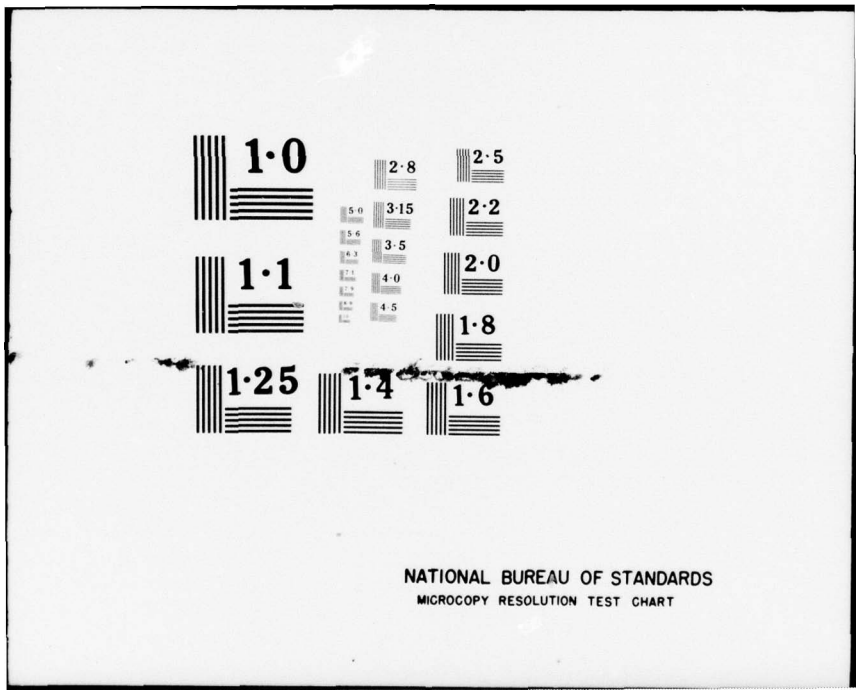
AD-A039 384 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF CIVIL ENG--ETC F/G 9/2
REPRESENTATION OF A COMPUTER-AIDED ITERATIVE DESIGN PROCESS.(U)
APR 77 M A TAMM, S J FENVES N00014-76-C-0354

UNCLASSIFIED R-77-5 NL

1 OF 1
ADA
039384

END
DATE
FILMED
6-77


1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
2·8
3·15
3·5
4·0
4·5
5·0
5·6
NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

REPRESENTATION OF A COMPUTER-AIDED ITERATIVE DESIGN PROCESS

by

Martin A. Tamm and Steven J. Fenves

A Technical Report of a Research Program
Sponsored by
THE OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
Contract N00014-76-C-0354

April, 1977

DEPARTMENT OF CIVIL ENGINEERING
CARNEGIE INSTITUTE OF TECHNOLOGY
Carnegie-Mellon University

R-77-5

REPRESENTATION OF A COMPUTER-AIDED ITERATIVE DESIGN PROCESS,

by

Martin A. Tamm and Steven J. Fenves

R-77-5

A Technical Report of a Research Program
Sponsored by
THE OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
Contract N00014-76-C-0354

April 1977

50 P.

406 386

ABSTRACT

A network representation of the iterative design process has been presented. The representation is closely based on that previously developed for non-iterative design checking. The representation uses two sets of data values, denoted as Level 1 and Level 2, in a manner similar to first-order iterative computational schemes. Certain data values, designated internally modifiable parameters, serve as a reference between successive cycles. Provisions are made for retaining certain data between cycles, so as to suppress calculations in some iterations, when it is felt that the results are insensitive to changes in certain data values.

The representation is illustrated through the application to a typical structural design problem.

Detailed data structures and transformation process are presented. In the case of data, it is shown that the subscripted nature of the data must be explicitly taken into account, whereas the non-iterative process can be represented in terms of generic, or unsubscripted, data. The recursive procedures for evaluating and modifying data have been extended to handle the two-level, subscripted data representation. Four additional processes are defined: those to ACCEPT or REJECT the current cycle of results, and those to RETAIN and RELEASE data between cycles.

ACKNOWLEDGEMENTS

This report is based on the Master's Thesis submitted by Mr. Martin A. Tamm in partial fulfillment of the requirements for the degree of Master of Science in Civil Engineering from Carnegie-Mellon University. The work was done under the supervision of Dr. Steven J. Fenves, Professor of Civil Engineering.

This study was sponsored by the Office of Naval Research under Contract N00014-76-C-0354, entitled "Computer Aids for Structural Analysis and Design."

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

1.1. Objective

The functional design of a structural system involves a number of iterative processes, each of which determines some values for parameters which define the system. Individual processes deal with the design of specific components, such as frame members, frame connections, or floor grid members, and each process consists, in general, of procedures for design, analysis, checking and selection.

In order to develop a computer based design capability, there must be a formal representation of the design process. Such a formal representation has been extensively developed for non-iterative design checking procedures (References [4],[5],[6], [7], [8], [9]). The representation is based upon a network of the data generated and used in the procedure. A network is formulated in which each datum is assigned a node and an oriented branch is drawn to the datum node from each of its direct ingredients. Associated with each derived datum is a transformation, or unit procedure, that must be executed to evaluate the datum. Basic parameters are those data which have no evaluation procedure and must be input.

Processing is formally expressed as an attempt to evaluate a given datum. In order to prevent unnecessary execution of transformations, a recursive execution procedure, called SEEK [5], is implemented, which evaluates missing direct or indirect ingredients of a datum only if they are found to be needed for evaluation of the datum sought. If a new value for a datum is input, a recursive updating procedure, called WARN [5], erases the values of all dependents, direct or indirect, of the datum.

It is the purpose of this report to present a network representation for the iterative structural design process based on that developed for non-iterative

design checking.

Individual design processes are specific not only in the aspect of the structure they design but also in the type of solution offered. Separate design processes must be considered for, say, member design using a concrete beam, a standard wide flange shape, or a plate girder. Throughout this report, examples will be drawn from the case of a gable frame structure subject to loads at the joints and distributed loads along the members, and which is to be designed elastically using standard wide flange shapes. The AISC Specification [1]criteria for the elastic design of members under combined axial and bending stress is to govern. The iterative procedure is restricted to the selection of the cross-section of the members satisfying the specification criteria. The loading, frame geometry, and member material are fixed unless changed by external input.

1.2. Organization of Report.

The basic concepts of the network representation of the cyclic design process are presented in Chapter 2. The data structures used to represent the data values in the design process and the transformations operating on the data are presented in Chapter 3. The processes dealing with the control of the design cycle and the evaluation and modification of the data are described in Chapter 4. A summary of the work and conclusions reached from the study are presented in Chapter 5.

CHAPTER 2. NETWORK REPRESENTATION OF CYCLIC DESIGN

In this chapter, the basic concepts of the network representation of cyclic, or iterative, design are presented. Succeeding chapters further elaborate on the data structures and procedures needed to implement the cyclic design process.

2.1. The Cyclic Network

The basic difference between a non-iterative and an iterative process is that for a single set of starting inputs the non-iterative process produces one value per datum (or element of a subscripted datum), whereas an iterative process produces repeated cycles of values for each datum. The generation of repeated cycles of values, each of which acts as the input for generating the next cycle, suggests a representation based on the acyclic network of data values instead of the cyclic network of data variables. Example networks of data variables and data values are shown in Figures 1.a and 1.b, respectively, for a hypothetical three-datum, two-cyclic-path network. In this way each value generated is associated not only with a datum but also with the cycle on which it is generated.

To obtain the representation of a network of values from a network of variables, it is necessary to define a set of data which are the first to be evaluated on a given cycle of calculation. These data, called the internally modifiable parameters, are evaluated from the previous cycle of calculation and serve as input parameters for the current cycle of calculation. For the set to be able to act as a reference point between successive cycles, it must satisfy the following two requirements:

1.) There must be one internally modifiable parameter on each separate cyclic path through the network of data variables; and

2.) There must be no more than one internally modifiable parameter on each path through the network of variables.

If the first requirement is not met, there will be an ambiguity in determining on which cycle a value was generated, and if the second requirement is not met, there will apparently be more cycles of calculation than values generated for each datum. In Figure 1.b, datum A is the first datum evaluated on each cycle, and thus, it is the internally modifiable parameter. Datum B would be acceptable as an alternate internally modifiable parameter, because it occurs on both cyclic paths in Figure 1.a. Datum C, by itself, could not be used as an internally modifiable parameter, because it occurs only on one of the two cyclic paths in the network, and it could not be used with datum A or B, because then there would be more than one internally modifiable parameter on the other path.

