SUPERELEMENT PLANNING, DATA MANAGEMENT, AND MULTIPLE BOUNDARY CONDITIONS

WRITTEN BY

ROBERT F. COOKE, PH.D
BOEING COMPUTER SERVICES COMPANY
565 ANDOVER PARK WEST
TUKWILA, WASHINGTON 98188

PRESENTED AT

MACNEAL-SCHWENDLER CORPORATION'S
USER CONFERENCE IN
PASADENA, CALIFORNIA
MARCH, 1986

ABSTRACT

This paper discusses three critical issues affecting finite element method (FEM) analysis of large NASTRAN models using Superelements: planning, data base management, and multiple boundary conditions. Significant progress has been made in each of these areas which in turn provides project management with greatly improved control, visibility, and efficiency (reduced analysis cost). A data base management modification is presented which reduces long-time disk requirements by 80% and facilitates combining subcases. A strategy and DMAP for combining subcases is presented. A Superelement spreadsheet is presented which displays on one sheet of paper the Superelement plan, bookkeeping information, and the implementation status of a multi-run job. This spreadsheet is the key to understanding the interrelationship between jobs, data bases, data base sets, disk files, magnetic tapes, SEID, and SEID operation. This visibility is required for planning, execution, restarting, and error analysis.

CONTENTS

SUPERELEMENT PLANNING, DATA MANAGEMENT, AND MULTIPLE BOUNDARY CONDITIONS

1.0 INTRODUCTION

The use of Superelements is almost intrinsically required when you have large models, large projects, limited machine resources, and/or special types of analysis. For a structure represented by a large model having 10,000 grid points or more (with several boundary conditions and a computer of your choice), significant problems must be addressed before and during the analysis.

First, analyzing a real model this size will require a staff of some 100 people who support and are supported by the FEM analysis. A staff of this size is typically organized into teams. Each team is responsible for a portion of the model, which may be indentified by Superelements, and supplies input accordingly. Each team expects output (data recovery) for only their area of responsiblity on a timetable determined by their individual needs and schedule.

To support each team's individual output requirements it is possible to run all data recovery (SEDR) in one job and leave it to the entire staff to sort through this one monstrous printout. This is not a practical approach since it could consist of some two to twenty boxes of paper. Thus it follows that many customized SEDR jobs are required to circumvent this situation. To do this successfully, we think of each user of data (e.g., stress analyst) as a customer. The customer is supplied SEDR upon request. Early in the project, as part of the planning phase, each user is requested to supply requirements for SEDR. Following completion of the NASTRAN execution and certification (that the

model appears good, e.g., no mechanisms), the scheduled SEDR are launched. Examination of these results usually suggests additional SEDR for the purposes of:

- Resolving problems as to modeling, execution, NASTRAN limitation, or actual designs
- Understanding load paths
- Studying critical areas

Stringent weight/cost design criteria or evolutionary structural arrangements tend to require more attention to design/analysis detail and hence more SEDR on an ad hoc basis. Thus, SEDR may be required over a period of several weeks whereas, the actual solution execution may only take several days.

Secondly, Superelement solutions do not loop on boundary conditions. Results must be suitably combined to avoid confusion by the staff. That is, engineers expect results for left/right sides not symmetric/antisymmetric half models.

Thirdly, there is a problem with the economics of data storage. Data storage on magnetic tape is cheap, but it is also slow and not as reliable as on disk. Disk storage is fast and reliable, but it is expensive and limited in size and availability. Obviously, we would like to use tape for the large, infrequently used data blocks and disk for small, frequently used data blocks. Recall that matrix crunching takes several days whereas SEDR may be sporadic, interactive and span several weeks.

Fourthly, the complexity involved when satisfying many customers as efficiently as possible (runs, boundary conditions, data base, DBSET, disk files, tape) causes bookkeeping problems which must be solved.

In light of these factors, specific methodologies and techniques have been developed which: (1) systematizes the planning/coordination process culminating in a solution coordination spreadsheet; (2) automates the processing of multiple boundary conditions utilizing DMAP; and (3) improves the economics of NASTRAN usage through effective data storage management.

This paper describes the methodologies and techniques developed as part of the disciplined approach found necessary to solve large NASTRAN Superelement problems.

2.0 MULTIPLE BOUNDARY CONDITIONS

Superelement solutions in NASTRAN (SOL 61, 64, 69) do not loop on boundary conditions. Only one boundary condition request (SPC) is honored per run. A technique has been developed to handle multiple boundary conditions using DMAP. Prior to developing this approach, several alternatives were considered but were not used for the reasons mentioned below. Each of these techniques either restricts flexibility in choosing the Superelement tree, or requires the execution of the entire model for each boundary condition.

- SOL 24 With Alters RF24D13A And RF24D13B

  This technique is not attractive for large models because it requires a DECOMP for each boundary condition for the entire model. Also, restarts to add load cases may trigger DECOMP. This method is attractive for small models, but is unattractive when checking out the model with symmetric boundary conditions before proceeding to antisymmetric conditions (conclusion based upon a bug encountered in Version 61).

- Cyclic Symmetry

  This technique requires that the CYJOIN grid points be added to the Residual Superelement. This restricts choices on Superelement definition, and would invariably cause a larger number of Superelement boundary degrees of freedom, resulting in increased machine CPU cost.

- MSC Application Note August, 1983

  This Note describes a DMAP alter applicable to SOL 61 if all changed constraints (SPC and MPC) are in the Residual Superelement. This is not economically attractive if we have configuration changes via SEEXCLUDE in which we would like to promote upstream grid points on the centerline of a symmetric half model.

- External Superelements

  This technique does not permit data recovery in the external Superelement.

The method developed to handle Superelement solutions for multiple boundary conditions involves a DMAP procedure that relies on an intimate understanding of how MSC/NASTRAN performs data recovery. The method developed performs a separate execution for each boundary condition. This can be done for both SOL 24 and SOL 61. SOL 61 is preferred if the model is large and can be partitioned such that at least half the model has unchanged constraints (SPC, MPC). This is to be expected for symmetric/antisymmetric boundary conditions. Only the SEID which have changed boundary conditions (as well as their downstream SEID) need to be executed for more than one boundary condition. This technique offers the greatest machine cost effectiveness and avoids the restrictions of the techniques mentioned earlier. A description of the data recovery process appears in Appendix A. The multiple boundary condition procedure is outlined in the following series of steps.

- Place as much of the model as possible (practical) that is common to all boundary conditions/ configurations in upstream Superelements, and execute SEMG, SEMA. This step is an economic (machine CPU) consideration and may not be desirable for small models.

- Execute SEMG, SEMA for each boundary condition and model configuration.

- Execute SELG, SELA, SEDR for the entire model, for each SEID, boundary condition, and configuration. Each job should save on a distinct data base UGVS, QGS, PG, and PJ. A data base printout should be checked to verify that UGVS is present for each Superelement. Subsequent data recovery requires only UGVS, QGS, PG, PJ, and the SEMG data bases. The large data blocks such as GOAT, KOO, and LOO are no longer required for additional data recovery. It is important that SEDR must be performed for each SEID.

- Combine results via DMAP

  The results are combined by using a DMAP sequence to solve the equation:

$$U_u = U_S * FACT_S + U_a * FACT_a$$

where:

U = solution vector (PJ, PG, QGS, UGVS)

FACT = DMI matrix specifying how subcases are combined

Subscripts are:

a = antisymmetric component

s = symmetric component

u = unsymmetric combination

The DMI factors specify how the component subcases are combined. See Appendix D1. Note that the DMI are diagonal matrices for purely symmetric boundary conditions.

The DMAP sequence (Appendix D2) loops on each SEID in SLIST (words 1, 8, 15, . . .) obtained from the SEMAP via module SEP3. Datablock ULV (or data name SDR1DONE) is stored to avoid triggering SDR1 in residual SEDR.

