 31-40, 41-50, 51-60)

Where the fields given above are defined as:

XP1,YP1,ZP1 – panel coordinates of first locater point
XG1,YG1,ZG1 – global coordinates of first locater point

XP2,YP2,ZP2 – panel coordinates of second locater point
XG2,YG2,ZG2 – global coordinates of second locater point

XP3,YP3,ZP3 – panel coordinates of third locater point
XG3,YG3,ZG3 – global coordinates of third locater point

The coordinates of the locater points (records P7a-c) must be preceded by the keyword LOCATE.

Theses records specify the panel locater point transformation. The coordinates of the three sets of points are used to size and locate the panel in the global coordinate system. The locater points may be any three points which are not colinear and thus form a triangle. The locater points need not lie on the panel itself, or be any of the defining points.

The action of the locater points is most easily understood if the locater triangle in panel coordinates is similar to the locater triangle in global coordinates. The size of the panel is altered (without changing the shape of the panel) until the panel locater triangle is congruent to the global locater triangle. The panel is then placed in the global coordinate system so that the panel locater triangle coincides with the global locater triangle.

More generally, if the triangles are not similar, the panel is sized according to the distance between the first and second locater points. The panel is then placed so that the first and second panel locater points coincide with the first and second global locater points, and the panel triangle is coplanar with the global triangle. The fourth example dataset (7.11.4) also provides examples of the use of locater points.

Example 1:

```
*234567890 12345678901234567890123456789012345678901234567890RULER
*
LOCATE
*XP       YP        ZP        XG        YG        ZG
0.        0.        0.        10.       0.        0.
0.        1.        0.        10.       2.        0.
0.        0.        1.        10.       0.        2.
```

This specifies a translation of +10 in the X direction plus a 2X increase in size.

Example 2:

```
*234567890 12345678901234567890123456789012345678901234567890RULER
*
```

LOCATE

*XP	YP	ZP	XG	YG	ZG
100.	10.	10.	300.	0.	0.
101.	10.	10.	300.7071	.7071	0.
100.	10.	0.	300.	0.	-1.

This specifies a translation of a point at (100.,10.,10.) to (300.,0.,0.) plus a rotation of 45 degrees about the Z-axis.

Record P8: JSPACE – <KEYWORD>

The keyword JSPACE precedes the specification of the number and distribution of elements in the J direction in record P8a. Only a single spacing range may be specified using this option. If the keyword and its associated spacing information is omitted, then no respacing is done in the J direction unless provided for by spacing range information with the AT keyword, and record P8a is not read. See Figure 7-4 for a definition of the J direction.

The spacing specified by JSPACE is overruled by the specification of a spacing range, using the AT keyword.

Record P8a: NJ, JSPACE

This record specifies the number of elements and spacing distribution to be used in respacing the panel in the J direction. This specification is used as a default spacing to be applied over the whole J extent of the panel. This specification will be ignored if the finer spacing control provided by the specification of a spacing range, using the AT keyword (record S7) is present.

NJ – Number of elements in J direction (col. 1-10)

The number of elements the panel is to have in the J direction. A maximum of 100 elements may be specified.

JSPACE – Spacing distribution in J direction (col. 11-20)

JSPACE = ±0: equal spacing
JSPACE = ±1: cosine spacing
JSPACE = ±2: sine spacing
JSPACE = ±3: equal spacing

Flag specifying the spacing of the element lattice of the panel in the J direction (Figure 7-6). All spacing distributions are defined with respect to the arc length along the panel surface. Changing the sign of the spacing parameter reverses the direction of the spacing distribution. This has an effect only on sine spacing (a positive value corresponds to elements concentrated at edge 1). Noninteger values of JSPACE produce a spacing distribution which is a weighted combination of the integer values bounding it (Figure 7-7).

Record P9: KSPACE – <KEYWORD>

The keyword KSPACE precedes the specification of the number and distribution of elements in the K direction in record P9a. Only a single spacing range may be specified using this option. If the keyword and its associated spacing information is omitted, then no respacing is done in the K direction unless provided for by spacing range information with the SECTION keyword, and record P9a is not read. Figure 7-4 for a definition of the K direction.

The spacing specified by KSPACE is overruled by the specification of a spacing range in the section definitions.

Record P9a: NK, KSPACE

This record specifies the number of elements and spacing distribution to be used in respacing the panel in the K direction. This specification is used as a default spacing to be applied over the whole K extent of the panel. This specification will be ignored if the finer spacing control provided in the section point definition (record S6) is used.

NK – Number of elements in K direction (col. 1-10)

The number of elements the panel is to have in the K direction. A maximum of 100 elements may be specified.

KSPACE – Spacing distribution in K direction (col. 11-20)

KSPACE = 0: equal spacing
KSPACE = ±1: cosine spacing
KSPACE = ±2: sine spacing
KSPACE = ±3: equal spacing

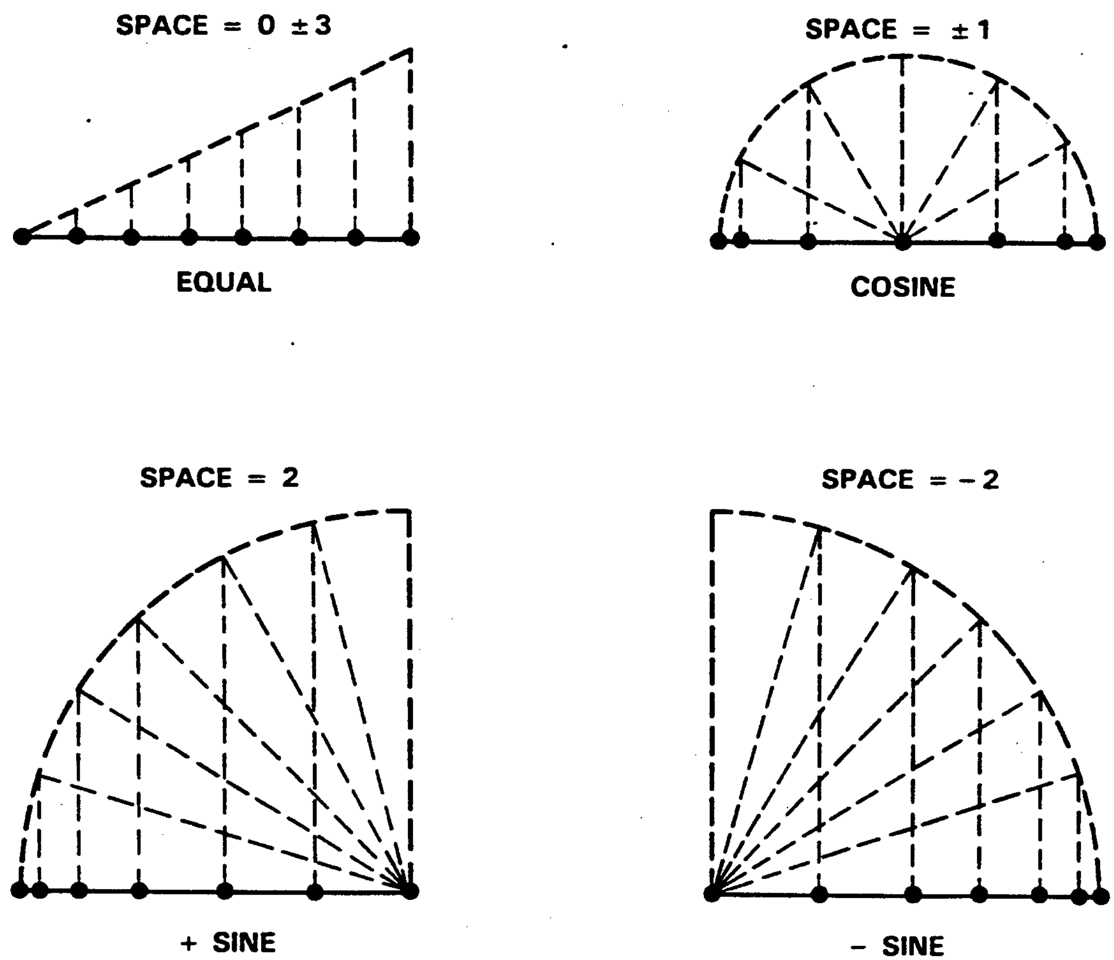


Figure 7-6 Basic Spacing Distributions

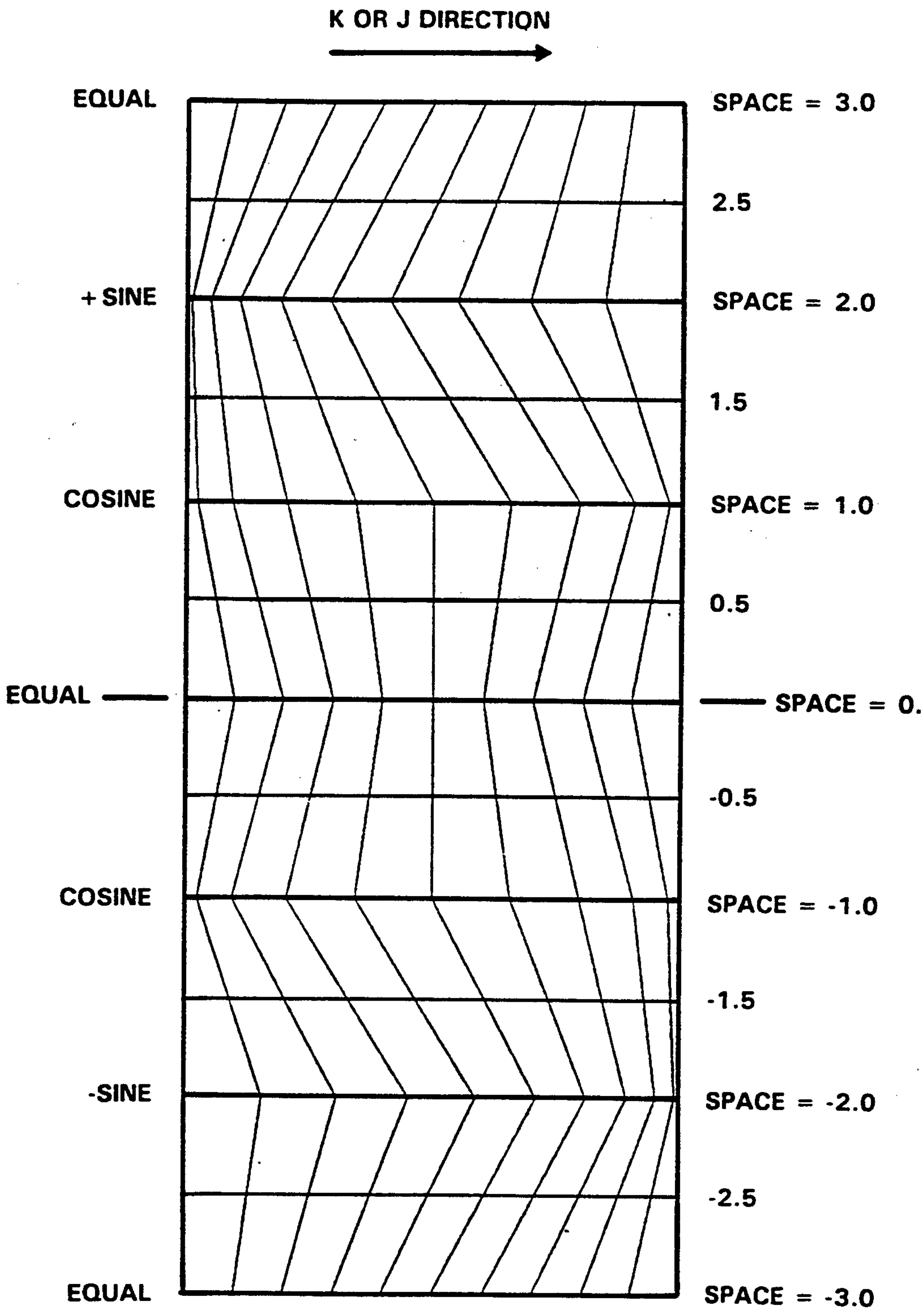


Figure 7-7 Blended Spacing Distributions

Flag specifying the spacing of the element lattice of the panel in the K direction (Fig. 4.6). All spacing distributions are defined with respect to the arc length along the panel surface. Changing the sign of the spacing parameter reverses the direction of the spacing distribution. This has an effect only on sine spacing (a positive value corresponds to elements concentrated at edge 4). Noninteger values of KSPACE produce a spacing distribution which is a weighted combination of the integer values bounding it (Fig. 4.7).

7.8 SECTION DATA

The section data records (S1-S13) contain the geometric definition of a section. Sections are used only to define panels and must follow the panel data records and be contained within a panel definition. At least two sections are expected to be present in each panel to define the panel lattice.

In addition to the basic coordinates of the section points, the section data may also include transformation data to translate, scale, and rotate the input points, and respacing data to alter the distribution of points along the curve defined by the input points and to redistribute points and to redistribute points in the J direction between sections. This permits the sections to be defined in their own coordinate systems, the Section Coordinate System (SCS), and transformed into the panel system. The use of the transformation or respacing capability is entirely optional and, if not used, the input points define a column of lattice points in the K direction of the panel.

Section Data Keywords

SECTION	Initiates section definition
RECTANGULAR	Precedes new section in rectangular coordinates
CYLINDRICAL	Precedes new section in cylindrical coordinates
XAXIS	Specifies X-axis for polar axis
YAXIS	Specifies Y-axis for polar axis
ZAXIS	Specifies Z-axis for polar axis
THREAD	Allows user to specify section thread point
AT	Specifies translation of section
SCALE	Specifies section scaling
TWIST	Specifies section rotation about an axis

Record S1: SECTION – <KEYWORD>

The keyword SECTION precedes the definition of the geometry of each section making up the panel. Each section consists of two parts, a geometric part made up of a list of points and an optional "thread point" specified by the THREAD keyword, and a transformation part in which translation, scaling, and rotation may be applied to the section points. No transformation is done if the second part is omitted or defaulted.

The sections making up the panel may define the panel lattice directly, or may be respaced in either or both of the J or K directions to suit the need. See 6.5 and 6.9 for information on the basic panel and section nomenclature and 6.10 for background on the respacing of sections and panels.

There are three basic options available when the keyword SECTION is used.

1. Define a new section using rectangular coordinates. This may be selected by entering the keyword RECTANGULAR in the next record.

2. Define a new section using cylindrical coordinates. This may be selected by entering the keyword CYLINDRICAL in the next record. Either the X, Y, or Z axis may be used as the polar axis for the cylindrical section. Records S4 and S5 describe this further. Cylindrical sections have special properties if respacing is specified.

3. Reuse the previous section defined. The repeated section may be translated, scaled in X, Y, or Z, and rotated. This option is selected if neither of the above options for defining a new section is selected. In other words, if the keyword, section definition, and thread point are omitted, then the section points, thread point, and any respacing data contained in the section definition of the previous section are used for the repeated section.

Both rectangular and cylindrical sections may be mixed within a panel definition.

Record S2: RECTANGULAR – <KEYWORD>

The keyword RECTANGULAR must precede the definition of a new section in rectangular coordinates.

Record S2a: X, Y, Z, KBREAK, NK, KSPACE

These record(s) contain the X,Y,Z coordinates of the points defining the section and optional flags for controlling the respacing process in the K direction.

X,Y,Z – coordinates of section points (col. 1-10, 11-20, 21-30)

The rectangular coordinates of a set of points defining the section in the section coordinate system.

KBREAK – break point in section curve (col. 31-40)

KBREAK = 0: no break point
KBREAK _ 0: break spacing

This flag specifies that the point be treated as a break point in the section curve. This is important only if the panel is to be respaced in the K direction. A break point means that the curves used to respace the panel in the K direction can have slope discontinuities as they pass through this point. See 6.10 for more information on panel respacing. Note that specifying a break point affects only the definition of the section shape and does not guarantee a lattice location there.

NK – number of elements in K direction (col. 41-50)

A nonzero value for NK specifies this point as a spacing range in the K direction and that NK elements are to be respaced between this point and the end of the spacing range. The spacing range runs from the current section definition point until the next spacing range is defined or the end of the section definition is reached. This guarantees that this pint will be a lattice point if no respacing in J is specified. The specification of a spacing range overrides the default panel spacing established under the keyword KSPACE for that spacing range.

KSPACE – spacing distribution for respacing (col. 51-60)

KSPACE = 0: equal spacing
KSPACE = ±1: cosine spacing
KSPACE = ±2: sine spacing
KSPACE = ±3: equal spacing

This flag specifies the distribution to be used for respacing of the element lattice within this spacing range in the K direction (Figure 7-6). All spacing distributions are defined with respect to the arc length along the panel surface. Changing the sign of the spacing parameter reverses the direction of the spacing distribution. This has an effect only on sine spacing (a positive value corresponds to elements concentrated at edge 4). Noninteger values of KSPACE produce a spacing distribution which is a weighted combination of the integer values bounding it (Figure 7-7).

Record S3: THREAD – <KEYWORD>

The keyword THREAD must precede the specification of the thread point. The keyword and thread point may be omitted, in which case the thread point is simply the origin (0.,0.,0.) of the section coordinates. The thread point may be thought of as a way to reorigin the section coordinates to the location of the thread point. It also has the additional significance that:

Translation of the section, using the AT keyword, is done by moving the thread point to a specified point in the panel coordinate system (PCS).

Scaling of the section, using the SCALE keyword, is done about the thread point.

Rotation of the section, using the TWIST keyword, is done about an axis that passes through the thread point.

The thread point is part of the section definition and is unchanged if the section is repeated.

Record S3a: X0, Y0, Z0 (col. 1-10, 11-20, 21-30)

This record contains the X, Y, Z coordinates for the section thread point. These are the rectangular coordinates of the point in the section coordinate system which is to be moved to the location in the panel coordinate system specified by the AT keyword. The thread point is also the point about which section scaling (using SCALE) and section rotation (using TWIST) is done. If the AT keyword is not used to specify the translation of the thread point is put at the origin of the panel system (default).

Record S4: CYLINDRICAL – <KEYWORD>

The keyword CYLINDRICAL must precede the definition of a new section in cylindrical coordinates. There are three possible choices for the polar axis to be used, the X, Y, or Z axis (Figure 7-8). The following record (S5, S7, or S9) specifies the polar axis to be used.

The coordinates that may be input are the polar coordinates, THETA (angle) and R (radius), and a coordinate in the direction of the polar axis (X, Y, or Z). The plane of the polar coordinates is normal to the polar axis. The polar angle (THETA) is measured positive in a right hand sense about the polar axis and is referenced to the axis following the polar axis in the permutation order XYZXYZ . . . etc. For the three choices of polar axis:

1. XAXIS – The polar angle is referenced to the Y axis. The +Z direction corresponds to +90 deg., +Y to 0 deg., -Z to –90 deg. and –Y to ±180 deg.

2. YAXIS – The polar angle is referenced to the Z axis. The +X direction corresponds to +90 deg., +Z to 0 deg., -X to –90 deg. and –Z to ±180 deg.

3. ZAXIS – The polar angle is referenced to the X axis. The +Y direction corresponds to +90 deg., +X to 0 deg., -Y to –90 deg. and –X to ±180 deg.

If respacing is specified in the section (K) direction, cylindrical sections possess special properties (6.10.4). The specified points are first interpolated in THETA, R, (X, Y, or Z) space by cylindrical arc length to generate an enriched cylindrical section with approximately 100 points. This enriched section is converted to rectangular coordinates and is then treated exactly as if it were a rectangular section for further operations. This procedure is done for two reasons:

1. The first interpolation, done in THETA, R, (X, Y, or Z) coordinates, has the property that only two points need be input to specify a circular arc. Near-circular sections can also be defined by fewer points than would be required for purely rectangular definitions.

2. The enrichment of the input points to approximately 100 interpolated points before respacing ensures the correct slope behavior at the ends of the arc.

The enrichment of the input points in cylindrical coordinates is done in a direction and range given by the end points. For example if the arc is specified from THETA = -90; to THETA = 90, the points will describe a 180 degree arc, through THETA = 0, with THETA increasing. If the arc is specified from THETA = -90 to THETA = -270, the points will describe a 180 degree arc, through THETA = -180, with THETA decreasing. Angles greater than 360 degrees may be used in order to specify the proper angle range.

