TECHNICAL RESEARCH REPORT

S Y S T E M S
R E S E A R C H
C E N T E R

Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

XBUILD: Pre-processor for Finite Element Analysis of Steel Bridge Structures

by M.A. Austin, S. Creighton, and P. Albrecht

TR 92-47

To appear in ASCE
Journal of Computing in Civil Engineering

XBUILD : Pre-processor for Finite Element Analysis of Steel Bridge Structures

By M.A. Austin, A.M. ASCE[1], S. Creighton[2], and P. Albrecht, M. ASCE[3]

ABSTRACT

Historically, the lack of interactive pre-processors to describe tedious finite element models has hindered the use of the finite element method for bridge analysis. The work performed in the present study mitigates this problem by providing an interactive graphically based pre-processor for the finite element analysis of highway bridges with high-speed engineering workstations. The name of the pre-processor is XBUILD. Version 1 of XBUILD is written in the C programming language, uses SunView graphics, and runs on SUN SPARCstations. The pre-processor gives engineers the choice of using both keyboard and mouse styles of interaction to describe and edit bridge geometries, set boundary conditions, define sub-structure and highway vehicle objects, and place trucks at desired positions on the bridge. The capabilities of XBUILD Version 1 are demonstrated by working through the step-by-step details of creating a finite element model for the FHWA-AISI test bridge.

Keywords : bridges, computer-aided design, pre-processor, finite elements, graphics.

---

[1] Assistant Professor, Department of Civil Engineering and Systems Research Center, University of Maryland, College Park, MD 20742, USA.

[2] Graduate Research Assistant, Department of Civil Engineering and Systems Research Center, University of Maryland, College Park, MD 20742, USA.

[3] Professor, Department of Civil Engineering, University of Maryland, College Park, MD 20742, USA.

INTRODUCTION

Despite three decades of research in finite element analysis methods, and significant advances in computer hardware and software, most engineers are still analyzing bridges with simplified pre-1960's methods. In current bridge engineering practice, for example, multi-girder highway bridges are analyzed in accordance with the standard specifications (1983) established by the American Association of State Highway Transportation Officials (AASHTO). These models assume that: (1) each girder is idealized as a two-dimensional beam acting separately from the other beams; (2) the idealized girder resists a fraction of the wheel loads as calculated with simplified formulas for wheel load distribution; and (3) in composite bridges, the idealized girder acts compositely with an effective width of the deck as calculated with a semi-empirical formula. At a time when slide rules were the only form of computational assistance, these assumptions were necessary simply to get the job done.

Today the picture has changed. Now that high-speed engineering workstations are readily available, conservative assumptions made for the sole purpose of simplifying analysis methods are no longer justifiable. Indeed, there is now strong evidence that bridge owner's are paying a high price for the engineer's convenience of reduced computational effort. Consider, for example, the two-span-continuous three-girder FHWA-AISI prototype bridge shown in Fig. 1. In a step towards improving the accuracy of bridge analysis, Elnahal, Albrecht and Cayes (1989) modeled the FHWA-AISI prototype bridge with the finite element analysis program `ANSYS` (Kohnke 1977). Finite element analysis methods consistently predicted the experimentally determined reactions and moments with an accuracy of ±10%. By comparison, the AASHTO analysis method predicted moments induced by

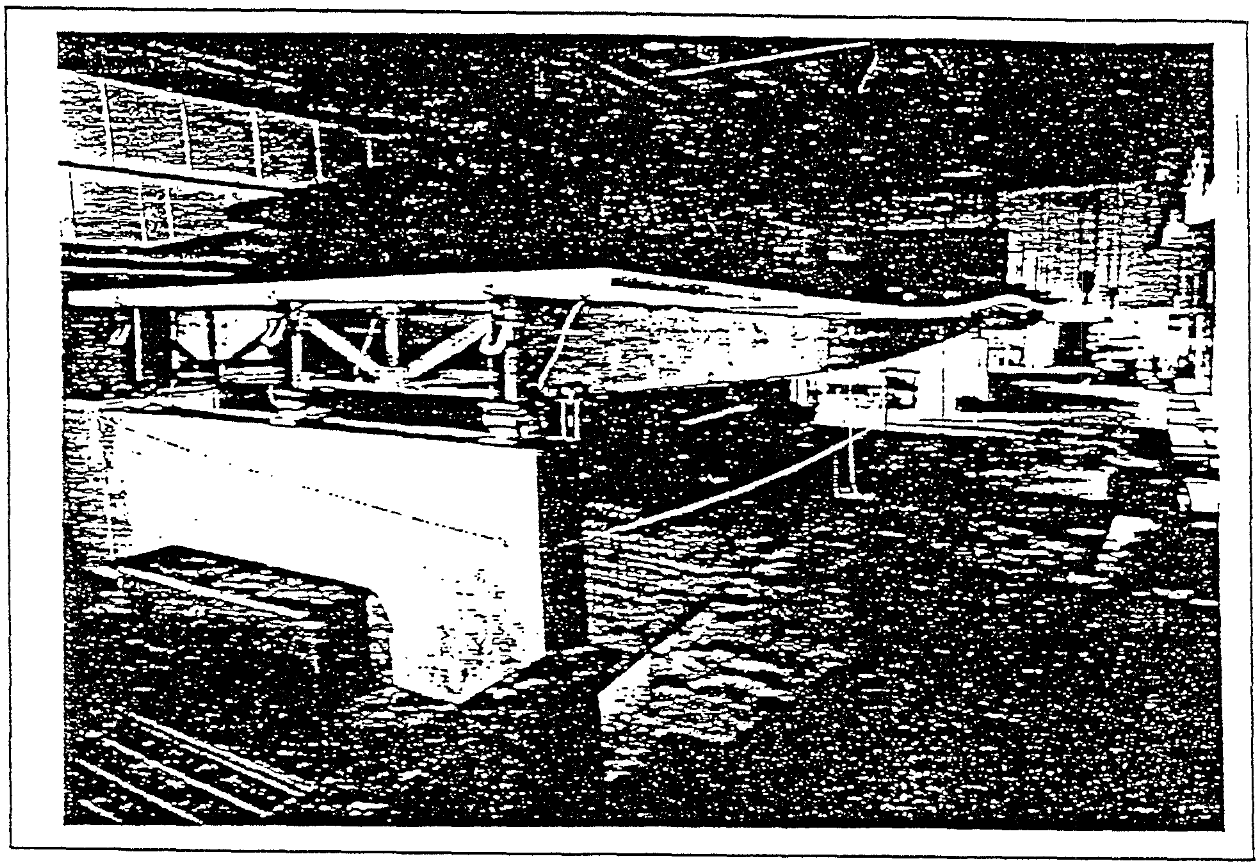

FIG. 1. - FHWA Test Bridge at Turner-Fairbanks Laboratory

truck oading up to two times larger than determined from the experiment.

Elnahal et al. (1989) also calculated the reactions and moments of the FHWA/AISI prototype bridge with the recently developed semi-empirical formulas for the distribution factor in slab-on-girder bridges (Nutt et al. 1987). The authors fully recognize that the prototype bridge with its three girders spaced 5.2m apart falls slightly outside the bounds for which the distribution factors of the Standard Specifications (1983) are valid, namely: four or more girders spaced 4.9m or less apart. Still, the empirical formulae predicted maximum moments 5 to 56 % higher than those determined from experimental work. And to make matters worse, these new distribution factors are only valid for interior girders.

Those for the exterior girders, which controlled the design of the prototype bridge, are still based on the AASHTO simple beam model. The findings clearly show the limitations of the distribution factor approach.

FINITE ELEMENT ANALYSIS OF BRIDGE STRUCTURES

Finite element analysis methods of bridges have so far failed to gain wide-spread acceptance among bridge engineers for three major reasons:

[1] Although several general-purpose finite element analysis programs are commercially available, such as ANSYS (Kohnke 1977), NASTRAN (McNeal-Schendler Corp. 1989), SAP (Wilson 1980), and STRUDL (GSTRUDL 1986), they all contain numerous bulky features not needed for bridge analysis. A notable exception is the CELL series of finite element programs for analysis of reinforced and pre-stressed bridges (William and Scordelis 1970, Scordelis et al. 1985).

[2] Most currently available bridge software systems, including the CELL series of programs, lack good public domain pre- and post-processor facilities to set up, edit, and query information on the response calculated from the finite element analyses.

[3] Special features for bridge analysis, such as automatic generation of moving truck loadings, are often missing.

In a recent NSF sponsored workshop for research needs in Short- and Medium-Span Bridges, Scordelis (1986) succinctly states the steps that must be taken to overcome these deficiences: "It has become evident to me that if these new analytical tools (referring to finite element analysis programs) are to be used by designers in the U.S., special purpose pre- and post-processing packages, written especially for bridge analysis, need to be developed as part of research projects undertaken in this field."