- Machine Generated DMI and SUBCASE

Special care needs to be taken because the data blocks being combined contain no subcase identification. It is easy to make a DMI or subcase mistake and get the wrong combined results. The approach developed avoids these human mistakes by using a FORTRAN program (Appendix D4) to read the subcase pointers and write the DMI and subcase for all jobs (S, A, U). However, even with automated procedures a manual check of displacements

for one grid in each subcase and SEID is mandatory. See Appendix D4 for the FORTRAN program, example input parameters (subcase pointers), and example DMI/subcase output.

A closely related problem is the handling of moveable Superelements for structures with moveable or removable parts, such as the flaps on an airplane or the bucket on a crane. For such structures it can be a simplification to have (in one printout) SORT2 type data recovery for all combinations of boundary conditions and model configurations. This is accomplished by:

- Using a unique SEID for each position (location) of the moveable part (configuration).

- Separate SEMG and SELG computer runs as required to satisfy uniqueness of both grid and element numbers. Uniqueness is required for input processing and partitioning the bulk data into Superelements (GEOMiS). Uniqueness is not required at the data block, data base level. Hence, separate models are not required for each position, just separate CORD cards defining the position.

- SEEXCLUDE as required.

It is noted that the temperature loads and enforced displacements which are not common to all boundary conditions require the combination of tables DIT and ETT via direct table input (DTI). In our experience, this has not been required or attempted.

3.0 DATA BASE MANAGEMENT

This section discusses DBSET modifications to enable:

1. Use of multiple boundary conditions and,

2. Economical use of database storage

Large models generate large amounts of data such that storage on disk for one week can exceed the CPU cost of generating the data. Storage requirements grow as the "3/2 power" with the number of grid points as shown in Table 3-1. The usual alternative is to save the data on a less costly media; dismountable packs or magnetic tapes rather than on-line disk. However, large multi-run substructured models having multiple boundary conditions and/or data recovery jobs may require hundreds of tape mounts involving tens of tapes. This reduces disk cost at the expense of labor, throughput, and reliability. Neither extreme is acceptable. The problem becomes more pronounced for larger models and for vector machines. This section extends the insight into the data blocks required for multiple boundary conditions and discusses the DBSET feature to split data bases. This reduces disk cost (up to 80%) and/or magnetic tape cost so that it is economical to support many SEDR jobs over an extended period of time.

3.1 UNMODIFIED MSC/NASTRAN DATA BLOCK STORAGE

Data blocks are presently stored by MSC/NASTRAN on a database via the DBSTORE command. In general all data blocks are directed to a DBSET defined as follows:

DBSET 1 -- contains data internal to a Superelement such as:

ECT, BGPDT . . .

KOO, LOO, GOAT

KFS, KSS, KSF

UGV, PG, QG

DBSET 2 -- Contains boundary data, or data required for downstream processing such as:

KAA, PA

3.2 PROBLEM WITH THE UNMODIFIED DATA BLOCK STORAGE CONVENTION

For our requirements of multiple boundary conditions and multiple SEDR, the following problems exist with the present convention:

1. The loads data blocks (PA, PG, . . .) occur in both DBSET 1 and 2 and are not uncoupled from the model data blocks. This could cause confusion or duplication of some data blocks.

2. DBSET 1 contains KOO, LOO, GOAT which is about 80% of the total data space (Table 3-1) and, is not needed after the initial SEDR (Table 3-3).

The data blocks KOO, LOO, GOAT comprise about 80% of the data space (Table 3-2). These data blocks are created in an SEMA operation. Subsequently, they are used only

for SELA and an "initial" or "branch" SEDR as shown in Table 3-3 and Appendix C. This suggests storing KOO, LOO, and GOAT on tape leaving all other data on disk.

3.3 THE SOLUTION: REDIRECT DATA BLOCKS

A DMAP alter has been developed to redirect selected data blocks as shown in Appendix D5, D6. This alter creates two DBSETS:

- DBSET 5 -- all load, displacement data
- DBSET 6 -- KOO, LOO, GOAT plus other miscellaneous data not needed after the first data recovery

These alters redirect specified data blocks to a different DBSET and nothing more. The user has control of disk files (DBij) and file size via the NASTRAN card which specifies each DBSET as a collection of DBij. This topic is discussed further in Appendix B.

These alters uncouple load data from model data for multiple boundary conditions and split KOO, LOO, GOAT onto their own database. This database represents 80% of the data space required, is infrequently used, and therefore should be stored on tape not disk.

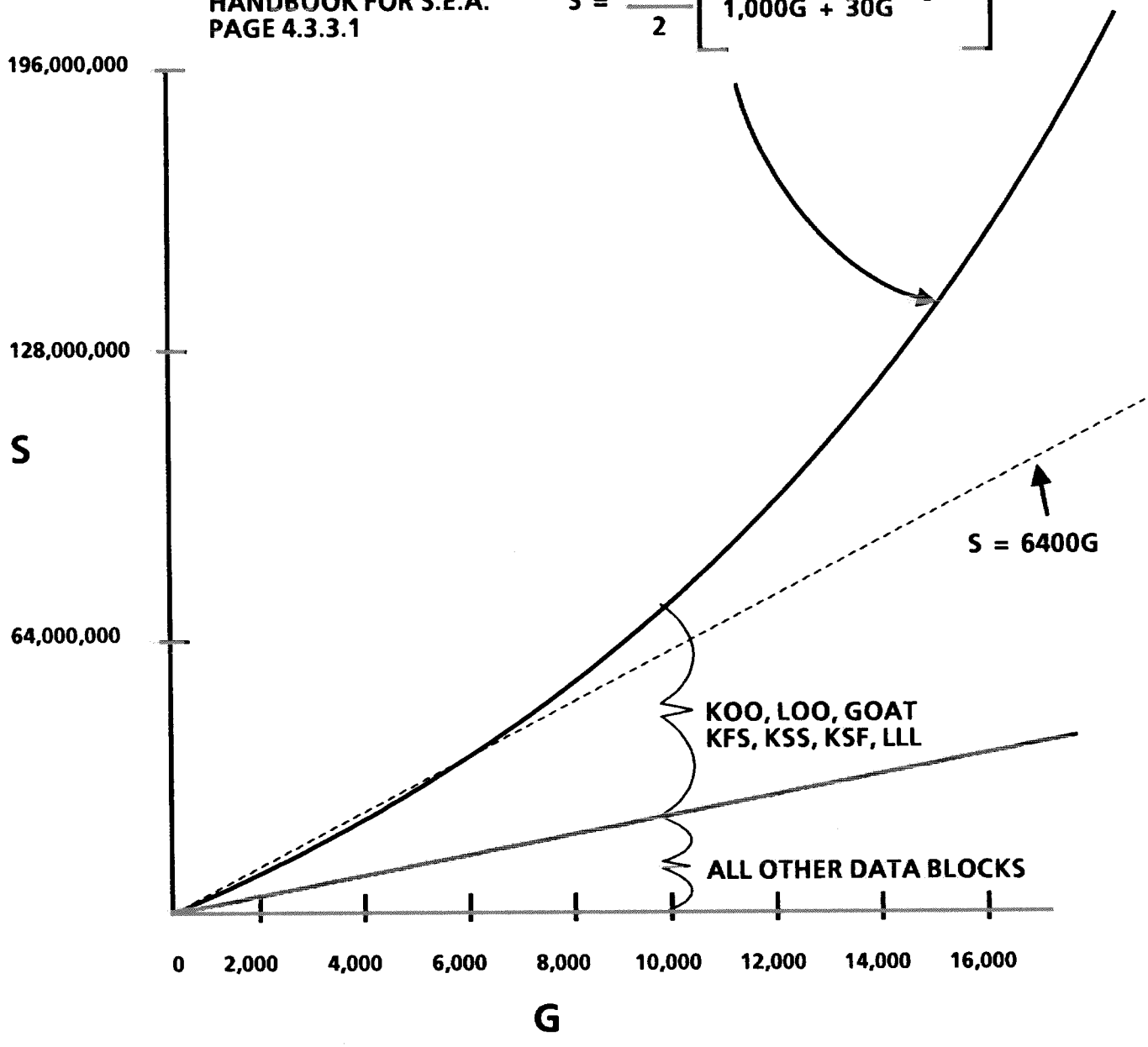
In summary, these modifications enable multiple boundary conditions and multiple disk based SEDR to be run over an extended period of time with 20% of the total disk cost associated with the unmodified convention.

