6-1-1999

Development of Control Hardware and Software for a Large-Scale Ship Hull and Deck Structure Test System

Megan Stefens

Stephen P. Pessiki

Richard Sause

DEVLOPMENT OF CONTROL HARDWARE AND SOFTWARE FOR A LARGE-SCALE SHIP HULL AND DECK STRUCTURE TEST SYSTEM

by

Megan Stefens

Stephen Pessiki

Richard Sause

ATLSS Report No. 99-05

June 1999

ATLSS is a National Center for Engineering Research on Advanced Technology for Large Structural Systems

117 ATLSS Drive
Bethlehem, PA 18015-4729

Phone: (610)758-3525
Fax: (610)758-5553

www.lehigh.edu/~inatl/inatl.html
Email: inatl@lehigh.edu

DEVLOPMENT OF CONTROL HARDWARE AND SOFTWARE FOR A LARGE-SCALE SHIP HULL AND DECK STRUCTURE TEST SYSTEM

by

Megan Stefens
Graduate Research Assistant
Civil and Evironmental Engineering

Stephen Pessiki
Associate Professor
Civil and Environmental Engineering

Richard Sause
Associate Professor
Civil and Environmental Engineering

ATLSS Report No. 99-05

June 1999

ATLSS is a National Center for Engineering Research on Advanced Technology for Large Structural Systems

117 ATLSS Drive
Bethlehem, PA 18015-4729

Phone: (610)758-3525
Fax: (610)758-5553

www.lehigh.edu/~inatl/inatl.html
Email: inatl@lehigh.edu

ACKNOWLEDGMENTS

This report is based upon work supported by the Naval Surface Warfare Center, Carderock Division under Contract No. N00167-97-K-0058. Additional support was provided by the Center for Advanced Technology for Large Structural Systems. Any opinions, findings and conclusions or recommendations expressed in this report are those of the authors and do not necessarily reflect the views of the Naval Surface Warfare Center, Carderock Division.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

ABSTRACT

The United States Navy has constructed four half-scale models of a naval surface combatant using fiber reinforced plastics (FRP) for the purpose of investigating the advantages of using fiber reinforced materials over traditional materials for ship construction. Lehigh University is conducting a series of tests on two of these models, which are composite ship hull and deck structures. The objectives of the test program include the execution of low-level tests to determine the stiffness properties of the hull structures and collapse-level tests to determine the strength and failure modes of the hull structures.

The focus of this report is the test system developed to execute the test program. The test system consists of the test fixture (i.e. the physical hardware used to apply forces to the hull structures, and the control system). The control system is the hardware and software used to orchestrate the tasks of load application and data acquisition during the tests.

The development of the control system software (control program) involves an analysis of the kinematics and statics of the test fixture to determine the forces applied to the hull structure. The net axial force and primary and secondary bending moments in the hull structure are critical parameters used in the control software. Specifically, these parameters are used in the portion of the program called the decision algorithm to make decisions regarding how the loading of the hull structure should proceed. These decisions are based on a set of loading objectives, one of which is to maintain a zero net axial force in the hull structure. The control software incorporates closed-loop displacement control of five independently operating actuators into an external loop that allows for user interaction. The user has the ability to change critical program parameters.

Testing to be performed on the hull structures by Lehigh University will determine whether hull structures constructed from fiber reinforced materials provide stiffness and strength properties comparable to those exhibited by their steel counterparts.

CHAPTER 1
INTRODUCTION

1.1 COMPOSITE SHIP HULL TEST PROGRAM

The United States Navy is investigating the use of fiber reinforced plastics (FRP) for construction of its naval surface combatants. Previously, the use of FRP was limited to small craft and naval minehunters. However, due to cost and quality improvements in fabrication of large composite structures, the use of FRP for surface ships has become more feasible.

Several advantages arise from fiber reinforced material construction (Nguyen and Critchfield, 1997). The use of FRP would result in hulls of increased performance due to reduced weight. In addition, the Navy foresees a reduction in life cycle costs because of the material's resistance to fatigue and salt water corrosion.

Four half-scale midship section models of a corvette class surface combatant were fabricated by four different processes in order to assess the applicability of these processes to naval combatant ship construction. The processed used were: an ultra-violet light curing resin with vacuum assisted resin transfer molding (UV-VARTM); a non-vacuum-bag consolidation of UV light curing prepreg system (UV-Prepreg); a non-vacuum-bag, non-autoclave consolidation of low temperature curing prepreg system (LTC-Prepreg); and, a patented vacuum assisted resin infusion process known as SCRIMP™ (Nguyen and Critchfield, 1997).

Lehigh University is presently conducting an experimental program consisting of a series of tests on two of the composite ship hull and deck structures. Testing will be performed on these models for the purpose of comparing them to their steel counterparts. The testing will include low-level tests to determine the stiffness properties of the hull structures, and collapse-level tests to determine the strength and failure modes of the hull structures. The two composite ship hull and deck structures to be tested by Lehigh University are shown in Figure 1.1. Each hull structure is approximately 26 feet long, 20 feet wide, 10 feet high and weighs about 20 kips. Figure 1.1(a) shows the LTC-Prepreg hull structure and Figure 1.1(b) is the hull structure fabricated by the UV-Prepreg process.

1.2 OBJECTIVES

The test program being performed by Lehigh University has the following objectives (Pessiki and Sause, 1997):

1. To design, fabricate, and assemble a test fixture for testing large-scale models of hull structures.
2. To conduct low-level load tests of the hull structures in order to determine their elastic flexibility in primary bending.
3. To conduct collapse-level tests of the hull structures in order to determine their ultimate strength and failure mode in primary bending.

The chief loading objective is to apply a primary sagging moment while maintaining zero net axial force in the hull structure. Under the action of a sagging moment, the deck is in compression and the keel is in tension. The LTC-Prepreg hull structure will be used as a calibration specimen to ensure that the test fixture and test control system can successfully accomplish the loading objectives.

1.3 OUTLINE OF REPORT

Figure 1.2 is a schematic drawing of the *test system*. The *test system* is comprised of the *test fixture* and the *control system*. The *test fixture* refers to the physical hardware used to provide forces and reactions to the hull structures. The *control system* refers to the hardware and software used to orchestrate the tasks of load application and data acquisition during the tests.

The focus of the research presented in this report is the development of the control system software (control program) and the coordination of the control system hardware, which together provide test control for the low-level and collapse-level tests. The control program, written in the BASIC computer language, incorporates closed-loop displacement control of five actuators into an external loop that allows for user interaction.

Chapter 2 describes the test system. This includes brief physical and functional descriptions of the test fixture and control system hardware and software. Chapter 3 presents the theory developed for the control program's decision-making algorithms. Chapter 4 describes in detail the hardware and software that make up the control system.

1.4 NOTATION

The following is a list of the notation used in this report. Note that vectors are written with bold text.