REQUIREMENTS FOR INTERACTIVE PRE- AND POST-PROCESSORS

An interactive pre- and post-processor dedicated to the design and analysis of bridge structures should provide engineers with the graphical and textual tools to: (a) view and manipulate the finite element model, and (b) assist in the interpretation of bridge design performance. Since the quantity of geometric information needed to describe a typical bridge structure may be enormous, graphical views of the finite element model are an indispensable means of detecting errors in the problem description.

The bridge pre- and post-processor should strike a balance between the attributes of "high functionality" and "ease of use." A practical way of achieving this objective is to design an interface that supports pulldown/popup menus, typing-at-the-keyboard, and picking styles of interaction. Also, mechanisms are needed to: (a) freely switch from one style of interaction to another, and (b) execute identical tasks via alternative paths (keyboard, menu or combinations thereof). Typing-at-the-keyboard commands are especially important because it is not practical to provide menus for all the commands needed to setup bridge design problems of a realistic size. The pre- and post-processor should employ icons and design objects that speak the same language as the bridge engineer (Heckel 1984). These design objects could include, for example, bridge diaphragms, traffic lanes, bridge decks, and HS20 trucks. Finally, pre- and post-processors should be developed on standard window systems, which are portable to a wide variety of PC's and engineering workstations. These window systems include SunView (SunView 1988) and X windows (Scheifler and Gettys 1986).

The writers surveyed several well-known finite element vendors and highway bridge consulting firms, and none had interactive pre- and post-processors dedicated to the analysis and design of highway bridges that met the above cited requirements.

PREVIOUS WORK IN PRE- AND POST-PROCESSORS

The precursor to the work reported in this paper is CSTRUCT, an interactive computer environment for the earthquake resistant design of two-dimensional planar steel structures (Austin et al. 1989). CSTRUCT handles linear and nonlinear structural analysis of truss and beam-column finite elements for both static and dynamic loadings. In the context of this paper, the term `beam-column` denotes a space frame element with two nodes. Each node is modeled with three translational and three rotational degrees of freedom.

CSTRUCT was developed with the philosophy that the engineer and computer should complement each other as they work together to setup, analyze and interpret the behavior for structural designs. As such, CSTRUCT employs a comprehensive and interactive command interpreter based on the UNIX tool YACC (Johnson 1975) to control the design and analysis processes via keyboard input. The result is a user interface that provides mechanisms for: (a) describing the design problem, (b) querying information about the design, (c) setting special performance criteria for unique structures, and (d) interpreting the attributes of design performance and behavior. The CSTRUCT graphical interface allows users to freely switch between typing-at-the-keyboard commands and those activated by selecting a pull-down menu item.

OBJECTIVES AND SCOPE

For the past two years, the writers have been formulating ideas and developing software for XBUILD (Creighton et al. 1990), a prototype pre-processor for generating three-dimensional finite element meshes, applying truck loadings, and specifying boundary conditions for straight, non-skewed, multi-girder highway bridges. XBUILD is written in the C programming language, uses SunView graphics (Sun Microsystems 1988), and is currently running on a SUN SPARCstation. Because its graphical interface borrows many ideas from CSTRUCT, readers are assumed to be familiar with the work reported by Austin et al. (1989). Discussion in this paper focuses exclusively on the new features XBUILD needs to handle three-dimensional, non-skewed steel bridge structures. The benchmark problem for testing XBUILD is a finite element model of the FHWA-AISI test bridge consisting of 341 nodes and 504 elements. The remainder of this paper describes features of the XBUILD graphical interface, procedures for setting up plate and beam-column finite element geometries, definition and manipulation of sub-structures, and facilities for defining and positioning truck object loadings. Finally, directions for continued research and development are given.

XBUILD GRAPHICAL USER INTERFACE

XBUILD's visual interface consists of a `parent` window and several `child` subwindows. The subwindows shown in Fig. 2 include: (1) a terminal emulator (TTY) for entering commands with the keyboard; (2) a canvas for displaying the `x-y` elevation of the structure; (3) a canvas for displaying the `x-z` plan of the structure; (4) a canvas for displaying the `z-y` section of the structure, in icon form; (5) a movement pad for the incremental update of viewing planes in positive and negative directions along an axis; and (6) pull-down menu

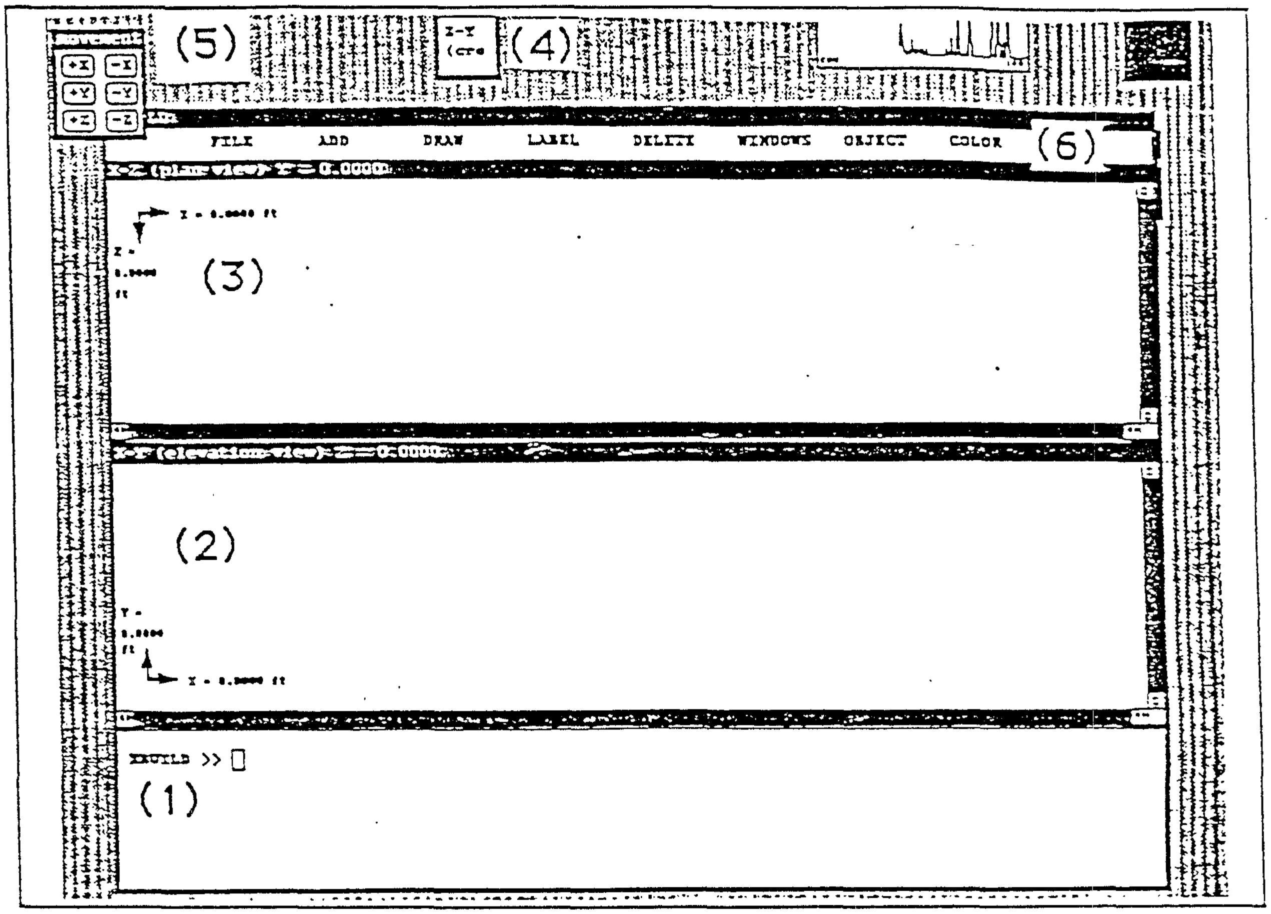

FIG. 2. - XBUILD Window Layout

items for ease of inputting data. The writers adopted the style of listing frequently used commands in the bar of pull-down menus located above the graphics and text windows. These include `FILE`, `ADD`, `DRAW`, `LABEL`, `DELETE`, `WINDOWS`, `OBJECT` and `COLOR`.