TABLE 3-1

DATABASE SPACE REQUIREMENTS

Formula:
HANDBOOK FOR S.E.A.
PAGE 4.3.3.1

$$S = \frac{3}{2}\left[1{,}000G + 30G^{\frac{3}{2}}\right]$$



G = Number Of Grid Points
S = Words [Long Word Machine]

TABLE 3-2

SIZE OF DATA BLOCKS FOR TYPICAL S.E.

EXAMPLE: GRIDS = 1,707 ELEMENTS = 6,600 BLOCK SIZE = 1,792 Words

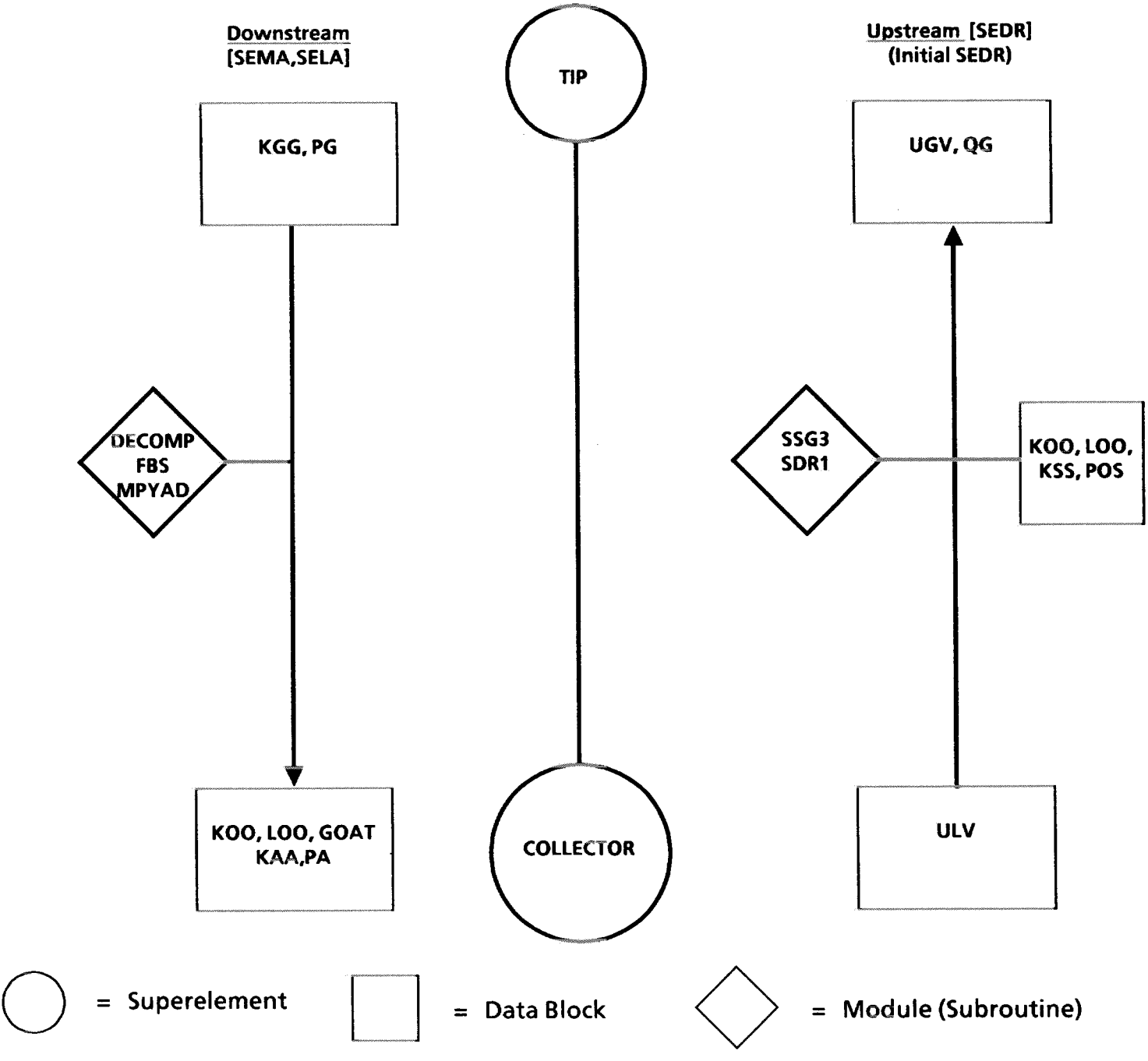
D.O.F. G = 10,242
S = 2,538
F = 7,704
O = 6,894
A = 810

% SPACE	DBSET	DATABLOCK	NUMBER OF BLOCKS*	
		ECTS	33	
		ESTS	102	
		GPECT	52	
		KELM, KDICT	208	
		OTHER	60	
9.1	1	SUBTOTAL	464	464
		KOO	149	
		LOO	1,469	
		GOAT	2,504	
		KFS, KSS, KSF	38	
		KGG	189	
85.4	6	SUBTOTAL	4,349	4,349
	2	KAA	263	263
		UGVS QGS	6	
		OTHER	19	
5.2	5	SUBTOTAL	25	25
100.0			TOTAL	5,101

* SMALL DATA BLOCKS NOT SHOWN INDIVIDUALLY

TABLE 3-3

BRANCH PROCESSING



4.0 PLANNING - SUPERELEMENT SPREADSHEET

Planning and organizing are the key to successful use of multi-run Superelements. In preparation of the plan attention should be given to:

- Objectives
- Assumptions
- Boundary conditions (which SEID have multiple SPC)
- Schedule -- availability/requirements for:
  - (a) model
  - (b) properties
  - (c) loads
  - (d) results (data recovery)
  - (e) restarts-add load cases, sensitivity studies
- Machine resources -- CPU, DISK
- Criticality -- in the event of a crisis, what becomes most important

One can now propose a multi-run plan which is a compromise based upon:

- Machine capability/availability

- Competing engineering groups (everybody cannot have their SEID in the Residual - impacts efficiency of sensitivity studies)

- NASTRAN Superelement rules:
  - (a) SETREE
  - (b) Operations (SEMG, SEMA, SELG, SELA, SEDR)
  - (c) Data base store/fetch

The plan itself specifies for each computer run the interrelationship between:

- SE operations (SEMG, SEMA, SELG, SELA, SEDR)
- SEID
- Read data base, disk file, magnetic tape VSN
- Write data base, disk file, magnetic tape VSN
- Data recovery requirements of each individual team/user.

This information is entered on a form (spreadsheet) similar to TABLE 4-1.

Each DBij must fit on a single logical disk device. Further, it is easier for the computer operator to manage the disk if the DBij are no larger than one-half of a logical device. This is achieved by creating several DBij for each DBSET and is discussed in Appendix E.

TABLE 4.1

MODEL: _____

DATE: _____

W = Write
R = Read

JOB STEP	JOB NAME	OPERATION	SEID
0		SEMAP	ALL
1		SEMG SEMA	TIPS
2		SEMG, MA SELG, LA	SYM C.L. ALL
3		SEMG, MA SELG, LA	ASYM C.L.
4		DMAP S ± A	ALL
5		SEDR	ANY
6			

VSN														
SEKTORS														
DBSET	1	1	2	5	6	1	2	5	6	1	2	5	6	5
Read DBij	16	15	14	13	12	11	10	9	8	11	10	9	8	5
	W													
	R	W	W		W									
	R		R		R	W	W	W	W					
	R		R		R					W	W	W	W	
								R				R		W
	R	R				R								R

●
●
●

This spreadsheet describes the interrelationship between job step, data base file, and data base set per DBMGR rules. The spreadsheet is vital in each of the following phases:

- Planning -- serves as a document for communication to the team
- Data preparation-- serves as the coding form for keying in JCL, CASE and EXEC control decks
- Execution -- charts progress
- Error analysis -- traces data base errors
- Historical log -- the attitude is emphasized, "if there is a serious problem, the plan failed, not NASTRAN," therefore, the effort needs to be planned carefully
- Model change -- depicts what can be salvaged, what must be rerun

5.0 SUMMARY

Superelements are a powerful technique for analyzing large models by automated substructuring.