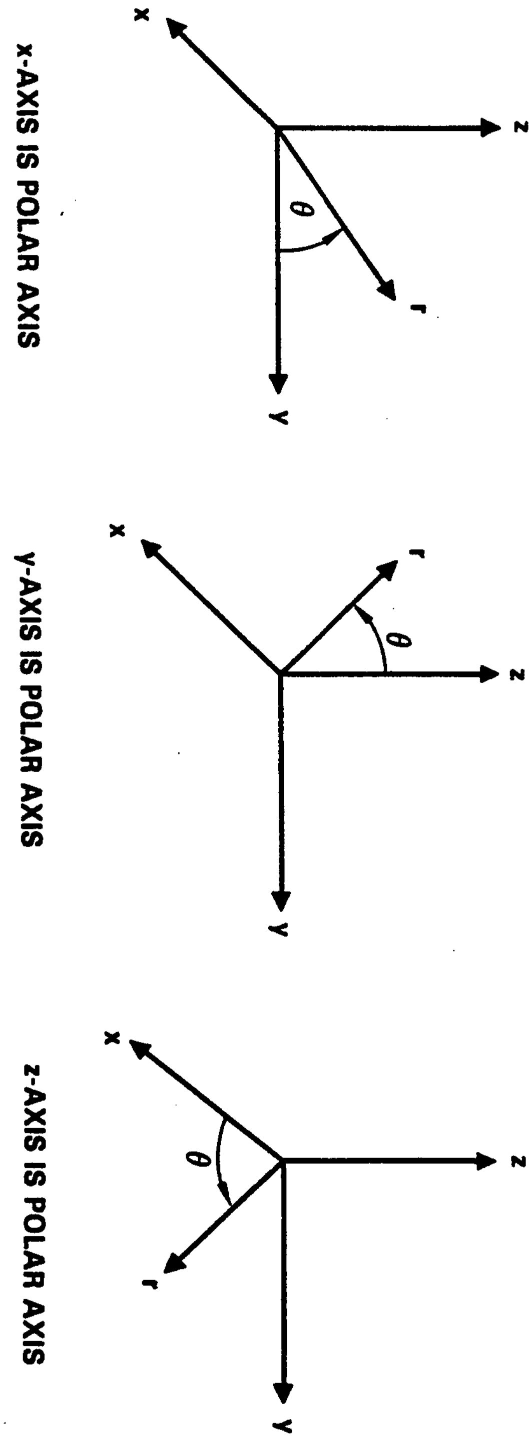


Figure 7-8 Polar Axes for Cylindrical Sections

Record S5, S7, or S9: XAXIS, YAXIS, or ZAXIS – <KEYWORD>

This record specifies the polar axis to be used for the cylindrical section to be defined. If one of the keywords XAXIS, YAXIS, or ZAXIS is not present, the polar axis will default to the X axis.

Record S5a, S7a, or S9a: THETA, R, (X, Y, or Z), KBREAK, NK, KSPACE

These record(s) contain the THETA, R, (X, Y, or Z) coordinates of the points defining the cylindrical section and flags for controlling the respacing process in the K direction. The choice of polar axis (X, Y, or Z) is controlled by record S4a. As many records may be entered as necessary to adequately describe the section shape up to a maximum.of 101.

THETA,R,(X,Y, or Z) – coordinates of the section (col. 1-10, 11-20, 21-30)

The cylindrical coordinates of a set of points defining the section in the section system. The polar angle, THETA, is measured in degrees. See record S4 for conventions for the polar angle.

KBREAK – break point in section curve (col. 31-40)

KBREAK = 0: no break point
KBREAK _ 0: break point

This flag specifies that the point be treated as a break point in the section curve. This is important only if the panel is to be respaced in the K direction. A break point means that the curves used to respace the panel in the K direction can have slope discontinuities as they pass through this point. See 6.10 for more information on panel respacing. Note that specifying a break point affects only the definition of the section shape and does not guarantee a lattice location there.

NK – number of elements in K direction (col. 41-50)

A nonzero value for NK specifies this point as a spacing range in the K direction and that NK elements are to be respaced between this point and the end of the spacing range. The spacing range runs from the current section definition point until the next spacing range is defined or the end of the section definition is reached. This guarantees that this point will be a lattice point if no respacing in J is specified. The specification of a spacing range overrides the default panel spacing established under the keyword KSPACE for that spacing range.

KSPACE – spacing distribution for respacing (col. 51-60)

KSPACE = 0: equal spacing
KSPACE = ±1: cosine spacing
KSPACE = ±2: sine spacing
KSPACE = ±3: equal spacing

This flag specifies the distribution to be used for respacing of the element lattice within this spacing range in the K direction (Figure 7-6). All spacing distributions are defined with respect to the arc length along the panel surface. Changing the sign of the spacing parameter reverses the direction of the spacing distribution. This has an effect only on sine spacing (a positive value corresponds to elements concentrated at edge 4). Noninteger values of KSPACE produce a spacing distribution which is a weighted combination of the integer values bounding it (Figure 7-7).

Record S6, S8, or S10: THREAD – <KEYWORD>

The keyword THREAD must precede the specification of the thread point. The thread point and its keyword may be omitted, in which case the thread point is simply the origin (0.,0.,0.) of the section coordinates. The thread point may be thought of as a way to reorigin the section coordinates to the location of the thread point. It also has the additional significance that:

Translation of the section, using the AT keyword, is done by moving the thread point to a specified point in the panel coordinate system (PCS).

Scaling of the section, using the SCALE keyword, is done about the thread point.

Rotation of the section, using the TWIST keyword, is done about an axis that passes through the thread point.

The thread point is part of the section definition and is unchanged if the section is repeated.

Record S6a, S8a, or S10a: THETA0,R0,(X0,Y0 or Z0) (col. 1-10, 11-20, 21-30)

This record contains the THETA,R,(X,Y, or Z) coordinates for the section thread point. These are the cylindrical coordinates of the point in the section coordinate system which is to be moved to the location in the panel coordinate system specified by the AT keyword. The thread point is also the point about which section scaling (using SCALE) and section rotation (using TWIST) is done. If the AT keyword is not used to specify the translation of the thread point, the thread point is put at the origin of the panel system (default).

Record S11: AT – <KEYWORD>

This record precedes the specification of the section translation and several flags controlling the respacing of the panel in the J direction. The translation of the section definition is done by specifying the location in the panel coordinate system where the section thread point is to be placed. The section points are translated along with the thread point (Figure 6-10). The keyword may be omitted, in which case the thread point is placed at the origin (0.,0.,0.) of the panel coordinate system. Respacing in the J direction is controlled by the JSPACE keyword if this record is omitted.

Record S11a: XTHREAD, YTHREAD, ZTHREAD, JBREAK, NJ, JSPACE

This record specifies the coordinates in the panel coordinate system (PCS) where the section thread point is to be placed and flags for controlling the respacing process in the J direction.

XTHREAD,YTHREAD,ZTHREAD – coordinates of thread point in PCS (col. 1-10, 11-20, 21-30)

The coordinates in the panel coordinate system of the point where the thread point of the section is to be placed. The section points are translated along with the thread point.

JBREAK – break specification for the section (col. 31-40)

JBREAK = 0: no break section
JBREAK _ 0: break section

This flag specifies that the section is to be treated as a break section. This is important only if the panel is to be respaced in the J direction. A break section means that the curves used to respace the panel in the J direction can have slope discontinuities at the point they cross this section. Note that specifying a break section affects only the definition of the panel shape and does not guarantee a lattice line location at this section.

NJ – number of elements in J direction (col. 41-50)

A nonzero value for NJ specifies this section as a spacing range in the J direction and that NJ elements are to be respaced between this section point and the end of the spacing range. The spacing range runs from the current section to the next spacing range or to the last section. This guarantees that points lying on this section curve will be lattice points. The specification of a spacing range overrides the default panel spacing established under the keyword JSPACE for that spacing range.

JSPACE – spacing distribution for respacing (col. 51-60)

JSPACE = 0: equal spacing
JSPACE = ±1: cosine spacing
JSPACE = ±2: sine spacing
JSPACE = ±3: equal spacing

This flag specifies the distribution to be used for respacing of the element lattice in this spacing range in the J direction (Figure 7-6). All spacing distributions are defined with respect to the arc length along the panel surface. Changing the sign of the spacing parameter reverses the direction of the spacing distribution. This has an effect only on sine spacing (a positive value corresponds to elements concentrated at edge 1). Noninteger values of JSPACE produce a spacing distribution which is a weighted combination of the integer values bounding it (Figure 7-7).

Record S12: SCALE – <KEYWORD>

This keyword precedes the specification of scaling factors to be applied to the section points. If the keyword is omitted no scaling is done and record S12a is not read.

The scaling factors are used to scale the section points independently in each of the three panel coordinate directions (see Figure 6-10). This scaling is done about the section thread point.

The keywords SCALE and TWIST may be repeated any number of times, and in any order, to transform the section points. The scaling and twisting operations are done in the order in which they are entered. Note that a scaling specification operates in the PCS on the section points after they have been transformed by any previous scaling and twisting operations.

Record S12a: XCALE,YSCALE,ZSCALE (col. 1-10, 11-20, 21-30)

This record specifies the three scaling factors to be applied to the section points. These are the factors by which the section points are scaled in each of the three coordinate directions. All scaling is done about the section thread point. Blank fields in this card for XSCALE, YSCALE, or ZSCALE default to scale factors of 1.0.

Example:

```
*234567890 1234567890123456789012345678901234567890123456789 0RULER
*
SCALE
*XSCALE   YSCALE   ZSCALE
  1.        3.3      .25
```

This specifies a scaling of the section points about the thread point such that the X values are unchanged, the Y values are scaled up by a factor of 3.3 and the Z values are scaled down by a factor of 4.

Record S13: TWIST – <KEYWORD>

The keyword TWIST precedes the definition of the axis of rotation (in the panel coordinate system) and twist angle used to rotate the section points. If the keyword is omitted no rotation is done and record S13a is not read. Rotations are done about an axis passing through the thread point in the section definition and parallel to the specified direction (see Figure 6-10).

The keywords TWIST and SCALE may be repeated any number of times, and in any order, to transform the section points. The twisting and scaling operations are done in the order in which they are entered. Note that a twisting specification operates in the PCS on the section points after they have been transformed by any previous scaling and twisting operations.

Record S13a: XAXIS, YAXIS, ZAXIS, TWIST

This record specifies the rotation axis direction and rotation angle used to rotate the section points.

XAXIS,YAXIS,ZAXIS – axis of rotation (col. 1-10, 11-20, 21-30)

The direction of the axis of rotation for twisting the section points, specified by a three component direction vector in the panel coordinate system. The rotation is done about an axis passing through the section thread point parallel to this direction.

The components of the direction vector may be treated like direction cosines, although the specified vector is normalized by the program anyway.
TWIST – angle of twist (degrees) (col. 31-40)

The angle in degrees by which the section points are to be rotated about the rotation axis. The positive sense of rotation is determined by the right-hand rule about the axis.

Example:

```
*234567890 1234567890123456789012345678901234567890l234567890RULER
*
TWIST
*XAXIS    YAXIS    ZAXIS    TWIST
  0.        1.       0.       10.
```

This specifies a 10 deg. Rotation of the section points about an axis parallel to the panel Y axis and passing through the thread point.

7.9 PROPELLER DATA

The propeller data (records R1-R4) begins with the first keyword PROPELLER, ending the panel data. Each propeller specification must be preceded by this keyword. If record R1 is omitted, no propeller effects are included and records R1-R4 are not read. A maximum of 8 propellers may be defined.

The propeller data is used by the program to alter the boundary conditions on the configuration surface to reflect the presence of one or more propeller slipstreams that modify the local onset flow. The onset velocity at all control points within a slipstream (upstream and downstream of the propeller plane) will be augmented by the velocity induced at that point by the propeller acting as if it were in free space. No account is taken of the interference effect of the configuration on the propeller. Note that there must be a sufficient control point density within the slipstream to properly reproduce the propeller effects (see Figure 7-10).

The slipstream model is based on the propeller size, direction, applied thrust and torque and a "cross-force" coefficient (modeling the lifting downwash effects from the cross flow on the propeller). The propeller and its slipstream are assumed to rotate on an axis whose direction is specified in the global coordinate system (Figure 7-9). The specification of propeller(s) affects only the flow conditions, not the AIC or solution matrices, and have no impact on restarting the program.

The program does not calculate any forces generated by the propeller(s) themselves, only their effect on the configuration. Velocities calculated at survey points within a propeller slipstream will include the slipstream velocity.

The propeller slipstream definition data is terminated by the keyword SURVEY, the end of file or a blank line.

Propeller Data Keywords

PROPELLER – Initiates propeller definition

Record R1: PROPELLER – <KEYWORD>

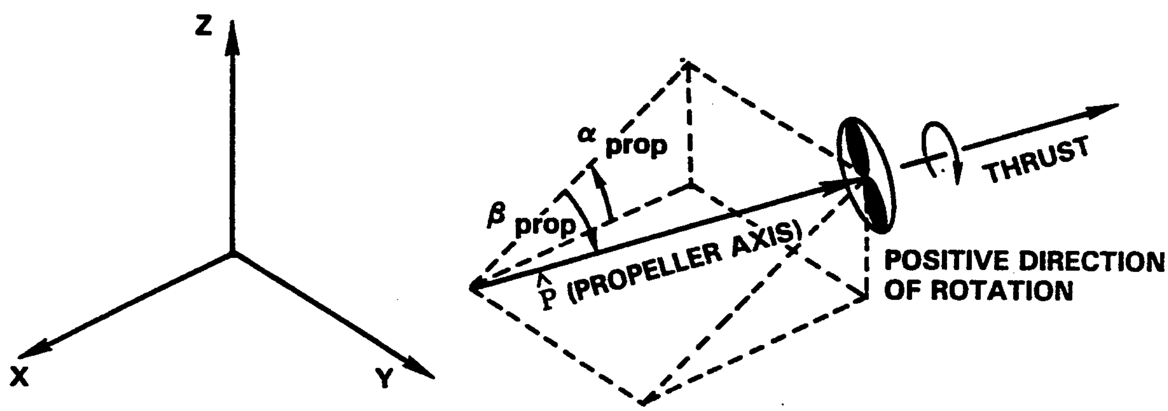
This keyword indicates the beginning of each propeller slipstream specification. It must precede the definition of each (nonimage) propeller. A maximum of 8 propellers may be defined.

Record R1a: XPROP,YPROP,ZPROP,ALPHAPROP,BETAPROP

This record specifies the location of the propeller center and the direction of the propeller axis.

XPROP,YPROP,ZPROP – coordinates of prop center (col. 1-10, 11-20, 21-30)

The global coordinates of the center of the propeller.



PROPELLER AXIS DIRECTION

$$\hat{P} = -\cos\alpha_{prop}\cos\beta_{prop}\,\hat{X} + \sin\beta_{prop}\,\hat{Y} + \sin\alpha_{prop}\cos\beta_{prop}\,\hat{Z}$$

Figure 7-9 Propeller Axis Definition in Global Coordinates

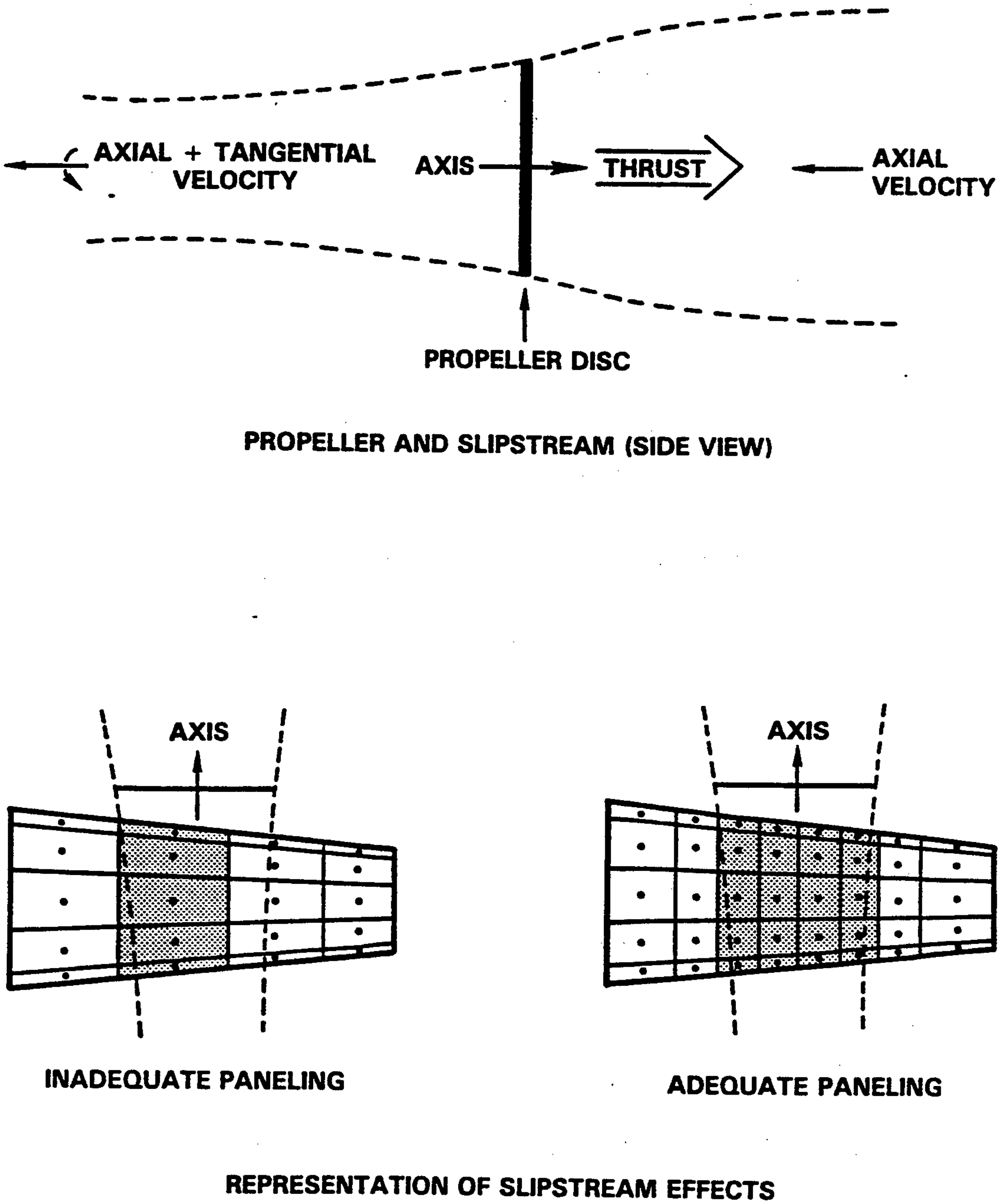


Figure 7-10 Propeller Slipstream

ALPHAPROP,BETAPROP – direction of propeller axis (col. 31-40, 41-50)

These two angles define the direction of the propeller's axis of rotation, referenced to the GCS (see Figure 7-9). The direction of the propeller axis is independent of the free-stream angle of attack or sideslip. The direction of the propeller axis determines both the positive direction of slipstream rotation and the direction of the slipstream velocity (i.e., the propeller axis points in the direction in which thrust is developed, and opposite to the direction of the slipstream axial velocity). For example, a propeller producing thrust in the –X direction, rotating about an axis parallel to the X axis, would be defined with ALPHAPROP = 0, BETAPROP = 0.

Record R1b: RPROP, DIR, PROPIMAGE

This record specifies the radius of the propeller, its direction of rotation and whether an image propeller exists across the X-Z plane of symmetry.

RPROP – propeller radius (col. 1-10)

The radius of the propeller in the same units as the definition of the configuration geometry.

DIR – direction of propeller rotation (col. 11-20)

DIR = +1: positive direction of rotation (default)
DIR = –1: negative direction of rotation

This flag specifies the direction of rotation of the propeller. The positive direction of spin is defined by the right-hand rule about the propeller axis (the direction of the propeller axis is defined by the propeller angles ALPHAPROP and BETAPROP). See Fig. 4.8.1.

PROPIMAGE – propeller image flag (col. 21-30)

PROPIMAGE = 0: propeller has no image
PROPIMAGE = 1: propeller has an image (default)

This flag specifies that the propeller has an image propeller, reflected across the X-Z plane of symmetry. The axis of rotation of the image propeller is defined by the reflection of the axis of the defined propeller. The image propeller's direction of rotation is reversed to preserve lateral symmetry (if the image propeller is to rotate in the same direction, both propellers must be explicitly defined).

Record R1c: CTHRUST, CTORQUE, CCROSSFORCE

This record specifies the non-dimensional coefficients for the thrust and torque applied to the propeller and the cross-force coefficient controlling the downwash generated by the propeller. These values must be determined from the results of other computations or from experiments. They should reflect propeller parameters appropriate to the flow conditions specified.

CTHRUST – propeller thrust coefficient (col. 1-10)

The propeller thrust coefficient defined as the ratio of propeller thrust (force parallel to propeller axis) to the area swept out by the propeller, per unit dynamic pressure (CTHRUST = THRUST/ (AREA*DYNAMIC PRESSURE)).