Generally, the internally modifiable parameters in an iterative procedure are the primary data being sought. All other data generated are used to determine if the values of the data being sought are converging or have converged to a desirable solution and/or to act as inputs for the next generation. In the Newton-Raphson method [3] for solving the equation, $F(x)=0$, the cyclic network consists of the data, x, $F(x)$, and $F'(x)$. The datum, x, is both the datum sought and the first datum to be evaluated on an iteration; thus, it is the internally modifiable parameter.

For most design processes, the network of values can be reduced to a two-cycle network. The first of these cycles is the cycle of current accepted values (designated Level 1), and the second, subsequently generated, cycle is the cycle of potential improved values (designated Level 2). All cycles generated previous to the current Level 1 values are outdated and need no longer be considered. Cycles of values subsequent to the current Level 2 values are not generated (see Fig. 1.b) until it has been determined that the Level 2 values are actually an improvement over the Level 1 values. The Newton-Raphson method, mentioned

previously, written in recursive form as

$$x_{i+1} = x_i - F(x_i)/F'(x_i), \qquad (1)$$

shows that x_{i+1} at Level 2 is generated from the Level 1 values of x_i, $F(x_i)$ and $F'(x_i)$. In programming such a procedure, the temporal subscript would not be used, and only the "current" level 1 and 2 values would be retained [3]. Higher-order processes, e.g. Richardson extrapolation, using several sets of previous values, are possible, but do not appear to be necessary for the design process.

It should be noted that the concept of "current" levels used in this report differs from the "permanence level" used in Reference [12] to denote information at various stages of design, checking, and construction.

Generally, an iterative process has inputs which remain unchanged from cycle to cycle. All data which are direct or indirect dependents of unchanging parameters only are acyclic ingredients; thus they are not re-evaluated to obtain improved values. Acyclic ingredient data only exist as accepted, or Level 1, values. These Level 1 values serve as ingredients to both the Level 1 and Level 2 values of cyclically dependent data. Figure 2 shows the general ingredience-dependence relationships between data values in the two-cycle network of values. Returning once more to the Newton-Raphson method, in the so-called Modified Newton-Raphson method:

$$x_{i+1} = x_i - F(x_i)/F'(x_o) \qquad (2)$$

the variable $F'(x_o)$ would be evaluated only once, and would thus be an acyclic ingredient of x.

2.2. Transformations

The design process is made up of unit procedures, called transformations, which are used to evaluate data from their direct ingredients. Transformations

can be of many types, including logical decisions, formulas, table searches, unquantifiable designer interaction, or any combination of these ([5], p. 1100).

For the purpose of processing, transformations are distinguished by their execution characteristics. In the organization of data into networks for evaluating criteria of the AISC Specification, two kinds of transformations were used: functions and decision tables ([9], p. 13). The decision table is characterized by the presence of dynamic ingredients: some boolean ingredients in the condition stub may not be needed to check some of the rules in the table, and the lists of ingredients differ for the various functions in the action stub ([6], p.9). Thus, the actual list of ingredients needed for evaluation must be determined as execution progresses. In contrast, the list of ingredients that must be present for executing a function transformation remains unchanged. All transformations which have no dynamic ingredients are termed routines. This includes explicit functions, searches through tabular data, and designer judgments which are based on a fixed set of ingredients.

Decision table transformations are also necessary in the iterative design process, not only for handling the extensive logic found in design checking but also for the decision logic found in other procedures. For example, Table 1 shows the logic involved in the example design selection procedure, to be described later.

The first action in Table 1 is a routine in which the designer uses his judgment in selecting a new shape, basing his judgment on the properties of the previous shape and the critical constraint values. The remaining actions in Table 1 are routines similar to those used in non-iterative design checking.

2.3. Multi-Valued and Subscripted Data

As presented in References [8] and [9], a transformation always produces a single datum. Two extensions of this restriction are needed for cyclic design.

First, in some cases two or more data items are so closely related that they are always evaluated together by a single transformation. An example is the datum 'standard wide flange shape' in Table 1. This datum actually consists of three individual data items, namely 'section designation', 'nominal depth', and 'section weight', evaluated together by the selection routines. Such data always occurring in groups will be referred to as multi-valued data.

Second, in dealing with a complete design, as opposed to a design check at a specific point, data have to be distinguished by their location, or subscript. General procedures for dealing with subscripted data are presented in References [8]and [11]. An implementation for the specific design example considered in this report is presented in Section 3.3.

2.4. Retained Data

Normally, a cyclically dependent datum is re-evaluated on each cycle of calculation. However, it is sometimes desirable to suppress the cyclic recalculation of certain data and to use the existing values of these data as ingredients over a number of iterations. This is done to reduce the amount of calculation, when it is expected that re-evaluation will produce little or no change in the given data, or when the cycle of calculation is still so approximate that using some values from previous cycles will produce no significant error.

For example, in structural design repeated selections of design variables are customarily performed without a re-analysis of the forces in the structure ([10], p.649). This is due to the fact that relative moments of inertia, upon which the distribution of member forces throughout the structure depends, are not expected to change drastically with each new selection of members. Thus, a re-analysis can be delayed until the final selection satisfying the design constraints is to be checked for consistency. Similarly, the loads on the structure, which are dependent on the repeatedly changing contribution of member weights, can be

left unchanged, if an adequate member weight contribution is assumed on the first cycle of calculation.

Retainable data are classified into sets which are intended to be retained together in order to suppress given sequences of calculations. Retained sets can be considered temporarily as acyclic ingredient data. As with true acyclic data, in every retained set the Level 1 values are held for more than one cycle of calculation and these values are used as inputs for generating both Level 1 and Level 2 values of dependent data as shown in Figure 3.

Unlike the Level 2 of true acyclic ingredient data, the Level 2 of retained cyclic data has a useful meaning. The Level 2 value of a retained datum can be generated from Level 1 values of ingredients not in the retained set, as shown in Figure 3. Using this representation of the Level 2, a comparison of Level 1 and Level 2 values can be made to observe how inaccurate the retained (Level 1) values have become. Of course, if the full sequence of suppressed calculations must be made to generate the Level 2 values for comparison, there is no sense in keeping the retained values. However, an indication of the inaccuracy of the full set of retained values can be obtained by only re-evaluating a few of the data.

If it is decided to release a set from retainment, the Level 1 values are discarded, and whatever Level 2 values have been generated are re-assigned as Level 1 values. It is also noted that Level 2 values of retained data are discarded whenever the Level 1 values of unretained data are discarded.

2.5. Application to Structural Design

The makeup of a single cycle in an iterative structural design process can be described through the categorizable groups of data generated and the procedures that generate them [11].

An appropriate beginning for a cycle of design is the group of data designated as the structural parameters. These are the data that define the

current design solution. All subsequent data generated on the cycle of calculation are used to determine the behavior and check the current design solution against the applicable design criteria. The structural parameters include design variables whose final values are found by iteration, and data which remain unchanged as input parameters. The design variables which define the cross-section of a wide flange shape are 'nominal depth' and 'section weight.' In the gable frame example structural parameters which remain unchanged are data such as 'grade of steel' and 'member length.' In terms of the terminology introduced earlier, the design variables 'nominal depth' and 'section weight' constitute the internally modifiable parameters, if these variables are the first ones evaluated in each design cycle (see Section 2.1)*. Unchanged parameters such as 'grade of steel' and 'member length' are the acyclic ingredient data defined previously.

The first step of calculation is structural processing, which determines values for structural attributes from structural parameters or other structural attributes. Structural attributes are properties of the structure which are needed in later steps of analysis and design. For example, the attributes 'cross-section area', 'moment of inertia', and 'radius of gyration', among others, are obtained by a table search from the parameters 'nominal depth' and 'section weight'. From 'moment of inertia' and ' member length' attributes are determined 'member stiffness' and,in turn, 'structural stiffness' attributes. ([13], p.397).

---

* It is to be noted that the first cycle need not start with the internally modifiable parameters given (for example, the designer may start with assumed relative moments of inertia). Also, upon completion of the process, when all constraints have been checked, the internally modifiable parameters need to be re-evaluated once more to produce the final design. For a general discussion of such initialization and termination problems of iterative processes, see Knuth, D.E., "Structured Programming with go to Statements", ACM Computing Surveys, Vol. 6, No. 4 (December 1974), pp. 261-302.

Next in the cycle of calculation is environmental processing, which determines the environmental attributes that express the demand or loading imposed by the environment on the structure. Inputs for this processing are the structural attributes and parameters defining the environment. For instance, the environmental parameter, 'region of the United States', and the structural attribute, 'slope of roof', may be used in a table lookup routine to get distributed snow loadings ([10], p.10).

