N78-32467

DYNAMIC STORAGE EXPANSION IN NASTRAN

Edwin N. Hess
Lockheed Electronics Co., Inc.

SUMMARY

Some functions of NASTRAN require a large block of working storage to execute. The method of meeting this requirement, because of insufficient data, has been to specify in advance an excessive amount to avoid a fatal exit. A method has been developed at the Lyndon B. Johnson Space Center (JSC) to calculate the amount of working space needed for the analysis and to inform the analyst of this data or, in the case of UNIVAC computers, to acquire this extra storage and continue the analysis.

INTRODUCTION

The design philosophy of NASTRAN dictated a completely open-ended design whenever possible. The use of a fixed dimension for large arrays was outlawed since this limited the size of the analysis that could be solved. Instead, modules were programmed to allocate space as required and to use spill logic to transfer data to scratch file if working space was limited.

The first public release of NASTRAN for the UNIVAC 1100 computers (level 11) assumed a limitation of direct addressing of 65,535 words. The HICORE system, which allowed indirect addressing of up to 262,143 words, was developed on level 12 NASTRAN at JSC. The UNIVAC computers were then competitive. As structures became larger and more complex, larger amounts of storage are required. The amount of working space for a particular analysis has been left to the analyst, with disastrous results. Either there was "insufficient core", leading to system fatal message 3008, or more working space than that required was attached and computer throughput and turnaround time suffered.

At JSC, fatal message 3008 has been changed to reflect the amount of working space required by the offending subroutine. Going beyond this, the branch to message 3008 was changed to branch to increase the space dynamically, and continue processing. The extra working storage required by a particular analysis is not necessarily that required to eliminate spill logic. We have found that spill may be economically advantageous in regards to extra time as opposed to an outrageous amount of working storage.

The subroutines mentioned will be either matrix subroutines which may be used by more than one module or module subroutines which are an exclusive part of the module. Utility and executive subroutines are not included except for changes necessary for the analyst's information. The term "working storage" is

used instead of core, and the word "problem" to mean analysis is avoided. A problem is encountered when an analysis fails.

FATAL MESSAGE 3008 (ref. 1)

System fatal messages usually consist of three parameters:

1. The message number
2. The data block name
3. The subroutine name

In the case of message 3008, the second parameter is not used, but is always set to zero, and the message reads:

SYSTEM FATAL MESSAGE 3008 - INSUFFICIENT CORE FOR SUBROUTINE NNN

Subroutine MSGWRT was altered to skip the FNAME call, which recovers the data block name, and the message rewritten to read:

SYSTEM FATAL MESSAGE 3008 - MMM ADDITIONAL CORE NEEDED FOR SUBROUTINE NNN

MODULE MODIFICATIONS

Most module and matrix subroutines have at least one branch to message 3008, but the majority of these are only safety valves and will not be taken if the working storage length is in the range of 20,000-25,000 words, which is the case when the UNIVAC is operating at the default core size of 65,536 words.

McCormick and Redner (ref. 2) studied the module core requirements and arrived at the following categories:

Group 0 - Modules which have no requirements of open core

Group 1 - Modules which require space for vectors or tables which do not exceed eight times the number of grid points in the model and do not provide spill.

Group 2 - Modules which require space for tables or matrices of variable size. Spill logic may be provided.

Group 3 - Modules for which the working space requirements are established by one or more matrix routines. Spill logic is usually present.

This information was used to determine which subroutines were likely to need additional working space. The decomposition subroutines real symmetric,

real unsymmetric, and complex require the largest space. The group 2 modules were also studied as to the working storage required.

Most subroutines have a preface section where file assignments and working space are calculated from table and matrix trailers. A calculation of working space is made with the result of a fatal message when insufficient. A simple change in these subroutines to add the second parameter in the call to subroutine message will give the analyst additional information of core requirements for subsequent analysis. Most of the matrix subroutines in group 3 were modified to state the amount of additional working space required.

Another method of calculating working storage is to read a record into working storage where the full record must be in core. If the number of words available is filled before the end of record is reached, the call to fatal message 3008 is taken. The following branch was added:

1. Reset the address of storage

2. Read the remainder of the record

3. The number of words read on the subsequent call(s) to read is the amount of insufficiency

This is necessary in modules such as TA1 (subroutine TA1A). This method is also used in subroutine XSORT as it prepares the continuation card dictionary. A correction was made by inserting a count, from which the space requirements could be calculated, of the continuation cards as they were read on the first pass through the bulk data cards.