CTORQUE – propeller torque coefficient (col. 11-20)

The propeller torque coefficient defined as the ratio of propeller torque (moment about propeller axis) to the area swept out by the propeller times the diameter of the propeller, per unit dynamic pressure (CTORQUE = TORQUE/(AREA*DYNAMIC PRESSURE*DIAMETER)).

CCROSSFORCE – propeller cross-force coefficient (col. 21-30)

The propeller cross-force coefficient defined as the ratio of propeller cross-force (force normal to propeller axis) to the area swept out by the propeller per unit dynamic pressure (CCROSSFORCE = CROSSFORCE/(AREA*DYNAMIC PRESSURE)). Note that the cross-force corresponds to the lift and/or side force on the propeller and is zero when the propeller axis is aligned with the free-stream. This parameter should be set based on propeller theory or experimental correlation.

7.10 SURVEY DATA

The survey data (records V1-V5) begins with the keyword SURVEY, ending the panel data or the propeller data, and initiating the reading of records defining the flow survey locations. The flow survey is optional, and is omitted if the keyword SURVEY is not present. The flow survey definition data is terminated by the end of file or a blank line.

The user must be careful if the flow survey locations are near the configuration surface. This is due to the constant strength doublet panels used on the body surface, which may induce unrealistic velocities if survey points are placed closer than an element length of the surface. See Ref. 4 for additional information on flow surveys.

Survey Data Keywords

SURVEY	Initiates survey definition
POINT	Precedes definition of survey points
PANEL	Initiates survey panel definition
CONTROL	Indicates control points as survey points
LATTICE	Indicates lattice points as survey points

Record V1: SURVEY – <KEYWORD>

The keyword SURVEY indicates the beginning of the definition of flow survey points at which the program is to calculate flow quantities. Any pending lists are terminated by this keyword.

The locations of the survey points may be specified in two ways.

1. The survey points may be defined by a list of arbitrary points. This option may be selected by entering the keyword POINTS in the next record.

2. A survey grid may be defined by a survey panel. A survey panel is specified in the same way as a configuration panel, taking advantage of the already established capability to generate surface grids of arbitrary shape. This option may be selected by entering the keyword PANEL in the next record. Either the control points or the lattice points of the survey panel may be selected as the survey locations, by entering the keyword CONTROL or the keyword LATTICE in the record following the PANEL keyword.

The two methods of defining survey locations may be used in combination, with one limitation. The keyword POINT, preceding a list of arbitrary survey points, can only be used once in the dataset and must come before a PANEL keyword. In other words there can only be one list of arbitrary survey points and it must come before any survey panels. As many survey panels as required may be defined, up to a maximum of 25. The total number of survey points (including the points used by survey panel grids) may not exceed 1000.

Record V2: POINT – <KEYWORD>

This keyword precedes records specifying an arbitrary list of survey points. This list may be terminated by the keyword PANEL or the end of file (or a blank line).

Record V2a: XSURV,YSURV,ZSURV (col. 1-10, 11-20, 21-30)

This record contains the global coordinates of points at which flow quantities are to be calculated. One point is entered per record, up to the maximum of 1000 survey points. Survey points should not lie on or very close to the surface of the configuration.

Record V3: PANEL – <KEYWORD>

This keyword initiates the definition of a survey panel. The survey panel is defined in the same way as a normal body panel, with the addition of another keyword. The choice of control points or lattice points as the survey points is determined by the keyword entered on the following record.

Record V4: CONTROL – <KEYWORD>

This keyword precedes the definition of a survey panel whose control points (center points) form the survey points. The survey panel is then defined in the same way as any other panel, using records P2-P9 and S1-S13. Panel parameters which are meaningless may be defaulted or entered with zeros. The number of survey points defined by the panel is the same as the number of control points in the panel (NJ x NK points).

Record V5: LATTICE – <KEYWORD>

This keyword precedes the definition of a survey panel whose lattice points (corner points) form the survey points. The survey panel is then defined in the same way as any other panel, using records P2-P9 and S1-S13. Panel parameters which are meaningless may be defaulted or entered with zeros. The number of survey points defined by the panel is the same as the number of lattice points in the panel (NJ + 1 x NK + 1 points).

7.11 EXAMPLE DATASETS

Four example datasets are presented in this section to illustrate the input dataset structure to QUADPAN. The first two datasets pertain to a very simple wing and demonstrate the use of rectangular sections and the respacing options. The third dataset contains an example of the use of cylindrical sections. The final dataset illustrates additional program capabilities on a more complex wing/body/fin configuration.

7.11.1 Example 1 – Rectangular Wing

The first example is a simple symmetrical cross section wing of rectangular planform, as illustrated in Figure 7-11. The wing aspect ratio is 2.0 with a chord of 1.0 and a 20% thickness/chord ratio.

This example illustrates the layout of the model geometry into panels and sections, and the assembly of that information into QUADPAN format. This dataset does not contain any transformations or respacing information, so the corner points of the resulting mesh are identical to the points which are input. The defined lattice will be far too crude to use, but will be respaced to more reasonable density in the next example.

The dataset is listed in Figure 7-13, and is shown with record numbers in columns 72-80 for instructional purposes. These are not used by the program and are not included in actual datasets. Comments have been used liberally to label fields in the input data, and are recommended for all datasets.

7.11.1.1 Model Layout –

One of the first things that must be done in order to create a QUADPAN model is panel layout. Referring to the drawing of the wing, the basic logic goes as follows:

- The configuration is laterally symmetric, so only the right side will be modeled and image panels will be used to represent the left side.
- In order to eliminate holes in the geometry, the wingtip will be closed by a simple flat closure.
- In order to have the wing produce lift a vortex wake, whose forward edge is coincident with the wing trailing edge and whose aft edge is far downstream, will be used.
- The presence of the wake panel at the trailing edge dictates that the wing must have panel edges at the trailing edge (panels must only abut at their edges). In addition, the wing will be made up of upper and lower surface panels to simplify the arrangement of the elements and strips of elements in the output.

7.11.1.2 Global Data –

The first section of the QUADPAN dataset is the global data, consisting of records G1-G5. The first record is the case title, which should be descriptive of the geometry. In the case of restarts, this information cannot change from run to run. The next record is the run ID, describing the flow conditions, or additional data about the run. The Mach number for this case is 0.2 and run flags are set to give a check run with all printout and a dump file. The abutment parameters are set to give an absolute matching tolerance of 0.002 for the panel abutment search. This value is somewhat large considering the size of this wing (X,Y,Z are all of order 1.0), but is sufficient for the sparse paneling used.

The reference quantities for this case are given in record G4, including the wingspan (which is 2.0 for the full wing) and the moment reference center (which has been placed on the quarter chord. The two flow conditions to be analyzed (after check runs are done and the run flag is changed to 1) are angles of attack of 0.0 and 5.0 degrees.

7.11.1.3 Defining The Geometry –

Four panels will be created to represent this geometry. These are shown in Figure 7-12.

Panel 1 – Upper Wing (from centerline to tip)
Panel 2 – Lower Wing (from centerline to tip)
Panel 3 – Wing Tip Closure (closes off tip)
Panel 10 – Wing Wake (with Kutta condition at wing T.E.)

The panel ID numbers may be selected by the user to aid in panel identification. In this case panel ID numbers 1-3 will be used for the wing and tip panels and panel ID number 10 will be used for the wake panel.

Panel 1 – Upper Wing

Each panel definition begins with the keyword PANEL, followed by the panel ID number and title. The next record (P2) selects the type of boundary condition, exterior surface, disposition of forces and existence of an image for the panel. In this case, the TYPE is set to 0 for a body panel with a hydrodynamic boundary condition (no flow through the surface). The WET flag is set to 1 so that the panel "positive" surface will correspond to the exterior surface. The FORCE and IMAGE flags are set to include panel forces into the total forces and to generate an image panel across the plane of symmetry.

The panel geometry can be input in several ways, but the most convenient method is to describe the panel such that the elements will be grouped into chordwise strips (leading edge to trailing edge) going from root to tip. In this case, with only one spanwise element, the latter distinction is lost, but example 2 will expand on this. This grouping is purely a function of the ordering of the input points and is only for the purpose of arranging the program output into a convenient form.

To group the elements in this fashion, the panel geometry will be described with chordwise sections. Two sections are needed to describe the wing panel – one at the centerline root, and one at the tip. This is illustrated in Figure 7-12. The keyword SECTION is used to start the section definition, along with the keyword RECT to indicate that a new section is being input in X, Y, Z coordinates. The three points used to describe the section are simply the lattice points on the upper wing running from leading edge to trailing edge along the centerline (Y = 0). The second section consists of the corresponding points at the tip (Y = 1.0).

The two sections define the K and J directions for the panel (see Fig. 4.10). The K direction always goes in the direction of the points in the section, while the J direction always runs from first to last section. The panel "positive" surface is defined by the cross product of the K and J directions, and corresponds to the exterior surface for WET = 1. Since this is indeed the exterior surface, the choice of WET = 1 is correct.

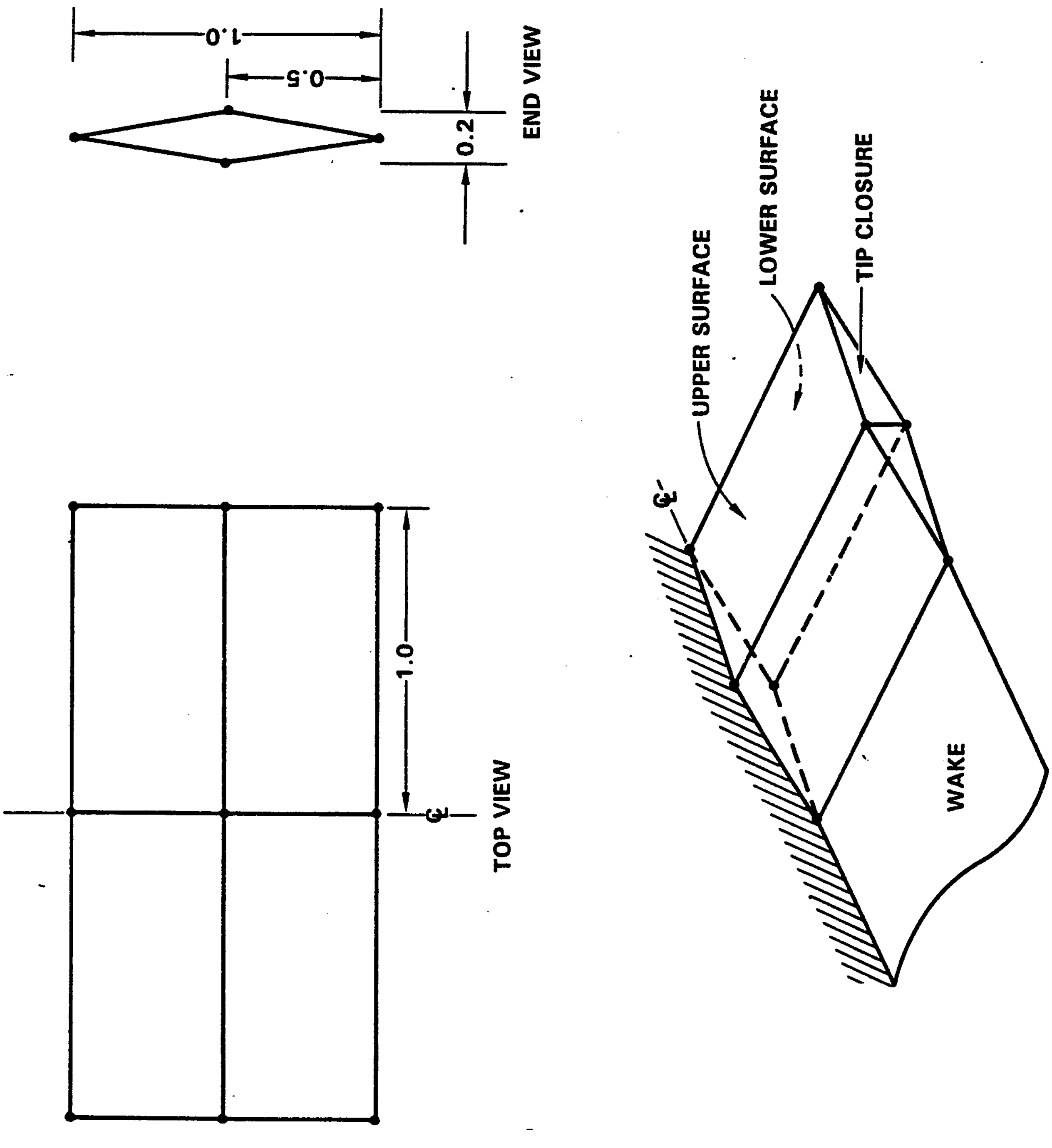


Figure 7-11 Rectangular Wing

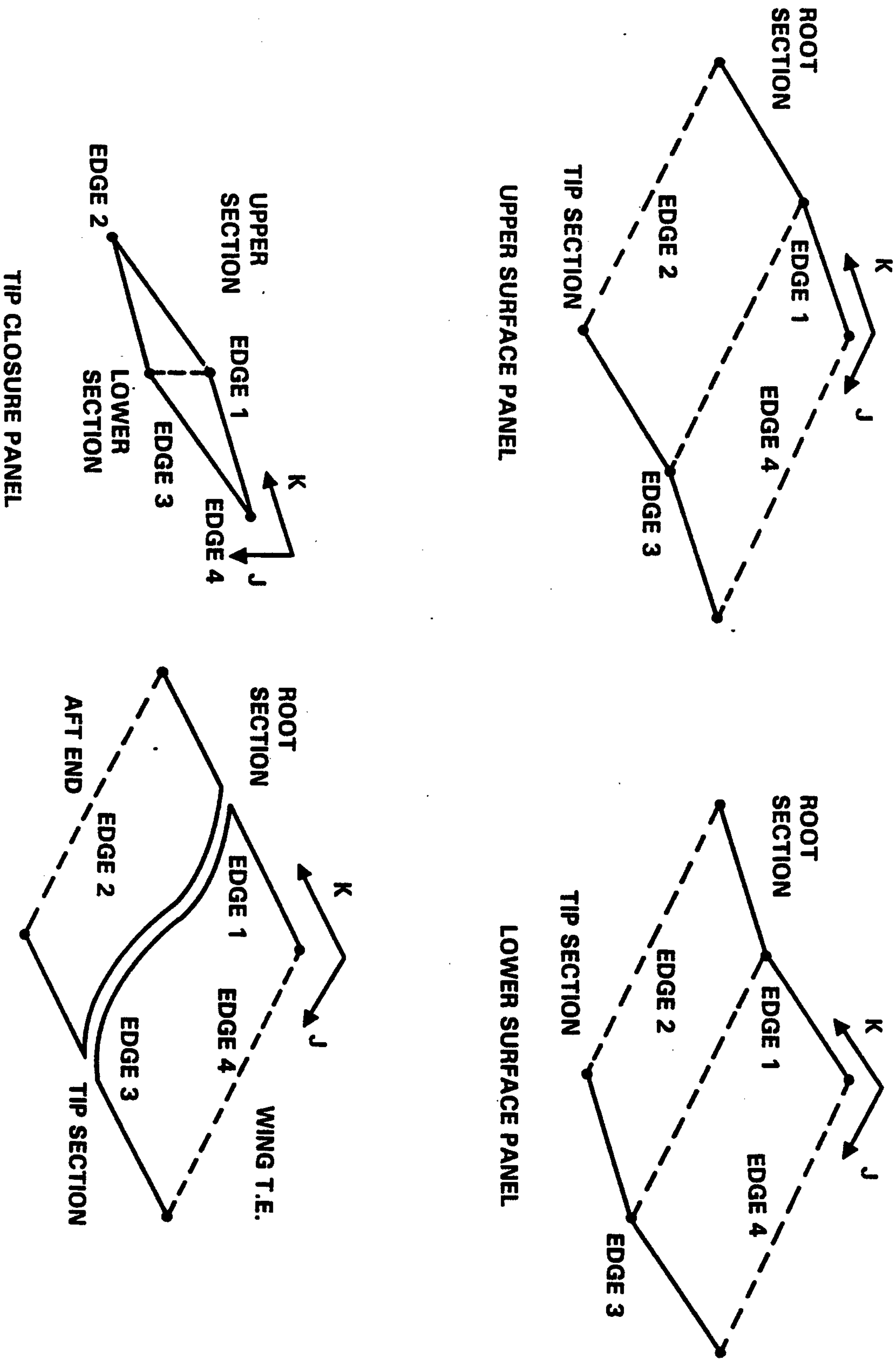


Figure 7-12 Layout of Panels

```
***********************************************************************
RECTANGULAR WING (SYM. SECTION)   5/12/83  NO RESPACING        EXAMPLE 1      G1
THIS IS A GEOMETRY GENERATION RUN WITH CHECKING                               G2
***********************************************************************
*MACH      RUN       PRINT     DUMP      ABSTOL    RELTOL
.2         0         0         1         .002      .1                          G3
*  REFERENCE DATA
*SREF      CBAR      WSPAN     XBAR      YBAR      ZBAR
2.0        1.0       2.0       .25       0.        0.                          G4
*  FLOW CONDITIONS
*ALPHA     BETA      OMEGAX    OMEGAY    OMEGAZ    VINF
0.                                                                             G5
5.
*****************************************
PANEL                                                                          P1
1          UPPER WING                                                          P2
*TYPE      WET       FORCE     IMAGE
0          1         1         1                                               P3
*
SECTION                                                                        S1
RECT                                                                           S2
*X         Y         Z         KBREAK    NK        KSPACE
0.0        0.0       0.0                                                       S2a
0.5        0.0       0.1                                                       S2a
1.0        0.0       0.0                                                       S2a
SECTION                                                                        S1
RECT                                                                           S2
*X         Y         Z         KBREAK    NK        KSPACE
0.0        1.0       0.0                                                       S2a
0.5        1.0       0.1                                                       S2a
1.0        1.0       0.0                                                       S2a
*****************************************
PANEL                                                                          P1
2          LOWER WING                                                          P2
*TYPE      WET       FORCE     IMAGE
0          -1        1         1                                               P3
*
SECTION                                                                        S1
RECT                                                                           S2
*X         Y         Z         KBREAK    NK        KSPACE
0.0        0.0       0.0                                                       S2a
0.5        0.0       -0.1                                                      S2a
1.0        0.0       0.0                                                       S2a
SECTION                                                                        S1
RECT                                                                           S2
*X         Y         Z         KBREAK    NK        KSPACE
0.0        1.0       0.0                                                       S2a
0.5        1.0       -0.1                                                      S2a
1.0        1.0       0.0                                                       S2a
*****************************************
PANEL                                                                          P1
3          TIP CLOSURE                                                         P2
*TYPE      WET       FORCE     IMAGE
0          1         1         1                                               P3
*
SECTION                                                                        S1
RECT                                                                           S2
*X         Y         Z         KBREAK    NK        KSPACE
0.0        1.0       0.0                                                       S2a
0.5        1.0       0.1                                                       S2a
```

Figure 7-13 Example 1 Dataset (1 of 2)

```
1.0       1.0       0.0                                                S2a
SECTION                                                                S1
RECT                                                                   S2
*X        Y         Z         KBREAK    NK        KSPACE
0.0       1.0       0.0                                                S2a
0.5       1.0       -0.1                                               S2a
1.0       1.0       0.0                                                S2a
****************************************
PANEL                                                                  P1
10        WING WAKE                                                    P2
*TYPE     WET       FORCE     IMAGE
4         1         0         1                                        P3
*
SECTION                                                                S1
RECT                                                                   S2
*X        Y         Z         KBREAK    NK        KSPACE
1.0       0.0       0.0                                                S2a
100.0     0.0       0.0                                                S2a
SECTION                                                                S1
RECT                                                                   S2
*X        Y         Z         KBREAK    NK        KSPACE
1.0       1.0       0.0                                                S2a
100.0     1.0       0.0                                                S2a
```

Figure 7-14 Example 1 Dataset (2 of 2)

Panel 2 – Lower Wing

This panel is described in an identical fashion as the upper wing panel to put the strips and elements in the same order. As can be seen in Figure 7-12, the panel "positive" surface, defined by the cross product of the K and J directions points upwards. This is incorrect – the exterior surface for this panel is the lower surface. The correct outward surface is selected by setting WET = -1, informing the program to use the opposite surface for the exterior.

Panel 3 – Tip Closure

The tip closure panel is described by two sections, corresponding to the outboard section of the upper wing panel and the outboard section of the lower wing panel. The panel "positive" surface, given by the cross product of the K and J panel directions, or the direction of the 1234 panel edge order, corresponds to the exterior surface. The choice of WET = 1 is correct for the tip panel.