Planning their use is not automated, but can be rendered routine through thorough organization and some experience. Visibility of the plan and the ensuing bookkeeping that is required, is accomplished by means of the Superelement spreadsheet. Multiple boundary conditions can be rendered routine by the automated procedures described. Long time disk requirements can be reduced up to 80% by redirecting large infrequently used data blocks (GOAT, KOO, LOO) onto their own data base/magnetic tape.

We encourage the MacNeal-Schwendler Corporation to:

- Incorporate provisions for redirecting GOAT, KOO, and LOO (plus a few other data blocks)

- Support multiple boundary conditions as a simple extension of automated substructuring

I gratefully acknowledge the following persons who contributed to the content or editing of this paper: Wayne Dimming (FORTRAN to generate DMI), Jefty Geller (use of techniques in a production environment), David Atler (Cyber Control Language procedures), John Nylander, Andy Mera, and Ervin Herness for editorial assistance. Also Mike Gockel, Carl Hennrich, Dean Bellanger and Jerry Joseph for Superelement training and consultation. Special thanks to Bill Mayer, Dennis Bjornson, and Ken Coke for their support in obtaining permission to publish this paper.

6.0 GLOSSARY

SUPERELEMENT (SEID) -- substructure of entire model

SETREE -- specifies the upstream/downstream SEID connections

SEPLAN -- multirun overview: which SEID, operations are executed in each computer run.

SPREADSHEET -- multirun detail: interrelationship between SE operation and data bases

DATA BLOCK -- matrix or table

DATA BASE (DBij) -- collection of data blocks

DATA BASE SET (DBSET) -- set of DBij

SPECIFIC DATA BLOCKS
- GOAT -- partitions of stiffness matrix
- KOO -- partitions of stiffness matrix
- LOO -- partitions of stiffness matrix
- UGVS -- displacements
- PG -- loads
- QGS -- constraint forces
- KSS, PSS, POS -- constraint dependent stiffness, loads

MODULE
- DECOMP, FBS, MPYAD -- modules perform matrix operations
- SSG3 -- upstream branch processing
- SDR1 -- upstream branch processing
- SEMG, SEMA -- SE Operations: SE matrix generation, assembly
- SELG, SELA -- SE Operations: SE load generation, assembly
- SEDR -- Data Recovery

APPENDICES

APPENDIX A

How does MSC/NASTRAN do data recovery?

The answer can be found by studying SOL 61 DMAP operations SEP4 thru SDR2. This is less than 100 statements out of 900..

a. Module SEDR searches the data base for UGVS and QGS, and sets parameter NOSDR1 to .TRUE. if both are found. Note that if the S-set is null then QGS is not stored. NOSDR1 is used later to decide if modules SSG3 and SDR1 are to be executed to calculate UGVS and QGS.

b. Initial data recovery

   NOSDR1 is .FALSE. and SSG3 and SDR1 are executed to calculate UGVS from the boundary displacement (ULV or ULVS), constraint forces (PSS, POS) and matrices KOO, LOO, GOAT.

c. Additional data recovery

   NOSDR1 is .TRUE. and we go directly to data recovery in SDR2 using whatever UGVS, QGS have been found on data base.

d. Conclusion

   Results can NOT be combined at the ULV level since the modules that follow use constraint related data blocks (PSS, POS) which are (in general) different for different boundary conditions.

   Results can be combined at the UGVS, QGS level. However, subsequent execution of SSG3, SDR1 must be prevented as both a safety measure and if the S-set is null.

   KOO, LOO, GOAT are not required after an initial SEDR.

e. Warning

   If SSG3/SDR1 are inadvertently executed and KOO is NOT available, null UGVS will result.

APPENDIX B

By default, NASTRAN partitions data blocks into:

a. DBSET 1 - internal data
b. DBSET 2 - external or boundary data

We focus on DBSET 1. It contains two types of data:

a. KOO, LOO, and GOAT - are only required for added load cases or initial data recovery.
b. EST, ECT,..... - are required for data recovery.

Inspection of the data base dictionary shows that just a few datablocks (KOO, LOO, and GOAT) use most of the space (see Table 3-1). A study of the Superelement solution sequence (DMAP, Appendix C) shows that KOO, LOO, and GOAT are only required for:

a. SEMA - generates KOO, LOO, and GOAT
b. SELA
c. Initial SEDR (SSG1, SDR1)
d. Restart to change model (multiple SEID on same data base)

Situations in which several data recovery restarts are required over an extended period of time suggest redirecting GOAT, KOO, and LOO to their own data base. This infrequently used data base can then be put on magnetic tape (or discarded) reducing expensive long-time permanent disk costs by some 80%. This also facilitates many disk-based customized data recovery jobs in which the speed of data base access is apparent, especially for interactive post processing.

Options for adding load cases are:

a. Use PARAM, SOLID
b. Uncouple load runs on separate data bases

It is felt that uncoupling sets of loads on separate data bases is more fail safe, especially if combining load cases via DMAP. Relevant data blocks are GEOM3S, PG, PJ, QGS, UGVS, and ETT and of less interest POS, PSS, PL, PA, and SLT. These are all relatively small and can be left on permanent disk. However, only PJ, PG, QGS, UGVS, and ETT are required for data recovery.

APPENDIX C

NASTRAN SUPERELEMENT SOLUTION SEQUENCE QUESTION:

What modules and data blocks are required for:
added load case?
change of boundary conditions?
data recovery?

MODULE		FUNCTION	DATA BLOCKS
	PREFACE	sorts bulk data	$GEOM_j$
	SEP1	generate S. E. map	-SEMAP
	SEP3	determines which SEID operation to process	-SLIST
SEMG	SEP2	partition GEOMi into GEOMiS for each SEID in SLIST	
	GP1	grid point tables	
	GP2	element connectivity	-ECT
	GP3	static loads	
	TA1	element summary (properties)	-EST
	EMG	Element matrix generation	-KELM
	EMA	Element matrix assembly	-KGG
SELG	SSG1	generate loads	-PJ
SEMA	SEMA	assemble matrices KJJ + sum of KAA	-KGG
	GP4	displacement sets, rigid element, enforced displacement	-USET,RG,YS
	UPARTN	partition constraints sets	-KOO
	DECOMP	decompose KOO	-LOO
	FBS	forwards backwards substitution	-GOAT
	MPYAD	form KAA from GOAT	-KAA

APPENDIX C (continued)

MODULE		FUNCTION	DATA BLOCKS
SELA	SELA	assemble load PJ + sum of PA	-PG
	SSG2	use KFS, KSS to partition loads	-POS,PSS
	SSG3	Use LLL, KLL, PL to find residual displacement	-ULV
SEDR	SEP4	fetch UGVS, QGS	
	SEDR	fetch ULV, set parameter NOSDR1	
	COND	ABC, NOSDR1 $ Jump to ABC if NOSDR1.LT.0	
	SSG3	use LOO, KOO, POS to find UOOVS	
	SDR1	use UOOVS, ULVS to compute internal displacement	-QGS,UGVS
	LABEL	ABC	
	SDR2	process specific data recovery request per case control	
		"Initial" Data Recovery	

CONCLUSIONS:

a. Initial SEDR computes internal displacements (UGVS) from the boundary values (ULVS), and uses SPC related data.

b. Subsequent SEDR skips over SSG3, SDR1 and does not require GOAT, KOO, LOO.

c. Results can be combined at the UGVS, QGS level for each Superelement, but not at any ULV level that involves changed constraints.