XBUILD's three canvas subwindows (x-y, x-z, and z-y) contain two coordinate arrows that are located at the subwindow's origin, and are parallel to its two principal axes. Each arrow may be extended or contracted in a direction parallel to its initial orientation. In the x-y subwindow, for example, the length of the x arrow is adjusted by depressing the right mouse button - this snaps the cursor to the tip of the right arrow - and dragging the arrow to the desired `x` coordinate. Views in the `z-y` clipping plane are automatically redrawn after the mouse button is released. Fig. 3 shows, as an example, a clipping plane of a bridge diaphragm located at $x = 10$ ft in the FHWA-AISI test bridge.

COMMAND LANGUAGE

Simple and frequently used commands may be executed by either selecting the appropriate menu item, or by typing an equivalent keyboard command. Both approaches result in the same post-command action. Lengthy or particularly complicated commands are entered with the keyboard alone, or with a combination of keyboard and graphics initiated commands. To this end, the UNIX tool YACC (Johnson 1975) was used to build an interactive command interpreter for defining problems. The YACC grammar in XBUILD has more than 150 rules. Rather than describe all of the command language's capabilities, this paper discusses YACC's basic ingredients. Most commands involve evaluation of arithmetic expressions. Since this is a standard application for YACC, details will be omitted here. Instead, the interested reader is referred to Austin et al. (1989) and to Chapter 8 of Kernighan and Pike (1983) for information. The remainder of the paper describes how to build numerical lists, specify engineering units, and create finite element models of bridge structures.

Numerical Lists. - A list is a sequence of zero or more elements of a given type. Complicated numerical lists of regularly and irregularly spaced items, such as node and element numbers, often need to be built in practical engineering design. Consequently, the XBUILD grammar allows designers to construct numerical lists for specifying `which` or `where` an action is to be applied to an object. Numerical lists may be created in several ways, some with

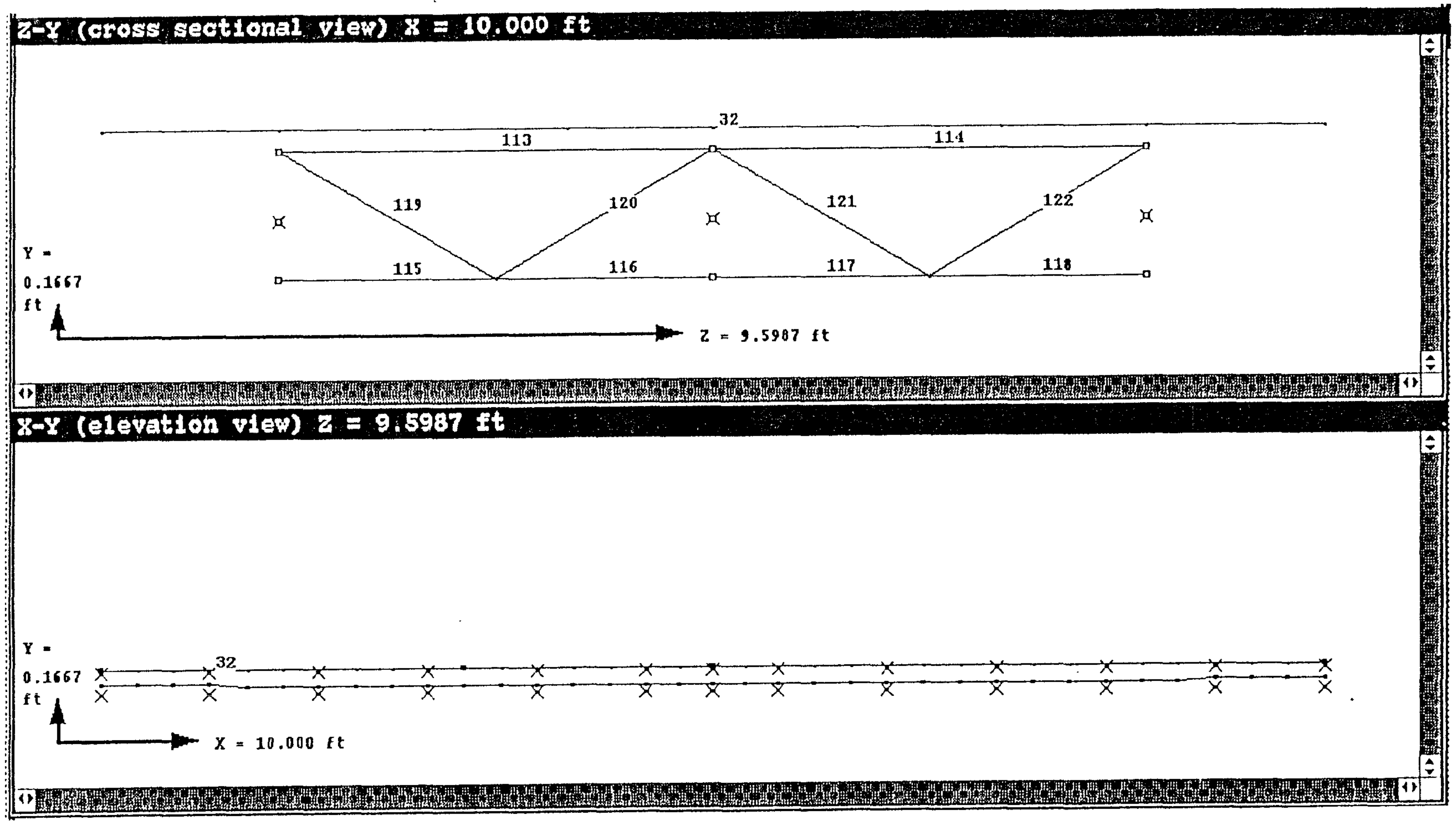

FIG. 3. - XBUILD's Clipping Plane at x = 10 ft

```
  <value> ::= <expr>                             (rule  1)
            | <quantity>                         (rule  2)
            ;

<numlist> ::= <value>                            (rule  3)
            | <numlist> ',' <value>              (rule  4)
            | [ <numlist> ]                      (rule  5)
            | <value> to <value>                 (rule  6)
            | [ <value> : <value> ]              (rule  7)
            | <value> to <value> by <value>      (rule  8)
            | [ ( <value> ~ <value> ) : <value> ] (rule  9)
            | [ <numlist> ';' <numlist> ]        (rule 10)
            ;
```

FIG. 4 - Numerical List Rules.

token names that closely mimic the English language, others with compact (but equivalent) forms using braces, parenthesis and semi-colons.

Fig. 4 summarizes the XBUILD grammar for constructing numerical lists. Rules 1 and 2 are the simplest cases. The patterns `<expr>` and `<quantity>` indicate that an engineering value may be the result of an arithmetic expression, or is an engineering quantity with units. The symbol `::=` is read as "is", and separates the left and right hand components of the grammar. The <> parentheses indicate that the command is essential, and the symbol | separates rules in the grammar. Rules 3 through 10 define the grammar for constructing numerical lists containing multiple elements. When a user types [**4,5,6,8**], for example, YACC uses rules 3,4 and 5 to match the input command. YACC then calls a function to generate a numerical list containing numbers 4,5,6, and 8. An important draw back of rules 3-5 is the burden it places on the user to specify a long list of regularly spaced numbers. To rectify this shortcoming, rules 6 and 7 allow a list to be generated by specifying its starting and ending values (i.e., range), with 1 being the implied separation increment.

```
  <quantity> ::= <length>                               (rule 11)
               | <force>                                (rule 12)
               | <area>                                 (rule 13)
               | <moment>                               (rule 14)
               | <linear_load>                          (rule 15)
               | <pressure>                             (rule 16)
               ;

    <length> ::= <expr> LENGTH                          (rule 17)
               ;

     <force> ::= <expr> FORCE                           (rule 18)
               ;

      <area> ::= <expr> AREA                            (rule 19)
               | <expr> LENGTH * LENGTH                 (rule 20)
               | <expr> LENGTH ^ 2                      (rule 21)
               ;

    <moment> ::= <expr> FORCE * LENGTH                  (rule 22)
               ;

<linear_load> ::= <expr> FORCE / LENGTH                 (rule 23)
               ;

  <pressure> ::= <expr> PRESSURE                        (rule 24)
               | <expr> FORCE / AREA                    (rule 25)
               | <expr> FORCE / LENGTH ^ 2              (rule 26)
               | <expr> FORCE / ( LENGTH * LENGTH )     (rule 27)
               ;
```

FIG. 5. - YACC Rules for Engineering Units.

For example, the numerical list 4,5,6,7,8 may be created by typing either **4 to 8** or [**4:8**]. Numerical lists may be concatenated to each other. Typing [**0 to 6 by 2; 6 to 12 by 3**], for example, generates the single list 0,2,4,6,9,12.