Panel 10 – Wing Wake

The wing wake panel uses an ID number of 10 to distinguish it from the body panels (this can be convenient in large datasets, to identify wake panels with a different set of numbers for clarity). The wing wake panel is used to enforce the Kutta condition on the wing by placing one of its edges coincident with the trailing edge. Any of the four edges of the panel could be used for establishing the Kutta condition, so long as the TYPE specifies the edge to be used. In order to keep the wake panel sections streamwise (for our convenience, to match the other panels) a TYPE = 4 will be used for this panel. In this case edge 4, corresponding to the first point in each section, will be placed at the wing trailing edge. The opposite edge of the wake will be placed 100 semispans downstream to avoid interference with the lifting wing. The wake panel is described by two sections, a centerline section and a tip section. Each section consists of one point on the wing trailing edge and another point at the aft edge of the wake.

The panel WET for a wake does not strictly need to be set, as a wake is wetted on both its "positive" and "negative" surface and the program will hook everything up correctly regardless of how WET is input. The WET flag does control the sense of positive GAMMA (the doublet strength) that comes out of the program, and can be set to give positive GAMMA for positive lift if the panel "upper" surface (as set by (K x J) and WET) corresponds to the direction of positive lift. This is a matter of convenience only.

7.11.2 Example 2 – Rectangular Wing With Respacing

The second example is based on the same symmetrical wing with rectangular platform used in example 1, as illustrated in Figure 7-15. The purpose of the second example dataset is to illustrate the use of panel respacing, repeated sections, and section transformations.

The layout of the model geometry is identical to the previous example and will consist basically of the same panels and sections. The difference between these datasets is that the QUADPAN panel respacing options have been used in example 2 to make a denser lattice of elements form the same geometry data. In this case, the lattice will be respaced to give:

- Three equally spaced elements spanwise on the wing.
- Five cosine spaced elements chordwise on the wing. This concentrates elements toward the leading and trailing edges.

This lattice is still not sufficiently refined for accurate analysis, where normally at least 8 elements should be used to define one side of a lifting surface.

The dataset is listed in Figure 7-16, and is shown with record numbers in columns 72-80 for instructional purposes. These are not used by the program and are not usually included in actual datasets. Comments have been used liberally to label fields in the input data, and are recommended for all datasets.

7.11.2.1 Model Layout -

The basic layout of the model is unchanged form the previous example. Four panels are used to represent the wing upper and lower surface, the tip closure, and the wing wake, The difference for this dataset is that panel respacing must be added to create a denser lattice from the sparse input geometry. It is important that all adjacent panel edges have the same number and distribution of elements. This matching of elements at panel edges is done to avoid holes in the model, and to maximize the accuracy of the analysis. It is also particulary important that the spacing of elements on the wing wake match the elements on the wing at the trailing edge. In addition, the respacing applied to the wing upper and lower surfaces must be done such that the lattice elements on the upper and lower surfaces near the trailing edge be identical (or nearly identical) in size. This is done to maximize the accuracy of the Kutta condition.

With only four panels to consider, this matching of elements at panel edges is simple. Since the points used to define the geometry match exactly at panel edges, all that is necessary is that the respacing used must agree in number and distribution.

7.11.2.2 Global Data –

The first section of the QUADPAN data set shown in Figure 7-16 is the global data, consisting of records G1-G5. This is the same information as provided for the first example dataset, except that the case title has been changed to reflect the updated geometry. The run flag has been changed to 1 to run the flow analysis.

7.11.2.3 Respacing the Geometry

The same four panels from example 1 will be used to represent this geometry. These are shown in Figure 7-12.

Panel 1 – Upper Wing (from centerline to tip)
Panel 2 – Lower Wing (from centerline to tip)
Panel 3 – Wing Tip Closure (closed off tip)
Panel 10 – Wing Wake (with Kutta condition at wing T.E.)

Panel 1 – Upper Wing

The panel definition for the upper wing is identical with that in the previous example with the exception of the JSPACE and KSPACE panel respacing specifications (records P8, P8A, and P9, P9A). These have been added to the panel definition immediately prior to the first SECTION keyword.

As discussed above, panel respacing will be used on the wing to obtain three elements equally spaced spanwise, and five elements cosine spaced chordwise. Since the directions spanwise and chordwise have no direct meaning in QUADPAN, where a panel can be described in a number of different ways, the user must first decide what spacing must be applied along the J and K panel directions. In this case spanwise corresponds to the J panel direction, and chordwise to the K panel direction (see Figure 7-12).

The optional keywords JSPACE and KSPACE are used to respace the panel. The spacing distribution for the J direction is specified to be 0. for equal spacing spanwise, and the spacing distribution for the K direction is specified to be 1 for cosine spacing chordwise (actually this gives cosine spacing based on arc length along the wing surface).

The respaced panel is shown in Figure 7-15. Notice that the wing no longer has a diamond cross section, instead it has a smooth parabolic surface. This is due the respacing process, where a spline curve has been fit through the input points to generate the respaced points. If it was desired to preserve the diamond cross section of the wing, a break point should be specified for the middle point in each wing section to permit a slope discontinuity there.

Panel 2 – Lower Wing

The lower surface panel is defined in a similar manner as it was the first example dataset, except that a repeated section is used to obtain the tip section by translation of the root section. In addition panel respacing has been specified for both the K and J panel directions to match the upper wing panel defined above. This respacing has been done differently from that used for the upper wing to illustrate the use of spacing intervals.

Looking at the lower wing panel definition in Figure 7-16, the first section is the same as that used in the previous dataset, except that section respacing has been specified by placing a 5 in the fifth field and a 1 in the sixth field of the record containing the first point in the section. This specifies that a spacing interval start with this point and that five elements be respaced in this interval using a cosine spacing distribution. The spacing interval extends to the next spacing interval specification, of to the end of the section. In this case, there are not further specifications so the interval goes to the end of the section.

The first section ends with the keyword, AT, which is used here to specify respacing in the J direction (from section to section). The AT keyword is normally used to define translation of the section points by moving the thread point (which defaults to the origin of the section coordinates if no THREAD keyword and coordinates are given) and the section points to correspond to the point specified by the AT keyword. In this case, since AT specifies (0.,0, 0.), no translation is done. The 3 in the fifth field and the 0. in the sixth field of the AT specification defines the beginning elements will be respaced with equal spacing distribution. The J respacing interval extends to the next specified interval or the last section, as it does in this case.

The second section is specified using a repeated section. This is done by omitting the keyword RECT or CYL and the section points. The previous section definition will be used, including all respacing information for the K direction. In order to move the previous section definition, which was defined at the wing root, the AT point for this section specifies that the section thread point (the origin, by default) be moved to the leading edge of the tip (0.,1.,0.).

Panel 3 – Tip Closure

As in the first example dataset, the tip closure panel is described by two sections, one from the outboard section of the upper wing panel and one from the outboard section of the upper wing panel and one from the outboard section of the lower wing panel. For this example the tip panel will remain a simple closure panel, and little or no detail of the tip flow is desired. In this case, only respacing in the chordwise direction is necessary. Since the chordwise direction corresponds to the K direction, this is done by adding the keyword KSPACE, and respacing with five elements using a cosine distribution.

Panel 10 – Wing Wake

The wing wake panel is identical with the wake panel in the first example with the addition of a respacing specification in the J direction. The elements on the wake must match the upper and lower surface wing elements to establish the Kutta condition on the wing trailing edge. Since the trailing edge of the wing corresponds to edge 4 on the wake, running in J direction, the keyword JSPACE is used to respace the wake panel, Respacing is specified to put three elements with equal spacing across the width of the wake panel.

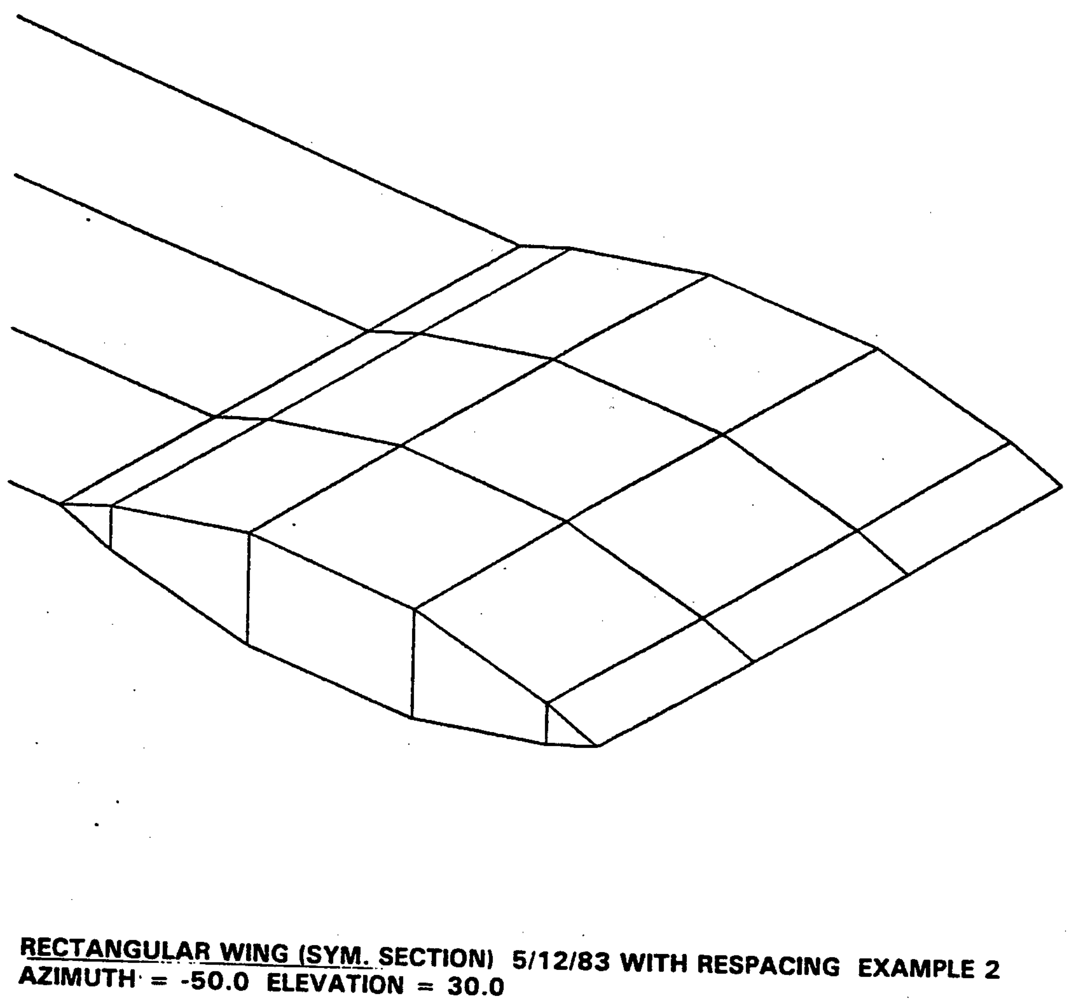


Figure 7-15 Rectangular Wing With Respacing

```
**************************************************************************
RECTANGULAR WING (SYM. SECTION)   5/12/83  WITH RESPACING    EXAMPLE 2      G1
ALPHA SWEEP (SYMMETRIC FLOW)               DUMP FILE - EX2B.DUMP.DATA       G2
**************************************************************************
*MACH       RUN        PRINT      DUMP       ABSTOL     RELTOL
.2          1          0          1          .002       .1                  G3
*  REFERENCE DATA
*SREF       CBAR       WSPAN      XBAR       YBAR       ZBAR
2.0         1.0        2.0        .25        0.         0.                  G4
*  FLOW CONDITIONS
*ALPHA      BETA       OMEGAX     OMEGAY     OMEGAZ     VINF
0.                                                                          G5
5.
******************************************
PANEL                                                                       P1
1           UPPER WING                                                      P2
*TYPE       WET        FORCE      IMAGE
0           1          1          1                                         P3
*
JSPACE                                                                      P8
*NJ         JSPACE
3           0.                                                              P8a
KSPACE                                                                      P9
*NK         KSPACE
5           1.                                                              P9a
*
SECTION                                                                     S1
RECT                                                                        S2
*X          Y          Z          KBREAK     NK         KSPACE
0.0         0.0        0.0                                                  S2a
0.5         0.0        0.1                                                  S2a
1.0         0.0        0.0                                                  S2a
*
SECTION                                                                     S1
RECT                                                                        S2
*X          Y          Z          KBREAK     NK         KSPACE
0.0         1.0        0.0                                                  S2a
0.5         1.0        0.1                                                  S2a
1.0         1.0        0.0                                                  S2a
******************************************
PANEL                                                                       P1
2           LOWER WING                                                      P2
*TYPE       WET        FORCE      IMAGE
0           -1         1          1                                         P3
*
SECTION                                                                     S1
RECT                                                                        S2
*X          Y          Z          KBREAK     NK         KSPACE
0.0         0.0        0.0                   5          1.                  S2a
0.5         0.0        -0.1                                                 S2a
1.0         0.0        0.0                                                  S2a
AT                                                                          S6
*X          Y          Z          JBREAK     NJ         JSPACE
0.0         0.0        0.0                   3          0.                  S6a
*
SECTION                                                                     S1
AT                                                                          S6
*X          Y          Z          JBREAK     NJ         JSPACE
0.0         1.0        0.0                                                  S6a
******************************************
```

Figure 7-16   Example 2 Dataset (1 of 2)

```
PANEL                                                              P1
3         TIP CLOSURE                                              P2
*TYPE     WET       FORCE     IMAGE
0         1         1         1                                    P3
*
KSPACE                                                             P9
*NK       KSPACE
5         1.                                                       P9a
*
SECTION                                                            S1
RECT                                                               S2
*X        Y         Z         KBREAK    NK        KSPACE
0.0       1.0       0.0                                            S2a
0.5       1.0       0.1                                            S2a
1.0       1.0       0.0                                            S2a
*
SECTION                                                            S1
RECT                                                               S2
*X        Y         Z         KBREAK    NK        KSPACE
0.0       1.0       0.0                                            S2a
0.5       1.0       -0.1                                           S2a
1.0       1.0       0.0                                            S2a
*******************************************
PANEL                                                              P1
10        WING WAKE                                                P2
*TYPE     WET       FORCE     IMAGE
4         1         0         1                                    P3
*
JSPACE                                                             P8
*NJ       JSPACE
3         0.                                                       P8a
*
SECTION                                                            S1
RECT                                                               S2
*X        Y         Z         KBREAK    NK        KSPACE
1.0       0.0       0.0                                            S2a
100.0     0.0       0.0                                            S2a
*
SECTION                                                            S1
RECT                                                               S2
*X        Y         Z         KBREAK    NK        KSPACE
1.0       1.0       0.0                                            S2a
100.0     1.0       0.0                                            S2a
```

Figure 7-17 Example 2 Dataset (2 of 2)

7.11.3 Example 3 – Cylindrical Fuselage

The third example is a fuselage-like body of revolution consisting of a cylindrical center section with ellipsoidal fore and aft bodies. The ellipsoidal ends are each half of an ellipsoid whose length/diameter ratio is 2:1. The configuration is illustrated in Figure 7-18. The purpose of this example dataset is to illustrate the use of cylindrical section to define panels.

The dataset is listed in Figure 7-20, and is shown with record numbers in columns 72-80 for instructional purposes. These are not used by the program and are not usually included in actual datasets. Comments have been used liberally to label fields in the input data, and are recommended for all datasets.

7.11.3.1 Model Layout –

The first decision to be made in model layout for this (or any) configuration concerns the use of symmetry. In this case, the fuselage is laterally symmetric, so only the right side will be modeled. Image panels will be used to represent the left side.

The fuselage geometry could be modeled in several ways:

- As three panel - a forebody, a centerbody, and an aftbody. This has the advantage that, if respacing is desired, the centerbody shape and lattice loctions are not affected by the splines and spacing distributions used to represent the forebody and aftbody. The disadvantage of this model is that the dataset is longer.

- As one panel – the forebody, centerbody, and aftbody are all combined into one panel. In order to preserve control of the shape and lattice spacing, break sections and spacing sections could be used. This has the advantage that it produces a shorter dataset.

For this example, the single panel representation will be used. The lattice spacing for this case will be selected so respacing gives:

- Six equally spaced elements around the right half of the circular cross section.

- A spacing distribution along the length of the body that gives six elements on the forebody (concentrated near the nose), four elements equally spaced on the centerbody, and six elements on the aftbody (concentrated near the aft end).

This element distribution should give reasonably accurate results for this geometry.

7.11.3.2 Global Data –

The global data for this dataset is similar to that for the first two examples, consisting of records G1-G5. The case title has been changed to be descriptive of the fuselage geometry. This case is set up for incompressible flow (MACH=0.) and the run flags are set to give a geometry generation run with minimum printout and a dump file. This run configuration should be used initially to validate the dataset in conjunction with the plotting utility. The abutment parameters are set to give an absolute matching tolerance of 0.002 for the panel abutment search.

The reference quantities for this case are given in record G4. The frontal area has been used for the reference area, and 1.0 for the remaining reference quantities. The moment reference center has been placed at the center of the fuselage. A single flow condition is specified – an angle of attack of 0.0 degrees.

7.11.3.3 Defining The Geometry –

As discussed above, a single panel will be created to represent the fuselage geometry. The completed mesh is shown in Figure 7-19 and the dataset is illustrated in Figure 7-20. A large explanatory comment has been used in the dataset for documentation. This can be helpful, especially when the paneling or spacing is difficult to follow.