APPENDIX D
COMBINING SUBCASES BY DMAP

APPENDIX D1

COMBINING LOADS

$$V_u = V_s\ FACT_s + V_a\ FACT_a$$

Configuration 1 (Subcases 9 ± 10 = S ± A)

```
DMI, SFACT1,   0,   1,   2,   1,   1,  20
               1,   1,  +1
               2,   2,  +1
               3,   3,  +1
               4,   4,  +1
               5,   5,  +1
               6,   6,  +1
               7,   7,  +1
               8,   8,  +1
               9,   9,  +1
              10,   9,  +1

DMI, AFACT1,   0,   2,   1,   1,  20,  20
               9,  10,  +1
              10,  10,  -1
```

APPENDIX D1 (continued)

COMBINING LOADS

$$V_u = V_s\ FACT_s + V_a\ FACT_a$$

Configuration 2 (Subcases 11 ±12, 14 ±15, 18 ±19 = S ± A)

```
DMI, SFACT2,   0,   1,   2,   1,   1,   20
              11,  11,  +1
              12,  12,  +1
              13,  13,  +1
              14,  14,  +1
              15,  15,  +1
              16,  16,  +1
              17,  17,  +1
              18,  18,  +1
              19,  19,  +1
              20,  19,  +1

DMI, AFACT2,   0,   2,   1,   1,  20,   20
              11,  11,  +1
              12,  12,  -1
              14,  14,  +1
              15,  15,  -1
              18,  19,  +1
              19,  19,  -1
```

APPENDIX D2
COMBINE RESULTS, TWO BOUNDARY CONDITIONS

```
$
$.....      DM7V62B       .....
$
$..... O N E  CONFIGURATION  .....
$
DBMGR //0/-1/2000////DB05/ $ GIVES 56,000 SEKTORS
$
DBMGR //2/0/0/15/1 $ ALPHABET.
MATPRN SFACTD,AFACTD// $ PRINT DMI
$.....
$
$...
DBMGR //10/0/0/-5///SDR1DONE $  INITIAL DATA RECOV. DONE
$...
DBFETCH /SEMAP,CCASECC,,,/0/0/1/-16 $
TABPT CCASECC// $
SEP3 CCASECC,SEMAP/SLIST/S,N,NP/S,N,XSEID $  XSEID=-1
TABPT SLIST// $
PARAM //ADD/V,N,SCONT/-5/XSEID $       SCONT = -6
PARAML SLIST//TRAILER/1/V,N,NSEID $      NO. SEID
$              JUMP  ENDSGL $
$.....$
$
LABEL TOPSGL $  AAAAAAAAAAAAAAAAAAAAAAAA
$
PARAM //ADD/V,N,SCOUNT/+7/SCONT $      GRAB WORDS 1,8,15,22,..
PARAML SLIST//DTI/1/V,N,SCOUNT//V,N,SEID $SEID=SLIST(SCONT)
PARAM //ADD/V,N,LPFLG/-1/SEID        $ LPFLG = SEID - 1
$
$.....
DBFETCH /SUGVS,SPG,SPJ,SPJ1,SQGS/0/SEID/1/-2 $  SYMM.CODE=S
DBFETCH /AUGVS,APG,APJ,APJ1,AQGS/0/SEID/1/-4 $  ASYM.CODE=A
$
MPYAD SUGVS,SFACT,/SCOMP $ UGVS
MPYAD AUGVS,AFACT,/ACOMP $ UGVS
ADD SCOMP,ACOMP/UGVS/ $
$
MPYAD  SPG,SFACT,/SPG $  PG
MPYAD  APG,AFACT,/APG $  PG
ADD SPG,APG/PG/  $
$
MPYAD  SPJ,SFACT,/SPJ $  PJ
MPYAD  APJ,AFACT,/APJ $  PJ
ADD SPJ,APJ/PJ/  $
$
MPYAD  SPJ1,SFACT,/SPJ1 $ PJ1
MPYAD  APJ1,AFACT,/APJ1 $ PJ1
```

```
ADD SPJ1,APJ1/PJ1/ $
$
PARAML QGS//PRES////V,N,NULLBLOK $ NULLBLOK = -1 IF PURGED
COND NULLQGS,NULLBLOK $
MPYAD SQGS,SFACT,/SQGS $ QGS
MPYAD AQGS,AFACT,/AQGS $ QGS
ADD SQGS,AQGS/QGS/ $
JUMP ENDQGS $
LABEL NULLQGS $
DBMGR //10/0/SEID/-5///QGSFLAG $ S-SET IS NULL
LABEL ENDQGS $
DBSTORE UGVS,PG,PJ,PJ1,QGS//0/SEID/-5 $
$.....
$
COND ENDSGL,LPFLG $
REPT TOPSGL,1000 $
LABEL ENDSGL $
END $
```

APPENDIX D3
COMBINE RESULTS, TWO BOUNDARY CONDITIONS, TWO CONFIGURATIONS

```
$
$.....      DM8V62B       .....
$
$..... T W O  CONFIGURATIONS .....
$
DBMGR //0/-1/2000////DB05/ $ GIVES 56,000 SEKTORS
$
DBMGR //2/0/0/15/1 $ ALPHABET.
MATPRN SFACTD,AFACTD// $ PRINT DMI
$.....
$
$...
DBMGR //10/0/0/-5///SDR1DONE $  INITIAL DATA RECOV. DONE
$...
DBFETCH /SEMAP,CCASECC,,,/0/0/1/-16 $
TABPT CCASECC// $
SEP3 CCASECC,SEMAP/SLIST/S,N,NP/S,N,XSEID $  XSEID=-1
TABPT SLIST// $
PARAM //ADD/V,N,SCOUNT/-5/XSEID $       SCOUNT = -6
PARAML SLIST//TRAILER/1/V,N,NSEID $     NO. SEID
$
$              JUMP  ENDSGL $
$.....$
$
LABEL TOPSGL $ AAAAAAAAAAAAAAAAAAAAAAAA
$
PARAM //ADD/V,N,SCOUNT/+7/SCOUNT $      GRAB WORDS 1,8,15,22,..
PARAML SLIST//DTI/1/V,N,SCOUNT//V,N,SEID $SEID=SLIST(SCOUNT)
PARAM //ADD/V,N,LPFLG/-1/SEID        $ LPFLG = SEID - 1
$
$.....
DBFETCH /SUGVS,SPG,SPJ,SPJ1,SQGS/0/SEID/1/-2  $ SYMM=UP.CODE=S
DBFETCH /AUGVS,APG,APJ,APJ1,AQGS/0/SEID/1/-4  $ ASYM=UP.CODE=A
DBFETCH /XUGVS,XPG,XPJ,XPJ1,XQGS/0/SEID/1/-12 $ SYMM=DN.CODE=X
DBFETCH /YUGVS,YPG,YPJ,YPJ1,YQGS/0/SEID/1/-14 $ ASYM=DN.CODE=Y
$
MPYAD SUGVS,SFACTU,/SCOMPU $ UGVS
MPYAD AUGVS,AFACTU,/ACOMPU $ UGVS
MPYAD XUGVS,SFACTD,/SCOMPD $ UGVS
MPYAD YUGVS,AFACTD,/ACOMPD $ UGVS
ADD5 SCOMPU,ACOMPU,SCOMPD,ACOMPD,/UGVS $
$
MPYAD SPG,SFACTU,/SPGU $  PG
MPYAD APG,AFACTU,/APGU $  PG
MPYAD XPG,SFACTD,/SPGD $  PG
MPYAD YPG,AFACTD,/APGD $  PG
ADD5 SPGU,APGU,SPGD,APGD,/PG  $
```

```
$
MPYAD SPJ,SFACTU,/SPJU $ PJ
MPYAD APJ,AFACTU,/APJU $ PJ
MPYAD XPJ,SFACTD,/SPJD $ PJ
MPYAD YPJ,AFACTD,/APJD $ PJ
ADD5 SPJU,APJU,SPJD,APJD,/PJ $
$
MPYAD SPJ1,SFACTU,/SPJ1U $ PJ1
MPYAD APJ1,AFACTU,/APJ1U $ PJ1
MPYAD XPJ1,SFACTD,/SPJ1D $ PJ1
MPYAD YPJ1,AFACTD,/APJ1D $ PJ1
ADD5 SPJ1U,APJ1U,SPJ1D,APJ1D,/PJ1 $
$
PARAML QGS//PRES////V,N,NULLBLOK $ NULLBLOK = -1 IF PURGED
COND NULLQGS,NULLBLOK $
MPYAD SQGS,SFACTU,/SQGSU $ QGS
MPYAD AQGS,AFACTU,/AQGSU $ QGS
MPYAD XQGS,SFACTD,/SQGSD $ QGS
MPYAD YQGS,AFACTD,/AQGSD $ QGS
ADD5 SQGSU,AQGSU,SQGSD,AQGSD,/QGS $
JUMP ENDQGS $
LABEL NULLQGS $
DBMGR //10/0/SEID/-5///QGSFLAG $ S-SET IS NULL
LABEL ENDQGS $
$
DBSTORE UGVS,PG,PJ,PJ1,QGS//0/SEID/-5 $
$.....
$ ... DBFETCH /GBGPDTS,GEQEXINS,GCSTMS,GUSET,/0/SEID/1/-16 $
$ ... DBFETCH /GSILS,GGPLS,,,/0/SEID/1/-16 $
$ ... MATGPR GGPLS,GUSET,GSILS,UGVS//H/G $ PRINT UGVS
$ ... VECPLOT PG,GBGPDTS,GEQEXINS,GCSTMS,,/QAG/V,Y,GRDPNT/0/+1/PGLOAD $
$
COND ENDSGL,LPFLG $
REPT TOPSGL,1000 $
LABEL ENDSGL  $
$
DBMGR //2/0/0/-5/0 $
$
END $
```