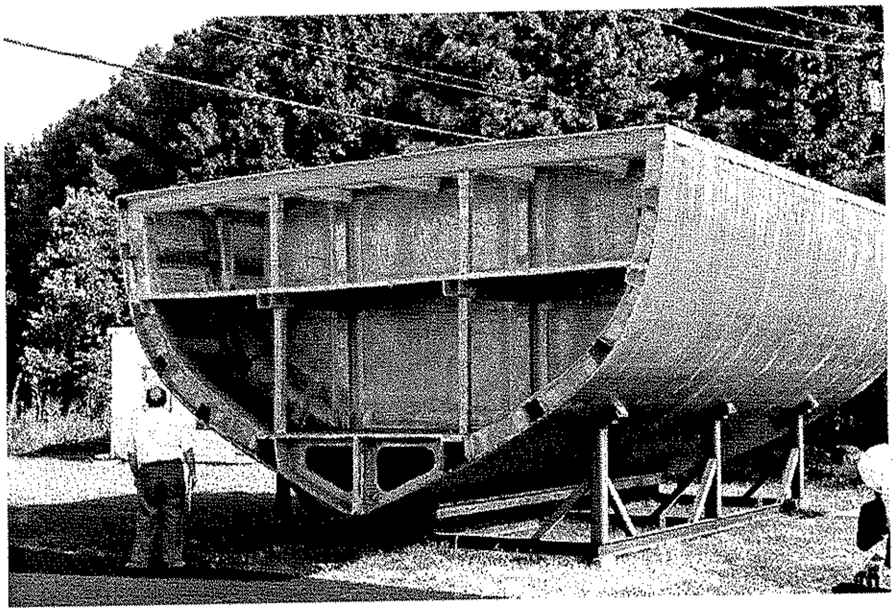
$\mathbf{A}$	=	actuator position vector
a_X	=	**i** component of **A**; the length of actuator or bottom link in X direction
a_Y	=	**j** component of **A**; the length of actuator or bottom link in Y direction
a_Z	=	**k** component of **A**; the length of actuator or bottom link in Z direction
$\mathbf{C'_N}$	=	vector from point r'_N to the north end of an actuator
c_1	=	**i** component of $\mathbf{C'_N}$; the length of actuator or bottom link in X direction
c_2	=	**j** component of $\mathbf{C'_N}$; the length of actuator or bottom link in Y direction
c_3	=	**k** component of $\mathbf{C'_N}$; the length of actuator or bottom link in Z direction
$\mathbf{C'_S}$	=	vector from point r'_S to the south end of an actuator
$\mathbf{D'_N}$	=	vector from point r'_N to bottom of vertical link
d_1	=	**i** component of $\mathbf{D'_N}$; the length of vertical link in the X direction
d_2	=	**j** component of $\mathbf{D'_N}$; the length of vertical link in the Y direction
d_3	=	**k** component of $\mathbf{D'_N}$; the length of vertical link in the Z direction
$\mathbf{E_N}$	=	vector from point r to the north end of a top actuator or bottom link
$\mathbf{E_{No}}$	=	vector defining original coordinates of the north end of a bottom actuator
$\mathbf{E_S}$	=	vector from point r to the south end of a top or bottom actuator
$\mathbf{E_{No}}$	=	vector defining original coordinates of the north end of a bottom actuator
e_1	=	**i** component of $\mathbf{E_{No}}$; X distance from r to north end of a bottom actuator

e_2 = **j** component of $\mathbf{E_{No}}$; Y distance from r to north end of a bottom actuator
e_3 = **k** component of $\mathbf{E_{No}}$; Z distance from r to north end of a bottom actuator
e_4 = **i** component of $\mathbf{E_{No}}$; X distance from r to north end of a bottom actuator
e_5 = **j** component of $\mathbf{E_{No}}$; Y distance from r to north end of a bottom actuator
e_6 = **k** component of $\mathbf{E_{No}}$; Z distance from r to north end of a bottom actuator
$(F_{BOT})_N$ = sum of the X direction forces in the bottom links
$(F_{BOT})_S$ = sum of the X direction forces in the bottom actuators
$(F_{TOP})_N$ = sum of the X direction forces in the top actuators at north grillage
$(F_{TOP})_S$ = sum of the X direction forces in the top actuators at south grillage
$(F_{TE})_N$ = total force in top east actuator at the north grillage
$(F_{TE})_S$ = total force in top east actuator at the south grillage
$(F_{TW})_N$ = total force in top west actuator at the north grillage
$(F_{TW})_S$ = total force in top west actuator at the south grillage
$(F_X)_{BNE}$ = X direction force in bottom north east horizontal link
$(F_X)_{BNW}$ = X direction force in bottom north west horizontal link
$(F_X)_{BSE}$ = X direction force in bottom south east actuator
$(F_X)_{BSW}$ = X direction force in bottom south west actuator
F_{XN} = X direction force in any actuator or bottom link at north grillage
$(F_X)_{NE}$ = X direction force in north east vertical link
$(F_X)_{NW}$ = X direction force in north west vertical link
$(F_{XN})_{TC}$ = X direction force in top center actuator at north grillage
$(F_{XN})_{TE}$ = X direction force in top east actuator at north grillage
$(F_{XN})_{TW}$ = X direction force in top west actuator at north grillage
F_{XS} = X direction force in any top or bottom actuator at south grillage
$(F_X)_{SE}$ = X direction force in south east vertical link
$(F_X)_{SW}$ = X direction force in south west vertical link
$(F_{XS})_{TC}$ = X direction force in top center actuator at south grillage
$(F_{XS})_{TE}$ = X direction force in top east actuator at south grillage
$(F_{XS})_{TW}$ = X direction force in top west actuator at south grillage
$(F_Y)_{BNE}$ = Y direction force in bottom north east horizontal link
$(F_Y)_{BNW}$ = Y direction force in bottom north west horizontal link
$(F_Y)_{BSE}$ = Y direction force in bottom south east actuator
$(F_Y)_{BSW}$ = Y direction force in bottom south west actuator
F_{YN} = Y direction force in any actuator or bottom link at north grillage
$(F_Y)_{NE}$ = X direction force in north east vertical link
$(F_Y)_{NW}$ = X direction force in north west vertical link
$(F_{YN})_{TC}$ = Y direction force in top center actuator at north grillage
$(F_{YN})_{TE}$ = Y direction force in top east actuator at north grillage
$(F_{YN})_{TW}$ = Y direction force in top west actuator at north grillage
F_{YS} = Y direction force in any top or bottom actuator at south grillage
$(F_Y)_{SE}$ = Y direction force in south east vertical link
$(F_Y)_{SW}$ = Y direction force in south west vertical link
$(F_{YS})_{TC}$ = Y direction force in top center actuator at south grillage
$(F_{YS})_{TE}$ = Y direction force in top east actuator at south grillage
$(F_{YS})_{TW}$ = Y direction force in top west actuator at south grillage

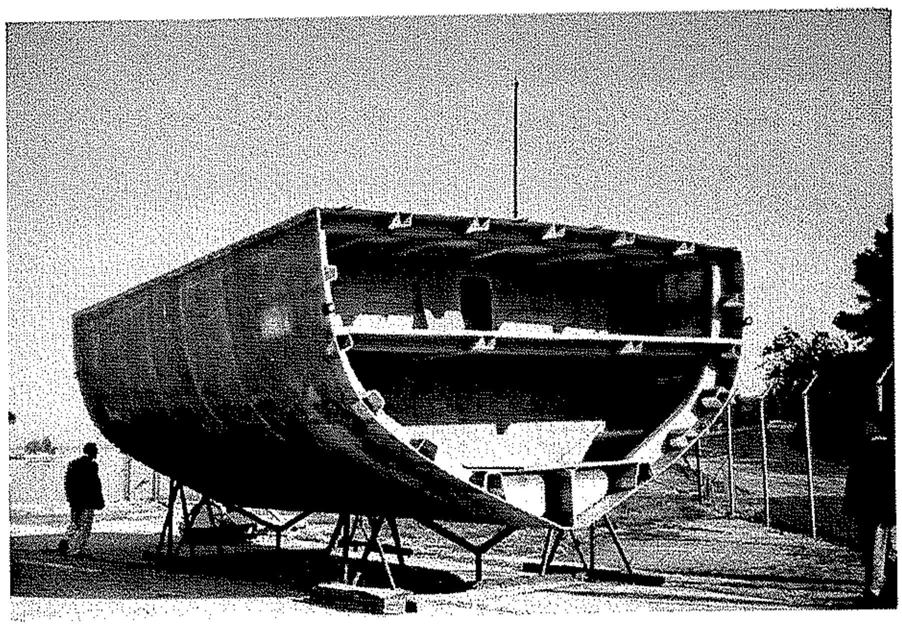
F_{ZN} = Z direction force in any actuator or bottom link at north grillage
F_{ZS} = Z direction force in any top or bottom actuator at south grillage
$(F_Z)_{BNE}$ = Z direction force in bottom north east horizontal link
$(F_Z)_{BNW}$ = Z direction force in bottom north west horizontal link
$(F_Z)_{BSE}$ = Z direction force in bottom south east actuator
$(F_Z)_{BSW}$ = Z direction force in bottom south west actuator
$(F_{ZN})_{TC}$ = Z direction force in top center actuator at north grillage
$(F_{ZN})_{TE}$ = Z direction force in top east actuator at north grillage
$(F_{ZN})_{TW}$ = Z direction force in top west actuator at north grillage
$(F_{ZS})_{TC}$ = Z direction force in top center actuator at south grillage
$(F_{ZS})_{TE}$ = Z direction force in top east actuator at south grillage
$(F_{ZS})_{TW}$ = Z direction force in top west actuator at south grillage
i, j, k = basis vectors for X, Y, Z global coordinate system
$\mathbf{i'}_N, \mathbf{j'}_N, \mathbf{k'}_N$ = basis vectors for x'_N, y'_N, z'_N local coordinate system at north grillage
$\mathbf{i'}_S, \mathbf{j'}_S, \mathbf{k'}_S$ = basis vectors for x'_S, y'_S, z'_S local coordinate system at south grillage
l_X = **i** component of **L**; the length of vertical link in the X direction
l_Y = **j** component of **L**; the length of vertical link in the Y direction
l_Z = **k** component of **L**; the length of vertical link in the Z direction
L = vertical link position vector
L1 = vector from point r to bottom of vertical link
L2 = vector from point r to top of vertical link
$L2_X$ = **i** component of **L2**; X distance from r to top of vertical link
$L2_Y$ = **j** component of **L2**; Y distance from r to top of vertical link
$L2_Y$ = **k** component of **L2**; Z distance from r to top of vertical link
M_{YN} = moment about Y axis at north grillage
M_{YS} = moment about Y axis at south grillage
M_{ZN} = moment about Z axis at north grillage
M_{ZS} = moment about Z axis at south grillage
$(M_\omega)_{MAX}$ = maximum moment due to self weight of hull structure and grillages
P = axial force in hull structure; average of P_N and P_S
P_N = axial force in hull structure measured at the north grillage
P_S = axial force in hull structure measured at the south grillage
P1, P2, P3 = three points whose coordinates define a plane
$P1_NX$ = X coordinate of point $P1_N$
$P2_NX$ = X coordinate of point $P2_N$
$P3_NX$ = X coordinate of point $P3_N$
$P1_NY$ = Y coordinate of point $P1_N$
$P3_NY$ = Y coordinate of point $P3_N$
$P1_SX$ = X coordinate of point $P1_S$
$P2_SX$ = X coordinate of point $P2_S$
$P3_SX$ = X coordinate of point $P3_S$
$P1_SY$ = Y coordinate of point $P1_S$
$P3_SY$ = Y coordinate of point $P3_S$
P1X = X coordinate of point P1
P2X = X coordinate of point P2