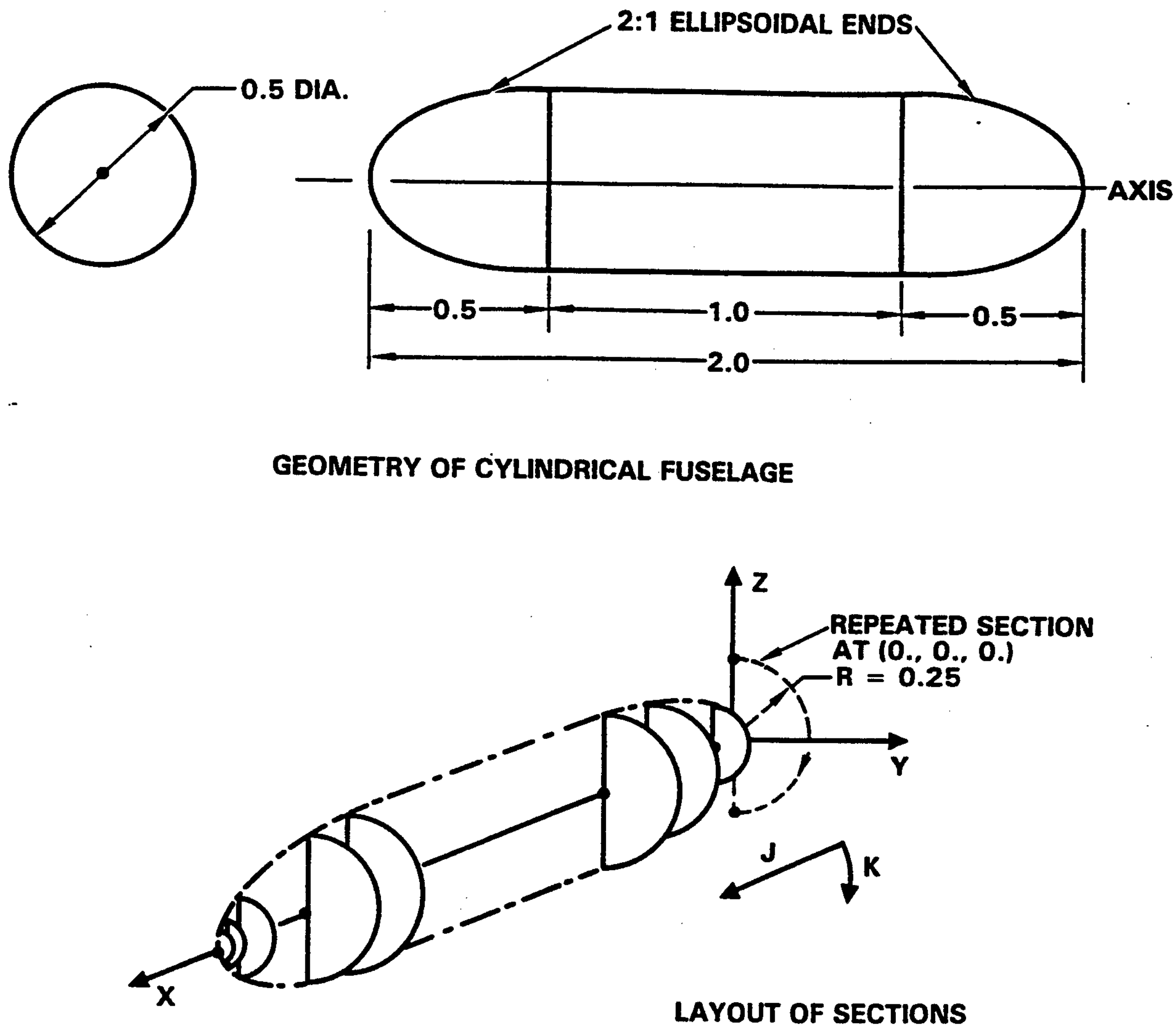
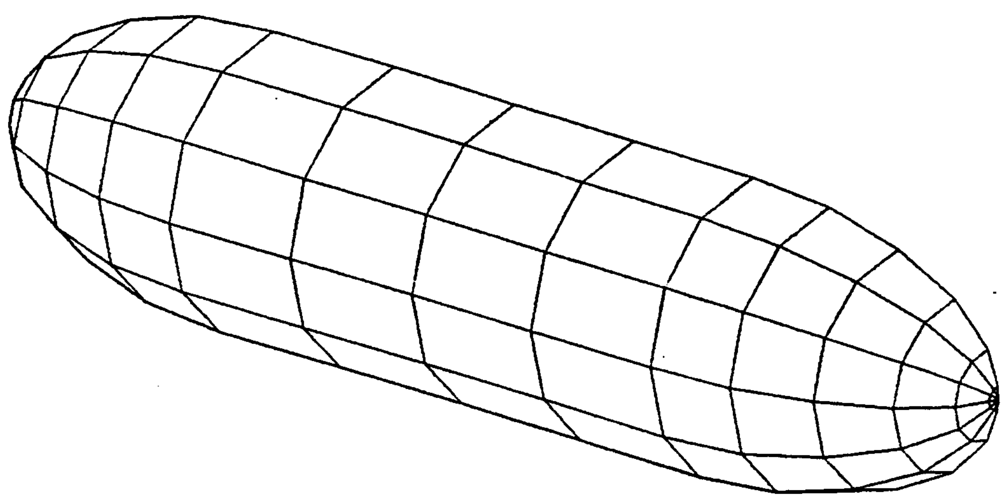


Figure 7-18 Cylindrical Fuselage



CYLINDRICAL FUSELAGE WITH ELLIPSOIDAL ENDS 5/12/83 EXAMPLE 3
AZIMUTH =-50.0 ELEVATION = 20.0

Figure 7-19 Mesh for Fuselage

```
*******************************************************************
CYLINDRICAL FUSELAGE WITH ELLIPSOIDAL ENDS     5/12/83       EXAMPLE 3      G1
THIS IS A GEOMETRY GENERATION RUN WITH NO CHECKING                          G2
*******************************************************************
*MACH       RUN        PRINT      DUMP       ABSTOL     RELTOL
0.          -1         0          0          .002       .1                  G3
*
*  REFERENCE DATA
*SREF       CBAR       WSPAN      XBAR       YBAR       ZBAR
0.19635     1.0        1.0        1.0        0.         0.                  G4
*
*  FLOW CONDITIONS
*ALPHA      BETA       OMEGAX     OMEGAY     OMEGAZ     VINF
0.
*                                                                           G5
*-----------------------------------------------------------*
* THIS DATASET DESCRIBES AN FUSELAGE LIKE BODY MODELLED WITH  *
* A SINGLE PANEL CONSISTING OF:                               *
*    AN ELLIPSOIDAL FOREBODY EXTENDING FROM X=0 TO X=.5,      *
*    A CYLINDRICAL CENTERBODY          FROM X=.5 TO X=1.5,    *
*    AN ELLIPSOIDAL AFTBODY EXTENDING  FROM X=1.5 TO X=2.0    *
*                                                             *
* THE DIAMETER OF THE CENTERBODY IS 0.5                       *
* ONLY THE STARBOARD HALF OF THE BODY IS EXPLICITLY DEFINED   *
*-----------------------------------------------------------*
*
PANEL                                                                       P1
1          FUSELAGE (STARBOARD SIDE)                                        P2
*TYPE      WET        FORCE      IMAGE
0          -1         1          1                                          P3
*
KSPACE                                                                      P9
*NK        KSPACE
6          0.                                                               P9A
*
* THE FIRST SECTION DEFINES THE BASIC CYLINDRICAL CROSS SECTION
*
SECTION                                                                     S1
CYLINDRICAL                                                                 S4
XAXIS                                                                       S5
*THETA     R          X          KBREAK     NK         KSPACE
 90.       0.25                                                             S5A
-90.       0.25                                                             S5A
AT                                                                          S11
*X         Y          Z          JBREAK     NJ         JSPACE
0.000      0.0        0.0                   6          2.                   S11A
SCALE                                                                       S12
*XSCALE    YSCALE     ZSCALE
0.0        0.0        0.0                                                   S12A
*
* THIS SECTION IS SCALED AND REPEATED TO OBTAIN THE SUCCEEDING SECTIONS.
*
SECTION                                                                     S1
AT                                                                          S11
*X         Y          Z          JBREAK     NJ         JSPACE
0.006      0.0        0.0                                                   S11A
SCALE                                                                       S12
*XSCALE    YSCALE     ZSCALE
0.156      0.156      0.156                                                 S12A
*
```

Figure 7-20 Example 3 Dataset (1 of 3)

```
SECTION                                                                       S1
AT                                                                            S11
*X        Y         Z         JBREAK    NJ        JSPACE
0.024     0.0       0.0                                                       S11A
SCALE                                                                         S12
*XSCALE   YSCALE    ZSCALE
0.309     0.309     0.309                                                     S12A
*
SECTION                                                                       S1
AT                                                                            S11
*X        Y         Z         JBREAK    NJ        JSPACE
0.095     0.0       0.0                                                       S11A
SCALE                                                                         S12
*XSCALE   YSCALE    ZSCALE
0.588     0.588     0.588                                                     S12A
*
SECTION                                                                       S1
AT                                                                            S11
*X        Y         Z         JBREAK    NJ        JSPACE
0.345     0.0       0.0                                                       S11A
SCALE                                                                         S12
*XSCALE   YSCALE    ZSCALE
0.951     0.951     0.951                                                     S12A
*
* BREAK SECTIONS HAVE BEEN USED FOR THE NEXT TWO SECTIONS TO ENSURE
* THAT THE CENTER BODY IS CYLINDRICAL, CONSTANT DIAMETER.
*
SECTION                                                                       S1
AT                                                                            S11
*X        Y         Z         JBREAK    NJ        JSPACE
0.500     0.0       0.0       1         4         0.                          S11A
SCALE                                                                         S12
*XSCALE   YSCALE    ZSCALE
1.0       1.0       1.0                                                       S12A
*
SECTION                                                                       S1
AT                                                                            S11
*X        Y         Z         JBREAK    NJ        JSPACE
1.500     0.0       0.0       1         6         -2.                         S11A
SCALE                                                                         S12
*XSCALE   YSCALE    ZSCALE
1.0       1.0       1.0                                                       S12A
*
SECTION                                                                       S1
AT                                                                            S11
*X        Y         Z         JBREAK    NJ        JSPACE
1.655     0.0       0.0                                                       S11A
SCALE                                                                         S12
*XSCALE   YSCALE    ZSCALE
0.951     0.951     0.951                                                     S12A
*
SECTION                                                                       S1
AT                                                                            S11
*X        Y         Z         JBREAK    NJ        JSPACE
1.905     0.0       0.0                                                       S11A
SCALE                                                                         S12
*XSCALE   YSCALE    ZSCALE
0.588     0.588     0.588                                                     S12A
*
SECTION                                                                       S1
```

Figure 7-21 Example 3 Dataset (2 of 3)

```
AT                                                                S11
*X         Y          Z          JBREAK    NJ      JSPACE
1.976      0.0        0.0                                         S11A
SCALE                                                             S12
*XSCALE    YSCALE     ZSCALE
0.309      0.309      0.309                                       S12A
*
SECTION                                                           S1
AT                                                                S11
*X         Y          Z          JBREAK    NJ      JSPACE
1.994      0.0        0.0                                         S11A
SCALE                                                             S12
*XSCALE    YSCALE     ZSCALE
0.156      0.156      0.156                                       S12A
*
SECTION                                                           S1
AT                                                                S11
*X         Y          Z          JBREAK    NJ      JSPACE
2.000      0.0        0.0                                         S11A
SCALE                                                             S12
*XSCALE    YSCALE     ZSCALE
1.000      0.0        0.0                                         S12A
```

Figure 7-22 Example 3 Dataset (3 of 3)

The panel definition begins with the keyword PANEL, followed by the panel ID number and title. The following record selects the boundary condition, exterior surface, disposition of forces, and existence of an image for the panel. In the case, the TYPE is set to 0 for a body panel with an impermeable boundary condition. The WET flag is set to –1 to obtain the proper exterior surface. The FORCE and IMAGE flags are set to include panel forces into the total forces and to generate an image panel across the plane of symmetry.

Since the fuselage cross section is circular throughout the body, two things can be done to simplify the dataset. First, cylindrical section definitions can be used. This has the advantage that, due to the special properties of cylindrical section, only two points need be input to describe a circular cross section. In this case, the points correspond to the end points of the half circle. The second thing that can be done is to represent the body with repeated sections, scaled and translated from the first circular cross section definition. This is done in the dataset, where the first cross section is entered with a radius of 0.25, corresponding to the maximum radius on the fuselage. This section is scaled down to a zero radius for the nose section, and again scaled and translated to obtain all succeeding sections back to the aft end of the body. See Figure 7-18. Each scaling is independent of the previous scalings, operating only on the basic section definition with a radius of 0.25.

In order to respace the forebody, a spacing interval in the J direction has been specified starting with the first section. This has been done by placing a 6 in the fifth field an a 2. in the sixth field of the AT specification of the section. This indicates that, in the is spacing interval which runs from the first section at X=0. to the section at X=0.5, six elements will be placed in a sine distribution which concentrates element near the first section (the nose).

The section at X=0.5 contains a break specification as well as another spacing interval. The break specification for this section is to force the splines used for respacing in the J direction to give straight lines on the centerbody. The spacing interval specification ends the forebody spacing interval and begins a spacing interval that puts four equally spaced elements along the centerbody. The section at X=1.5 contains a break specification, ends the centerbody spacing interval and begins the final spacing interval along the aftbody. The spacing for the aftbody is symmetrical to the spacing used on the forebody. Again, six elements have been specified, but this time a –2. distribution has been used, to concentrate elements near the last section (the aft end).

Respacing around the circular cross section is done using the KSPACE keyword. Six elements, using and equal spacing distribution, have been specified.

As can be seen in Figure 7-18, the panel "positive" surface, defined by the cross product of the K and J directions points inwards. This is incorrect, and the correct outward surface is selected by setting WET=-1, informing the program to use the opposite surface for the exterior. The completed surface mesh, after respacing is shown in Figure 7-19.

The only disadvantage of this choice of panel definition is that the sections are defined around the body in "hoops," ordering the elements in the output in the same fashion. This could be somewhat of an inconvenience if the pressure distribution along the body is the subject of the analysis. The alternative way to define the body, with sections running along the length of the body from nose to tail, has the disadvantage that cylindrical sections can no longer be used. In this case, a number or rectangular section "stringer" would have to be defined at various circumferential angles to ensure that the cross section is circular.

7.11.4 Example 4 – Fuselage, Wing, and Tail

This example is a typical, although brief, dataset which describes a complete configuration. It represents the sort of dataset which the user will encounter in practice. A given configuration can be described in a variety of different ways because of the flexibility of the QUADPAN input package. The dataset in this example is intended to illustrate a wide range of input features. Therefore, some portions of the configuration could have been defined as easily without using all the features which have been used. The user should remember that the panel and section scaling, translating, and rotating features have been provided for convenience when a configuration has been defined in multiple reference frames. In many practical situations, complex models can be constructed without using any of these features.

The configuration on which this example is based is shown in Figure 7-23 and has the following characteristics:

- The wings and fin have identical symmetric biconvex sections and have zero thickness at the tips.
- The fuselage is a circular cylinder with a conical nose.
- There is a short boattail which tapers to the diameter of the exhaust.

Since the configuration is symmetric across the X-Z plane, the geometry will be defined by only explicity enterig the right hand side. The use of symmetry will also save large amounts of computer time when running the case.

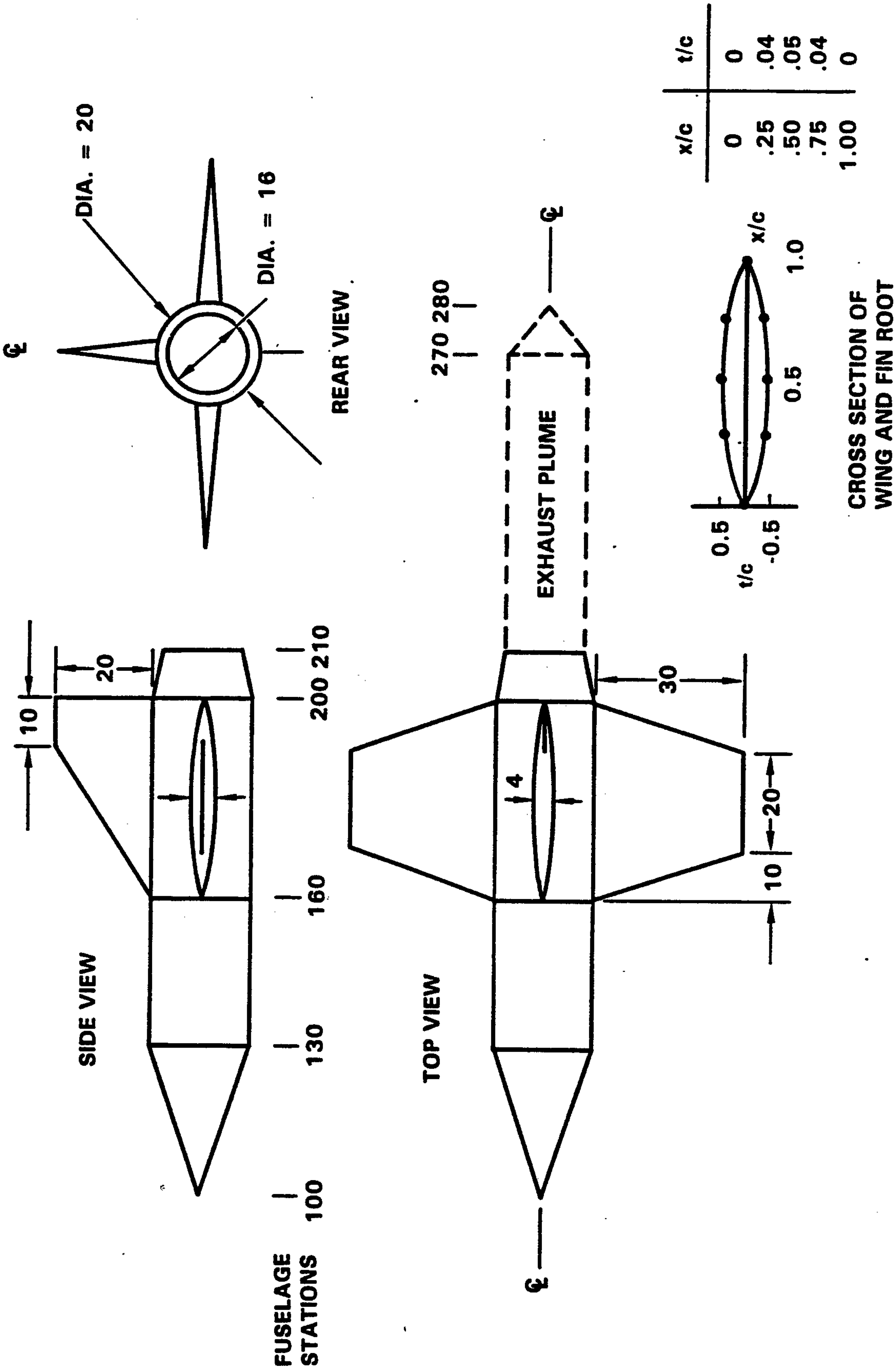


Figure 7-23 Complex Test Case

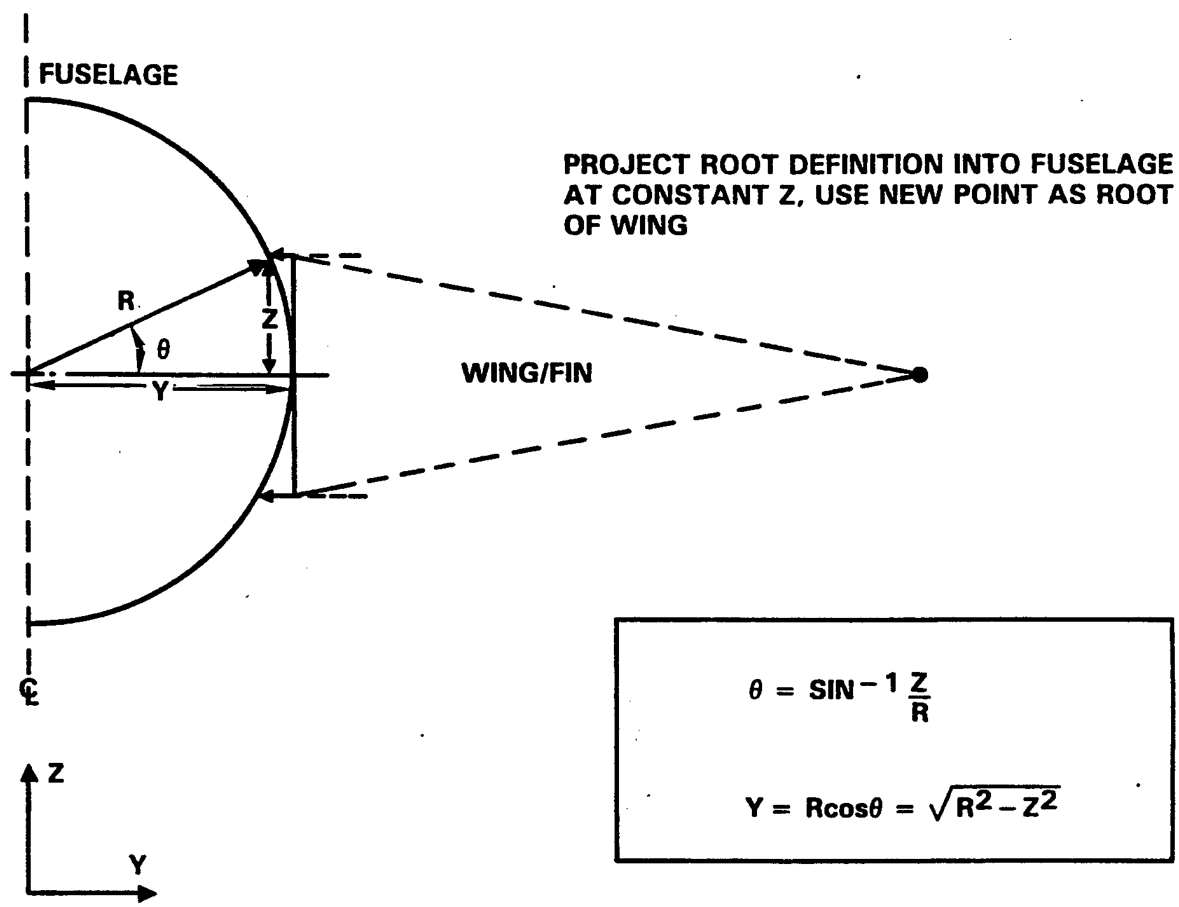


Figure 7-24 Definition Wing/Body Intersection

7.11.4.1 Model Layout –

As a result of the options available to do geometry generation with QUADPAN, there are many possible arrangements of panels that may be used to represent this configuration. The arrangement used in this example is based on the following considerations:

- The wing and fin are most conveniently modeled as separate panels because the direction of the sections which are convenient for defining them is different from the section direction which is convenient for the fuselage.

- The need to have wakes attached to the trailing edges of the lifting surfaces suggests that the wings and fins be separate surfaces. In addition, the upper and lower surfaces of the wing are separate panels because that is how the wing sections are defined.

- Since the vertical fin is on the plane of symmetry, only the right hand side is explicitly defined, with the left hand side generated as its image panel.

- The fuselage is most easily treated by separating the wing junction region from the forebody. In this way the entire forebody can be generated from one panel using a spacing interval to ensure element contiguity between the forebody and midbody. The midbody must necessarily be constructed as separate upper and lower panels since the cutout for the wing would be prohibited in a single panel.