Engineering Units. - Quantities in bridge design are often specified in different engineering units. The command language recognizes the quantities of length, force, area, force times length (moment), force per unit length (linear load), and force per area (pressure or stress). It accepts most SI and US units for these quantities. The rules shown in Fig. 5 define each quantity. For example, 5 *sqin*, 5 *in*^2, and 5 $in * in$ are all acceptable ways of expressing five square inches. Similarly, 10 *psi*, 10 lb/in^2 and 10 $lb/(in * in)$ each express the same pressure. Thus, units of a quantity are defined by either single word descriptors or arithmetic combinations of sub-units.

FINITE ELEMENT MODELING IN XBUILD

The current version of XBUILD supports two-node beam-column elements and four-node quadrilateral plate elements. The beam-column elements have three translational and three rotational degrees of freedom per node. Each node of a four-node plate element has three translational and two rotational degrees of freedom. XBUILD also provides special nodes with constrained degrees of freedom called offset nodes. Beam offset nodes are connected to plate nodes with rigid links to model the fully composite action between the steel girders and the concrete deck within the linear regime. As such, displacements of the beam nodes are a linear function of displacements at the plate nodes, and only the latter appear in the global stiffness matrix (Cook et al. 1974).

For small to moderate sized problems having possibly irregular mesh geometries, models are created by first specifying the coordinates of nodes and offset nodes, and then attaching elements to these nodes. Both types of nodes are input with the keyboard. This is generally faster than using a mouse, and the coordinates can be specified to the precision of the computer. Typing the command **add node**

```
XBUILD >>
XBUILD >> add node
XBUILD_add_node >>
```

moves XBUILD into a mode for adding coordinates. XBUILD responds with the `add_node` prompt, as shown. Nodal coordinates are then entered with commands of the type:

```
XBUILD_add_node >> x <numlist> y <numlist> z <numlist>
```

where <numlist> matches rule 3 in Fig. 4 and x, y, and z are identifiers for numerical lists containing the nodal coordinates along the x, y and z axes, respectively. Similarly, nodal offsets are specified by first moving into an `add nodal_off` mode, and then typing commands of the form:

```
XBUILD >>
XBUILD >> add nodal_off
XBUILD_add_nodal_off >> x <numlist> y <numlist> z <numlist>
```

In both cases, XBUILD generates a three-dimensional rectangular block of nodal coordinates. When one coordinate attribute is a single numerical value, instead of a general numerical list, XBUILD generates a rectangular mesh of nodes. XBUILD generates a line of nodes when two coordinate attributes are single numerical values. As will be shown below, beam-column and plate finite elements may be attached to the nodes by using keyboard input or graphically defining nodal regions with the mouse.

BRIDGE PRE-PROCESSOR : FHWA TEST BRIDGE EXAMPLE

The features of XBUILD are demonstrated in this section with selected steps needed to create a finite element mesh of the FHWA-AISI test bridge. Creighton et al. (1990) give a complete sequence of commands, with commentary.

Generation of Quadrilateral Plate Element Mesh. - The deck of the FHWA-AISI bridge is modeled with a rectangular mesh of four-node quadrilateral plate elements shown in Fig. 6. In the x direction, the plates are 40 in long from x = 0 to x = 600 in, 36 in long from x = 600 in to x = 744 in, and 40 in long from x = 744 to x = 1344 in. In the z direction, the plates are 33.56 in wide from z = 0 to z = 33.56 in, 27.208 in wide from z = 33.56 to z = 196.808 in, and 33.56 in wide from z = 196.808 to z = 230.368 in. The mesh of plates lies on the y = 29.707 in plane. All plates are 4 in thick and made of 7.6-ksi concrete.

For large, regularly shaped finite element meshes such as the FHWA-AISI bridge, XBUILD provides a special mechanism to generate concurrently plate elements and nodes with the keyboard. The following sequence of commands specifies the thickness of all plates, assigns to the plates the properties stored in `CON7.6` (predefined concrete material names are stored in the XBUILD symbol table), sets the default unit of length to inch, and generates 315 nodes and 272 quadrilateral plate elements for the bridge deck.

```
XBUILD >> add plate4
XBUILD_add_plate4 >> set plate4 thickness to 4 in
XBUILD_add_plate4 >> set plate4 material to "CON7.6"
XBUILD_add_plate4 >> set length units to in
XBUILD_add_plate4 >> x [0 to 600 by 40; 600 to 744 by 36; 744 to 1344 by 40]
                     z [[0~33.56:33.56] ; [33.56~196.808:27.208] ; [196.808~230.368:33.56]]
                     y 29.707
  INFO_add_plate4 >> No. of nodes generated  = 315
  INFO_add_plate4 >> Total No. of nodes      = 315
  INFO_add_plate4 >> No. of plates generated = 272
  INFO_add_plate4 >> Total No. of nodes      = 272
XBUILD_add_plate4 >>
```

Fig. 6 shows the plan and elevation views of the resulting finite element mesh.

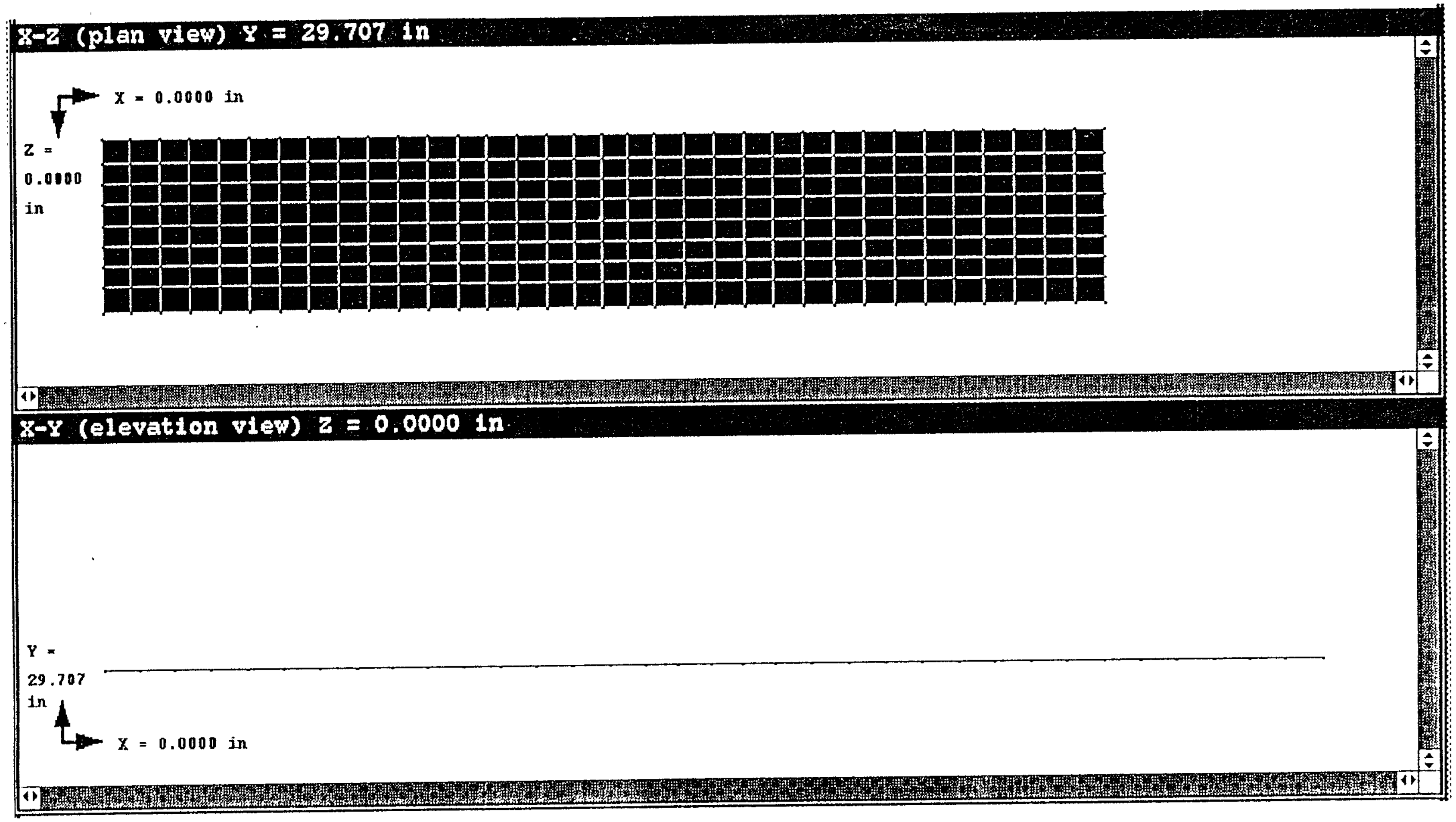
X-Z (plan view) Y = 29.707 in
X = 0.0000 in
Z = 0.0000 in
X-Y (elevation view) Z = 0.0000 in
Y = 29.707 in
X = 0.0000 in