APPENDIX D4
FORTRAN GENERATED DMI/SUBCASE

```
      PROGRAM CASECDS(INPUT,TAPE1,TAPE2,OUTPUT,TAPE3=INPUT,TAPE4=OUTPUT,
     1TAPE5,TAPE6,TAPE7,TAPE8,TAPE9,TAPE10)
C
C     PROGRAM - CASECDS
C     DATE - 21JUNE84
C     BY - DIMMIG
C     PURPOSE - CREATE CASE CONTROL AND DMI CARDS FOR NASTRAN
C          BASED ON USER SUPPLIED LOAD DATA
C      = = > SEE CASPROC FOR FILES USED
C
C     UPDATE - 21JUNE84
C     BY - GELLER
C        - CORRECTED DMI GENERATION SECTION AND USYM
C          FILE OUTPUT.
C
C
C
C     UPDATED - 2/20/85
C
C     BY DIMMIG
C
C        - MODIFIED TO CREATE RECTANGULAR DMI'S TO GET RID OF DUMMY
C          CASES REDUCING CRUNCH TIME & SPACE ON FBS
C
C
C MODIFIED BY DIMMIG TO ALLOW DIFFERENT MODEL TITLES
C
C
      DIMENSION SID1(999),SID2(999),LABEL(999,4),DUMLAB(4),GRIDS(100)
      INTEGER SID1,SID2,PLABEL,SLABEL,ALABEL,DUMMY,DUMLAB,ALPHA,FACTOR,F
     1LAG,GRIDS,SID,FLAG1,TIT
      DATA DUMMY/99999/
      DATA DUMLAB/8HDUMMY CA,8HSE    ,8H      ,8H      /
      DATA NOLABEL,PLABEL,SLABEL,ALABEL/10H       ,10HPURE SYM. ,10HS
     1YM. COMP ,10HASYM. COMP/
      DATA SID1,SID2/999*0,999*0/
      NGRIDS=0
      FLAG1=0
      WRITE(4,1011)
 1011 FORMAT(/* INPUT A MODEL TITLE (UP TO 8 CHARACTERS).*)
      READ(3,1012) TIT
 1012 FORMAT(A8)
      IF(EOF(3)) 200,210
  200 CONTINUE
      TIT=8HGARBAGE
  210 CONTINUE
    5 CONTINUE
```

```
      NGRIDS=NGRIDS+1
      READ(1,*) GRIDS(NGRIDS)
      IF(EOF(1)) 25,5
   25 CONTINUE
      NGRIDS=NGRIDS-1
      NCASE=0
   10 CONTINUE
      NCASE=NCASE+1
      READ(2,1006) SID1(NCASE),SID2(NCASE),(LABEL(NCASE,J),J=1,4)
 1006 FORMAT(2I8,5X,4A8)
      IF(EOF(2)) 20,15
   15 CONTINUE
      SID=SID1(NCASE)
  130 CONTINUE
      IF(FLAG1.EQ.1) GO TO 120
      IF(NGRIDS.EQ.0) FLAG1=1
      IF(NGRIDS.EQ.0) WRITE(4,1008)
 1008 FORMAT(/* NO GRIDS INPUT ON FILE 'GRIDS'*/* THEREFORE NO 'FORCE0'
     1FILE CREATED!*)
      IF(NGRIDS.EQ.0) GO TO 120
      DO 110 I=1,NGRIDS
      WRITE(10,1007) SID,GRIDS(I)
 1007 FORMAT(*FORCE  *,2I8,*      0.    1.    0.    0.*)
  110 CONTINUE
      IF(SID2(NCASE).EQ.0) GO TO 120
      IF(SID.EQ.SID2(NCASE)) GO TO 120
      SID=SID2(NCASE)
      GO TO 130
  120 CONTINUE
      NTOTAL=NTOTAL+1
      IF(SID2(NCASE).NE.0) NTOTAL=NTOTAL+1
      GO TO 10
   20 CONTINUE
      NCASE=NCASE-1
      WRITE(4,1002) NCASE
 1002 FORMAT(/* NUMBER OF CASES READ IN = *,I3)
C
C  WRITE OUT THE PURELY SYMMETRIC LOAD CASES
C
      II=0
      NSYM=0
      DO 40 I=1,NCASE
      IF(SID2(I).NE.0) GO TO 40
      II=II+1
      NSYM=NSYM+1
      WRITE(5,1000) SID1(I),SID1(I),TIT,SID1(I),(LABEL(I,J),J=1,4),PLABE
     1L,II
 1000 FORMAT(*$*/*SUBCASE *,I8/*  LOAD = *,I8/*  LABEL = *,A8,*.*,I8,*
     1.*,4A8,*$ *,A10/*$ SUPER= 100,*,I3/*$*,40(*-*))
 1009 FORMAT(*$*/*SUBCASE *,I8/*  LOAD = *,I8/*  LABEL = *,A8,*.*,I8,*
     1 +*,I8,*.*,4A8,*$ *,A10/*$ SUPER= 100,*,I3/*$*,40(*-*))
 1010 FORMAT(*$*/*SUBCASE *,I8/*  LOAD = *,I8/*  LABEL = *,A8,*.*,I8,*
     1 -*,I8,*.*,4A8,*$ *,A10/*$ SUPER= 100,*,I3/*$*,40(*-*))
   40 CONTINUE
```

```
      WRITE(4,1001) NSYM
 1001 FORMAT(/* NUMBER OF PURELY SYMMETRIC CASES OUTPUT = *,I3)
C
C  WRITE OUT THE ASYMMETRIC COMPONENT CASES WITHOUT DUMMY SYMMETRIC
C  COMPONENTS
C
      II=0
      NASYM = 0
      DO 50 I=1,NCASE
     II=II+1
C     WRITE(6,1000) SID1(I),DUMMY,TIT,DUMMY,(DUMLAB(J),J=1,4),NOLABEL,II
      IF(SID2(I).EQ.0) GO TO 50
      II=II+1
      NASYM = NASYM + 1
      WRITE(6,1000) SID2(I),SID2(I),TIT,SID2(I),(LABEL(I,J),J=1,4),ALABE
     1L,II
   50 CONTINUE
      WRITE(4,1003) NASYM
 1003 FORMAT(/* NUMBER OF ASYMMETRIC COMPONENTS = *,I3)
C
C  WRITE OUT THE PURE SYMMETRIC CASES, THE SYMMETRIC COMPONENTS AND
C   NO DUMMY ASYMMETRIC CASES
C
      II=0
      DO 60 I=1,NCASE
      II=II+1
      IF(SID2(I).EQ.0) WRITE(7,1000) SID1(I),SID1(I),TIT,SID1(I),(LABEL(
     1I,J),J=1,4),PLABEL,II
      IF(SID2(I).NE.0) WRITE(7,1000) SID1(I),SID1(I),TIT,SID1(I),(LABEL(
     1I,J),J=1,4),SLABEL,II
      IF(SID2(I).EQ.0) GO TO 60
C     II=II+1
C     WRITE(7,1000) SID2(I),DUMMY,TIT,DUMMY,(DUMLAB(J),J=1,4),NOLABEL,II
   60 CONTINUE
C
C  WRITE OUT ALL THE CASES
C    1) PURE SYMMETRIC
C    2) SYMMETRIC COMPONENTS
C    3) ASYMMETRIC COMPONENTS
C
      II=0
      DO 70 I=1,NCASE
      II=II+1
      IF(SID2(I).EQ.0) WRITE(8,1000) SID1(I),SID1(I),TIT,SID1(I),(LABEL(
     1I,J),J=1,4),PLABEL,II
      IF(SID2(I).NE.0) WRITE(8,1009) SID1(I),SID1(I),TIT,SID1(I),SID2(I)
     1,(LABEL(I,J),J=1,4),SLABEL,II
      IF(SID2(I).EQ.0) GO TO 70
      II=II+1
      WRITE(8,1010) SID2(I),SID2(I),TIT,SID1(I),SID2(I),(LABEL(I,J),J=1,
     14),ALABEL,II
   70 CONTINUE
C
C  WRITE OUT THE DMI FILE
```

```
C
C
   NSYM=NSYM+NASYM
   ALPHA=1HS
   WRITE(9,1004) ALPHA,NSYM,NTOTAL
   ALPHA=1HA
   WRITE(9,1004) ALPHA,NASYM,NTOTAL
 1004 FORMAT(*DMI,*,A1,*FACT,0,2,1,1,,*,I3,*,*,I3)
   WRITE(9,1104) NTOTAL
 1104 FORMAT(*$*/*$ ULV DMI FOR TOTAL # OF LOADCASES*/*$*/*DMI,ULV,0,2,1,
1,1,,1,*,I3/*$*)
   LC=0
   LCSYM = 0
   LCASYM = 0
   DO 80 I=1,NCASE
   WRITE(9,1101)SID1(I),SID2(I)
   LCSYM = LCSYM + 1
   LC=LC+1
   ALPHA = 1HS
   FACTOR = 4H+1.0
   WRITE(9,1005) ALPHA,LC,LCSYM,FACTOR
   IF(SID2(I).EQ.0)GO TO 80
   LCASYM=LCASYM+1
   ALPHA=1HA
   WRITE(9,1005) ALPHA,LC,LCASYM,FACTOR
   LC=LC+1
   ALPHA=1HS
   WRITE(9,1005) ALPHA,LC,LCSYM,FACTOR
   FACTOR=4H-1.0
   ALPHA=1HA
   WRITE(9,1005) ALPHA,LC,LCASYM,FACTOR
  80 CONTINUE
 1005 FORMAT(*DMI,*,A1,*FACT,*,I8,*,*,I8,*,*,A4)
 1101 FORMAT(*$*/,*$ SYM SID = *,I6,3X,*ASYM SID = *,I6,/*$*)
C
   END$
```