- The fuselage aft of the wing must be divided into upper and lower panels so that the side edges of the wing wake can abut panel edges. This is essential for the correct calculation of the velocity on the side of the fuselage. Note that this is a situation in which no other panel arrangements are possible because the wake panel must abut other panels only at panel edges. Since the fuselage adjacent to the wing is also divided into upper and lower panels it is natural to combine the aftbody with the wing junction region so that one panel describes the upper half of the body aft of the leading edge and another describes the corresponding lower half of the body. The panel must have two spacing intervals with a break at the trailing edge of the wing root to force the elements on the wing, midbody, and aftbody to be continguous even though the panel boundaries are not. This is an example of using one panel with spacing ranges instead of several panels.

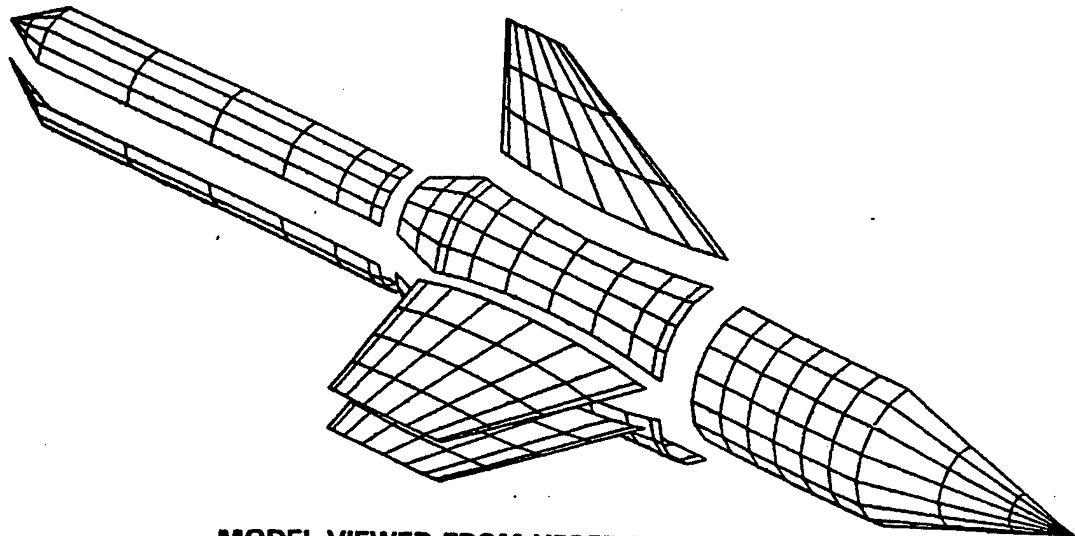
- The exhaust is modeled with a force free solid body. This has been broken into upper and lower panels like the aftbody so that accurate force data could be obtained on the exhaust plume if desired.

The partitioning of the configuration into panels is shown in Figure 7-24.

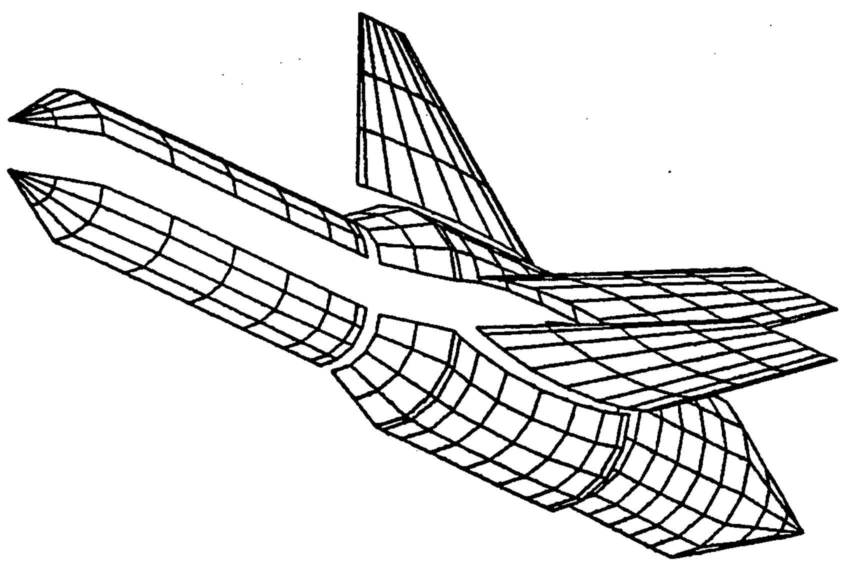
7.11.4.2 Defining the Geometry –

As a result of this panel layout, ten panels have been used to represent this configuration:

Panel 1 - Forebody
Panel 3 - Upper Midbody
Panel 4 - Lower Midbody
Panel 5 - Upper Wing
Panel 6 - Lower Wing
Panel 7 - Wake for Wing
Panel 8 - Vertical Fin
Panel 9 - Vertical fin Wake
Panel 20 - Upper Plume
Panel 21 - Lower Plume



MODEL VIEWED FROM UPPER FRONT QUARTER



MODEL VIEWED FROM LOWER AFT QUARTER

Figure 7-25 Mesh for Test Case

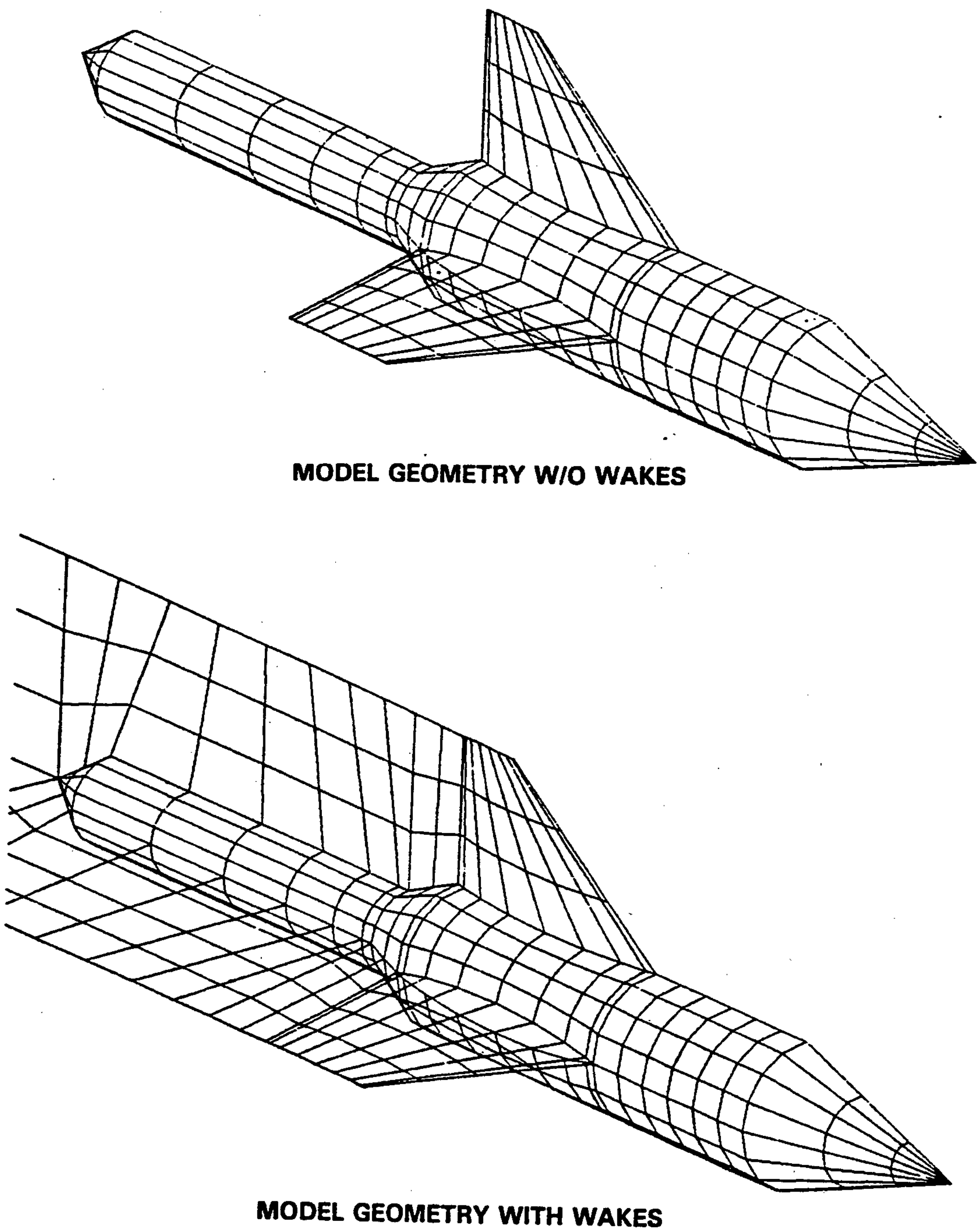


Figure 7-26 Wake Geometry

Panel 1 – Forebody

Since the forebody is a body of revolution, it is most conveniently modeled using a series of cylindrical sections. The sections are translated to appropriate positions with AT points. Since only two points define a circle using cylindrical coordinates, this option is very convenient for bodies of revolution or shapes which are nearly bodies of revolution. Because the panel is defined using azimuthal sections, the K direction is azimuthal, and the J direction is longitudinal.

To ensure that the elements in the forebody panel will be contiguous with the elements on the upper and lower midbody, a spacing interval in the K direction is used. In the last section of the forebody, a third point at THETA = 0., defines the point which separates the upper and lower midbody panels at the leading edge of the wing. This point is a spacing point so that it will be an element corner point. The spacing information in the section definition overrides the spacing information following KSPACE, but the total number of elements must be the same.

A spacing interval in the J direction has been used to produce the shoulder between the conical nose and the cylindrical body. Specifiying JBREAK = 1 puts a break in slope of the interpolated curve but does not force the mesh to have a slope discontinuity at the desired point since the mesh might straddle the breakpoint. Only by requiring that the shoulder be both the end of a spacing interval and a break point is the mesh forced to duplicate the shoulder.

Panel 3 – Upper Midbody

The midbody is comprised of upper and lower panels. The side edges of the upper body are contoured for the intersection with the fin and upper wing surface. The lower body is contoured to form the intersection with the lower wing surface.The wing/body junction in this example is specified in the drawing. It is frequently the case that the geometrical information provided to describe the configuration is insufficient to define these intersections. The user should then make reasonable geometric approximations to generate this data.

Since the midbody is a body of revolution, it is most conveniently defined with azimuthal sections. The penalty for this is that a section must be provided for each point used to define the wing root, so that the definition of the side edge of the body is identical to the definition of the wing root. If this is not done, the elements on the wing will not precisely abut the elements on the body. The same requirement applies to the fin/body intersection. Defining the midbody with longitudinal sections eases that problem but is inconvenient because the body is not described with longitudinal sections so these would have to be generated manually.

The upper midbody is defined with cylindrical sections at the locations where the wing and fin roots are defined. The end points of each section are the points used to define the wing and fin, transformed into the polar coordinate system used to define the body. Consider first the upper midbody panel. The lower edge of the panel abuts the upper wing surface over part of its length and the lower midbody panel for the remainder of its length. To make the elements along the adjacent edges contiguous, the intersection of the trailing edge of the wing and the body is explicity defined as the end of a spacing interval. The 8 elements ahead of this point will be contiguous with the 8 elements on the wing and the 2 elements specified aft of this point will be contiguous with elements on the lower midbody panel. To make the elements contiguous, the spacing as well as the number of elements must be consistent on the adjacent panels. Therefore, cosine spacing is used in the first spacing interval (JSPACE = 1) to match the wing, and equal spacing (JSPACE = 0) in the second to match the lower midbody panel.

The root trailing edge point must also be specified as a break point because the upper wing surface and lower midbody are separate panels so that there is a slope discontinuity between them. If this slope discontinuity were not preserved on the upper midbody, the definition of the edge of the upper midbody would not be identical to the definition of the edge formed by the upper surface and the aft part of the lower midbody. This would result in a mesh with gaps.

Scaling parameters following the keyword SCALE are used to produce the decrease in body diameter at the aft end of the fuselage.

Panel 4 – Lower Midbody

The same considerations apply to the lower midbody panel as to the upper midbody panel. The definition of the lower midbody panel differs from that of the upper midbody panel in that locater points are used. The use of locater points is entirely optional and the configuration in the present example could have been easily modeled without them. This would have been accomplished by defining the lower midbody panel in the same way as the upper midbody panel, but with the section definitions revised to describe the lower quadrant of the body.

The illustrative use of locater points in this example exploits the fact that the fin root section is identical to the wing root section. The lower midbody panel can, therefore, be constructed by rotating the upper midbody panel 90 degrees around the X-axis, and replacing the contour for the wing intersection with a straight line. In this way, the cutout of the fin on the upper midbody panel becomes the cutout for the lower surface of the wing on the lower midbody panel.

The first step in defining the lower panel is to start with the upper panel but remove the cutout for the wing. The section definitions in the lower panel are identical to the section definitions in the upper panel except that edge 3 (the edge which abuts the wing surface) is at THETA=0, RADIUS=10. The AT points have been removed because the translation they performed can be done with the locater points.

The use of locater points in this example involves starting with the upper midbody panel and locating it on the vehicle by selecting three points on the panel as it is defined and specifying the location of these three points in the global coordinate system. Imagine starting with the upper midbody panel defined without AT points so that the forward edge is at X=0, rather than at X=160. The point which is at the wing root leading edge is mapped to the desired location of the forward lower corner of the lower midbody panel. This is at X,Y,Z coordinates 160,0,-10. Next the corner of the upper midbody panel at the wing root trailing edge is mapped to the desired location of the lower aft corner of the lower midbody panel (coordinates 200,0,-10). Since the length of the line connecting these two points in panel coordinates is equal to the length of the line connecting these two points in global coordinates, no stretching is done. The last step is a rotation about the line connecting these two points in global coordinates until the plane defined by all three points in panel coordinates maps onto the plane defined by all three points in global coordinates. The third point is selected to be the upper forward corner of the upper midbody panel. The desired orientation of the lower midbody panel is obtained by mapping the plane formed by the lower forward corner, lower aft corner, and upper forward corner of the upper midbody panel onto the lower forward corner, lower aft corner, and upper forward corner where the lower midbody panel is to be placed. This is accomplished by selecting global location of the third point to the upper forward corner of the lower midbody panel (coordinates 160,10,0).

Many other choices of locater points could have been used to produce the same positioning of the panel. While the locater points used in this example are on the panel this need not be so in general. Also the three points in panel coordinates all mapped onto the points specified in global coordinates. In general only the first two points specified map onto corresponding points in both coordinate systems.

All spacing interval and breakpoint considerations which apply to the upper midbody panel apply to the lower midbody panel.

Panels 5,6, and 7 – Wing Group

The upper and lower surface of the wing are separate panels which are defined using identical techniques. Each panel is defined with two sections, one at the root and one at the tip. AT points are used to locate the sections in global space, with SCALE parameters used to establish the chord length and thickness. The root section is a nonplanar space curve which describes the intersection of the wing with the circular body. After translation with the AT points has been done, the points are identical to the points used to describe the side edge of the portion of the upper midbody panel which is adjacent to the wing. The points are expressed in rectangular coordinates here, whereas they are expressed in cylindrical coordinates in the definition of the body panel.

In order to generate contiguous elements at the wing/body intersection, the number and spacing of the elements must be identical in adjoining parts to the two components. The wing is defined with longitudinal sections so that the K direction runs fore to aft. The body is defined with sections in the azimuthal direction so it is the J direction of the body panel which runs fore to aft. Therefore, the K spacing of the wing must match the J spacing of the body. The flexibility of panel orientation in QUADPAN means that any edge of one panel can abut any edge of another panel. Hence, care must be taken to determine the J and K directions on each panel so that the spacing of adjoining panels can be made compatible.

The wing wake is defined as a panel with sections at the root and tip. Edge 4 of the wake abuts the trailing edge of the wing, so the wake is a TYPE 4 panel. This edge of the wake must be defined with exactly the same set of points used to define the trailing edge of the wing so that precise alignment of the wake with the wing is obtained.

The inboard edge of the wake conforms to the side of the fuselage, and is spaced to be compatible with the fuselage so that the abutment will be found. See Figure 7-26. The pressure on the fuselage will be incorrect if this is not done. The side edge of the wake adjoins both the aft portion of the midbody panels and the body which is used to simulate the exhaust plume. The first spacing interval is used to match the wake to the aft spacing interval in the upper and lower midbody panels. The next two spacing intervals match the wake to the two spacing intervals on the body which models the exhaust, and the final spacing interval extends the wake a large distance downstream.

Spacing intervals on the outboard edge of the wake are provided to deep the elements on the wake from becoming too highly skewed. Instead of duplicating the spacing intervals used on the inboard edge on the outboard edge, the intervals over boattail and the exhaust plume were combined into one interval and a fractional spacing was used to produce reasonably shaped elements. The fractional spacing used here is a combination of sine and equal spacing.

Panels 8 and 9 – Vertical Fin Group

The construction of the vertical fin is identical to that of the wing. The panel coordinate system has the same orientation as the global coordinate system since locater points are not used. Therefore, the thickness of the fin is produced by nonzero values of the Y coordinate in the section definitions. The points defining the root section, the number of elements and the spacing must all match the adjoining part of the midbody to produce a mesh with contiguous elements. AT points are used to translate the sections to the correct locations and the SCALE keyword is used to scale the tip section to obtain the correct chord.

The vertical fin wake (see Figure 7-26) is defined in essentially the same fashion as the wake for the wing. Whereas the wake for the wing is defined entirely in global coordinates, the fin wake is defined with AT points. This is entirely for illustrative purposes.

It is important to notice that the wake for the vertical fin has an image even though it lies on the plane of symmetry. See Figure 7-27. Any wake which is on the plane of symmetry should have an image of the rest of the panels in the configuration have images. While the program would run and give correct answers if IMAGE were set to zero, about four times as much computer time would be used.

Panels 10 and 11 – Exhaust Plume

The exhaust plume consist of upper and lower panels. The panels have TYPE set to 0 because they have hydrodynamic boundary conditions associated with them. The parameter FORCE is set to 0 so that these panels do not contribute to the total forces and moments.