FIG. 6. - Plate Element Mesh for Modeling Bridge Deck

Generation of Bridge Girders. - The three girders of the FHWA-AISI test bridge are modeled as beam-column elements. To connect the beam-column elements of the girders to the plate elements of the deck, the user imposes master-slave constraints with XBUILD's offset node feature, i.e:

```
XBUILD >> add nodal_off
XBUILD_add_nodal_off >> x [0 to 600 by 40; 600 to 744 by 36; 744 to 1344 by 40]
                        z 33.56 to 196.808 by (27.208*3)
                        y 12.51
  INFO_add_nodal_off >> No of nodal offset nodes generated = 105
  INFO_add_nodal_off >> Total No. of nodal offsets         = 105
XBUILD_add_nodal_off >>
```

Cross section and elevation views of the offset nodes are shown in Fig. 7. Beam-column elements are added by first entering the `add bc_elmt` mode, and then using either the mouse or keyboard to specify nodal connectivities for the beam-column elements. For example, the elevation view of Fig. 7 shows the definition of a rectangular box region for adding beam-column elements. The rectangular box is obtained by depressing the left mouse button, dragging, and then releasing the button. All nodes within the box are numerically ordered, and a series of beam-column elements are generated between the nodes. The result is shown in the `x-y` elevation view of Fig. 8.

Generation of First Bridge Diaphragm. - The vertical cross-braces, located midway between the girders on each diaphragm, are assumed to be zero force members and are omitted from the model. The command sequence

```
XBUILD >> add nodal_off
XBUILD_add_nodal_off >> x 0 y 1.707 to 25.707 by 24  z 33.56 to 196.808 by (27.208*3)
XBUILD_add_nodal_off >> <
XBUILD_add >> node
```

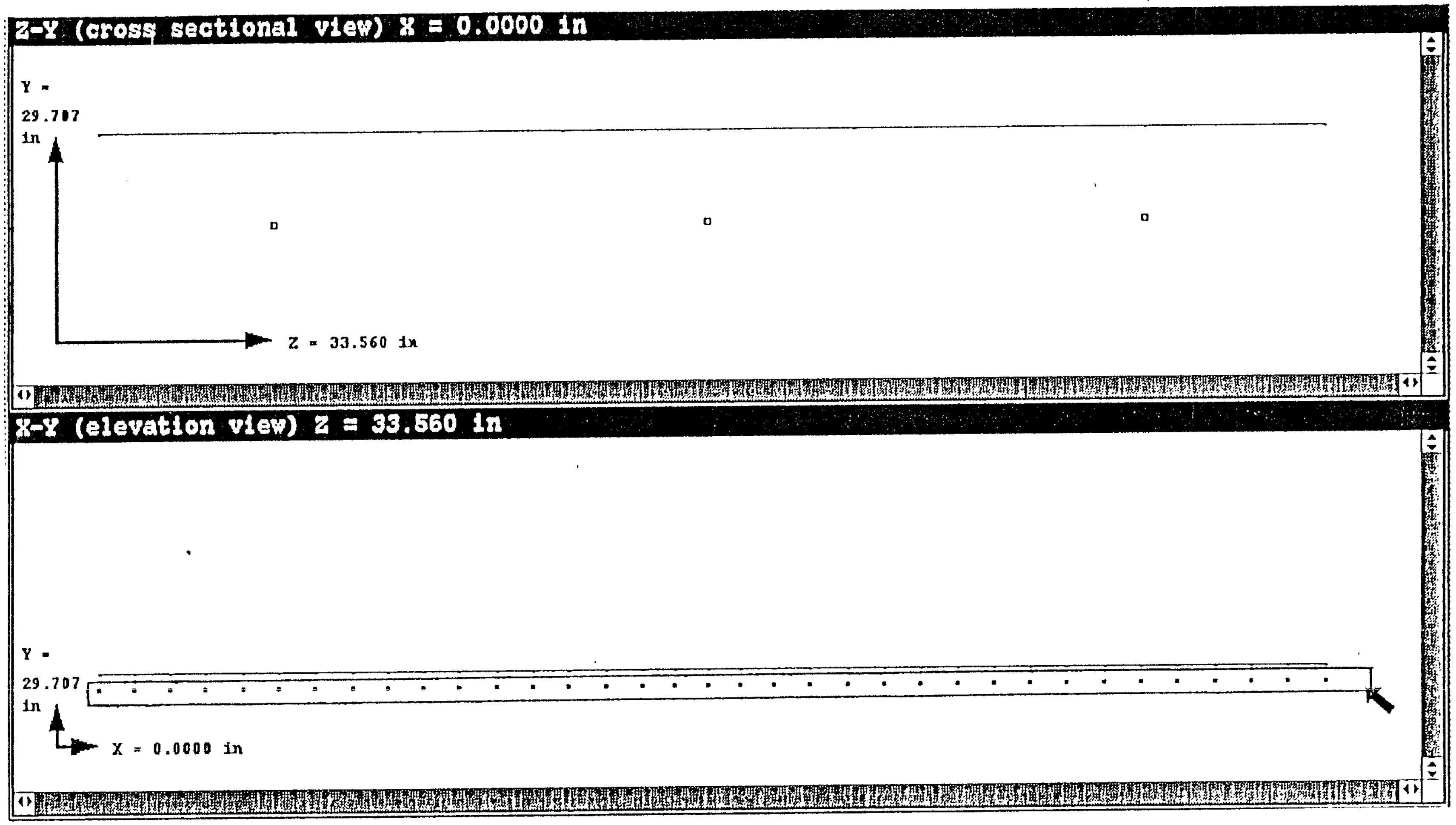
Z-Y (cross sectional view) X = 0.0000 in
Y =
29.707
in
Z = 33.560 in
X-Y (elevation view) Z = 33.560 in
Y =
29.707
in
X = 0.0000 in

FIG. 7. - Addition of Girder Elements with Mouse

```
XBUILD_add_node >> x 0   y 1.707   z 74.621 to 156.243 by (156.243-74.621)
XBUILD_add_node >> <
XBUILD_add >> bc_elmt
XBUILD_add_bc_elmt >>
```

generates nodes and nodal offsets shown in the cross sectional view of Fig. 8, and moves XBUILD into a program state for adding beam-column elements. Horizontally oriented beam-column elements are most conveniently added by defining a rectangular box of nodes, as described above. Beam-column elements whose nodes are difficult to grab with the mouse are added via keyboard input using the command syntax:

```
XBUILD >> add bc_elmt
XBUILD_add_bc_elmt >> from node <expr> to node <expr>
```

where the first and second instances of `<expr>` are nodal connectivities. For example, the following sequence of keyboard commands sets the beam-column section type to `FHWA_DIA` (predefined section type stored in XBUILD symbol table) and defines the cross braces for the diaphragm.

```
XBUILD_add >> bc_elmt
XBUILD_add_bc_elmt >> set bc_elmt section to "FHWA_DIA"
XBUILD_add_bc_elmt >> from Nodal_off 109 to Node 316
  INFO_add_bc_elmt >>  No of elmts generated =    1
  INFO_add_bc_elmt >>  Total No. of elms     =  109
XBUILD_add_bc_elmt >> from Nodal_off 110 to Node 316
  INFO_add_bc_elmt >>  No of elmts generated =    1
  INFO_add_bc_elmt >>  Total No. of elms     =  110
XBUILD_add_bc_elmt >> from Nodal_off 110 to Node 317
  INFO_add_bc_elmt >>  No of elmts generated =    1
  INFO_add_bc_elmt >>  Total No. of elms     =  111
XBUILD_add_bc_elmt >> from Nodal_off 111 to Node 317
  INFO_add_bc_elmt >>  No of elmts generated =    1
  INFO_add_bc_elmt >>  Total No. of elms     =  112
```