INPUT

```
   1000          SYMMETRIC GUST
   1002          SYMMETRIC GUST
   1200   1201   ROLL MANEUVER
   1212   1213   ROLL MANEUVER
```

DMI - OUTPUT

```
DMI,SFACT, 0, 2, 1, 1, 4, 6
DMI,AFACT, 0, 2, 1, 1, 2, 6
$
```

```
$
$
$
$  SYM SID -   1000   ASYM SID =      0
$
DMI,SFACT,     1,          1,+1.0
$
$  SYM SID -   1002   ASYM SID =      0
$
DMI,SFACT,     2,          2,+1.0
$
$  SYM SID -   1200   ASYM SID = 1201
$
DMI,SFACT,     3           3,+1.0
DMI,SFACT,     4           4,+1.0
DMI,AFACT,     3           4,+1.0
DMI,AFACT,     4           4,- 1.0
$
$
$  SYM SID -   1212   ASYM SID = 1213
$
DMI,SFACT,     5           5,+1.0
DMI,SFACT,     6           6,+1.0
DMI,AFACT,     5           6,+1.0
DMI,AFACT,     6           6,- 1.0
```

SUBCASE - OUTPUT

```
$
SUBCASE 1000
    LOAD =   1000
    LABEL = E X A M P L E  .     1000.
       $ PURE SYM.
$   SUPER = 100,   1
$  . . . . . .
$
SUBCASE 1002
    LOAD =   1002
    LABEL = E X A M P L E  .     1002.
       $ PURE SYM.
$   SUPER = 100,   2
$  . . . . . .
$
SUBCASE 1200
    LOAD =   1200
    LABEL = E X A M P L E  .     1200  +     1201.
              $
 ASYM. COMP
$   SUPER = 100,   3
$  . . . . . .
$
SUBCASE 1201
```

```
    LOAD =    1201
    LABEL = E X A M P L E  .      1200  -      1201.
                $
 ASYM. COMP
$   SUPER = 100,   4
$  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
$
SUBCASE 1212
    LOAD =    1212
    LABEL = E X A M P L E  .      1212  +      1213.
                $
 ASYM. COMP
$   SUPER = 100,   5
$  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
$
SUBCASE 1213
    LOAD =    1213
    LABEL = E X A M P L E  .      1212  -      1213.
                $
 ASYM. COMP
$   SUPER = 100,   6
```