P3X = X coordinate of point P3
$P1X_o$ = original X coordinate of point P1
$P2X_o$ = original X coordinate of point P2
$P3X_o$ = original X coordinate of point P3
P1Y = Y coordinate of point P1
P2Y = Y coordinate of point P2
P3Y = Y coordinate of point P3
$P1Y_o$ = original Y coordinate of point P1
$P2Y_o$ = original Y coordinate of point P2
$P3Y_o$ = original Y coordinate of point P3
P1Z = Z coordinate of point P1
P2Z = Z coordinate of point P2
P3Z = Z coordinate of point P3
$P1Z_o$ = original Z coordinate of point P1
$P2Z_o$ = original Z coordinate of point P2
$P3Z_o$ = original Z coordinate of point P3
r = origin of global coordinate system
r'_N = origin of north grillage local coordinate system
$\mathbf{R_N}$ = vector from point r to point r'_N
r'_N = origin of south grillage local coordinate system
$\mathbf{R_S}$ = vector from point r to point r'_S
t_{ij} = entry i,j of transformation matrix $\mathbf{T_N}$ or $\mathbf{T_S}$
$\mathbf{T_N}$ = transformation matrix for north grillage
$\mathbf{T_S}$ = transformation matrix for south grillage
$\mathbf{u'_N}$ = vector normal to the plane of the north grillage
$\mathbf{u'_S}$ = vector normal to the plane of the south grillage
$\mathbf{v'_N}$ = vector from point $P1_N$ to $P2_N$
$\mathbf{v'_S}$ = vector from point $P1_S$ to $P2_S$
$\mathbf{w'_N}$ = vector from point $P2_N$ to $P3_N$
$\mathbf{w'_S}$ = vector from point $P2_S$ to $P3_S$
X_C = X translation of point C at a given loading step
$X1_N$ = transducer which measures $\Delta P1_N X$
$X2_N$ = transducer which measures $\Delta P2_N X$
$X3_N$ = transducer which measures $\Delta P3_N X$
x'_N, y'_N, z'_N = north grillage local coordinate system axes
$X1_S$ = transducer which measures $\Delta P1_S X$
$X2_S$ = transducer which measures $\Delta P2_S X$
$X3_S$ = transducer which measures $\Delta P3_S X$
x'_S, y'_S, z'_S = south grillage local coordinate system axes
$Y1_N$ = transducer which measures $\Delta P1_N Y$
$Y3_N$ = transducer which measures $\Delta P3_N Y$
$Y1_S$ = transducer which measures $\Delta P1_S Y$
$Y3_S$ = transducer which measures $\Delta P3_S Y$

β	=	angle that X direction grillage transducer makes with the horizontal
$\Delta P1X$	=	change in X direction coordinate of point P1
$\Delta P2X$	=	change in X direction coordinate of point P2
$\Delta P3X$	=	change in X direction coordinate of point P3
$\Delta P1Y$	=	change in Y direction coordinate of point P1
$\Delta P2Y$	=	change in Y direction coordinate of point P2
$\Delta P3Y$	=	change in Y direction coordinate of point P3
$\Delta P1Z$	=	change in Z direction coordinate of point P1
$\Delta P2Z$	=	change in Z direction coordinate of point P2
$\Delta P3Z$	=	change in Z direction coordinate of point P3
θ_{YN}	=	rotation about Y axis at north grillage
θ_{YS}	=	rotation about Y axis at south grillage
θ_{ZN}	=	rotation about Z axis at north grillage
θ_{ZS}	=	rotation about Z axis at south grillage
ω	=	combined distributed weight of grillages and hull structure



(a) LTC-Prepreg hull structure



(b) UV-Prepreg hull structure

Figure 1.1 Photographs of hull structures to be tested by Lehigh University

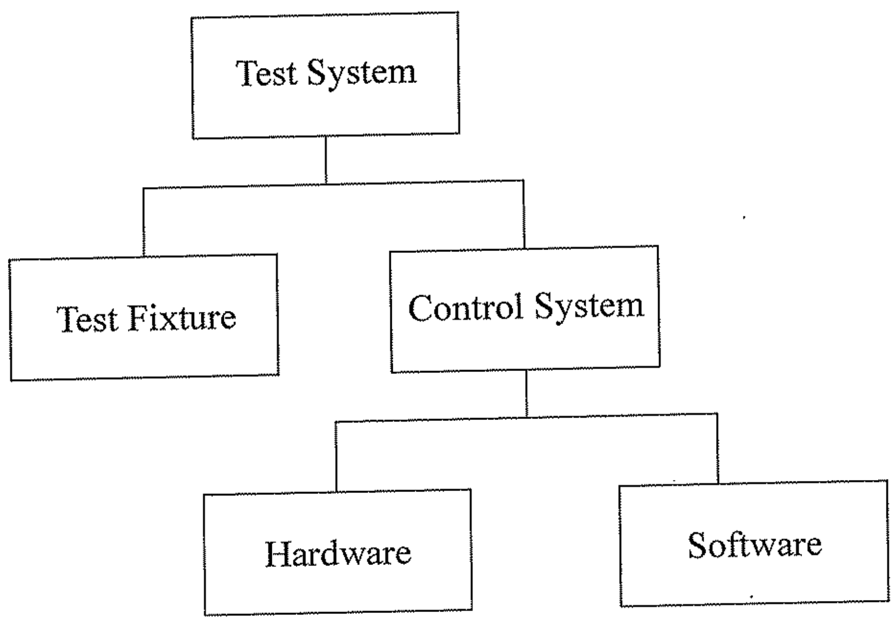


Figure 1.2 Schematic of test system

CHAPTER 2
TEST SYSTEM

2.1 INTRODUCTION

This chapter presents an overview of the test system. Included is a description of the test fixture and the notation associated with its components. Also described here are the loading objectives and an explanation of how the configuration of the test fixture accomplishes those objectives.

2.2 SCHEMATIC AND NOTATION

A schematic drawing of the elevation view of the test fixture is shown in Figure 2.1. The hull structure is connected to a rigid steel grillage at each end through an attachment fixture (not shown). Each grillage is comprised of 11 vertical W40×249 steel sections. Seven W sections are positioned vertically in one plane, and four W sections positioned horizontally in another plane are connected to the vertical sections.