Structural analysis then takes the structural and environmental attributes to generate the structural response. For example, equivalent joint loads (environmental attributes) and structural stiffnesses (structural attributes) are used in computing joint displacements, and joint displacements and member stiffnesses are used in turn to get member end forces ([13], p. 397). Then, the member end forces and member distributed forces are used to obtain the magnitude of the critical bending and axial forces along each member.

Constraint processing, or design checking, produces constraint values, which indicate if and to what degree the design satisfies the applicable design criteria. Constraint values include the results of criterion interaction equations, such as AISC equations 1.6-1a, 1.6-1b and 1.6-2 ([2], p. 5-22) boolean data which indicate whether a given criterion is satisfied ([8], p.5) and data that indicates the controlling mode of failure, such as yielding or stability. In an intermediate step constraint processing generates inputs for criterion constraint equations. These inputs reflect either an action toward failure or a resistance to failure ([11], p. 483). For the AISC combined stress criterion, actions toward failure include axial and bending stresses 'f_a' and 'f_b', and resistances to failure include the allowable axial and bending stresses, 'F_a' and 'F_b'. The "action" and "resistance" data are, in turn, derived from a combination of structural attributes and structural response.

The final step in a cycle of design is the selection of new values for the design variables for starting the next cycle of design. The selection procedure attempts to choose a design of reduced weight or cost that still satisfies the design criterion. A selection procedure provided by the AISC Manual [1] uses modified forms of the constraint equations to generate values for guiding the choice of the 'section weight' parameter from column selection tables. The iterative use of this procedure is intended for an unchanged value of nominal depth ([10], p. 649). To select sections of different nominal depth an optional procedure can be provided, by which a designer uses judgment to select a member of less weight which is expected to be satisfactory, as shown in Table 1.

The relative dependences among groups of data generated in the structural design cycle is shown in Figure 4, with an arrow pointing to each dependent group from its ingredient groups [11]. This diagram reflects the pattern of connectivity of the detailed network of data. In order to further illustrate this relationship, the information network of Figure 4 is redrawn in Figure 5 to correspond to the general format introduced in Figure 1.

As can be seen from Figure 5, the data making up the set of design variables will reference every cyclic path in the detailed ingredience-dependence network, and thus are an appropriate set of internally modifiable parameters for the iterative structural design process.

In terms of the terminology introduced earlier, all of the processes up to the selection of new design variables occur at Level 1. The selection process attempts to generate new design variables at Level 2.

An improvement exists if the Level 2 value of 'section weight' is less than the Level 1 value. At this point the Level 2 values are reassigned as Level 1 values, and the previous Level 1 values are discarded. Then, a new cycle of Level 2 values can be generated.

As indicated in Section 2.4, it is not necessary to execute all steps of computation on every cycle. Re-analysis (evaluation of structural response) and re-evaluation of loads (action data) can be bypassed by retaining the corresponding data from an earlier cycle.

CHAPTER 3. DATA STRUCTURES

This chapter describes the data structures needed to represent both the data values in the cyclic design process and the transformations operating on the data.

3.1. Representation of Subscripted Data

For the proper operation of an iterative design process, a storage scheme is needed which provides easy access by processors operating on the network of data. This can be done by means of a global data array ([6], p. 467). This array provides space for the multiple values of subscripted data, as well as for the two status levels (Level 1 and Level 2) on which values can exist. For example, the data, 'nominal depth', 'section weight', and 'moment of inertia', are subscripted by member, and joint loads are subscripted first by joint number and second, by the component of load. These space requirements can be met by a two-dimensional array, with a typical value accessible as DATA (ADDRESS, LEVEL). The LEVEL is specified directly as 1 or 2. However, to specify the ADDRESS, the following data structure must be implemented.

The total number of subscripts of any kind is NTS. A vector RS(j), j=1,..., NTS, gives the (fixed) Range for each Subscript, j (e.g. if j=1 corresponds to the subscript 'MEMBER', RS(1)=4). For external identification, SUBNAME(j) contains an alphameric name (e.g., SUBNAME(1) = 'MEMBER').

Throughout the processing, the vector SUB(j), j=1,..., NTS, stores the current values of the subscripts (e.g., if SUB(1) = 3, we are currently processing the 3rd member).

Each datum, referenced by its numerical designation, i, has associated with it the following:

1) an alphameric name, NAME (i) (e.g. NAME(1) = 'DEPTH');

2) the Number of Subscripts applicable to the datum, NS(i);

3) a list DS(i,n), i=1,..., NS(i) of the applicable Datum Subscripts (e.g. DS(1,1)=1 indicates that the first subscript (n=1) of variable 'Depth' (i=1) is 'Member'(j=1)).

4) the location of the first element of the datum, FIRST(i)

Then, for a specified datum, i, and a set of specified subscripts held in SUB(j), the address of the value of interest is

$$\text{ADDRESS} = \text{FIRST}(i) + [\text{SUB}(\text{DS}(i,1)) - 1] + \sum_{n=2}^{\text{NS}(i)} [\text{SUB}(\text{DS}(i,n)) - 1][\prod_{m=1}^{n-1} \text{RS}(\text{DS}(i,m))]$$

The allocation of storage space in the DATA array (i.e. assignment of values for FIRST(i)) is dependent on the range of subscripts in the particular problem being processed.

It is intended that this storage scheme be used during iterative processing only. Once the final desired values have been generated, these values can be transferred into the overall design project data base (such as those described in [7] and [12]). Then, the space in the arrays may be re-allocated for performing the next iterative design process. For long-term storage, it would be wasteful to provide the double storage space (Level 1 and Level 2) that is needed during iterative processing.

3.2. Representation of Cyclic Data

A number of additional data structures are needed to represent the status of the various types of data introduced in Chapter 2.

First, in order to represent the current status of any data value, an array of presence flags, which indicate which values exist in storage ([6], p. 469), is needed. For each ADDRESS in DATA, PR(ADDRESS)=0, 1, 2, or 3, depending on

whether a value is undefined or exists on Level 1, Level 2 or both levels, respectively.

Second, in order for a processor to identify which data are Internally Modifiable Parameters, an indicator, IMP, is defined, so that IMP(i)=1 for each datum, i, which is an internally modifiable parameter, and IMP(i)=0 for each datum, i, which is not. Likewise, in order to identify those data which are in the set of acyclic ingredient data, an indicator, SET, is defined, so that SET(i)=1 for each datum, i, which is acyclically dependent, and SET(i)=0 otherwise.

Third, to implement the ability to retain sets of data, the elements of each set, k, can be listed in an array, ELEMENT(k,n), and each set can be identified by an alphameric name in SETNAME(k). Descriptive names for the sets discussed above would be 'RESPONSE' and 'LOADING'.

While a cyclically dependent datum, i, is being retained, SET(i)=0 is changed to SET(i)=k as an indication. The set, k=1, is the set of acyclic ingredient data (SETNAME(1)='ACYCLIC'), which can be viewed as a set of permanently retained data.

Finally, in order to exercise control over the selection of design variables (see Section 4.1), an indicator of active subscripts, SACT(j) is defined. If SACT(j) = 1, the design cycle can be completed for the restricted subscript, j. Unrestricted subscripts are designated by SACT(j)=0.

3.3. Representation of Transformations

The overall representation of transformations follows closely that given in Reference [4]. For a datum, i, the designated decision table transformation is indicated by a non-zero entry in the array, T(i), and a designated transformation routine is indicated by a non-zero entry in the array, S(i). For both types of transformations, the list of ingredient data items is stored in an array INGRED (s,n), where s is either a table or routine designation.

The detailed representation of decision table transformations follows that given in Reference [6]. The Condition Stub is represented as CS(c,T(i))=k, i.e. the cth condition of table T(i) can be obtained as the value of some datum, k. The transformation routine, s, that is the single action for rule r in the Action Stub is indicated by AS(r,T(i))=s. Finally, the Condition Entry is stored in an array CE(c,r,T(i)), with entries representing 'yes', 'no' or 'immaterial'.

CHAPTER 4. PROCESSES

This chapter details the various processes involved in the control of the design cycle, the evaluation and modification of design data, and the treatment of retained data.

4.1. Control of Design Cycle

Revising the status of the two active cycles of values entails either rejecting the most recently calculated cycle of values (Level 2) or accepting them as the new currently accepted values (Level 1). The decision of which of these two actions to take is an external process, using either the designer's judgment or some program in the design system (e.g. the designer's acceptance of all Level 2 member selections which are lighter in weight than previous selections and which still satisfy the design criteria, and rejection of all others). The actual change of status is made by executing either the ACCEPT or REJECT procedure. These procedures are represented in Tables 2 and 3, respectively.