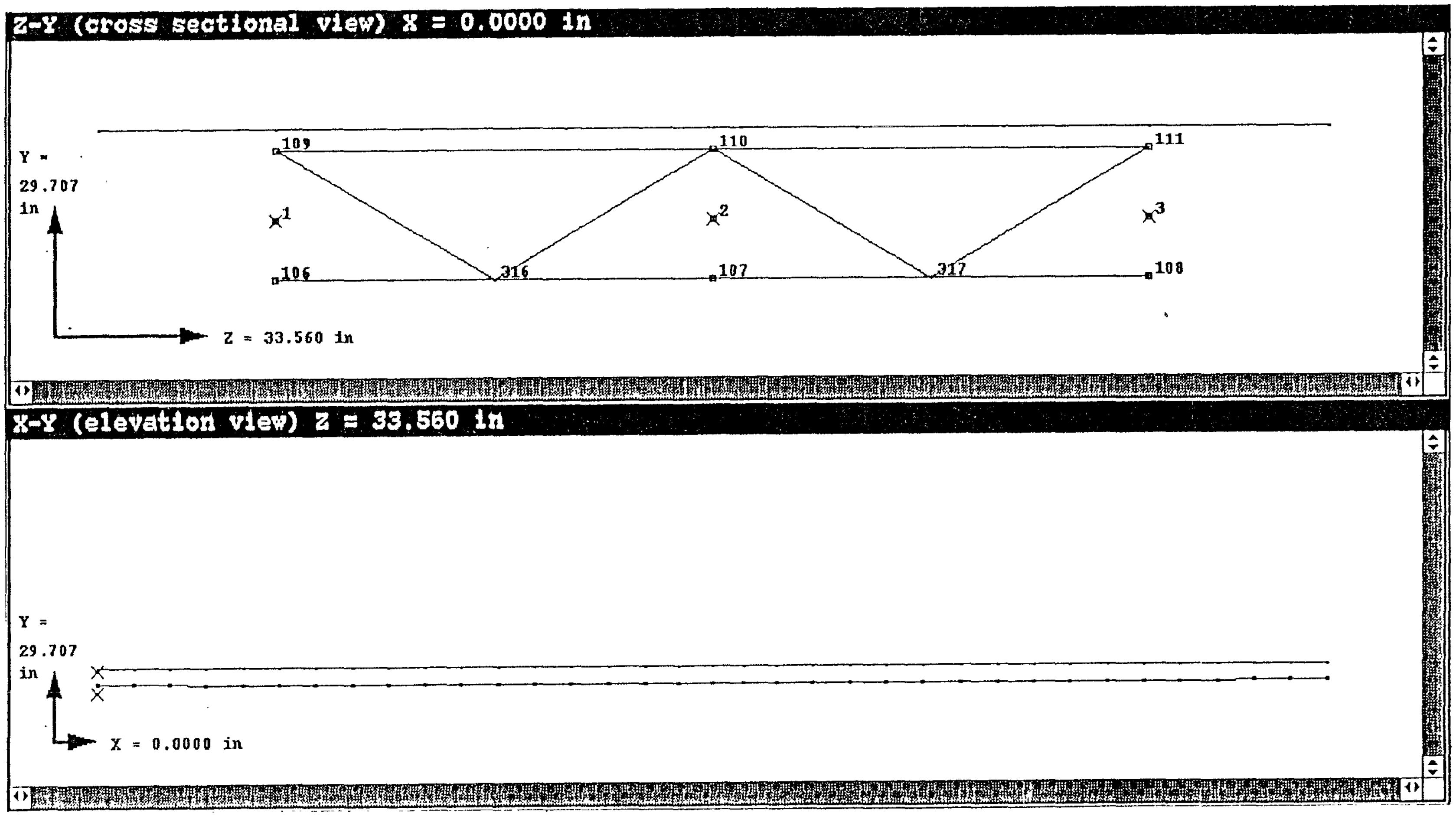

FIG. 8. - Diagonal Cross Braces are Added with Language

The combined results of the mouse and keyboard input are shown in the cross section view of Fig. 8.

STRUCTURAL OBJECTS

A structural object is a collection of items such as nodes, elements, loads, and boundary conditions that can be given a name, duplicated, queried, and modified. The duplication feature allows the user to position an object any place in the model as often as needed. The diaphragms of the FHWA Bridge are a good example for illustrating the duplication feature. Rather than adding nodes, offset nodes, and elements for each of the bridge's eleven diaphragms, one diaphragm is added, and given an object name, say, **dia**. Copies of the object are then placed at the other ten diaphragm positions along the bridge length.

Creation of "dia" (Diaphragm) Object. - Substructure objects consisting of nodes, offset nodes, and elements may be created, named, and stored in the XBUILD symbol table. New objects are created by first moving into the `create object` mode, providing a name for the object, and then using the mouse and keyboard to declare node and element numbers belonging to the object. For example, the diaphragm object shown in the cross sectional view of Fig. 9 is initiated with the command sequence:

```
XBUILD >> create object
XBUILD_create_object >> start object "dia"
```

Groups of horizontally and vertically oriented nodes and beam-column elements needed for "dia" may be grabbed with the mouse, as already demonstrated. Elements that are difficult to grab uniquely, are added to the object definition with keyboard commands; e.g. `include bc_elmt 109`. A summary of object contents is displayed by simply typing:

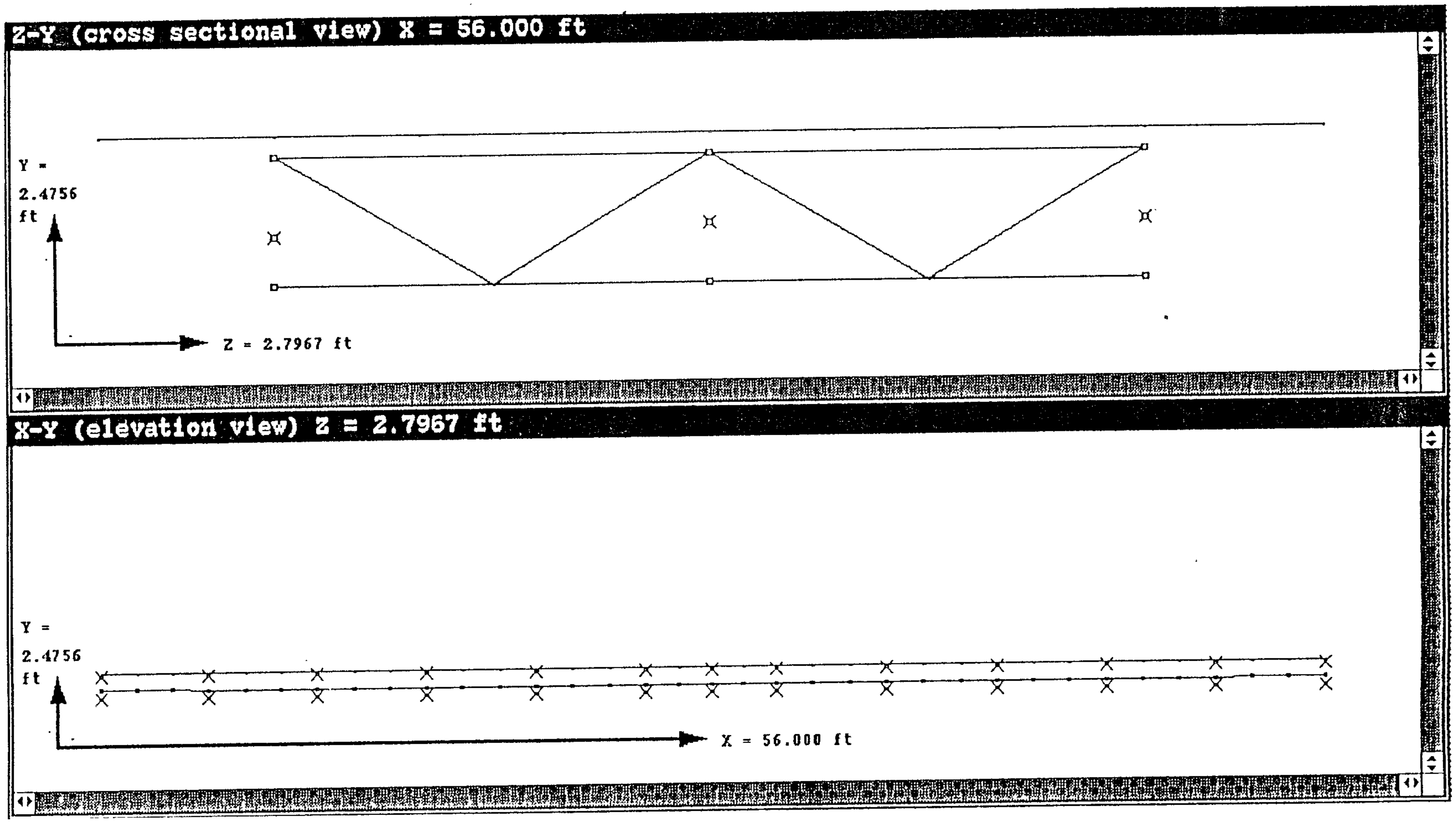
Z-Y (cross sectional view) X = 56.000 ft
Y = 2.4756 ft
Z = 2.7967 ft
X-Y (elevation view) Z = 2.7967 ft
Y = 2.4756 ft
X = 56.000 ft

FIG. 9. - Duplication of first Diaphram along Bridge.

```
XBUILD_create_object >> print object "dia"
  INFO >> Node Nos    : -111 -110 -109 -108 -107 -106  316  317
  INFO >> Bc elmt Nos :  103  104  105  106  107  108  109  110  111  112
XBUILD_create_object >>
```

Here nodes and offset nodes are distinguished by prefixing the latter group by a minus sign.