Each grillage is suspended by two vertical pinned links from an overhead frame (not shown). The grillages are linked together by a series of actuators and links. Above the hull structure, three 600 kip capacity actuators span the length of the hull structure in parallel. Below the hull structure, two 1000 kip capacity actuators are attached to the south grillage. These actuators are in turn attached to two bottom links that are restrained against any movement except translation in the X direction. A second set of bottom links connects the center links to the north grillage. The overhead frame reacts against the strong wall at the south end of the test fixture and is supported by four columns attached to the strong floor.

All actuators and links are given names which refer to their relative locations in space. The notation is illustrated in Figure 2.2. The top actuators are called TE, TC and TW, for Top East, Top Center, and Top West, respectively. The bottom actuators are BSW, or Bottom South West, and BSE, or Bottom South East. The bottom links attached to the north grillage are called BNE and BNW, while the second set of bottom links are BCE and BCW. The vertical links are named NE and NW at the north grillage, and SE and SW at the south grillage. These acronyms are used as subscripts for variables. For example, the total force in actuator BSE is written $(F)_{BSE}$.

The clevises on the actuators and links employ spherical bearings which are intended to allow free rotation about the Z axis. However, these bearings also allow a limited range of free rotation about the Y axis. Figure 2.3(a) is a photograph of a clevise and bearing, showing the primary (Z) and secondary (Y) axes of rotation. Figure 2.3(b) is a view of the clevise which shows the limited amount of secondary rotation permitted. Note that when the actuator is in position in the test fixture, the clevise will be rotated 90 degrees about the primary axis of rotation from the position shown in the figure. In Figures 2.1 and 2.2 and in following figures, a circle is used to represent the clevise in the XY plane, where rotation occurs about the Z axis (i.e. primary axis of rotation). A diamond is used to represent the clevise in the XZ plane, where rotation occurs about the Y axis (i.e. secondary axis of rotation).

2.3 SPECIMEN LOADING

2.3.1 Objectives

The main loading objective is to apply primary bending moment M_Z to the hull structure. If the top actuators act in tension and the bottom actuators act in compression, the result is a primary sagging moment M_Z causing compression in the deck and tension in the keel. In order to maintain a constant applied moment throughout the hull structure, it is required that the shear in the Y direction be equal to zero.

Another objective is to keep the axial force applied to the hull structure approximately equal to zero. Roughly speaking, if the total force in the top actuators is equal and opposite to the total force in the bottom actuators, then no net axial force is applied to the hull structure. As a result, the elastic neutral axis remains at the centroid of the cross-section.

In order to accomplish these loading objectives, it is required that when the top actuators are to be moved (extended or retracted), they must all be moved the same amount. Likewise, when the bottom actuators are to be moved, they must both be moved the same amount. The reasons for this requirement will be explained later.

2.3.2 System Statics

A simplified explanation of the statics of the loading system is presented here. Consider that the top actuators act in tension and the bottom actuators act in compression. The north grillage rotates through some angle θ_{ZN} about the Z axis, the south grillage rotates through an angle θ_{ZS}, and the hull structure is subjected to a positive bending moment M_Z. As shown in Figure 2.4(a), all actuators and links are free to rotate in the X-Y plane. The resultant forces are shown in Figure 2.4(b). As a result, the actuator and link forces can have X and Y components which are illustrated in the free body diagrams of Figure 2.4(c).

Now consider that a bending moment occurs about the Y axis as in Figure 2.5(a). The reason for the occurrence of M_Y will be discussed later. Figures 2.5(b) and (c) show that in this situation, a Z component of force can develop as well in the top actuators. A similar situation occurs in the bottom actuators as well. Therefore, if the hull structure is subjected to both M_Z and M_Y, each actuator and link may have X, Y and Z components of force. Note that the forces on the hull structure, which are required for equilibrium, are not shown in Figures 2.4 and 2.5.

Figure 2.6 shows a free body diagram of the test fixture and hull structure, with the top actuators acting in tension and the bottom actuators acting in compression. $(F_{TOP})_N$ is the total X-direction force exerted on the north grillage by the top actuators. $(F_{TOP})_S$, $(F_{BOT})_N$, and $(F_{BOT})_S$ are similar X direction forces. Clearly, $(F_{TOP})_N$ and $(F_{TOP})_S$ are always equal and opposite because the top actuators must exert the same force on both the north and south grillages. The arrangement of the bottom links and actuators causes $(F_{BOT})_N = (F_{BOT})_S$. Figure 2.7 illustrates the kinematics of the bottom actuators and links due to extension of the bottom actuators. The four pins are labeled A, B, C, and D. Points A and D are free to move in any direction. Points B and C are restrained against translation in the Y direction by the rollers shown and are restrained against Z translation by rollers in the XZ plane not shown in the figure. The statics of the system are illustrated in Figure 2.8. The free body diagrams show that the X-direction forces in the north links are equal to the X-direction forces in the south actuators and therefore $(F_{BOT})_N = (F_{BOT})_S$.

Although Y and Z components of force may exist, they are typically small compared to the X component. (For the vertical links, the Y component is largest). The reason for this is that the actual expected grillage rotations about the Y and Z axes are very small (less than 5 degrees) and therefore the rotations of the actuators and links will be small. In the simplified explanation of the statics of the system presented here, only the X force component of the actuators and the Y components of the vertical links are considered. As explained above, $(F_{TOP})_N = (F_{TOP})_S$ and $(F_{BOT})_N = (F_{BOT})_S$. Later, in Chapter 3, all components of force in each actuator and link are considered in the explanation of the theory behind the control algorithm.

The chosen actuator configuration allows the loading objectives to be accomplished. Refer to Figure 2.9(a), in which the top actuators act in tension and the bottom actuators act in compression. The combined weight of the hull structure and grillages is distributed along the length L shown and is called ω. F_{YN} is the sum of the forces in the vertical links at the north grillage, while F_{YS} is the sum of the forces in the links at the south grillage. Summing moments about point A, we have

$$(F_{TOP})_N \cdot y_1 - (F_{TOP})_S \cdot y_1 + (F_{BOT})_N \cdot y_2 - (F_{BOT})_S \cdot y_2 + \omega L \cdot (\frac{L}{2}) - F_{YS} \cdot (L) = 0.$$

Because $(F_{TOP})_N$ and $(F_{TOP})_S$ are equal and opposite forces and $(F_{BOT})_N$ and $(F_{BOT})_S$ are likewise equal and opposite, the above equation reduces to

$$F_{YS} = \frac{\omega L}{2}.$$

From summing forces in the Y direction,

$$F_{YN} = \frac{\omega L}{2}.$$

Thus, the shear in the Y direction is due only to the self-weight of the hull structure.

Figure 2.9(b) shows a free body diagram of each grillage and of the hull structure with some assumed forces. The sum of the X direction forces in the top actuators is given as 500 kips and the total X direction force in the bottom actuators and links is 400 kips. It is noted that this state of forces does not satisfy the condition of zero net axial force in the hull structure. The shear and moment diagrams that result from this loading configuration are given in Figure 2.10. The shear and moment diagrams due to self-weight are shown in Figures 2.10(a) and (b). As illustrated in the free body diagrams of Figure 2.9(b), the applied moment about the Z axis at the north grillage, M_{ZN}, is equal to the applied moment at the south grillage, M_{ZS}, because there is no shear in the Y direction due to loading. Thus, the applied moment diagram is as shown in Figure 2.10(c). Then the total moment diagram, given in Figure 2.10(d), is the superposition of the applied moment (Figure 2.10(c)) and self-weight moment diagrams (Figures 2.10(b)). However, if the maximum moment due to self-weight $(M_{\omega})_{MAX}$ is small compared to the applied moment, then the actual moment diagram can be approximated by Figure 2.10(c). Therefore, neglecting self-weight, we can say that the shear in the Y direction is zero and M_Z is constant. Zero shear and constant moment results from the condition that $(F_{TOP})_N = (F_{TOP})_S$ and $(F_{BOT})_N = (F_{BOT})_S$. This condition is achieved by the actuator configuration even for the example presented here where a net axial force exists in the hull structure. The same situation occurs if the net axial force is zero.