Some care must be taken when using the results given by this message. On a large static analysis we found the following storage requirements:

TA1	80,000
RBMG2	90,000
SSG3	103,000

TYPICAL SUBROUTINE (SDCOMP)

Decomposition of a symmetric matrix is performed in steps by rows. The row under consideration is called a pivotal row. The contribution of the pivot row into each row of the resulting matrix is dependent upon the active (nonzero) column elements of that row and are combined with the corresponding column positions of the other rows. All computations can occur without spill if sufficient space is available to contain a triangular matrix whose row dimension is equal to the maximum number of active columns. When sufficient space is not available, the spill logic divides the triangular matrix into spill groups containing consecutive rows which will fit into the available space. It is expected that a reasonable compromise between time and space can

be realized by requesting sufficient working storage to contain a triangular matrix with a dimension equal to the average number of active columns. This scheme would allow for the majority of processing to be contained in core and allow spill for the larger pivotal rows (see fig. 1).

Current preface processing of SDCOMP involves the organization of working storage and the determination of spill groups. Before beginning computational processing, statistics gathered during the preface are printed for the user's information. The statistics reported include:

- o Maximum number of active columns
- o Space required to eliminate spill
- o Number of spill groups
- o Average number of rows in each spill group

Tests on a Space Shuttle analysis were made to determine the costs of spill to conform to available core, as shown in the following table. A decreasing benefit was derived after a certain point which shows eliminating spill is not beneficial. The figure for additional core should be that needed for the average column.

Storage size total (K)	Percent increase	Time in SDCOMP	Percent decrease
65		1520	
80	22	992	53
137*	71	657	50

*Required to eliminate spill.

The choice of subroutine SDCOMP as being typical was made because of the completeness of calculating optimum working space. This same method is used in subroutine GENVEC, which is a slave of both real unsymmetric and complex decompositions. Spill has not been calculated into requirements of any other subroutines at this time.

DYNAMIC CORE ALLOCATION

The ability to dynamically extend main storage without terminating an execution is available on the UNIVAC 1100 computers. This function has been successfully implemented at JSC.

The calls to message 3008 were changed to call a computer dependent subroutine EXPAND, reset necessary parameters, and return to the beginning of the

subroutine. Figure 2 shows the subroutine SDCOMP flow as regards dynamic expansion.

The design requirements of subroutine EXPAND were as follows:

- Provide for levels of expansion
- Access the UNIVAC 1100 function MCORE$
- Place a limit on expansion
- Provide for moving the contents of reserved storage
- Restore the contents of reserved storage
- Inform the user of the expansion

Levels of expansion were provided in case a matrix subroutine (SDCOMP) needed additional working storage after a module subroutine (INVPWR) had requested additional working storage and, as in these cases, the module subroutine reserves a section of storage not available to the matrix subroutine. Figure 3 shows a typical map of working storage area.

The limit on main storage is required by the addressable limit of 262,143 or by the computer facility.

The subroutine that requires additional working space calls EXPAND with the following parameters:

- Address of working storage
- Additional storage required
- Length of working space currently available
- The calling subroutine name

An additional entry into subroutine EXPAND (SHRINK) is called before exiting to provide for the restoration of the contents of the reserved area to its original position and to reduce the level index.

A new call to the 1100 executive (LCORE$) was made on each reentry into the main Module driver subroutines XSEMii. This provided for the release of core to its default value following each module.

MODIFICATION TECHNIQUES

Executive and Utility Modifications

The use of the system data block (ref. 3, section 2.4.1.8) was expanded to store the following data:

31 Current length of main storage

35 Maximum length of main storage

36 Default length of main storage

57 First level length of assigned main storage

58-61 Subsequent level lengths of assigned main storage

Two functions were added to the computer dependent subroutine MAPFNS (ref. 3, section 5.4.7) to execute the executive requests to MCORE$ and LCORE$. These were labeled GETCOR and RELCOR, respectively.

Subroutine MSGWRT was modified to skip the call to subroutine FNAME for message 3008 and to write the modified message. This change is computer independent.

An additional line was placed in subroutines XSEMii to call subroutine RELCOR on each return from a module execution.