Before the change in status of any data can take place, the conditions of Tables 2 and 3 must be properly met. No change in status is made if all the internally modifiable parameters (the design variables) are not present at Level 2 (Rule 1 of both Tables). This is simply an indication that there are no Level 2 values to ACCEPT or REJECT. Knowing that the Level 2 internally modifiable parameters are present is important in ACCEPTing, to prevent the existing Level 1 values from being erased without having any new values to replace them or any starting inputs to generate replacements.

In an iterative design process in which the data are subscripted and in which the selection procedure generates at one time new values for the design variables corresponding to only one specified value of a given subscript (such as the procedure of Table 1, which selects at one time the 'nominal depth' and

'section weight' corresponding to one value of the 'MEMBER' subscript), repeated cycles of calculations can be performed for generating only the data values that correspond to the specified value of the subscript (e.g. for the specified member). ACCEPT and REJECT can be executed for restricted subscript values by specifying the subscript restrictions in SACT(j).

To generate data values corresponding to a specified value of a given subscript without requiring generation of data values corresponding to all values of the given subscript, all data not subscripted by that given subscript must have been previously retained. Data that do not have the given subscript and which are dependent on data that do have this subscript will be dependent on ingredient data values over the full range of values of the given subscript ([11], pp. 487-489). For instance, if the given subscript is the 'MEMBER' subscript, data such as structural stiffness, which are not subscripted by 'MEMBER', will be dependent on the member stiffnesses over the full range of subscript values for 'MEMBER'. Because it is desired to iteratively generate values for only those data elements corresponding to the single specified value of 'MEMBER', all dependent data such as structural stiffness must have been retained. In the example discussed previously, retaining the sets, 'RESPONSE' and 'LOADING', will retain all data not subscripted by 'MEMBER' (as well as some that are).

If the required sets of data have not been retained, the ACCEPT and REJECT procedures are exited without altering the status of any data (Rule 2 of both Tables). Either the required sets of data must be retained, or the subscript restriction removed, before again attempting to change the status of the existing cycles of values.

In the actual acceptance of the Level 2 cycle of values as Level 1, the change in status of a value is simply accomplished by the operation, DATA(ADDRESS, 1)=DATA(ADDRESS, 2), and by changing PRESENCE(ADDRESS)=2 or 3 to PRESENCE(ADDRESS)=1.

Any elements of an unretained datum for which only a Level 1 value exists are revised by changing PRESENCE(ADDRESS)=1 to PRESENCE(ADDRESS)=0. PRESENCE(ADDRESS)=3 is changed to PRESENCE(ADDRESS)=1 for elements of retained data.

In the actual rejection of the Level 2 cycle of values, the revision affects only the Level 2 values and is accomplished by setting PRESENCE(ADDRESS)=1 for all unretained values.

4.2. Data Evaluation

It has been shown in a previous paper that evaluating data in an ingredience-dependence network is most efficiently done by means of a recursive execution procedure ([5], pp. 1103-1105). The essence of the efficiency of recursive execution is that an ingredient is not evaluated unless it is missing and found to be necessary to complete evaluation of the datum being sought. Flexibility in input of data other than the basic input parameters is also attained, because the recursive procedure searches ingredient by ingredient along a path in the network until a value that is present is found, and then a sequence of transformation executions are performed leading back to the original datum sought ([5], p. 1104).

Recursive execution can occur over the two active cycles of values. A pair of recursive procedures, SEEK1 and SEEK2, are used to seek the Level 1 and Level 2 values, respectively. The decision logic of these procedures is shown in Tables 4 and 5, respectively. The original recursive SEEKing procedure, for operating on an acyclic network, is shown on p. 1104 of Ref. [5] .

The operation of SEEK1 is as follows. If the Level 1 value is already present, the procedure is simply exited (Rule 1, Table 4). If the sought datum is an internally modifiable parameter, the Level 1 value must be present, or else an error condition exists (Rule 2). This is because the Level 1 internally modifiable parameters act as input parameters for the two-cycle network of values. Error also results from encountering a missing input parameter, which is identified as a datum with no designated transformation for evaluation

(Rule 4). If the datum is to be evaluated by a decision table transformation (Rule 3), processing is completed by procedures which handle the decision logic of the table transformation (which will be discussed further). To complete the determination of a datum available by a routine, the ingredients associated with the routine must be present at Level 1. If not, SEEK1 must be used repeatedly to obtain the missing ingredients (Rule 5, repeatedly). When all ingredients are present at Level 1, the routine is executed (Rule 6).

Aside from the attempt to obtain Level 2 values instead of Level 1 values, the logical operation of SEEK2 differs from that of SEEK1 in the following considerations. If a datum is acyclic, a Level 2 value is not to be sought (Rule 2, Table 5). If a datum is an internally modifiable parameter, all ingredients needed for Level 2 evaluation (Rule 6) are sought on Level 1 (Rule 5). For other data, ingredients needed for Level 2 evaluation (Rule 9) are sought on Level 2 (Rule 7) or on Level 1 (Rule 8), depending on whether the ingredient is or is not of the same set of data as the dependent datum. For an unretained datum, all acyclic and retained ingredients are of a different set, and thus, sought on Level 1. For a retained datum, ingredients of different sets are unretained data, acyclic data, and data of other retained sets.

If the evaluating transformation of a datum is a decision table, the SEEK1 and SEEK2 procedures call on the SEEK1ACTION and SEEK2ACTION procedures, represented in Tables 6 and 7, to isolate the governing rule of the transformation table and to evaluate the datum using the transformation routine associated with the rule, respectively. In the operation of the SEEKACTION procedures, if the governing rule of the transformation table has not been found, the unchecked rules are checked (Rule 2, repeatedly, of Tables 6 and 7) until a governing rule is found or it is determined that a governing rule does not exist. Lack of a governing rule is an error condition (Rule 1 of both Tables). Once a governing rule is found, each SEEKACTION procedure proceeds with logic identical to that

found in the respective SEEK procedure for attempting to execute a routine. (Rules 3 and 4 in SEEK1ACTION correspond to Rules 5 and 6 in SEEK1; and Rules 3 through 7 in SEEK2ACTION correspond to Rules 5 through 9 in SEEK2.)

To actually check if a rule governs, SEEK1ACTION and SEEK2ACTION call on the respective procedures, CHECK1RULE and CHECK2RULE, shown in Tables 8 and 9. Rule checking consists of matching the material conditions. A material condition is one for which the Condition Entry value is 'Yes' or 'No' but not 'immaterial' for the given rule, r. In matching, the boolean datum in the Stub has its value compared with the corresponding value of the Condition Entry. If all material conditions of the rule are successfully matched (i.e. Condition Entry and Condition Stub values are equal, for each condition), then the rule is the governing one (Rule 2 of Tables 8 and 9). Otherwise, the rule does not govern (Rule 1 of both Tables). If there are still material conditions which have not been matched, then the first still unmatched condition is processed. Because the boolean data in the Condition Stub are ingredients of the sought datum, the value used in matching must be of the appropriate level (Rule 3, Table 8; Rules 3, 5, 7, Table 9). A boolean datum with the needed value missing is SEEKed on the required level (Rule 4, Table 8; Rules 4, 6, 8, Table 9).

Whenever a SEEK, SEEKACTION, or CHECKRULE procedure is suspended in order to SEEK an ingredient datum, essential information about the dependent datum and its processing is stored in a stack ([4], p. 471). Included in the stored information are the datum, i; the LEVEL on which the datum is being sought; the current subscript values in SUB(j); boolean values indicating whether the suspended procedure is SEEK, SEEKACTION, or CHECKRULE; and, if applicable, the rule and condition of the transformation being processed. Whenever an exit from a SEEK procedure is achieved, the stack is checked to see if there is any suspended processing to complete; and, if so, the most recently suspended procedure is re-initiated. When an exit from CHECKRULE is achieved, the SEEK-

ACTION procedure is re-initiated for the same datum. An exit from SEEKACTION causes an immediate exit from the SEEK procedure for the same datum.

4.3. Initialization

The recursive execution procedure allows the first cycle of calculation to be started with inputs other than values for the internally modifiable parameters. Any appropriate combination of starting inputs can be made at Level 1 to generate the first values of the internally modifiable parameters at Level 2. (Level 1 internally modifiable parameters exist only as inputs and not as generated data.)

In the example, to direct the first selection of values for the design variables, 'nominal depth' and 'section weight', a number of sets of inputs may be entered. For instance, to reflect the desire to use the modified equation 1.6-2 in the first approximate selection, the following inputs can be made: 'Make selection based on designer judgment'=No, '($f_a/F_a \leq .15$)'=Yes (See Table 1). A value for 'nominal depth' must be input to determine the appropriate AISC Manual column selection table to use [1]. If use of the modified equation 1.6-2. is desired, '(F_a/F_b)=1' and a reasonable value for the bending factor, 'B', is input ([10], p. 649). To obtain values of critical member bending moment and axial force to be used in modified equation 1.6-2, values for relative moments of inertia are input ([10], p. 854). These indicated inputs, including structural attributes, structural parameters, constraint values, and resistances illustrate the wide variety of inputs that can be used in the starting cycle of calculation to generate the first full set of design variables. (The example inputs are not intended to show a complete set of starting inputs, having left out such basic input parameters as 'grade of steel', 'member lengths', 'external loads', et cet. which, by definition, must all be input).