APPENDIX D5
DBSET 5 - LOADS, DISPLACEMENT

```
$
$.....
$
$   NASTRAN ALTER VER 62B  SOL 61
$
$   PUT LOADS AND DISP. ON DBSET = 5
$
$   CHECK FOR: QGSFLAG = NULL S-SET
$
$         SDR1DONE = ULV NOT PRESENT
$
$
$.....
$
ALTER 43, 43  $
 DBSTORE GEOM3//V,Y,SOLID/0/5/NP $
ALTER 45, 45  $
 DBSTORE EDT//SOLID/0/5/NP $
ALTER 47, 47  $
 DBSTORE DIT//SOLID/0/5/NP $
ALTER 197,197  $
 DBSTORE GEOM3S//V,Y,SOLID=0/SEID/5 $
ALTER 253,253  $
 DBSTORE ETT,SLT//SOLID/SEID/5 $
ALTER 330,330  $
 DBSTORE PJ,PTELEM//SOLID/SEID/5 $
$XXXXXXXXXXXXXXXXXXXXXX USET
ALTER 351,351 $
 DBSTORE USET//MODEL/SEID/1 $
 DBSTORE USET//MODEL/SEID/2 $
$XXXXXXXXXXXXXXXXXXXXXX USET
ALTER 352,352  $
 DBSTORE YS//SOLID/SEID/5 $
ALTER 530,530  $
 DBSTORE PG//SOLID/SEID/5 $
ALTER 538,539  $
 DBSTORE PSS,QR,POS,PL//SOLID/SEID/5 $
 DBSTORE PA//SOLID/SEID/5 $
ALTER 550,550  $
 DBSTORE ULV//SOLID/SEID/5 $
ALTER 657,657  $
 DBSTORE UGVS,QGS//SOLID/SEID/5 $
ALTER 95, 95  $
 DBFETCH /EDT,GEOM3,,DIT,MATPOOL/SOLID/0//5 $
ALTER 315,315  $
 DBFETCH /ETT,SLT,,,/SOLID/SEID//5 $
ALTER 520,520  $
```

```
 DBFETCH /PJ,,,,/SOLID/SEID//5 $
ALTER 560,560  $
 DBFETCH /OLB,,,,/SOLID/PEID/1/5 $
ALTER 590,590  $
 DBFETCH /ULV,OLB,DIT,EDT,/SOLID/0//5 $
ALTER 599,599  $
 DBFETCH /SLT,,,,/SOLID/0//5 $
ALTER 630,630  $
 DBFETCH /PSS,PJ,POS,YS,QR/SOLID/SEID//5 $
ALTER 634,634  $
 DBFETCH /PJ,,,,/SOLID/SEID//5 $
ALTER 659,659  $
 DBFETCH /UGVS,QGS,PJ1,OLB1,/SOLID/SEID//5 $
ALTER 664,664  $
 DBFETCH /GEOM3S,ETT,,,/SOLID/SEID//5 $
ALTER 687,687  $
 DBFETCH /KSLT,FQGE,FDLT,,/SOLID/0/1/5 $
ALTER 770,770  $
 DBFETCH /PG,,,,/SOLID/SEID//5 $
ALTER 331,331 $
 DBMGR  //10/SOLID/SEID/5///SELGDONE $
ALTER 543,543 $
 DBMGR  //10/SOLID/SEID/5///SELRDONE $
ALTER 546,546 $
 DBMGR  //10/SOLID/SEID/5///SELRNG $
ALTER 527,527 $
 SELA PJ,SLIST,EMAP,EQEXINS/PG/PA/MAPS/SOLID/0/S,N,NP/SEID/5 $
ALTER 587,587 $
 SEP4 CASECC,PCDB,EMAP,XYCDB/DRLIST/APP/UGVS/SOLID/5/S,N,NP/
        S,N,SEID/S,N,NOLOAD/PUGV/QGS $
$
$.....
$
$  ONLY NEED FOR SEUPPLOT
$
ALTER 218,218 $ STORE EQEXINX,ECTX,BGPDTX,SILX .
ALTER 229,229 $ FETCH EQEXINX,ECTX,BGPDTX,SILX .
ALTER 798,798 $ FETCH EQEXINX,ECTX,BGPDTX,SILX .
$.....
$
$----------------------------------------------
$                    -
$  BUG - FIX  IF  S-SET  NULL   -
$                    -
$       IF S - SET  NOT NULL,  QGSFLAG = 0   -
$       IF S - SET   IS NULL,  QGSFLAG = -1   -
$                    -
$ OTHERWISE:  QGS IS NULL,              -
$       SSG3  SDR1 ARE,RE------DONE,      -
$       UGVS IS NULL UNLESS KOO IS AVAIL.   -
$                    -
 ALTER 657 $          AFTER DBSTORE QGS   -
  DBMGR //11/SOLID/SEID/0/S,N,QGSFLAG//QGS $     -
  COND NOQGS,QGSFLAG $                -
```

```
 JUMP YESQGS $                    -
 LABEL NOQGS $                    -
 DBMGR  //10/SOLID/SEID/5///QGSNULL $      -
 LABEL YESQGS $                   -
$                                 -
$---------- SKIP SSG3,SDR1 IF QGSNULL FOUND -
$---------- SKIP SSG3,SDR1 IF SDR1DONE FOUND -
$                                 -
ALTER 649 $              BEFORE SSG3   -
 DBMGR //11/SOLID/SEID/0/S,N,QGSFLAG//QGSNULL $  -
 COND LNOSDR1,QGSFLAG $           -
 DBMGR //11/SOLID/SEID/0/S,N,SDR1FLAG//SDR1DONE $ -
 COND LNOSDR1,SDR1FLAG $          -
$                                 -
$-------------------------------------------------
$                                 -
```

APPENDIX D6
DBSET 6 - KOO, LOO, GOAT

```
$
$.....
$
$   NASTRAN ALTER VER 62B  SOL 61
$
$ PUT GOAT=GO,KOO,LOO ON DBSET=6  NEED FOR LOADS, INITIAL SEDR
$ PUT KGG(GPSP1)    ON DBSET=6  NEED FOR LOADS, INITIAL SEDR
$
$.....
ALTER 139,139 $ KTT
ALTER 145,145 $
 DBSTORE GO//MODEL/SEID/6 $
ALTER 151,151 $ GO=GOAT
DBMGR  //9/MODEL/SEID/MODEL/SEID/6/GO/GOAT $
ALTER 155,155 $
 DBSTORE GOAT//MODEL/SEID/6 $
ALTER 160,160 $ GOQ=GOAQ
DBMGR  //9/SOLID/SEID/SOLID/SEID/6/GOQ/GOAQ $
ALTER 164,164 $
 DBSTORE GOAQ//MODEL/SEID/6 $
ALTER 343,343 $  DBSTORE KGG  NEED FOR GPSP1(LOADS)
 DBSTORE KGG//MODEL/SEID/6 $
ALTER 374,374 $
 DBSTORE GM//MODEL/SEID/6 $
ALTER 381,381 $
 DBSTORE KFS,KSS,KSF//MODEL/SEID/6 $
ALTER 390,390 $
 DBSTORE KOO,KVV//MODEL/SEID/6 $
ALTER 413,413 $
 DBSTORE LOO,UOO//MODEL/SEID/6 $
ALTER 441,441 $
 DBSTORE GOAT//MODEL/SEID/6 $
ALTER 466,466 $
 DBSTORE KLL//MODEL/SEID/6 $
ALTER 472,472 $
 DBSTORE LLL,ULL//MODEL/SEID/6 $
ALTER 499,499 $
 DBSTORE DM//MODEL/PEID/6 $
$ .....
ALTER 114,114 $
 DBFETCH /BUSET,,,,/MODEL/PEID/1/1 $
ALTER 140,140 $
 DBFETCH /GOAT,,,,/MODEL/SEID//6 $
ALTER 148,148 $
 DBFETCH /GO,,,,/MODEL/SEID//6 $
ALTER 157,157 $
 DBFETCH /GOQ,,,,/SOLID/SEID//6 $
```

```
ALTER 349,349 $   DBFETCH KGG  NEED FOR GPSP1(LOADS)
 DBFETCH  /KGG,,,,/MODEL/SEID//6 $
ALTER 394,394 $
 DBFETCH  /GO,,,,/MODEL/SEID//6 $
ALTER 411,411 $
 DBFETCH  /KOO,,,,/MODEL/SEID//6 $
ALTER 436,436 $
 DBFETCH  /LOO,UOO,,,/MODEL/SEID//6 $
ALTER 502,502 $
 DBFETCH  /GM,GOAT,DM,,/MODEL/PEID//6 $
 DBFETCH  /USET,,,,/MODEL/PEID//1 $
ALTER 519,519 $
 DBFETCH /KOO,LOO,KLL,LLL,KFS/MODEL/PEID//6 $
ALTER 625,625 $
 DBFETCH /GM,KFS,KSS,GOAT,/MODEL/V,N,PEID//6 $
 DBFETCH  /USET,,,,/MODEL/PEID//1 $
ALTER 651,651 $
 DBFETCH /KOO,LOO,,,/MODEL/PEID//6 $
$ .....
ALTER 440,440 $
 DBMGR   //5/MODEL/SEID/0/6//GO $
```

APPENDIX E
MAXIMUM SIZE OF A DBij

Planning - SE Spreadsheet

The maximum size of the DBij must be estimated as part of the planning and allocation of an adequate number of DBij on each DBSET and so entered in the spreadsheet.

a. Choose SETREE and run SEMAP to validate tree and obtain time and space estimates.
b. Select which SEID are to be run in each job step.
c. For each job step:

Estimate size of data base for each DBSET to enable choosing the size of each DBij. Considerations are:

1. Each DBij must fit on one logical disk device.
2. Size should be just under integer multiple of that which will fit on one tape.
3. Rules of thumb for CYBER (60 bit per word) are:

   (a) 1792 NASTRAN words per block (Buffsize)
   (b) 64 word per sector
   (c) 28 sectors per block (28 x 64 = 1792)
   64 million words per physical disk (CYBER 885). In general, allow 100 words per grid for models with less then 10,000 grids.

4. Resolve permanent storage into disk and tape in consideration of: duration of storage, frequency of access, and availability of disk resources. Since the data base format is random - sequential (1), all data bases for a given run must be available on disk for the duration of the run. If disk space is limited, the way it is configured (scratch, permanent) may constrain where tapes copied.

5. Pre-assign taped and disk device (machine dependent).