Matrix and Module Subroutine Modifications

When a call to subroutine EXPAND is necessary, all files must be closed before returning to the beginning of the subroutine to reexecute the preface. The GINO buffers will be reassigned. Care must be taken that files opened previously be closed without rewind and reopened without rewind. This is the case of the PG file in module SSG1 (Static Solution Generator, Phase 1). This load vector file is opened in subroutine SSG1 and the load vectors written by subroutine EXTERN for the external load vectors and by subroutine EDTL for the element deformation and temperature load vectors. Either EDTL or EXTERN may require extra storage.

If a matrix subroutine is denied the use of a section of upper storage, the additional storage requested must be at least as large as the total length of the GINO buffers to prevent GINO error 1151 (buffer overlaps a previously assigned buffer). GINO will remember the address of the buffer in the reserved area and prohibit this area to be used as a buffer again.

RESULTS

Dynamic expansion has been successfully demonstrated in static and normal modes analysis from the following modules and/or subroutines. These subroutines reflect the size required by the analysis by their varying needs and hence are calls to the prime candidates for calculating the required working storage and the improved message 3008:

Module	Subroutines
READ	INVPWR
SDR2	SDR2C
	SDR2D
SSG1	EDTL
RBMG2	DECOMP
SSG3	MPYAD
XSORT	XSORT

The following subroutines have had the call to subroutine MESAGE changed but have not called for increased storage and are therefore untested.

RCOVB	TRDIA2
AMG	TRHT
PARTN	FCNTL
TRD	TRNSP
TRDIA	INVP3

GENVEC (preface for both DECOMP and CDCOMP)

All of the above subroutines should calculate the storage requirements and relay this information to the analyst on all computers.

CONCLUDING REMARKS

The improvement in user fatal message 3008 is a useful tool to the analyst and is a guide to total main storage requirements of an analysis. After this improvement to the message, a zero value of additional storage is an alert to the system programmer that the minor change in the offending subroutine is desirable.

For the sake of keeping the computer independence, it is recommended that a call to the computer dependent subroutine EXPAND be made in all cases of insufficient main storage. This subroutine would then directly make the call to subroutine MESAGE for those computers where the insufficiency is fatal.

All of the areas of storage insufficiency have not been discussed; others are anticipated as the structural models get larger and other paths through NASTRAN, particularly dynamic analysis, are explored.

Further work on the storage requirements should include earlier detection of insufficient size. Table trailers are sparsely used and could, in some cases, be used to cause an earlier demise of an analysis that has insufficient main storage.

REFERENCES

1. The NASTRAN User's Manual. NASA SP 222(03), July 1978, Section 6.2.2.

2. McCormick, C. W., and Redner, K. H.: Study of Modifications Needed for Effective Operation of NASTRAN on IBM Virtual Storage Computers. NASA CR-2527, April 1975.