4.4. Data Input

External data are entered not only for the purpose of starting the first cycle of calculation, but also on any cycle of calculation for the purpose of redirecting the succeeding cycles, when it appears that the design variables are not converging toward a desirable solution. For instance, if overly large member cross-sections are being required to meet the design constraints, it is reasonable to input a higher grade of steel, so that future cycles of calculation will produce better selections. At any point in the iterative process external inputs are entered only as Level 1 values.

There are situations in which the INPUT procedure, shown in Table 10, cannot enter a value for a datum. The first of these occurs when the datum for which the input is to be entered has been retained (Rule 1, Table 10). This is simply because the existing value of a retained datum is considered to be deliberately held until the datum is released.

The second situation occurs when there are existing Level 2 values for unretained data (Rule 2). This is because all Level 2 values are directly or indirectly dependent on all Level 1 values, so that the modification of any Level 1 values would invalidate existing Level 2 values. The desire to keep or discard the Level 2 values should first be confirmed by ACCEPTing or REJECTing them. The presence or absence of any Level 2 values can be established by checking the internally modifiable parameters, since these are the first Level 2 values to be generated.

4.5. Erasing Data

Upon successful input of a new value for a datum, all Level 1 dependents of the modified datum, as well as Level 2 values of retained dependents become invalid and must be erased (Rule 3, Table 10). For example, in a modification

of 'grade of steel', the invalidated data would include the yield stress, 'F_y'; allowable axial compression stress, 'F_a'; allowable bending stress, 'F_b'; and the constraint values from equations 1.6-1a, 1.6-1b, and 1.6-2.

The selective erasing of just those data that are direct or indirect dependents of the modified datum is done by the recursive procedure, WARN represented in Table 11. (The recursive WARNing procedure for operating on an acyclic network is shown on p. 1105 of Reference [5]). To guarantee that all direct and indirect dependents of a modified datum are erased, a given dependent datum is not erased until all its direct dependents are erased.

If a given dependent to be erased has, itself, retained dependents with Level 2 values (Rule 1, Table 11) or unretained dependents with Level 1 values (Rule 2), then these dependents of the dependent must be WARNed before the dependent, itself, is erased (Rule 3). All dependents which are internally modifiable parameters can be ignored, because the Level 2 of unretained data is eliminated before the WARNing of dependents begins. If a given dependent to be erased is a retained datum, its own dependents of the same retained set must have their Level 2 values erased (Rule 4) before its own Level 2 value is erased (Rule 5). The Level 2 value of a retained datum has as its only dependents other Level 2 values of the retained set.

Whenever the WARNing of a dependent is suspended to WARN one of its dependents, the datum designation, i, and the specified subscript values in SUB(j) are put into a stack. Upon exiting the WARN procedure, the datum for which WARN was most recently suspended is re-initiated.

4.6. Retaining and Releasing Cyclically Dependent Data

The retaining of data is handled by the procedure, RETAIN shown in Table 12, which processes retainable data according to the sets that are to be retained together. In the operation of RETAIN there is a restriction against actually retaining data, while unretained Level 2 values exist (Rule 1, Table 12). The

purpose of this is to eliminate, in a simple manner, the need to search for certain Level 2 values that would be invalidated. These values include the existing Level 2 values of the newly retained data, which should now be dependents of Level 1 ingredients of unretained data, and the existing Level 2 of dependents of the newly retained data, which should now be dependents of the Level 1 of the retained data. Once the existing Level 2 values are eliminated by ACCEPTing or REJECTing, data can be retained (Rule 2).

The procedure for retaining data is complemented by the procedure for releasing data from retainment, called RELEASE shown in Table 13. The RELEASE procedure can be used for any set that has been RETAINed except the set of acyclically dependent data, which are intended to be permanently retained (Rule 1, Table 13). The actual releasing of data from retainment requires the erasure of the data's Level 1 values, which have been held over a number of cycles of calculation and cannot be valid as unretained values of cyclically dependent data. Any existing Level 2 values of the data while retained become the Level 1 values of the data upon release, because the Level 2 of a retained datum is generated on the same cycle as the Level 1 of an unretained datum.

The re-emergence of the dependents of the formerly retained data on current ingredients has the same effect as inputing new data. In each case the invalidation of an existing Level 1 value causes the invalidation of all existing Level 2 values, which should be ACCEPTed or REJECTed first (Rule 2, Table 13), and in each case all the Level 1 (and retained Level 2) dependents of an invalidated value should be WARNed.

CHAPTER 5. SUMMARY AND CONCLUSIONS

5.1. Summary

A network representation of the iterative design process has been presented. The representation is closely based on that previously developed for non-iterative design checking. The representation uses two sets of data values, denoted as Level 1 and Level 2, in a manner similar to first-order iterative computational schemes. Certain data values, designated internally modifiable parameters, serve as a reference between successive cycles. Provisions are made for retaining certain data between cycles, so as to suppress calculations in some iterations, when it is felt that the results are insensitive to changes in certain data values.

The representation is illustrated through the application to a typical structural design problem.

Detailed data structures and transformation processes are presented. In the case of data, it is shown that the subscripted nature of the data must be explicitly taken into account, whereas the non-iterative process can be represented in terms of generic, or unsubscripted, data. The recursive procedures for evaluating and modifying data have been extended to handle the two-level, subscripted data representation. Four additional processes are defined: those to ACCEPT or REJECT the current cycle of results, and those to RETAIN and RELEASE data between cycles.

5.2. Conclusions

The provision of storing values on Level 1 and Level 2 for each datum generated in an iterative structural design process allows the comparison of the values generated on two adjacent cycles of calculation to decide whether or not the design variables generated on the succeeding cycle represent an improvement over the design variables of the preceeding cycle. If Level 2 is found to be an improvement over the Level 1, it replaces the preceeding cycle of values as the

Level 1, and the Level 2 is opened up for another cycle of calculation, if desired.

The data called the internally modifiable parameters are designated as the input parameters of each cycle of values. These data are usually the design variables which are being determined by the iterative process. The importance of internally modifiable parameters for processing is that they provide a reference point beyond which recursive SEEKing and WARNing procedures do not search for ingredients or dependents. Without these reference parameters these procedures could enter on an endless loop of recursion on one of the cyclic paths in the ingredience-dependence network.

The data structures and processes turned out to be considerably more complex than those previously developed for non-iterative processes. Also, because of the large volume of subscripted data involved, the complexity of possible initialization methods that may be used, and the variety of manners in which data may be entered, modified, retained and released, a system based on the representation presented would have to be significantly re-structured for each different design process.

Whether futher generalizations or simplifications can be attained will be known only after the representation presented herein has been programmed and tested on a number of representative design processes.

REFERENCES

1. American Institute of Steel Construction, Manual of Steel Construction, 7th ed., New York, 1970.

2. American Institute of Steel Construction, "Specification for the Design, Fabrication, and Erection of Structural Steel Buildings," New York, 1969.

3. Fenves, S. J., Computer Methods in Civil Engineering, Prentice-Hall, Inc., 1967.

4. Fenves, S. J., "Processing of Design Specifications Using a Recursive Decision-Table Processor," Information Processing 71, North-Holland Publishing Company, 1972.

5. Fenves, S. J., "Representation of the Computer-Aided Design Process by a Network of Decision Tables," Computers and Structures, Vol. 3, pp. 1099-1107, Pergamon Press, 1973.

6. Goel, S. K., Fenves, S. J., "Computer-Aided Processing of Design Specifications," Journal of the Structural Division, ASCE, Vol. 97, No. ST1, pp. 463-479, January, 1971.

7. Hatfield, F. J., Fenves, S. J., "The Information Organizer: A System for Symbolic Data Manipulation." Computers and Structures, Vol. 1, pp. 85-102, Pergamon Press, 1971.

8. Nyman, D. J., Fenves, S. J., Wright, R. N., "Restructuring Study of the AISC Specification," Civil Engineering Studies, Structural Research Series No. 393, University of Illinois, January, 1973.

9. Nyman, D. J., Fenves, S. J., "An Organizational Model for Design Specifications," Department of Civil Engineering, Carnegie-Mellon University, September, 1973.