2.3.3 Biaxial Bending

Although the chosen actuator configuration facilitates constant moment M_Z throughout the hull structure, it also permits secondary bending moment M_Y to occur. Consequently, the hull structure may be subjected to biaxial bending. M_Y results when the forces in the top actuators are not equal to each other and/or when the forces in the bottom actuators are not equal to each other. This secondary moment exists without shear in the Z direction, i.e. is constant along the length of the hull structure. To illustrate this, Figure 2.11(a) is a plan view (or X-Z plane) of the test structure with only top actuator forces shown. V_{ZN} and V_{ZS} are the shear forces in the Z direction at the north and south grillages, respectively, the existence of which we are investigating. Summing moments about point A,

$$(F_{TE})_N \cdot z_1 - (F_{TE})_S \cdot z_1 + (F_{TW})_N \cdot z_2 - (F_{TE})_S \cdot z_2 - V_{ZS} \cdot L = 0.$$

Since $(F_{TE})_N = (F_{TE})_S$ and $(F_{TW})_N = (F_{TW})_S$, this equation reduces to

$$V_{ZS} = 0.$$

By summing forces in the Z direction, we see that V_{ZN} is also equal to zero. A similar situation results from an analysis of the bottom actuators and links.

Figure 2.11(b) illustrates a case in which the forces in the top actuators are unequal. The free body diagrams show that because the shear in the Z direction is zero and the forces applied to the north and south grillages by the actuators are the same, the moment about Y is constant.