Duplication and Positioning of Structural Objects. - The object `dia` (diaphragm) is duplicated and placed longitudinally along the bridge. The command sequence:

```
XBUILD_create_object >> set length units to ft
XBUILD_create_object >> copy "dia" x 10 to 50 by 10
XBUILD_create_object >> copy "dia" x 56
XBUILD_create_object >> copy "dia" x 112-50 to 112 by 10
```

sets units of length to feet. In the first span, diaphragms are placed at x=10,20,30,40, and 50 ft. One copy of the diaphragm is placed at the bridge center; at x=56 ft. In the second span, copies of the diaphragm are symmetrically positioned with the first span. Visual feedback of the duplicated diaphragms is given in the elevation view of Fig. 9.

TRUCK OBJECTS

The AASHTO guidelines (1983) for live load were developed in 1944. The guidelines state that live loads on highway bridges shall consist of standard trucks or lane loads that are equivalent to truck trains. Today, the standard class of highway loading, designated HS20, consists of a tractor truck with semi-trailer of 72,000 lbs total load on three axles, or a corresponding lane load of 640 lbs per linear foot of load lane combined with a concentrated load of 18,000 lbs for moment and 26,000 lbs for shear. The loads are assumed to be uniformly distributed over a 10 ft width within a 12 ft wide design traffic lane. The traffic lanes are placed in such numbers and positions, across the entire bridge roadway width measured between curbs, so as to produce the maximum stress in the member under consideration. AASHTO does not require modeling of individual wheel loads.

XBUILD provides for the application of individual wheel loads, and dead and live loads uniformly distributed along a line, or over an area. However, because the bridge deck is now modeled as a finite element plate mesh, there is no longer a good reason for not modeling standard truck loadings with individual wheel loads. The implementation of live loads includes: (a) a simple language for describing vehicle geometry, axle loadings, and wheel positions, (b) procedures for reading and storing vehicle definitions in a symbol table, and (c) a set of functions for positioning and moving vehicles on the bridge deck.

Consider, for example, the `HS20` truck shown in Fig[4]. 10. The origin of the truck coordinate system is located at the center of the front axle. The model assumes that axles are oriented perpendicular to the center line of the vehicle, and all wheels belong to an axle. Consequently, wheel positions are determined from the position of the parent axle and the perpendicular distance of the wheel to the vehicle centerline. These ideas are reflected in Fig. 11, which is an abbreviated version of the `HS20` truck and geometry. It is written in simple language with a small set of keywords - axle, wheel, perimeter, and so on - for identifying and dimensioning components of the vehicle in a hierarchal manner. Several features are worth noting. First, all vehicle, axle, and wheel definitions are identified by name. Loads are applied at the axle level, and divided equally among wheels on an axle.

[4] This figure is taken from the Standard Specifications for Highway Bridges, Fourteenth Edition. Copyright 1990, American Association on State Highway and Transportation Officials, Washington, D.C., Used by permission.

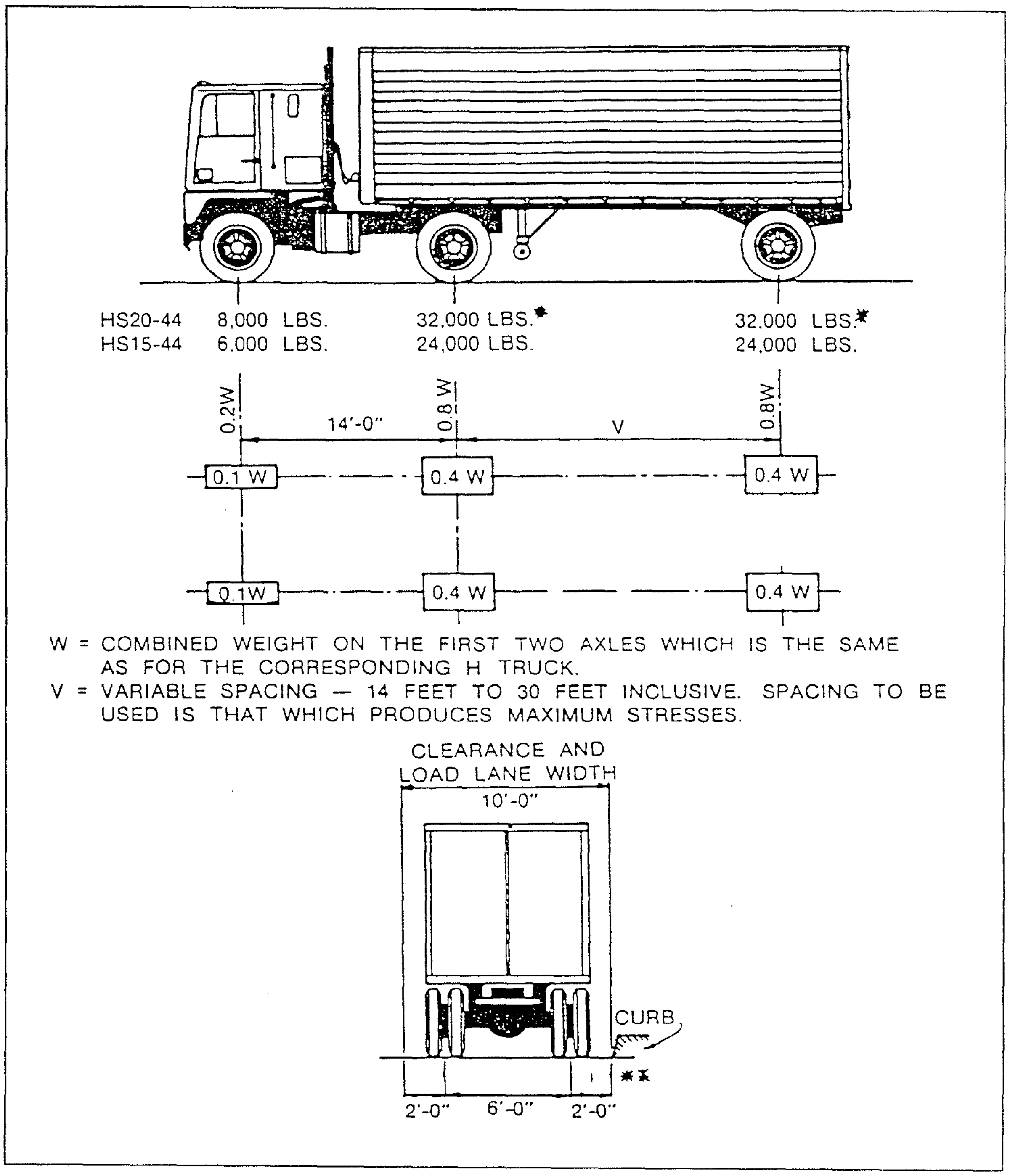
HS20-44
HS15-44
8,000 LBS.
6,000 LBS.
32,000 LBS.*
24,000 LBS.
32,000 LBS.*
24,000 LBS.
0.2W
0.8W
0.8W
14'-0"
V
0.1 W
0.4 W
0.4 W
0.1W
0.4 W
0.4 W
W = COMBINED WEIGHT ON THE FIRST TWO AXLES WHICH IS THE SAME AS FOR THE CORRESPONDING H TRUCK.
V = VARIABLE SPACING — 14 FEET TO 30 FEET INCLUSIVE. SPACING TO BE USED IS THAT WHICH PRODUCES MAXIMUM STRESSES.
CLEARANCE AND LOAD LANE WIDTH
10'-0"
CURB
**
2'-0"
6'-0"
2'-0"

FIG. 10. - HS20 Truck Geometry and Axle Loads

The dimensions and positions of all wheels are defined as part of an axle. Finally, the language supports the building of lists of coordinate positions. In the HS20 definition, a

```
/* Data File for Standard Truck Definition */

truck_perimeter = { (2.5,5), (-48,5), (-48,-5), (2.5,-5) }
truck("HS20") {
      perimeter {ft} { truck_perimeter }
      axle {
          name  "axle1";
          origin distance 0.0 ft;
          load   8 kips;
          wheel { name  "front_right";
                  center distance  3 ft;
                  diameter    1 ft;
                  width      .5 ft;
          }
          wheel { name "front_left";
                  center distance  -3 ft;
                  diameter  1 ft;
                  width .5 ft;
          }
      }
      axle {
          name  "axle2";
          origin distance -14 ft;
          load  32 kips;

          .... details of wheel positions deleted ....
      }
      axle {
          .... details of axle position and wheel loadings deleted ....
      }
}
```

FIG. 11. - Abbreviated Truck Object Datafile

list of coordinate positions for the truck perimeter is built and temporarily stored in the symbol table as `truck_perimeter`. It is subsequently referenced by name and given units of feet within the `HS20` truck definition.