3. The NASTRAN Programmer's Manual. NASA SP 223(03), July 1978.

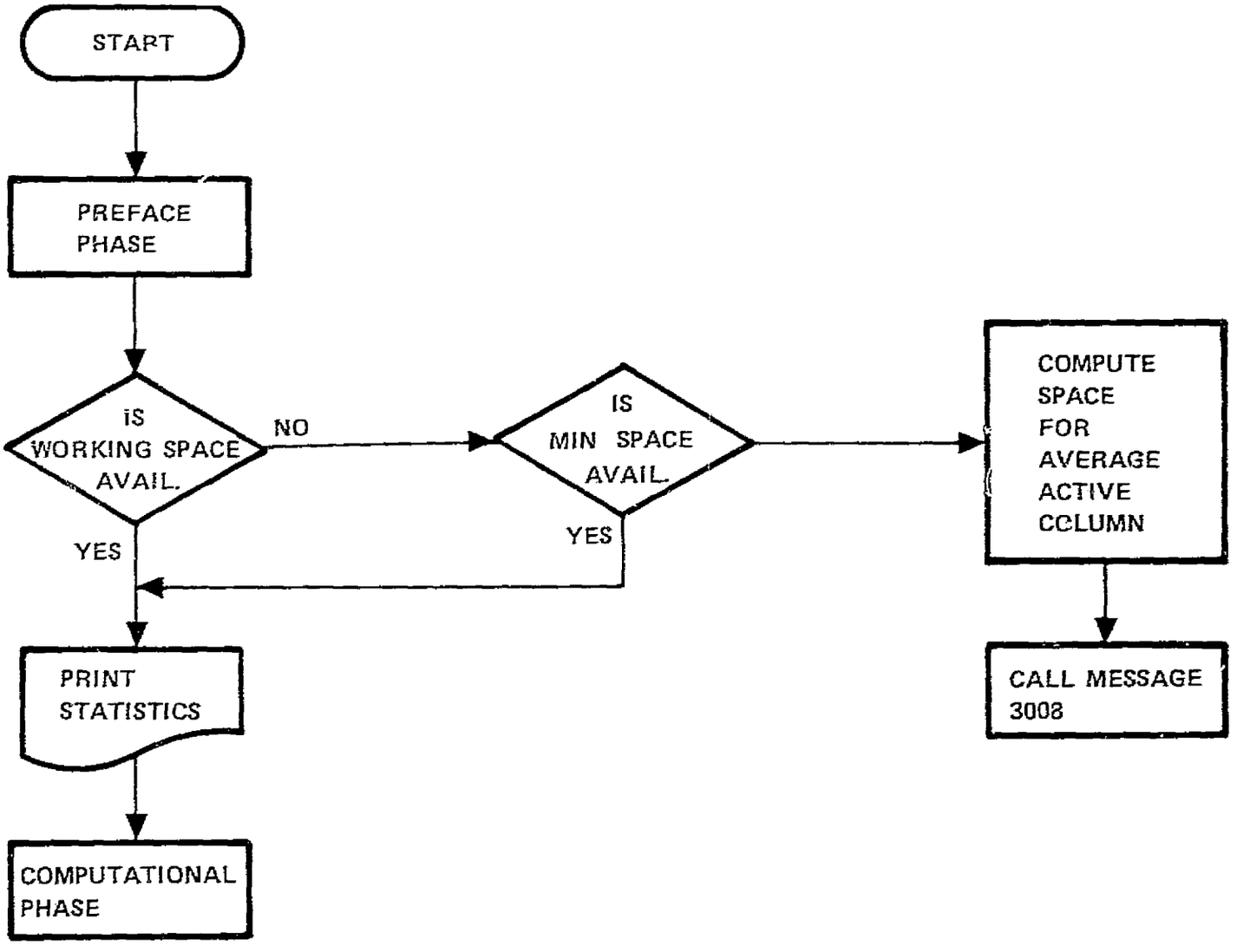


Figure 1. — Storage Calculation With Spill Optimization (SDCOMP).