10. Salmon, C. G., Johnson, J. E., Steel Structures, Design and Behavior, Intext Educational Publishers, Scranton, Pennsylvania, 1971.

11. Wright, R. N., Boyer, L. T., Melin, J. W., "Constraint Processing in Design," Journal of the Structural Division, ASCE, Vol. 97, No. ST1, January, 1971, pp. 481-494.

12. Fenves, S. J., "Representation of Information in the Design-Construction Process", Proceedings 6th Congress of CIB, Budapest, 1974, pp. 231-238.

13. Hsieh, Y. Y., Elementary Theory of Structures, Prentice-Hall, 1970.

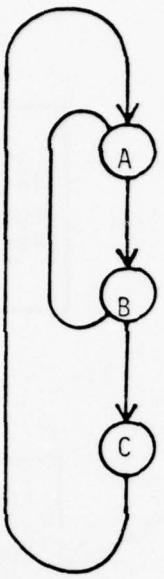


Figure 1a Cyclic Network of Data Variables

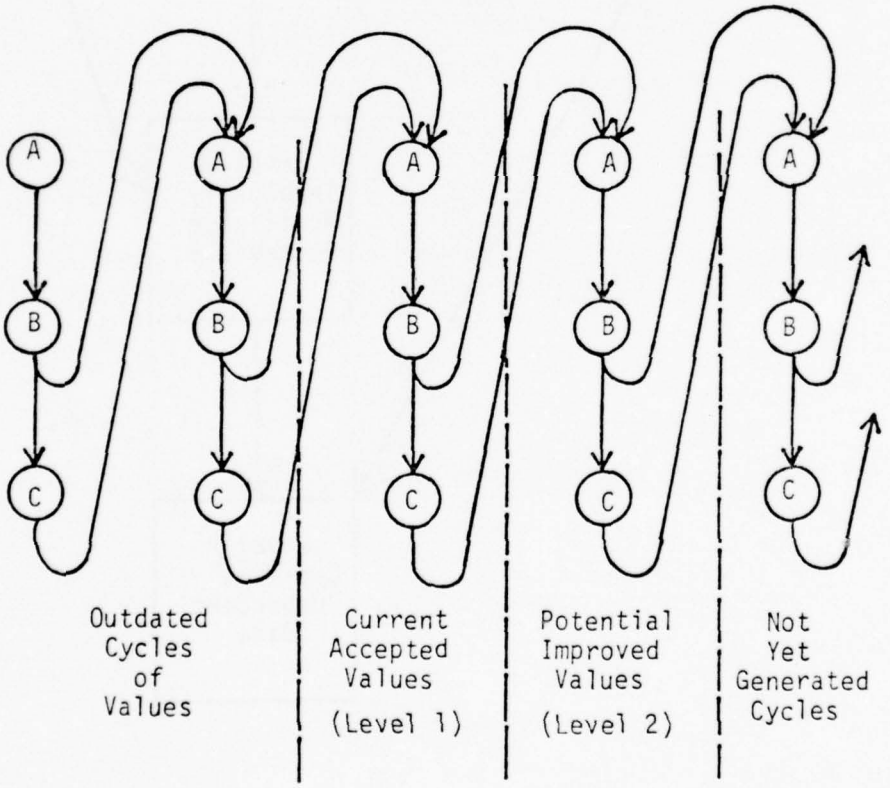


Figure 1b Acyclic Network of Data Values

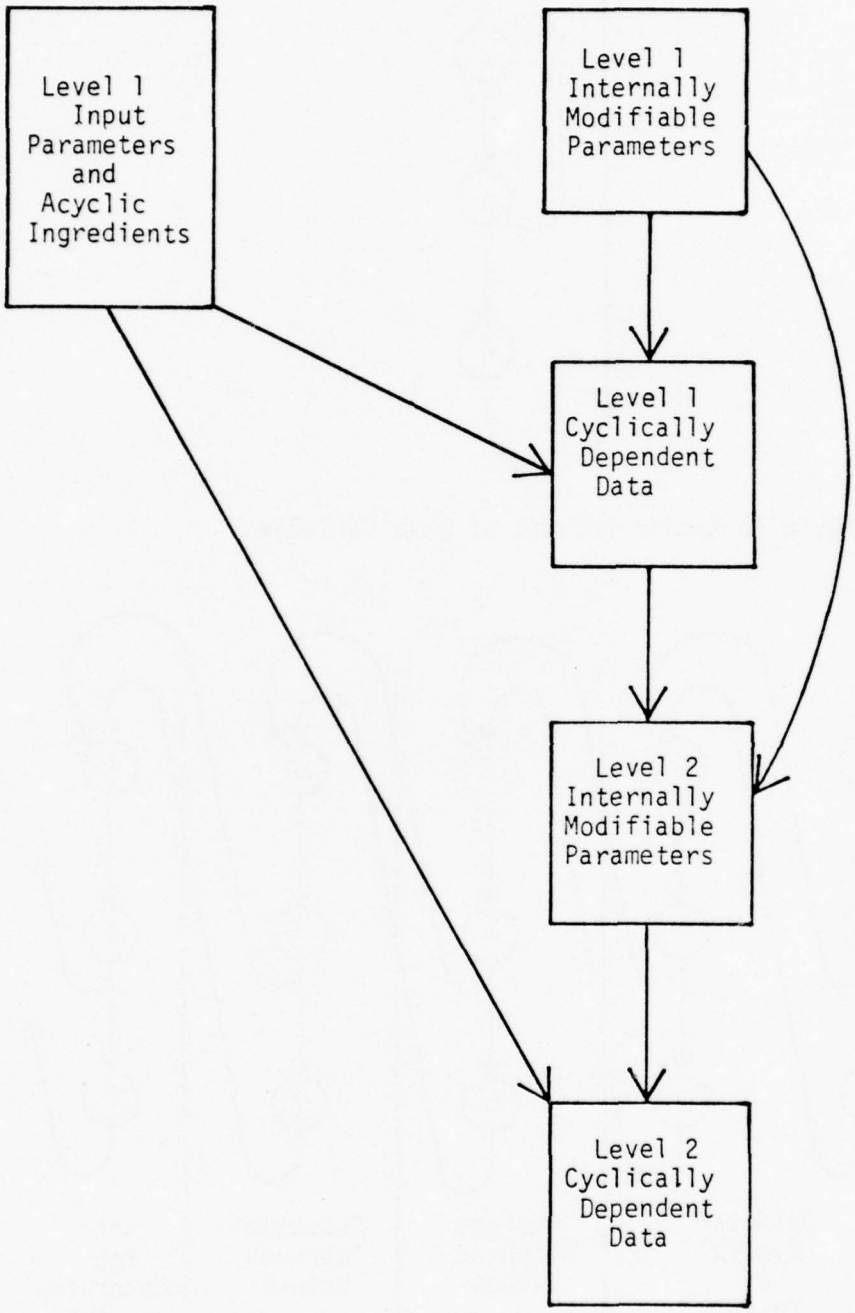


Figure 2 Ingredience-Dependence Relationships Among Data Values

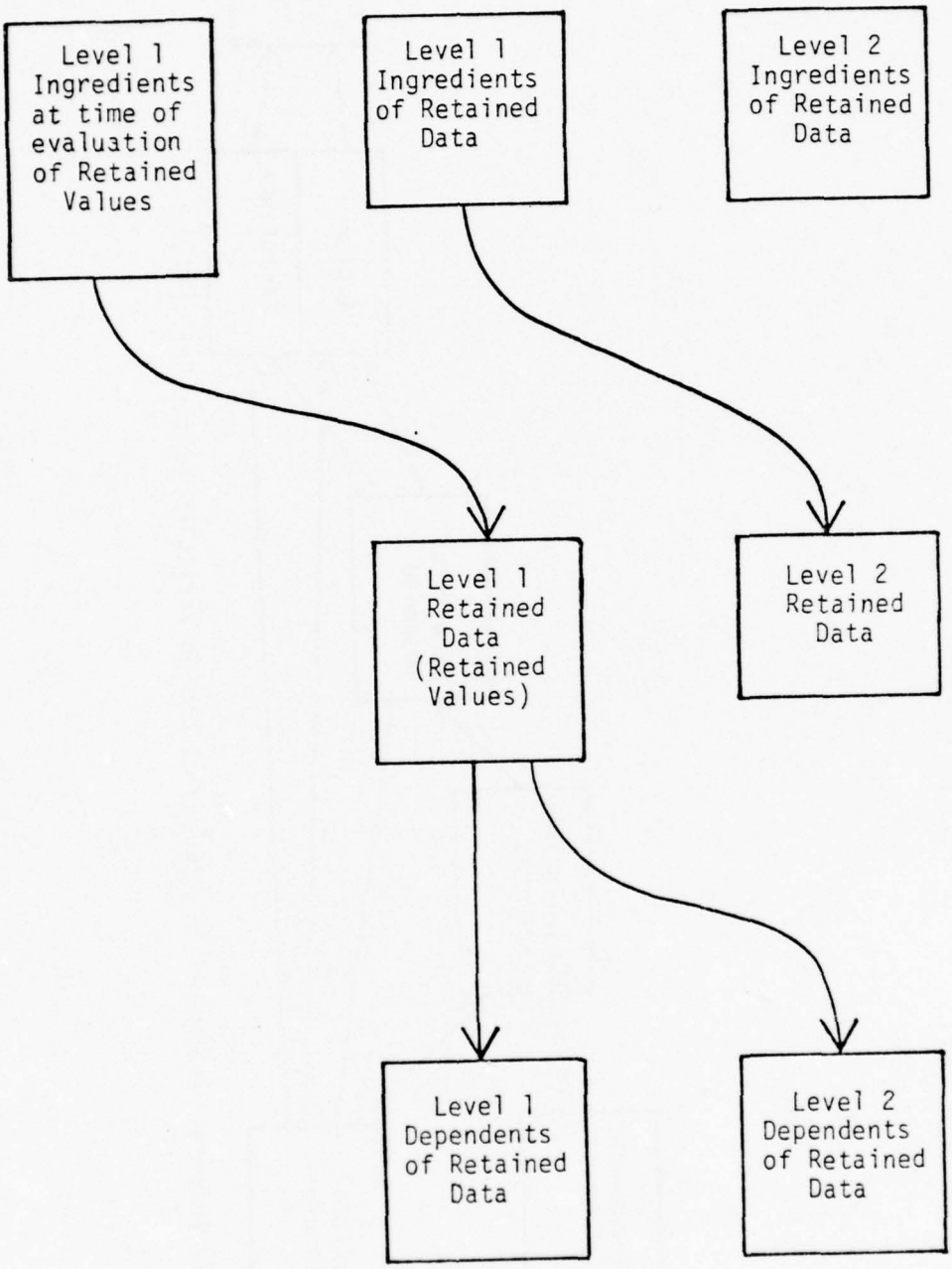


Figure 3 Ingredience-Dependence Relationships Between a Set of Retained Data and Data Not in the Set

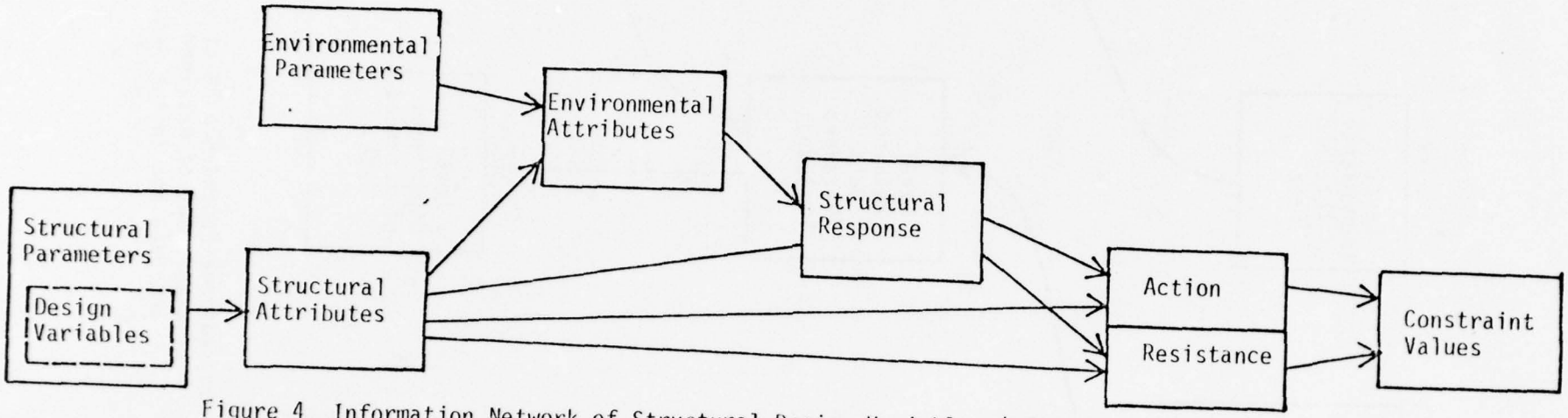