The contents of Fig. 11 are stored in a file called `truck.data`. Standard truck loading definitions are read from `truck.data` and stored in the XBUILD symbol table as part of the programs startup procedure.

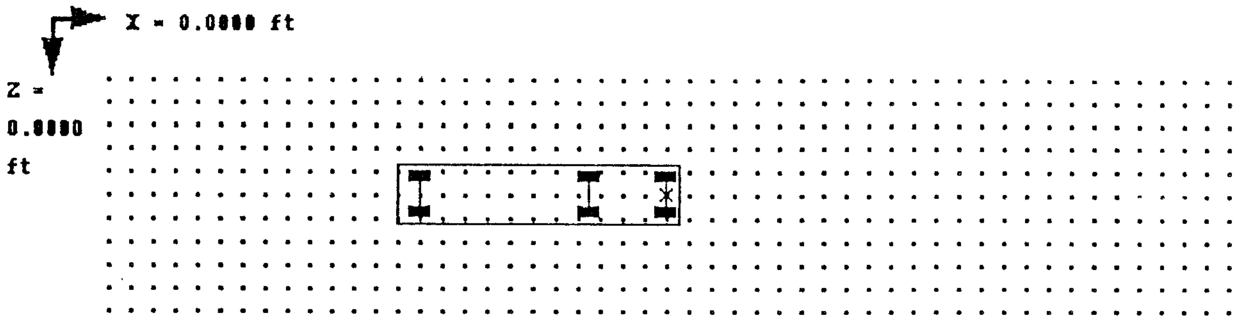

FIG. 12. - Schematic of Truck Vehicle Loading on Bridge Deck

Positioning and Movement of Highway Vehicles. - Once defined, trucks may be identified by name, and positioned on the bridge as a complete object. For example, the command sequence:

```
XBUILD_add_plate4 >>
XBUILD_add_plate4 >> truck "HS20"
  INFO_add_plate4 >> 8.0 kips <-- -14 ft --> 32.0 kips <-- -44.0ft --> 32.0 kips
XBUILD_add_plate4 >>
XBUILD_add_plate4 >> set length units to ft
XBUILD_add_plate4 >> place truck "HS20" x 100 ft z 20 ft
```

displays HS20 truck axle positions relative to the truck coordinate origin, along with total weight on each axle. The coordinate origin of the HS20 truck is then positioned at bridge coordinates x = 100 ft, z = 20 ft, as shown in Fig. 12. A second application of lists of coordinate positions is to generate path sequences for moving trucks. For example, the command:

```
XBUILD >> move truck "HS20" {ft} {(175, 15), (100,20), (75,30)}
```

sets units of length to feet, and sequentially positions an `HS20` truck at coordinates (175,15), (100,20) and (75,30). The authors are currently extending the set of list operations to replicate the functionality of **`<numlist>`**.

CONCLUSIONS AND FUTURE WORK

The authors have developed a pre-processor for: (a) generating three-dimensional finite element meshes of straight, non-skewed multi-girder steel highway bridges; (b) specifying boundary conditions, section sizes and material types; and (c) defining and applying truck loadings. The pre-processor was successfully used to create input files in a format acceptable to the finite element analysis program ANSYS.

The pre-processor is but one component of a long-term research effort to formulate design methodologies and write computer software tools that fully exploit the capabilities of engineering workstations. Several extensions to the work reported in this paper are currently underway. Phase 1 of software development for a finite element package written exclusively in C is near completion (Austin et al. 1992). Version 2 of the pre-processor, now in preparation, has facilities for manipulating and editing a finite element mesh viewed from an arbitrary eye position (Hudson et al. 1992). The writer's plan is to run the pre- and post-processor, and finite element program on separate engineering workstations. Interprocess Communication (IPC) facilities (Joy et al. 1983) will be used as a basis for developing a message protocol between the pre- and post-processors, and the finite element program. When completed, it is anticipated that designers will interact at the pre- and post-processor graphical interface, and build and send messages to the finite element program, requesting summaries of response information as needed for graphical and textual display.

ACKNOWLEDGEMENTS

The writers would like to acknowledge helpful discussions with Dr. P. Van Der Walt and Prof. A.C. Scordelis during the early stages of the project. This work was supported in part by a FHWA Fellowship to the second writer, NSF's Engineering Research Centers Program : NSFD CDR 8803012, and NSF Research Initiation Grant NSF BCS 8907722. Their support is gratefully acknowledged. The views expressed in this paper are those of the authors, and not necessarily those of FHWA or NSF.

APPENDIX I. - REFERENCES

[1] Austin, M.A., Mahin, S.A., and Pister, K.S. (1989), "CSTRUCT : Computer Environment for the Design of Steel Structures," *Journal of Computing in Civil Engineering*, ASCE, 209-227.

[2] Austin, M.A., Voon, B.K. and Sondhi, J., (1992), "Data Structures and Algorithms for a C-Based Finite Element Program," *SRC Technical Report*, University of Maryland, College Park, MD 20742, (In Preparation).

[3] Creighton, S., Austin, M.A, and Albrecht, P. (1990), "Pre-Processor for Finite Element Analysis of Highway Bridges," Technical Research Report TR 90-54, Systems Research Center, University of Maryland, College Park, MD 20742.

[4] Cook, R.D., Mallkus, D.S., and Plesha, M.E. (1974), **Concepts and Applications of Finite Element Analysis**, *John Wiley and Sons.*

[5] Elnahal, M.K., Albrecht, P., and Cayes L. (1989), "Load Distribution in a Two-Span Continuous Bridge",Report FHWA-RD-89-101, Federal Highway Administration, Washington D.C.

[6] "GTSTRUDL User's Manual Volume II." (1986), GTICE Systems Laboratory, School of Engineering, Georgia Institute of Technology, Atlanta, Georgia.

[7] Heckel P. (1984), **The Elements of Friendly Software Design**, Warner Books, New York.

[8] Hudson, C.A., and Austin, M.A. (1992), "Finite Element Analysis and Design of Bridges in a Distributed Computing Environment," *8th National Conference on Computing in Civil Engineering*, Dallas, Texas.

[9] Johnson, S.C. (1975),"YACC - Yet another Compiler Compiler," *Computer Science Technical Report 32*, AT&T Bell Laboratories, Murray Hill, New Jersey.

[10] Joy, W., Fabry, R., and Leffler, S. (1983), **A 4.3BSD Interprocess Communication Primer**, Computer Systems Research Group, University of California, Berkeley, CA 94720.

[11] Kernighan, B.W., and Pike, R. (1983), **The UNIX Programming Environment**, Prentice-Hall Software Series.

[12] Kohnke, P.C. (1977), "ANSYS Theoretical Manual," Swanson Analysis Systems, Inc.

[13] McNeal-Schendler Corp. (1989), "MSC NASTRAN Application Manual", Los Angelos, California.

[14] Nutt, R.V., Zokaie, T., and Schamber, R.A. (1987), "Distribution of Wheel Loads on Highway Bridges," NCHRP Report, Transportation Research Board, National Research Council, Washington D.C.

[15] Scheifler, R., and Gettys, J. (1986), "The X Window System," *ACM Transactions on Graphics*, Vol. 2, pp. 79-109.

[16] Scordelis, A.C., Chan, E.C., Ketchum, M.A., and Van Der Walt, P. (1985), "Computer Programs for Prestressed Concrete Box Girder Bridges," Report No. UCB/SESM 85-02, Department of Structural Engineering and Structural Mechanics, University of California, Berkeley, CA 94720.

[17] Scordelis, A.C. (1986), "State-of-the-Art: Bridge Analysis (Statics) : Analysis of bridges - Past, Present and Future," Proceedings of a Workshop on Research Needs for Short and Medium Span Bridges, Computech Engineering Services, Sponsored by the National Science Foundation, Washington, DC.

[18] "Standard Specifications for Highway Bridges." (1983), 13th Edition, American Association of State Highway and Transportation Officials, Washington, D.C.

[19] "SunView 1 Programmer's Guide." (1988), Sun Microsystems, Copyright 1982..1988, Sun Microsystems, Inc.

[20] William, K.J., and Scordelis, A.C. (1970), "Computer Program for Cellular Structures of Arbitrary Plan Geometry," Structural Engineering and Structural Mechanics Report No. UC-SESM 70-10, University of California, Berkeley.

[21] Wilson, E.L., and Habibullah, A. (1980), "SAP 80 : Structural Analysis Program," *Users Guide*, Computers and Structures Inc, University Avenue, Berkeley, California.