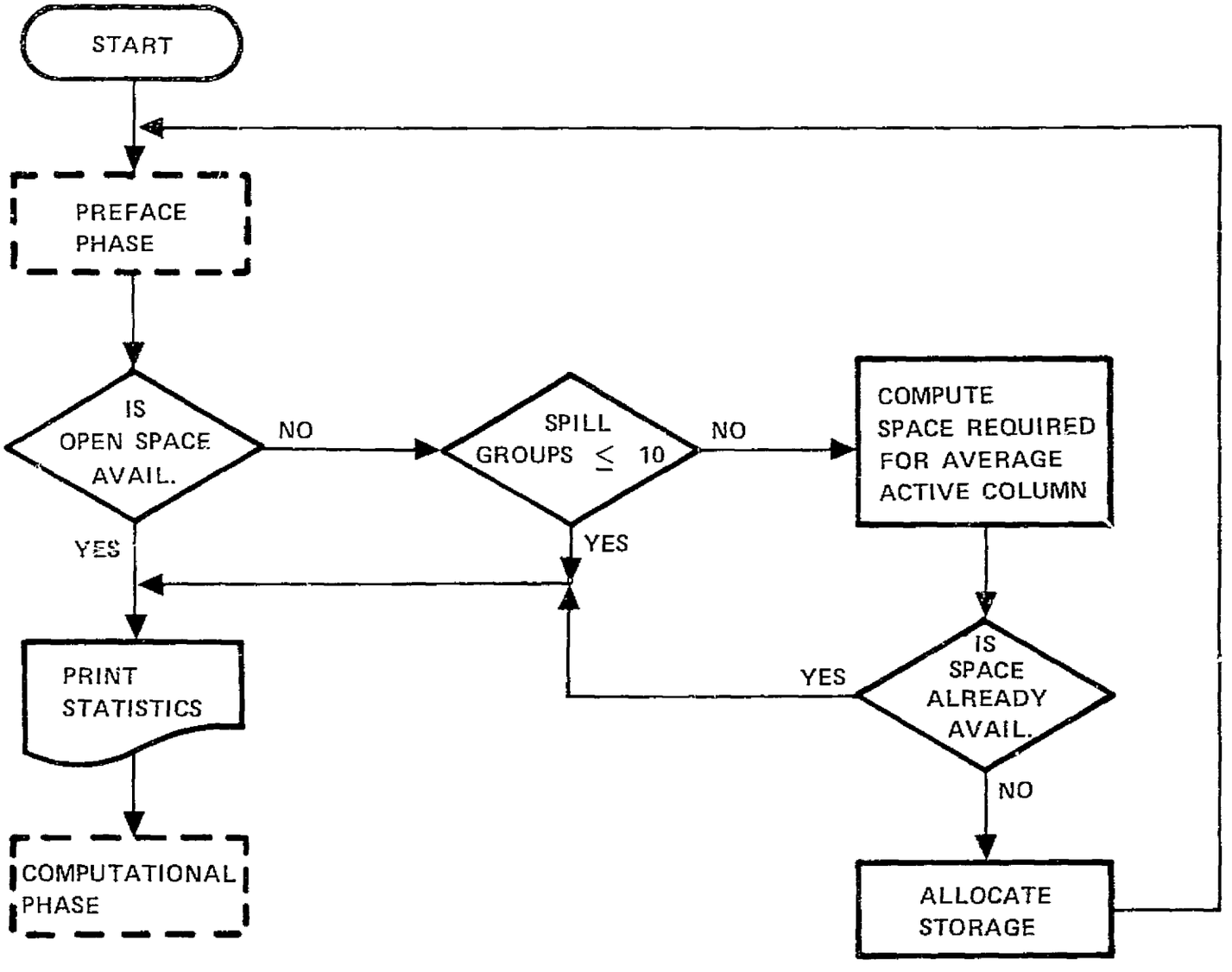
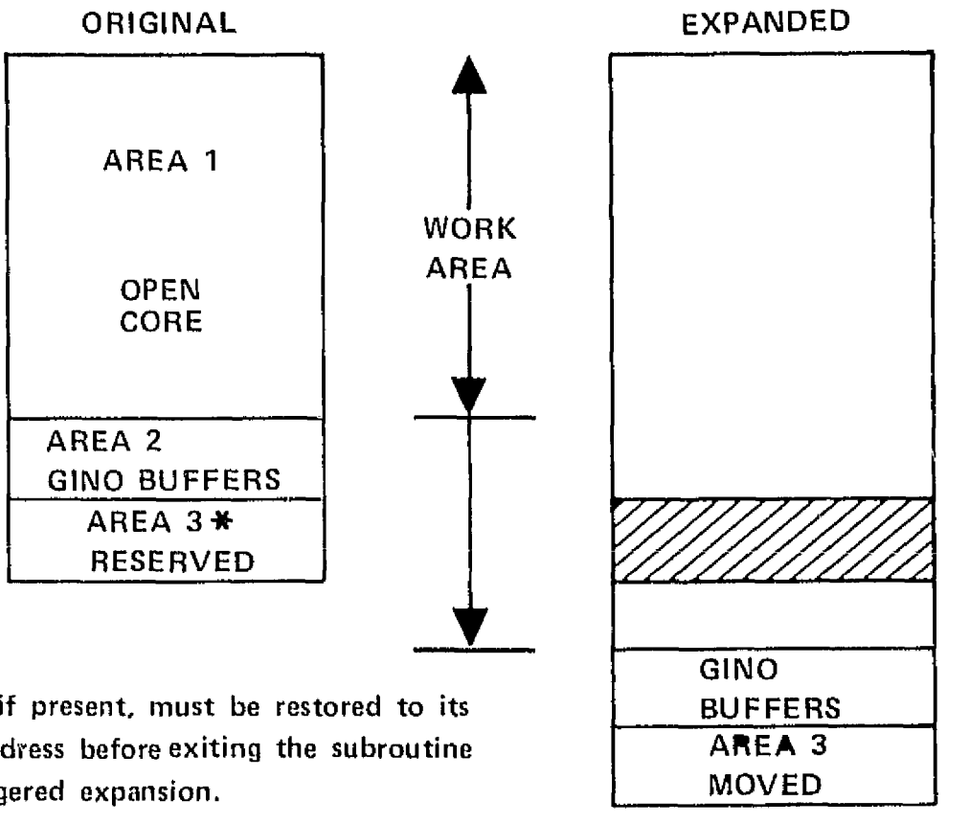


Figure 2. — Dynamic Expansion With Spill Optimia Optimization (SDCOMP).



*This area, if present, must be restored to its original address before exiting the subroutine which triggered expansion.

Figure 3. – Open Core Allocation Subroutine Any.