The geometry of these panels is developed in the same way as the upper midbody panel. Upper and lower panels were used so that the program would include the wake from the wing in calculating the pressure on the body. Since the forces produced by these panels are not used this is not critical. However, the forces on the panels may be of interest in assessing the validity of the model, in which case establishing the abutment with the edge of the wake is important.

```
***************************************************************************
COMPLEX TEST CASE, EXAMPLE 4 (7/29/83)  MACH 0.4                               G1
ALPHA SWEEP, NO DEFLECTIONS, NO PERMEABILITY                                   G2
***************************************************************************
*
*MACH      RUN        PRINT      DUMP       ABSTOL     RELTOL
.4         1          0          0          .001       .1                      G3
*
*  REFERENCE DATA
*SREF      CBAR       WSPAN      XBAR       YBAR       ZBAR
2666.      35.111     80.        171.222    0.                                 G4
*
*  FLOW CONDITIONS
*ALPHA     BETA       OMEGAX     OMEGAY     OMEGAZ     VINF
0.                                                                             G5
4.                                                                             G5
8.                                                                             G5
*
**************************************************************
*      THIS DATASET DESCRIBES A CYLINDRICAL FUSELAGE WITH
*      A CONICAL FOREBODY, CENTERLINE FIN AND SYMMETRICAL
*      WING.
*
*      ONLY HALF THE BODY IS DEFINED.
*      THE NOSE OF THE BODY IS LOCATED AT THE POINT
*      (100,0,0) IN THE GLOBAL COORDINATE SYSTEM.
**************************************************************
*
*
*
**************************************************************
*     FOREBODY PANEL
*       THIS PANEL EXTENDS FROM THE NOSE TO THE ROOT
*       OF FIN AND WING.
*
PANEL                                                                          P1
1.         FOREBODY                                                            P2
*TYPE      WET        FORCE      IMAGE
 0.        -1.        1.         1.                                            P3
*
KSPACE                                                                         P9
*NK        KSPACE
 8.         0.                                                                 P9a
*
SECTION                                                                        S1
RECTANGULAR                                                                    S2
*X         Y          Z          KBREAK     NK         KSPACE
0.         0.         0.                                                       S2a
AT                                                                             S11
*XT        YT         ZT         JBREAK     NJ         JSPACE
100.       0.         0.                    4.         2.                      S11a
*
SECTION                                                                        S1
CYLINDRICAL                                                                    S4
XAXIS                                                                          S5
*THETA     RADIUS     X          KBREAK     NK         KSPACE
 90.       10.00                            8.         0.                      S5a
-90.       10.00                                                               S5a
AT                                                                             S11
*XT        YT         ZT         JBREAK     NJ         JSPACE
```

Figure 7-27 Example 4 Dataset (1 of 9)

```
130.        0.         0.         1.         6.         0.                               S11a
*
SECTION                                                                                 S1
CYLINDRICAL                                                                             S4
XAXIS                                                                                   S5
*THETA      RADIUS     X          KBREAK     NK         KSPACE
 90.        10.00                            4.         0.                              S5a
  0.        10.00                            4.         0.                              S5a
-90.        10.00                                                                       S5a
AT                                                                                      S11
*XT         YT         ZT         JBREAK     NJ         JSPACE
160.        0.         0.                                                               S11a
*
*
***********************************************************************
*   UPPER MIDBODY PANEL
*       INCLUDES WING/BODY JUNCTION, AND BOATTAIL.
*       CIRCULAR SECTION BODY PANEL WITH SIDE EDGES
*       CONTOURED TO JOIN THE FIN (ON THE UPPER EDGE)
*       AND THE WINGS (ON THE LOWER EDGE)
*       AFT SECTION JOINS LOWER MIDBODY PANEL
*
PANEL                                                                                   P1
3.          UPPER MIDBODY                                                               P2
*TYPE       WET        FORCE      IMAGE
 0.         -1.        1.         1.                                                    P3
*
KSPACE                                                                                  P9
*NK         KSPACE
 4.          0.                                                                         P9a
*
SECTION                                                                                 S1
CYLINDRICAL                                                                             S4
XAXIS                                                                                   S5
*THETA      RADIUS     X          KBREAK     NK         KSPACE
 90.        10.00                                                                       S5a
  0.        10.00                                                                       S5a
AT                                                                                      S11
*XT         YT         ZT         JBREAK     NJ         JSPACE
160.        0.         0.         1.         8.         1.                              S11a
*
SECTION                                                                                 S1
CYLINDRICAL                                                                             S4
XAXIS                                                                                   S5
*THETA      RADIUS     X          KBREAK     NK         KSPACE
 80.7931    10.00                                                                       S5a
  9.2069    10.00                                                                       S5a
AT                                                                                      S11
*XT         YT         ZT         JBREAK     NJ         JSPACE
170.        0.         0.                                                               S11a
*
SECTION                                                                                 S1
CYLINDRICAL                                                                             S4
XAXIS                                                                                   S5
*THETA      RADIUS     X          KBREAK     NK         KSPACE
 78.4630    10.00                                                                       S5a
 11.5370    10.00                                                                       S5a
AT                                                                                      S11
*XT         YT         ZT         JBREAK     NJ         JSPACE
180.        0.         0.                                                               S11a
```

Figure 7-28 Example 4 Dataset (2 of 9)

```
*
-ELTION                                                                     S1
CYLINDRICAL                                                                 S4
XAXIS                                                                       S5
*THETA     RADIUS    X         KBREAK    NK        KSPACE
 80.7931   10.00                                                            S5a
  9.2069   10.00                                                            S5a
AT                                                                          S11
*XT        YT        ZT        JBREAK    NJ        JSPACE
190.       0.        0.                                                     S11a
*
SECTION                                                                     S1
CYLINDRICAL                                                                 S4
XAXIS                                                                       S5
*THETA     RADIUS    X         KBREAK    NK        KSPACE
 90.       10.00                                                            S5a
  0.       10.00                                                            S5a
AT                                                                          S11
*XT        YT        ZT        JBREAK    NJ        JSPACE
200.       0.        0.        1.        2.        0.                       S11a
*
SECTION                                                                     S1
AT                                                                          S11
*XT        YT        ZT        JBREAK    NJ        JSPACE
210.       0.        0.                                                     S11a
SCALE                                                                       S12
*XSCALE    YSCALE    ZSCALE
1.         0.6       0.6                                                    S12a
*
*
*************************************************************
*    LOWER MIDBODY PANEL
*       INCLUDES WING/BODY JUNCTION, AND BOATTAIL.
*       CIRCULAR SECTION BODY PANEL WITH SIDE EDGES
*       CONTOURED TO JOIN THE WING (ON THE UPPER EDGE)
*       AND THE LOWER CENTERLINE (ON THE LOWER EDGE)
*       AFT SECTION JOINS UPPER MIDBODY PANEL
*       SINCE THE FIN AND WING HAVE THE SAME SECTION, THIS PANEL
*       CAN BE MADE BY ROTATING THE UPPER PANEL 90 DEG. AROUND THE
*       X AXIS, AND REMOVING THE CUTOUT FOR THE WING. LOCATER
*       POINTS ARE USED TO PERFORM THIS ROTATION. THE LOCATER
*       POINTS ARE ALSO USED TO PERFORM A TRANSLATION, ELIMINATING
*       THE NEED FOR THE AT POINTS.
*
PANEL                                                                       P1
4.         LOWER MIDBODY                                                    P2
*TYPE      WET       FORCE     IMAGE
 0.        -1.       1.        1.                                           P3
*
*LOCATER POINTS USED TO POSITION THE PANEL IN THE GLOBAL SYSTEM
*
LOCATE                                                                      P7
*XPNL      YPNL      ZPNL      XGBL      YGBL      ZGBL
0.         10.       0.        160.      0.        -10.                     P7a
40.        10.       0.        200.      0.        -10.                     P7b
0.         0.        10.       160.      10.       0.                       P7c
*
KSPACE                                                                      P9
*NK        KSPACE
 4.         0.                                                              P9a
```

Figure 7-29 Example 4 Dataset (3 of 9)

```
*
SECTION                                                              S1
CYLINDRICAL                                                          S4
XAXIS                                                                S5
*THETA     RADIUS     X          KBREAK     NK       KSPACE
 90.       10.00                                                     S5a
  0.       10.00                                                     S5a
AT                                                                   S11
*XT        YT         ZT         JBREAK     NJ       JSPACE
0.         0.         0.         1.         8.       1.              S11a
*
SECTION                                                              S1
CYLINDRICAL                                                          S4
XAXIS                                                                S5
*THETA     RADIUS     X          KBREAK     NK       KSPACE
 80.7931   10.00                                                     S5a
 0.        10.00                                                     S5a
AT                                                                   S11
*XT        YT         ZT         JBREAK     NJ       JSPACE
10.        0.         0.                                             S11a
*
SECTION                                                              S1
CYLINDRICAL                                                          S4
XAXIS                                                                S5
*THETA     RADIUS     X          KBREAK     NK       KSPACE
 78.4630   10.00                                                     S5a
 0.        10.00                                                     S5a
AT                                                                   S11
*XT        YT         ZT         JBREAK     NJ       JSPACE
20.        0.         0.                                             S11a
*
SECTION                                                              S1
CYLINDRICAL                                                          S4
XAXIS                                                                S5
*THETA     RADIUS     X          KBREAK     NK       KSPACE
 80.7931   10.00                                                     S5a
  0.       10.00                                                     S5a
AT                                                                   S11
*XT        YT         ZT         JBREAK     NJ       JSPACE
30.        0.         0.                                             S11a
*
SECTION                                                              S1
CYLINDRICAL                                                          S4
XAXIS                                                                S5
*THETA     RADIUS     X          KBREAK     NK       KSPACE
 90.       10.00                                                     S5a
  0.       10.00                                                     S5a
AT                                                                   S11
*XT        YT         ZT         JBREAK     NJ       JSPACE
40.        0.         0.         1.         2.       0.              S11a
*
SECTION                                                              S1
AT                                                                   S11
*XT        YT         ZT         JBREAK     NJ       JSPACE
50.        0.         0.                                             S11a
SCALE                                                                S12
*XSCALE    YSCALE     ZSCALE
1.         0.6        0.6                                            S12a
*
*
```

Figure 7-30 Example 4 Dataset (4 of 9)

```
*******************************************************
*     WING GROUP.
*       UPPER AND LOWER SURFACES AND WAKE.
*
PANEL                                                                    P1
5.         UPPER SURFACE WING                                            P2
*TYPE      WET        FORCE      IMAGE
 0.         1.         1.         1.                                     P3
*
JSPACE                                                                   P8
*NJ        JSPACE
 4.         0.                                                           P8a
KSPACE                                                                   P9
*NK        KSPACE
 8.         1.                                                           P9a
*
*  ROOT SECTION DEFINITION
*  THE SECTION LINE IS CAMBERED IN Y TO FIT TO CYLINDRICAL BODY.
*
SECTION                                                                  S1
RECTANGULAR                                                              S2
*AIRFOIL COORDINATES
*X         Y          Z
0.         10.        0.                                                 S2a
10.        9.871      1.6                                                S2a
20.        9.798      2.0                                                S2a
30.        9.871      1.6                                                S2a
40.        10.        0.                                                 S2a
AT                                                                       S11
*XT        YT         ZT         JBREAK     NJ         JSPACE
160.       0.         0.                                                 S11a
*
*   TIP SECTION DEFINITION
*
SECTION                                                                  S1
RECTANGULAR                                                              S2
*AIRFOIL COORDINATES
*X         Y          Z
0.         40.        0.                                                 S2a
40.        40.        0.                                                 S2a
AT                                                                       S11
*XT        YT         ZT         JBREAK     NJ         JSPACE
170.       0.         0.                                                 S11a
SCALE                                                                    S12
*XSCALE    YSCALE     ZSCALE
.5         1.         1.                                                 S12a
*
*
*
PANEL                                                                    P1
6.         LOWER SURFACE WING                                            P2
*TYPE      WET        FORCE      IMAGE
 0.        -1.         1.         1.                                     P3
*
JSPACE                                                                   P8
*NJ        JSPACE
 4.         0.                                                           P8a
KSPACE                                                                   P9
*NK        KSPACE
 8.         1.                                                           P9a
```

Figure 7-31 Example 4 Dataset (5 of 9)

```
*
*  ROOT SECTION DEFINITION
*  THE SECTION LINE IS CAMBERED IN Y TO FIT TO CYLINDRICAL BODY.
*
SECTION                                                                   S1
RECTANGULAR                                                               S2
*AIRFOIL COORDINATES
*X          Y          Z
0.          10.        0.                                                 S2a
10.         9.871      -1.6                                               S2a
20.         9.798      -2.0                                               S2a
30.         9.871      -1.6                                               S2a
40.         10.        0.                                                 S2a
AT                                                                        S11
*XT         YT         ZT         JBREAK     NJ         JSPACE
160.        0.         0.                                                 S11a
*
*   TIP SECTION DEFINITION
*
SECTION                                                                   S1
RECTANGULAR                                                               S2
*AIRFOIL COORDINATES
*X          Y          Z
0.          40.        0.                                                 S2a
40.         40.        0.                                                 S2a
AT                                                                        S11
*XT         YT         ZT         JBREAK     NJ         JSPACE
170.        0.         0.                                                 S11a
SCALE                                                                     S12
*XSCALE     YSCALE     ZSCALE
.5          1.         1.                                                 S12a
*
*
*  WAKE PANEL FOR WING
*    EDGE 4 MATCHES AFT EDGE OF UPPER AND LOWER WING SURFACES.
*
PANEL                                                                     P1
7.          WAKE FOR WING                                                 P2
*TYPE       WET        FORCE      IMAGE
 4.         1.         1.         1.                                      P3
*
JSPACE                                                                    P8
*NJ         JSPACE
 4.          0.                                                           P8a
KSPACE                                                                    P9
*NK         KSPACE
 8.          1.                                                           P9a
*
*  ROOT SECTION DEFINITION
*
SECTION                                                                   S1
RECTANGULAR                                                               S2
*AIRFOIL COORDINATES
*X          Y          Z          KBREAK     NK         KSPACE
200.        10.        0.                    2.         0.                S2a
210.        6.         0.         1.         6.         2.                S2a
270.        6.         0.         1.         2.         0.                S2a
230.        0.         0.         1.         1.         0.                S2a
1000.       0.         0.                                                 S2a
*
```

Figure 7-32 Example 4 Dataset (6 of 9)

```
*   TIP SECTION DEFINITION
*
SECTION                                                                      S1
RECTANGULAR                                                                  S2
*AIRFOIL COORDINATES
*X         Y         Z         KBREAK    NK        KSPACE
190.       40.       0.                  10.       2.5                       S2a
280.       40.       0.        1.        1.        0.                        S2a
1000.      40.       0.                                                      S2a
*
*
*
****************************************************************
*     VERTICAL FIN GROUP
*       INCLUDES RIGHT HAND SURFACE AND WAKE.
*
PANEL                                                                        P1
8.         VERTICAL FIN                                                      P2
*TYPE      WET       FORCE     IMAGE
 0.        -1.        1.        1.                                           P3
*
JSPACE                                                                       P8
*NJ        JSPACE
 3.         0.                                                               P8a
KSPACE                                                                       P9
*NK        KSPACE
 8.         1.                                                               P9a
*
*  ROOT SECTION DEFINITION
*  THE SECTION LINE IS CAMBERED IN Z TO FIT TO CYLINDRICAL BODY.
*
SECTION                                                                      S1
RECTANGULAR                                                                  S2
*AIRFOIL COORDINATES
*X         Y         Z
0.         0.        10.                                                     S2a
10.        1.6       9.871                                                   S2a
20.        2.0       9.798                                                   S2a
30.        1.6       9.871                                                   S2a
40.        0.        10.                                                     S2a
AT                                                                           S11
*XT        YT        ZT        JBREAK    NJ        JSPACE
160.       0.        0.                                                      S11a
*
*   TIP SECTION DEFINITION
*
SECTION                                                                      S1
RECTANGULAR                                                                  S2
*AIRFOIL COORDINATES
*X         Y         Z
0.         0.        30.                                                     S2a
10.        0.        30.                                                     S2a
AT                                                                           S11
*XT        YT        ZT        JBREAK    NJ        JSPACE
190.       0.        0.                                                      S11a
*
*
*
PANEL                                                                        P1
9.         VERTICAL FIN WAKE                                                 P2
```

Figure 7-33 Example 4 Dataset (7 of 9)

```
*TYPE       WET         FORCE       IMAGE
 4.         -1.          1.          1.                                             P3
*
JSPACE                                                                              P8
*NJ         JSPACE
 3.          0.                                                                     P8a
*
*  ROOT SECTION DEFINITION
*
SECTION                                                                             S1
RECTANGULAR                                                                         S2
*AIRFOIL COORDINATES
*X          Y           Z           KBREAK      NK          KSPACE
40.         0.          10.         1.          2.          0.                      S2a
50.         0.          6.          1.          6.          2.
110.        0.          6.          1.          2.          0.                      S2a
120.        0.          0.          1.          1.          0.                      S2a
840.        0.          0.                                                          S2a
AT                                                                                  S11
*XT         YT          ZT          JBREAK      NJ          JSPACE
160.        0.          0.                                                          S11a
*
*   TIP SECTION DEFINITION
*
SECTION                                                                             S1
RECTANGULAR                                                                         S2
*AIRFOIL COORDINATES
*X          Y           Z           KBREAK      NK          KSPACE
10.         0.          30.                     10.         2.5                     S2a
90.         0.          30.         1.          1.          0.                      S2a
810.        0.          30.                                                         S2a
AT                                                                                  S11
*XT         YT          ZT          JBREAK      NJ          JSPACE
190.        0.          0.                                                          S11a
*
*
*************************************************************
*     EXHUST PLUME   PANEL
*       INCLUDES UPPER AND LOWER PLUME SURFACES.
*       SOLID FORCE FREE BODY USED TO SIMULATE THE EXHAUST
*       PLUME.
*
PANEL                                                                               P1
10.         UPPER PLUME                                                             P2
*TYPE       WET         FORCE       IMAGE
 0.         -1.         0.          1.                                              P3
*
KSPACE                                                                              P9
*NK         KSPACE
 4.         0.                                                                      P9a
*
SECTION                                                                             S1
CYLINDRICAL                                                                         S4
XAXIS                                                                               S5
*THETA      RADIUS      X           KBREAK      NK          KSPACE
 90.        6.                                                                      S5a
  0.        6.                                                                      S5a
AT                                                                                  S11
*XT         YT          ZT          JBREAK      NJ          JSPACE
210.        0.          0.          1.          6.          2.                      S11a
```

Figure 7-34 Example 4 Dataset (8 of 9)

```
*
SECTION                                                                    S1
AT                                                                         S11
*XT      YT      ZT      JBREAK   NJ     JSPACE
270.     0.      0.      1.       2.     0.                                S11a
*
SECTION                                                                    S1
RECTANGULAR                                                                S2
*X       Y       Z       KBREAK   NK     KSPACE
0.       0.      0.                                                        S2a
AT                                                                         S11
*XT      YT      ZT      JBREAK   NJ     JSPACE
280.     0.      0.                                                        S11a
*
*
PANEL                                                                      P1
11.      LOWER PLUME                                                       P2
*TYPE    WET     FORCE   IMAGE
 0.      -1.     0.      1.                                                P3
*
KSPACE                                                                     P9
*NK      KSPACE
 4.      0.                                                                P9a
*
SECTION                                                                    S1
CYLINDRICAL                                                                S4
XAXIS                                                                      S5
*THETA   RADIUS  X       KBREAK   NK     KSPACE
  0.     6.                                                                S5a
-90.     6.                                                                S5a
AT                                                                         S11
*XT      YT      ZT      JBREAK   NJ     JSPACE
210.     0.      0.      1.       6.     2.                                S11a
*
SECTION                                                                    S1
AT                                                                         S11
*XT      YT      ZT      JBREAK   NJ     JSPACE
270.     0.      0.      1.       2.     0.                                S11a
*
SECTION                                                                    S1
RECTANGULAR                                                                S2
*X       Y       Z       KBREAK   NK     KSPACE
0.       0.      0.                                                        S2a
AT                                                                         S11
*XT      YT      ZT      JBREAK   NJ     JSPACE
280.     0.      0.                                                        S11a
*
```

Figure 7-35 Example 4 Dataset (9 of 9)

8. PANEL ABUTMENTS

8.1 INTRODUCTION

This chapter provides information on panel edge abutments and the automatic procedure used in QUADPAN to establish panel abutments and element neighbors. The limitations on panel abutments are presented, along with guidelines for setting the input parameters controlling the abutment search. Finally, the panel abutment list in the output print from each QUADPAN run is discussed.

8.2 ABUTMENTS

Before further discussion it is important to understand the terminology and relevance of panel abutments. It should be realized that panel abutments deal only with the "developed" surface lattice, after geometry generation is complete and a regular lattice is defined.

Input geometries for QUADPAN are represented by a lattice of quadrilateral surface elements. These surface elements are grouped into a number of PANELS in order to simplify the task inputting the geometry, and to organize the large number of elements required to define that geometry into smaller, more manageable pieces. The division of the configuration surface into panels is illustrated in Figure 6-2 and Figure 6-3.

As described in 6.4.2, it is desirable that the surface lattice for QUADPAN be as continuous as possible, with each element contiguous with its neighboring elements. This is illustrated in Figure 6-4. These element neighbors are used in determining the surface velocity, by a finite difference procedure on the surface potentials at the nearby elements. They are also used to set up Kutta conditions at wake shedding lines, and to properly treat wake side edges that touch other body surfaces.

If only a single panel were used to define the entire geometry, with a topologically rectangular lattice of elements, there would be no problem in defining an element's immediate neighbors, but only simple geometries could be described. Instead, the configuration is described by a collection of panels, and the task of locating neighboring elements is make more complex for those elements that lie on the edges of the panels. In this case, neighboring elements will be located on the edges of other nearby panels.

An ABUTMENT is defined by the contiguous elements on neighboring panel edges. Panel abutments consist of the elements on the edge of a panel that are contiguous to, or touching, edge elements on another panel, the same panel, or the image of the same panel across the plan of symmetry.

8.3 RULES ON PANEL ABUTMENTS

Panels are regarded by the program as topologically rectangular, with four panel edges and a row and column grid of quadrilateral elements, regardless of the actual geometrical shape. Abutments concern the elements that lie on the panel edges.

The following ground rules apply to panel abutments:

- Panels may only (only!!) abut (touch) one another at their edges.
- Panels edges that are degenerate (zero length) are defined as not abutting, with no neighbors.
- Any number of panels may abut a panel edge, however, no more than four elements may abut (touch) one another along a common element edge. Figure 6-8.

Except for these limitations, panels may abut in any fashion. Edges may abut themselves or other edges in the same panel or different panels.

8.4 AUTOMATIC ABUTMENT PROCEDURE

The program contains an automatic procedure to search all panel edges to establish contiguous neighboring elements across panel boundaries. This procedure will normally find all contiguous abutments in the configuration, relieving the user of the burden of specifying connecting panel edges and elements in all but a few extreme cases.