Figure 4 Information Network of Structural Design Variables (after Ref. [11]).

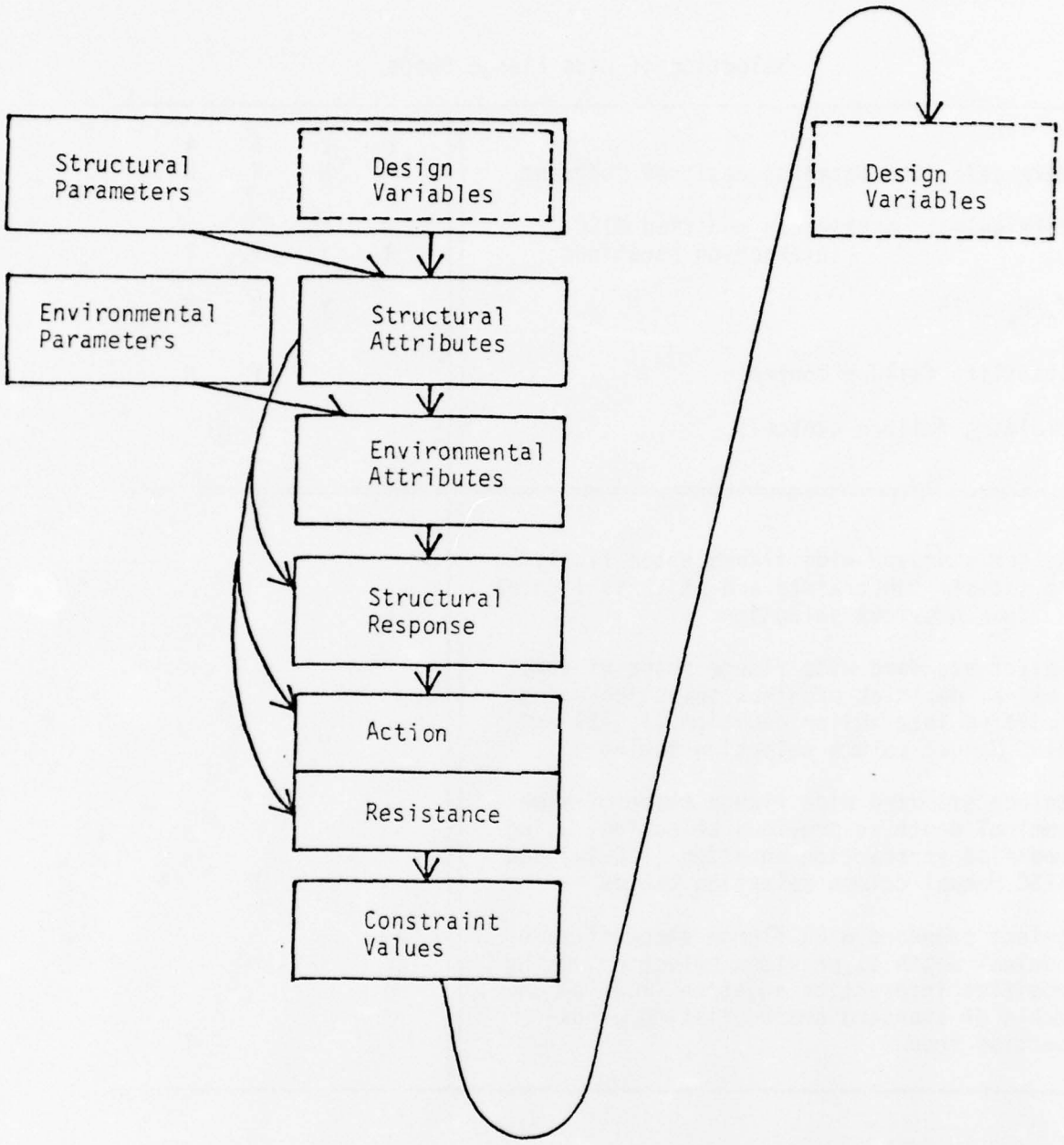


Figure 5 Cyclic Flow of Information in Structural Design

Table 1

Selection of Wide Flange Shape

	1	2	3	4
Make selection based on designer judgment	Y	N	N	N
Make selection based on modified AISC interaction equations	N	Y	Y	Y
$f_a/F_a \leq .15$		Y	N	N
Stability failure controls			Y	N
Yielding failure controls			N	Y
Select standard wide flange shape likely to satisfy constraints and which is lighter than previous selection	Y			
Select standard wide flange shape of same nominal depth as previous selection using modified interaction equation (1.6-2) and AISC Manual column selection tables		Y		
Select standard wide flange shape of same nominal depth as previous selection, using modified interaction equation (1.6-1a) and AISC Manual column selection tables			Y	
Select standard wide flange shape of same nominal depth as previous selection, using modified interaction equation (1.6-1b) and table of standard shapes listing cross-section area				Y