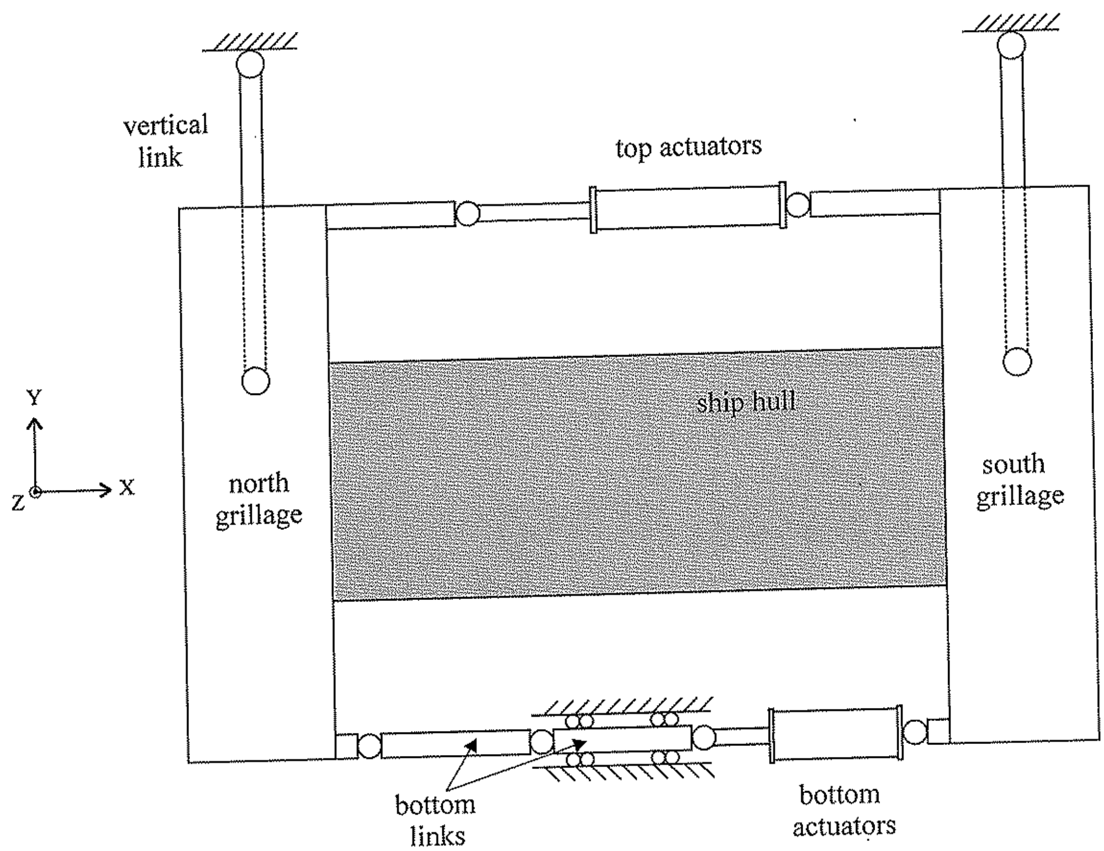


Figure 2.1 Schematic of test setup

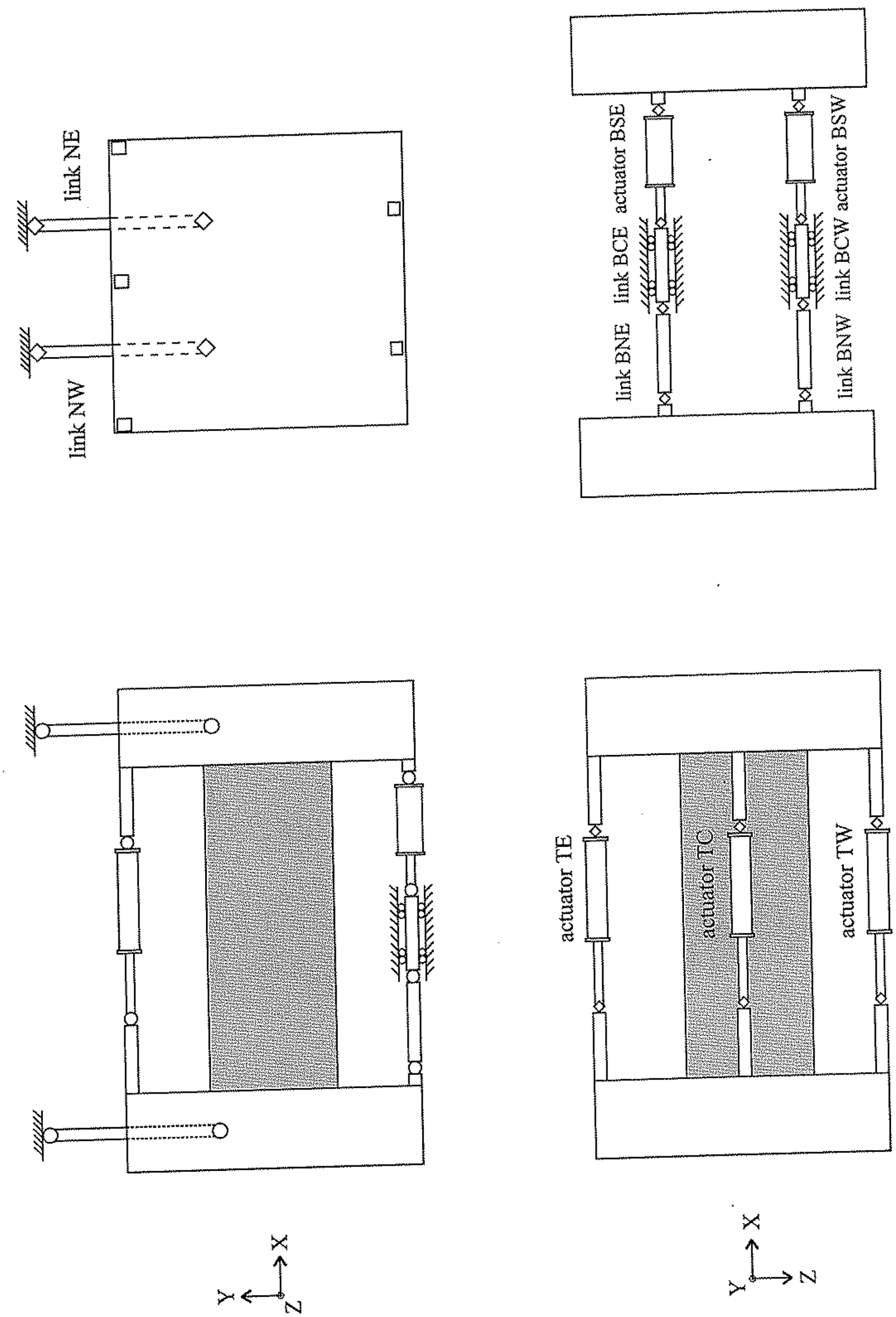
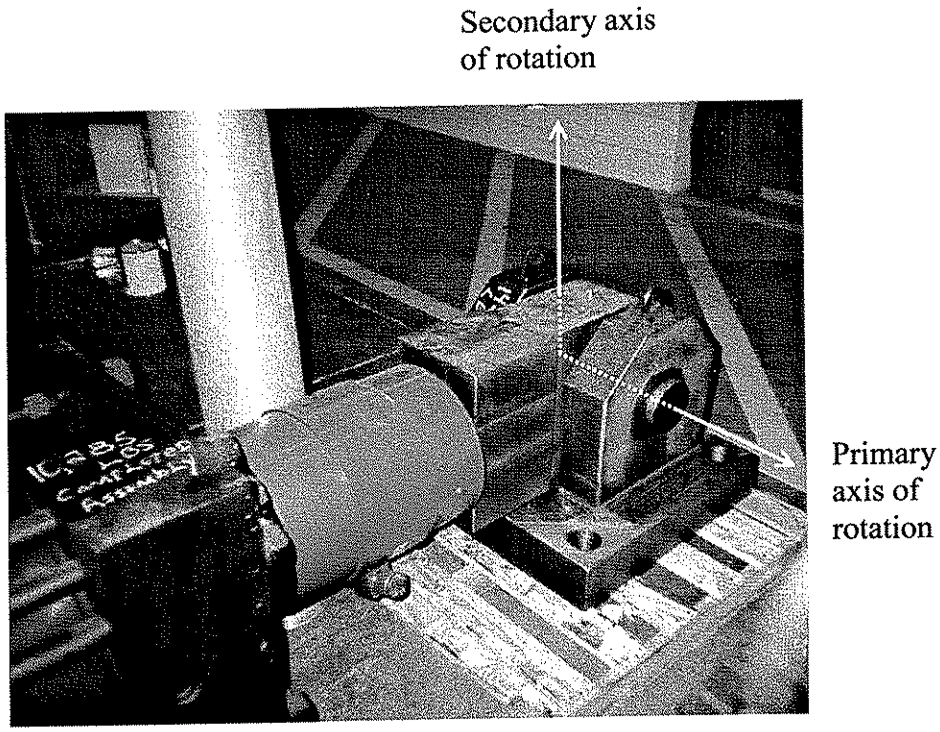
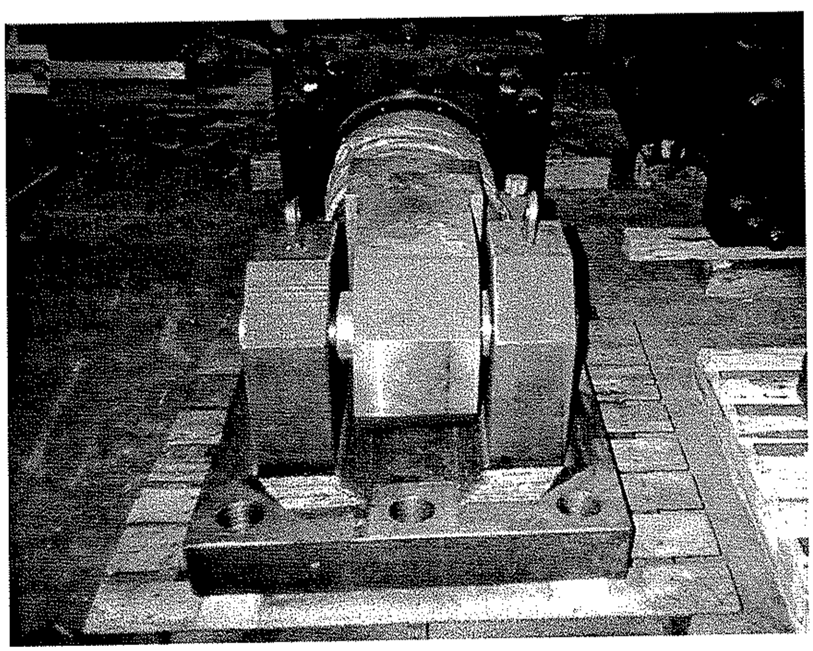