The panel abutment procedure works by searching for panel edge elements whose side midpoints (the side midpoint lies between that element's lattice point at the panel edge) are within a specified distance of one another (See Figure 8-2). As a result of this search technique elements which are contiguous (i.e., that line up with one another) will be found to abut. In addition to checking all of the user defined panel edges, the abutment search also checks for neighboring elements that lie across the plane of symmetry when the configuration is laterally symmetric.

As mentioned above, degenerate panel edges do not abut other panel edges The abutment procedure identifies degenerated panel edges by comparing the panel edge length to the absolute geometric tolerance distance – ABSTOL (discussed below). The panel edge length is defined by the sum of the lengths of the sides of the edge elements, and the edge is degenerate if this length is less than ABSTOL.

8.4.1 Abutment Search Distance

As discussed above, the automatic abutment procedure uses a geometric tolerance distance in its search to establish the panel edge element abutments. Element side midpoints on abutting elements must be within this tolerance distance. In order to make the abutment search as reliable as possible, especially in complex cases where the element sizes may vary over a large range, the search distance used by the program is not constant. Instead, the search distance is set by two user-specified parameters, ABSTOL and RELTOL.

- ABSTOL is the absolute search tolerance, specified in global coordinate units.
- RETOL is the relative search tolerance, specified as a fraction of the length of the element edge.

All abutting element side midpoints must be within the tolerance distance defined by the smaller of ABSTOL and RELTOL* (element edge length). The ABSTOL parameter gives an upper limit to the search distance, while RELTOL provides a locally scaled limit to the search distance that is proportional to the length of the element edge whose neighbors are being sought.

8.4.2 Setting the Abutment Parameters

The automatic abutment procedure is controlled by two user-specified parameters, ABSTOL and RELTOL, that set the tolerance distance used for the abutment search. Both of these parameters have defaults, but some adjustment is normally necessary for each case. It is recommended that ABSTOL be used as the primary means to control the abutment search.

ABSTOL is the absolute search tolerance, specified in global coordinate units. This parameter is the primary means available to control the automatic abutment procedure. ABSTOL is normally set sufficiently small that there are no abutment warnings or errors due to unintended neighbors, and sufficiently large that all the abutting edges are found by the program. For configurations where the geometry is defined precisely, with no gaps, ABSTOL may be set quite small (within reason, as discussed below). When there are gaps, ABSTOL must be adjusted, and /or user-specified abutments used to find all the abutting edges or avoid ambiguous or unintended abutments.

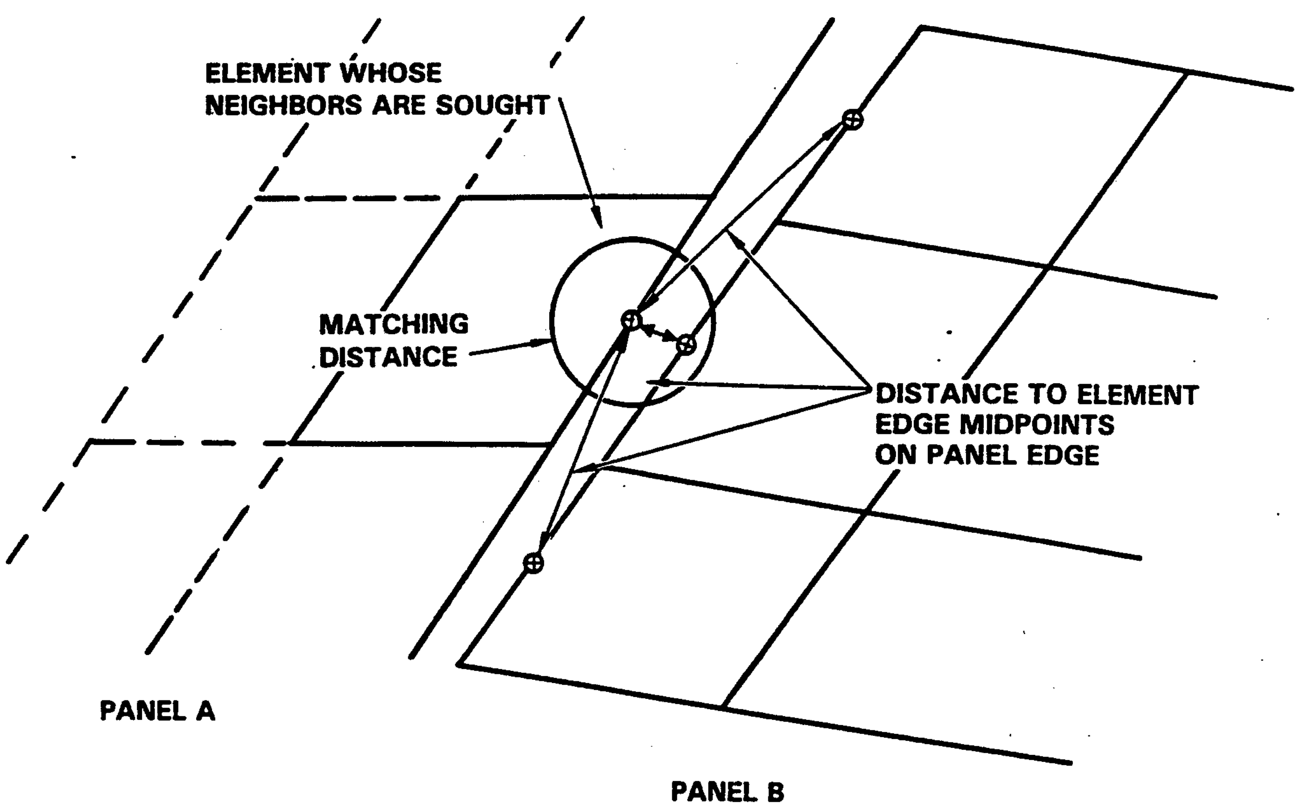


Figure 8-1 Operation of Abutment Search

As a result of limited numeric precision in any computer, there is a lower limit to the value that ABSTOL may be given, This lower limit must be larger than the product of the available machine precision (as a fraction of 1) and the maximum X, Y, and Z coordinate value at any panel edge. For the IBM computer the available precision is roughly 5.0E-7. If, for example, the configuration were 1,000 units long in the X direction, ABSTOL should be set larger than 5.0E-4, with 0.005 as a good starting value.

RELTOL is the relative tolerance distance, as a fraction of the element edge length. This parameter provided a local scaling to the abutment tolerance distance. This parameter defaults to 0.1, and normally should be given this value. RELTOL acts to limit the minimum angle between panels before the abutment search determines that they are connected, such as at the tip edge of upper and lower wing panels near the trailing edge. This parameter should never be given a value higher than 0.5.

8.5 USER-SPECIFIED ABUTMENTS

In the great majority of cases the automatic procedure will establish the element connectivity without user intervention. This will be true especially where the input geometry is relatively gapless, and the abutment search) tolerance is set small. Some cases, however, may require the user to restrict the search space for the automatic abutment search to eliminate possible abutment ambiguities.

Like any automatic procedure, the abutment search can sometimes fail. This normally happens when it finds too many nearby elements at an element edge. This happens in two situations:

- If more than two body type elements (or wake type elements) abut at a panel edge the program issues a warning message because there is not obvious single neighboring element to use to calculate the surface velocity. In this ambiguous situation the program will ignore all neighboring elements at that element edge, basing velocities on backward gradients calculated without crossing the panel edge. The accuracy at this junction is reduced, exactly the same as when noncontiguous elements are used. This is only a problem where the gradients across the junction are significant.

- If more than four neighboring panels abut at an edge, the program issues a warning message because the program limitation of four abutting panels is exceeded. This may need to be fixed before the run should be done. In this situation, like the previous one, the program will ignore all neighboring elements at that element edge, and velocities will be calculated without crossing the panel edge. This results in a reduction of accuracy at this junction, which is only a problem where the gradients across the junction are significant.

These situations usually a rise when there are small gaps between edges, and the abutment search tolerance has had to be increased. This is also sometimes a problem at the side edges of wing panels where very small elements are used near a thin trailing edge.

To eliminate such abutment ambiguities, it may be necessary to restrict the search space used by the abutment procedure. The search space, which normally arranges over all the panel edges in the configuration should check for abutting elements. This can be done using the optional keyword ABUT (0). The use of a user-specified abutment does not guarantee that panel edges abut, it merely specifies which edges may be searched with the automatic procedure.

8.5.1 Application of User-Specified Abutments

In addition to their use in fixing abutment errors, user-specified abutments may also be used to control some aspects of the potential flow calculation. The coarsely paneled swept wing, illustrated in

Figure 8-2, is an example. Only three spanwise columns of panels have been used, which concentrates too few elements near the tip to capture the details of the flow in that region. If the gradient around the tip is high, which will be the case for large angles of attack, the velocities on the outboard strip of elements may be adversely affected by the coarse spacing distribution used near the wingtip (the spanwise velocities will be affected and may be too high).

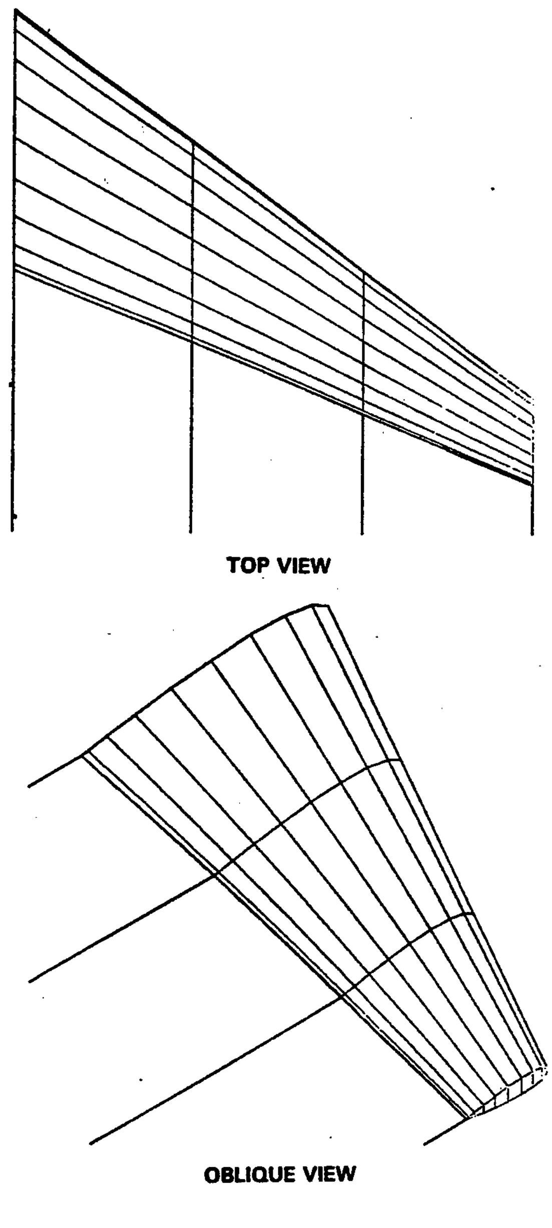


Figure 8-2 Swept Wing With Coarse Spanwise Paneling

The most accurate way to model this configuration is to place several narrow strips of elements near the wing tip and around the wing tip closure to properly represent the gradient there. However, if the actual details of the flow near the tip are not required, the additional element s might not be needed. Since the tip effects are localized, it is undesirable to have QUADPAN spread them over one-third of the wing by using the potential on the tip to calculate the velocity on the outboard wing elements. The best technique for this crude geometric model is to prevent the code from using the using tip elements to calculate the velocities on the wing surface by restricting the abutment search at the tip. In this way, the user can partially overcome deficiencies resulting from an inadequate number of elements at the wing tip.

This can be accomplished if the panels are forced to be noncontiguous by "disconnecting" the upper and lower wing panel tip edges from the tip. The search space for those panels can be altered using the ABUT keyword to specify 0 for the abutting panel(s) on those edges. It should be pointed out, however, that if the paneling were sufficiently concentrated in the tip region to capture the gradients, the user would probably do as well, or better, by letting the program connect the panels together.

8.6 PANEL ABUTMENT EXAMPLE

The operation of the automatic panel abutment procedure is illustrated in Figure 8-3. This case consists of three flat panels of different size, denoted as:

Panel A ID-> 11	modeled as an NK = 2 by NJ = 6 element array
Panel B ID-> 12	modeled as an NK = 3 by NJ = 3 element array
Panel C ID-> 13	modeled as an NK = 2 by NJ = 4 element array

The edge numbers and local J and K indices within each panel are shown in the figure.

The Panel Abutment List produced by QUADPAN for this three panel geometry is shown in Figure 8-4. The abutment parameters for this case are shown preceding the abutment list (ABSTOL = .002, RELTOL = .1). The abutments for each of the edges of the panels is listed in a compressed form, given by the starting and ending J and K indices of the abutting elements on both sides of the abutment. For example, Edge 2 of PANEL A (11) has been found to abut elements on edge 3 of PANEL B (12) and edge of 2 of PANEL C (13). Notice that a number of the edges have been found that do not abut any other panels. In addition, edge 3 of PANEL C and edge 2 of PANEL B have not been found to abut. This is due to noncontiguous elements at those edges.

Although use of the Abutment List can be confusing, because the user must consider the orientation of panel and its edges, it is one of the essential tools for the debugging and validation of QUADPAN datasets.

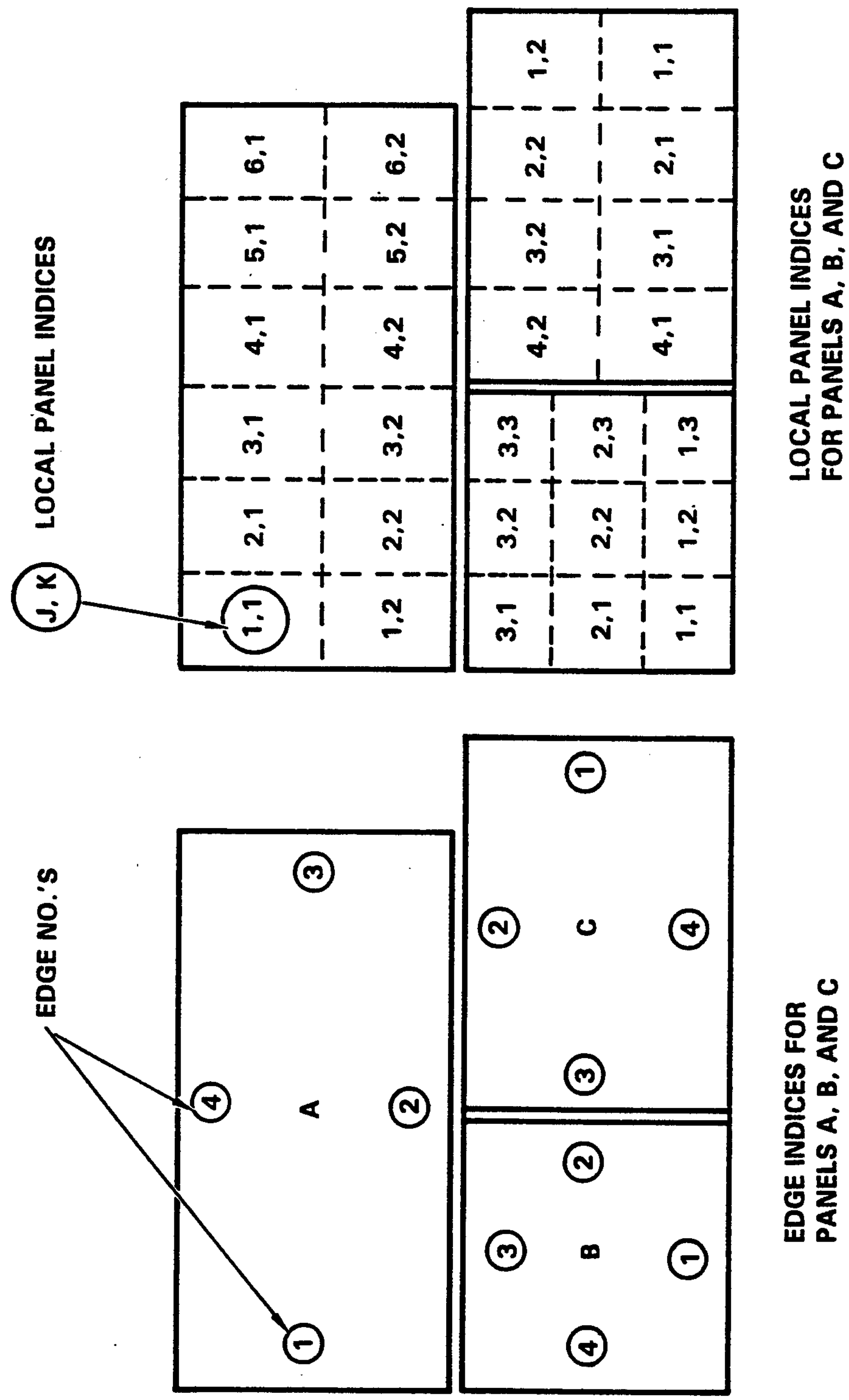


Figure 8-3 Abutment Example

```
EXAMPLE ABUTMENTS FOR 3 PANELS

PANEL ABUTMENT CHECK

   SEARCH PARAMETERS USED FOR ESTABLISHING PANEL EDGE ABUTMENTS
     ABSTOL =  0.2000E-02     (MAXIMUM ABSOLUTE MATCHING DISTANCE BETWEEN ELEMENT EDGE MIDPOINTS)
     RELTOL =  0.1000E+00     (MAXIMUM RELATIVE MATCHING DISTANCE BETWEEN ELEMENT EDGE MIDPOINTS
                               AS A FRACTION OF THE ELEMENT EDGE LENGTH)


** PANEL ABUTMENT LIST **

*-----------------------------------------------------------------------------------------------------
ABUTMENTS FOR PANEL ID->   11        NJ =   6       NK =   2                         PANEL A

  EDGE     J,  K  TO  J,  K  ABUTS  PANEL  EDGE    J,  K  TO  J,  K

    1      1   1      1   2            NO ABUTMENTS FOUND

    2      1   2      3   2          12     3     3   1      3   3                   PANEL B
    2      4   2      6   2          13     2     4   2      2   2                   PANEL C

    3      6   2      6   1            NO ABUTMENTS FOUND

    4      6   1      1   1            NO ABUTMENTS FOUND
*-----------------------------------------------------------------------------------------------------
ABUTMENTS FOR PANEL ID->   12        NJ =   3       NK =   3                         PANEL B

  EDGE     J,  K  TO  J,  K  ABUTS  PANEL  EDGE    J,  K  TO  J,  K

    1      1   1      1   3            NO ABUTMENTS FOUND

    2      1   3      3   3            NO ABUTMENTS FOUND

    3      3   3      3   1          11     2     3   2      1   2                   PANEL A

    4      3   1      1   1            NO ABUTMENTS FOUND
*-----------------------------------------------------------------------------------------------------
ABUTMENTS FOR PANEL ID->   13        NJ =   4       NK =   2                         PANEL C

  EDGE     J,  K  TO  J,  K  ABUTS  PANEL  EDGE    J,  K  TO  J,  K

    1      1   1      1   2            NO ABUTMENTS FOUND

    2      1   2      1   2            NO ABUTMENTS FOUND
    2      2   2      4   2          11     2     6   2      4   2                   PANEL A

    3      4   2      4   1            NO ABUTMENTS FOUND

    4      4   1      1   1            NO ABUTMENTS FOUND
END OF PANEL ABUTMENT CHECK
```

Figure 8-4 Example Abutments List

9. REFERENCES

1. Neill, D.J., Herendeen, D.L., Venkayya, V.B., “ASTROS Enhancements, Volume I - ASTROS User's Manual,” WL-TR-95-3004, May 1995.

2. Johnson, E.H. and Venkayya, V.B. "Automated Structural Optimization System (ASTROS), Theoretical Manual, "AFWAL-TR-88-3028, Vol. 1 December 1988.

3. Love, M.H., “Software Design Document for The Aerodynamic Analysis for the Design Environment,” FZM 8399, 17 June 1996.

4. Love, M.H., “Aerodynamic Analysis for the Design Environment Final Report Theoretical and Application Studies Document,” FZM 8536, 31 July 1998.

5. Love, M.H., “Aerodynamic Analysis for the Design Environment User's Document,” FZM 8538, 24 July 1998.

6. Neill, D.J., Herendeen, D.L., Venkayya, V.B., “ASTROS Enhancements, Volume II - ASTROS Programmer's Manual,” WL-TR-95-3006, May 1995.