Table 2

ACCEPT - Accept Level 2 values as new Level 1 values, subject to restrictions noted in SACT(j)

	1	2	3
All Internally Modifiable Parameters present at Level 2	N	Y	Y
Required sets retained (for restrictions in SACT(j))		N	Y
For unretained data (SET(i)=0), erase Level 1 values and assign Level 2 values as Level 1. For retained data (SET(i)≠0), erase Level 2 values.			Y
Exit	Y	Y	Y

Table 3

REJECT - Erase Level 2 values, subject to restrictions in SACT(j)

	1	2	3
Internally Modifiable Parameters present at Level 2	N	Y	Y
Required sets retained (for restrictions in SACT(j))		N	Y
For unretained data (SET(i)=0), erase Level 2 values.			Y
Exit	Y	Y	Y

Table 4

SEEK1 - Seek datum i at Level 1 for specified values in SUB(j)

	1	2	3	4	5	6
Present at Level 1 (PR(ADDRESS)=1 or 3)	Y	N	N	N	N	N
Internally Modifiable Parameter (IMP(i)=1)		Y	N	N	N	N
Table Indicated (T(i)≠0)			Y	N	N	N
Routine Indicated (S(i)≠0)				N	Y	Y
All ingredients at Level 1					N	Y
Set r=0, SEEK1ACTION i			Y			
Execute S(i) with Level 1 Ingredients, mark Level 1 value present						Y
SEEK1 missing Ingredient					Y	
Error		Y		Y		
Exit	Y	Y	Y	Y		Y

Table 5

SEEK2 i - Seek datum i at Level 2 for specified values in SUB(j)

	1	2	3	4	5	6	7	8	9
Present at Level 2 (PR(ADDRESS)= 2 or 3)	Y	N	N	N	N	N	N	N	N
Datum acyclically dependent (SET(i)=1)		Y	N	N	N	N	N	N	N
Table Indicated (T(i)≠0)			Y	N	N	N	N	N	N
Routine Indicated (S(i)≠0)				N	Y	Y	Y	Y	Y
Internally Modifiable Parameter (IMP(i)=1)					Y	Y	N	N	N
All Ingredients at Level 1					N	Y			
All Ingredients of same set (i.e. SET(i')= SET(i) for ingredient i') are at Level 2							N	Y	Y
All Ingredients of different set (i.e. SET(i')≠SET(i) for ingredient i') are at Level 1								N	Y
Set r=0, SEEK2ACTION i			Y						
Execute S(i) with Level 1 Ingredients, mark Level 2 value present						Y			
Execute S(i) with Ingredients at needed Level, mark Level 2 value present									Y
SEEK1 missing Ingredient value					Y			Y	
SEEK2 missing Ingredient value							Y		
Error				Y					
Exit	Y		Y	Y		Y			Y
Exit, do not SEEK2 acyclic datum		Y							

Table 6

SEEK1ACTION - Isolate governing rule, r, of table, T(i), for Level 1 evaluation of datum, i, by action routine, AS(r, T(i))

	1	2	3	4
Governing Rule found	N	N	Y	Y
Any unchecked Rules	N	Y		
All ingredients at Level 1			N	Y
Set r=r+1		Y		
CHECK1RULE r, i		Y		
Execute AS(r, T(i)) with Level 1 Ingredients mark Level 1 value present				Y
SEEK1 missing Ingredient			Y	
Error	Y			
Exit	Y			Y

Table 7

SEEK2ACTION - Isolate governing rule, r, of table, T(i), for Level 2 evaluation of datum, i, by action routine, AS(r,T(i))

	1	2	3	4	5	6	7
Governing Rule found	N	N	Y	Y	Y	Y	Y
Any unchecked Rules	N	Y					
Datum i Internally Modifiable Parameter (IMP(i)=1)			Y	Y	N	N	N
All Ingredients at Level 1			N	Y			
All Ingredients of same set as datum, i (i.e. SET(i')=SET(i) for ingredient i') are at Level 2					N	Y	Y
All Ingredients of different set than datum, i, (i.e. SET(i')≠SET(i) for ingredient i') are at Level 1						N	Y
Set r=r+1		Y					
CHECK2RULE r, i		Y					
Execute AS(r,T(i)) with Level 1 Ingredients, mark Level 2 value present				Y			
Execute AS(r,T(i)) with Ingredients at needed Level, mark Level 2 value present							Y
SEEK1 missing Ingredient value			Y			Y	
SEEK2 missing Ingredient value					Y		
Error	Y						
Exit	Y			Y			Y

Table 8

CHECK1RULE - Check if rule, r, of table, T(i), governs, for Level 1 evaluation

	1	2	3	4
Any unsuccessful matches	Y	N	N	N
All Material Conditions matched		Y	N	N
Condition datum present at Level 1			Y	N
SEEK1 Condition datum				Y
Match Condition using Level 1 value			Y	
This rule governs		Y		
This rule doesn't govern	Y			
Exit	Y	Y		

Table 9

CHECK2RULE - Check if rule, r, of table, T(i), governs, for Level 2 evaluation

	1	2	3	4	5	6	7	8
Any unsuccessful matches	Y	N	N	N	N	N	N	N
All Material Conditions matched		Y	N	N	N	N	N	N
Datum i Internally Modifiable Parameter (IMP(i)=1)			Y	Y	N	N	N	N
Condition datum, i', of same set as datum, i (SET(i')=SET(i))					N	N	Y	Y
Condition datum present at Level 1			Y	N	Y	N		
Condition datum present at Level 2							Y	N
SEEK1 Condition datum				Y		Y		
SEEK2 Condition datum								Y
Match Condition using Level 1 value			Y		Y			
Match Condition using Level 2 value							Y	
This rule governs		Y						
This rule doesn't govern	Y							
Exit	Y	Y						

Table 10

INPUT i - Input value for datum, i, for specified values in SUB(j)

	1	2	3
Datum retained (SET(i)≠0 or 1)	Y	N	N
Any unretained data at Level 2		Y	N
Input value at Level 1, mark value present at Level 1			Y
WARN all Dependents which are not Internally Modifiable Parameters			Y
Exit			Y
Exit, do not input values for retained data	Y		
Exit, ACCEPT or REJECT Level 2 values first		Y	

Table 11

WARN - Erase Dependents of datum, i, and datum, i, itself

	1	2	3	4	5
Datum, i, retained (SET(i)≠0 or 1)	N	N	N	Y	Y
All retained Dependents erased at Level 2	N	Y	Y		
All non-retained, non-Internally Modifiable Parameter Dependents erased at Level 1		N	Y		
All Dependents of the same set erased at Level 2				N	Y
WARN Dependent	Y	Y		Y	
Erase Level 1 value			Y		
Erase Level 2 value					Y
Exit			Y		Y

Table 12

RETAIN k - Retain the Level 1 values of all data in the set, k

	1	2
Any unretained data at Level 2	Y	N
For each datum in set, k, mark SET(i)=k		Y
Exit		Y
Exit, ACCEPT or REJECT Level 2 values first	Y	

Table 13

RELEASE k - Release from retainment the Level 1 values of all the data in set, k

	1	2	3
Set, k, is the set of ACYCLIC data (k=1)	Y	N	N
Any unretained data at Level 2		Y	N
WARN non-set Dependents of all the data in the set			Y
For all data in the set, erase all existing Level 1 values, re-assign any existing Level 2 values as Level 1, and mark SET(i)=0			Y
Exit			Y
Exit, do not release ACYCLIC data	Y		
Exit, ACCEPT or REJECT Level 2 values first		Y	

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Representation of a Computer-Aided Interative Design Process	5. TYPE OF REPORT & PERIOD COVERED: Interim
	6. PERFORMING ORG. REPORT NUMBER: R-77-5
7. AUTHOR(s): Martin A. Tamm, Steven J. Fenves	8. CONTRACT OR GRANT NUMBER(s): N00014-76-C-0354 new, Task No. NR 064-536
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, Pa. 15213	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Standard Mechanics Program, ONR, Arlington, Va. 22217	12. REPORT DATE: April, 1977
	13. NUMBER OF PAGES: 44
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited for any purpose of the United States government.

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Structural Design
Computer-Aided Design Interation
Data Bases

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

From Abstract

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)