Figure 2.2 Notation

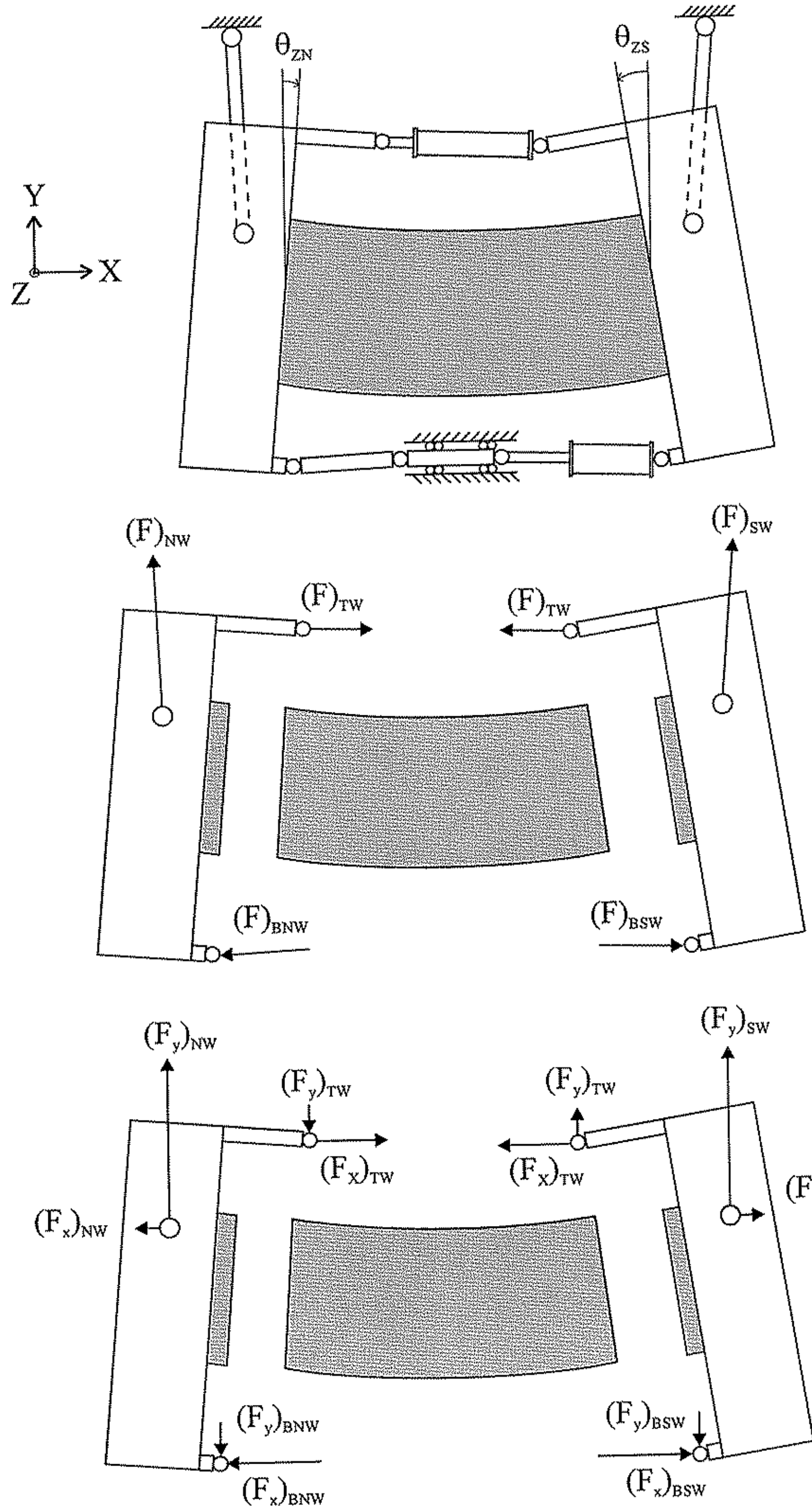


(a) Primary (Z) and secondary (Y) axes of rotation



(b) Limited secondary rotation permitted

Figure 2.3 Actuator clevise and bearing

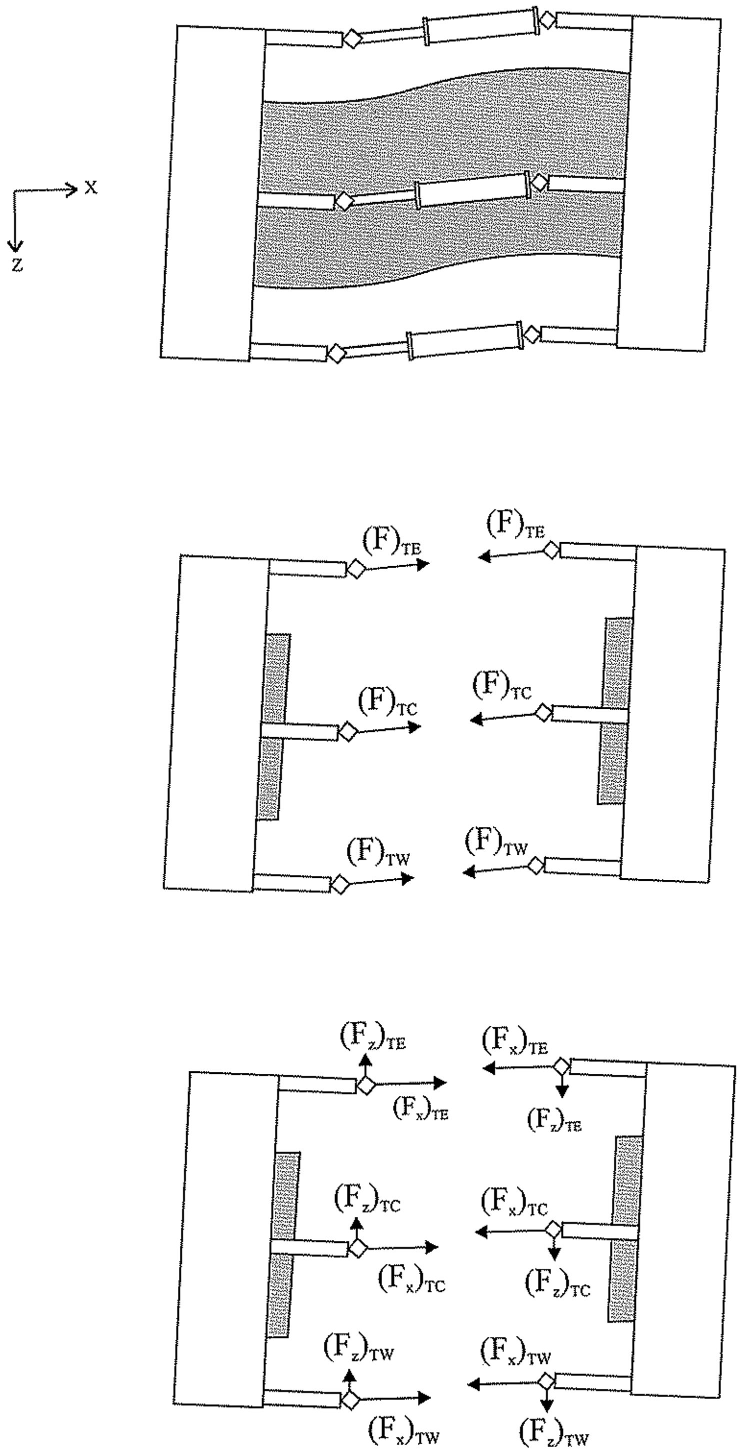


(a) Rotation about Z axis

(b) Actuator force resultants

(c) X and Y force components

Figure 2.4 X and Y actuator and link force components due to rotation about Z axis



(a) Rotation about Y axis

(b) Actuator force resultants

(c) X and Z actuator force components

Figure 2.5 X and Z actuator force components due to rotation about Y axis

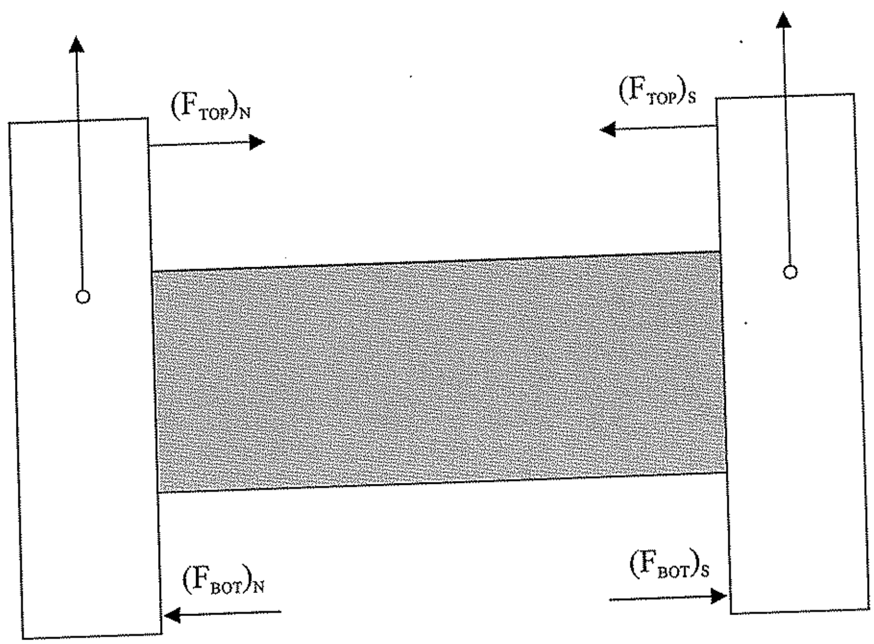


Figure 2.6 Illustration of X direction forces

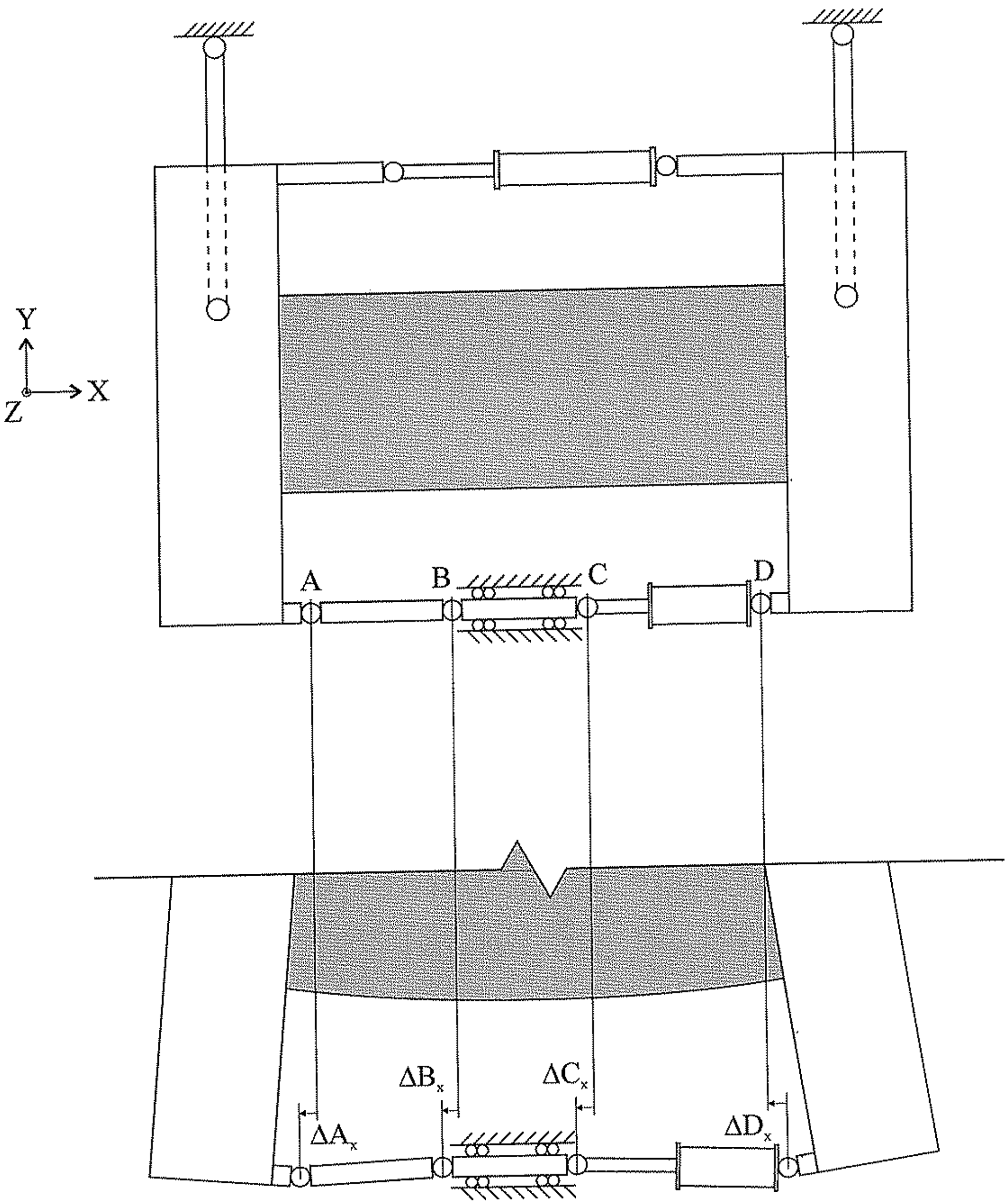
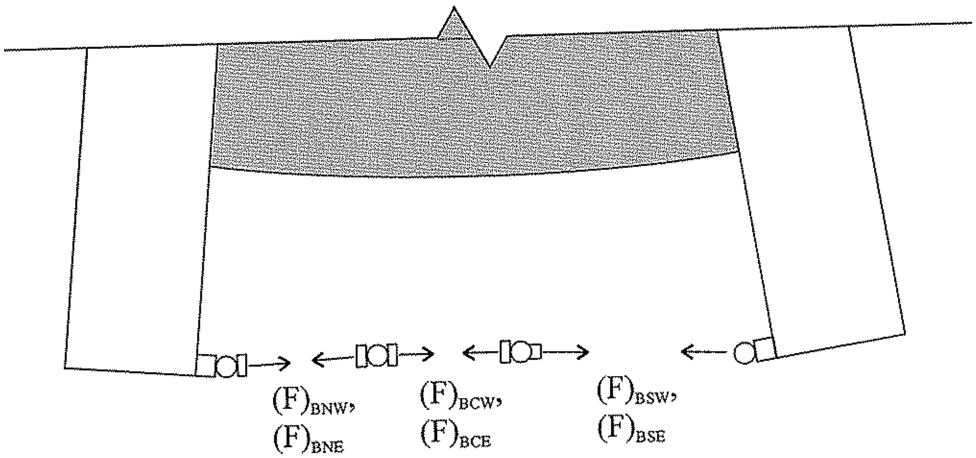


Figure 2.7 Kinematics of bottom actuators and links



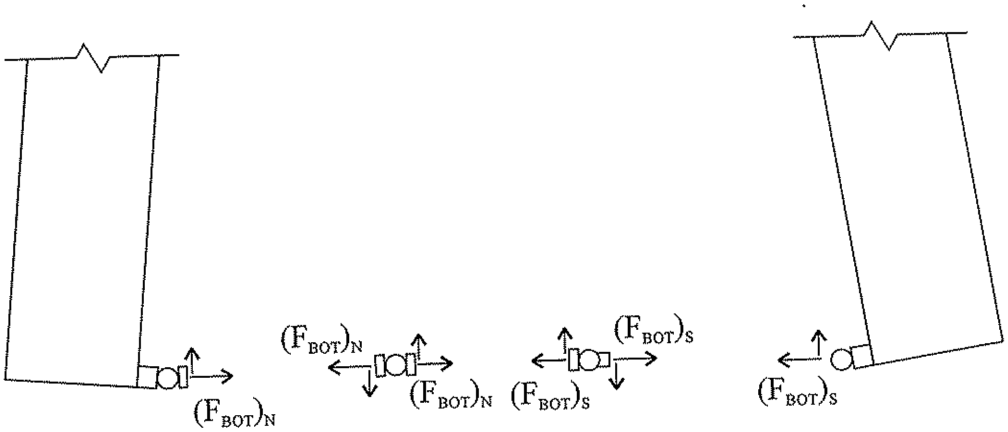


Figure 2.8 Statics of bottom actuators and links

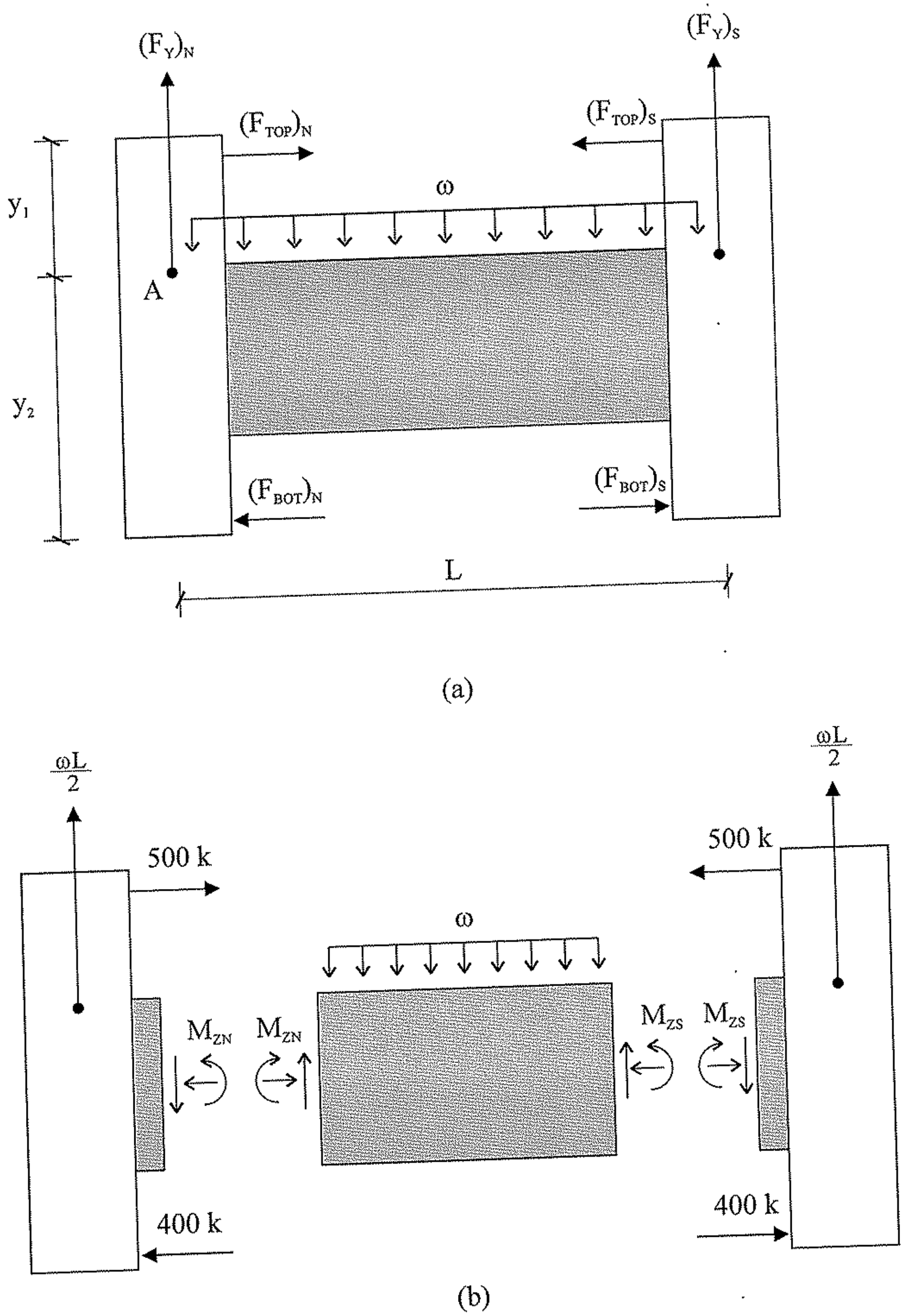


Figure 2.9 XY plane equilibrium

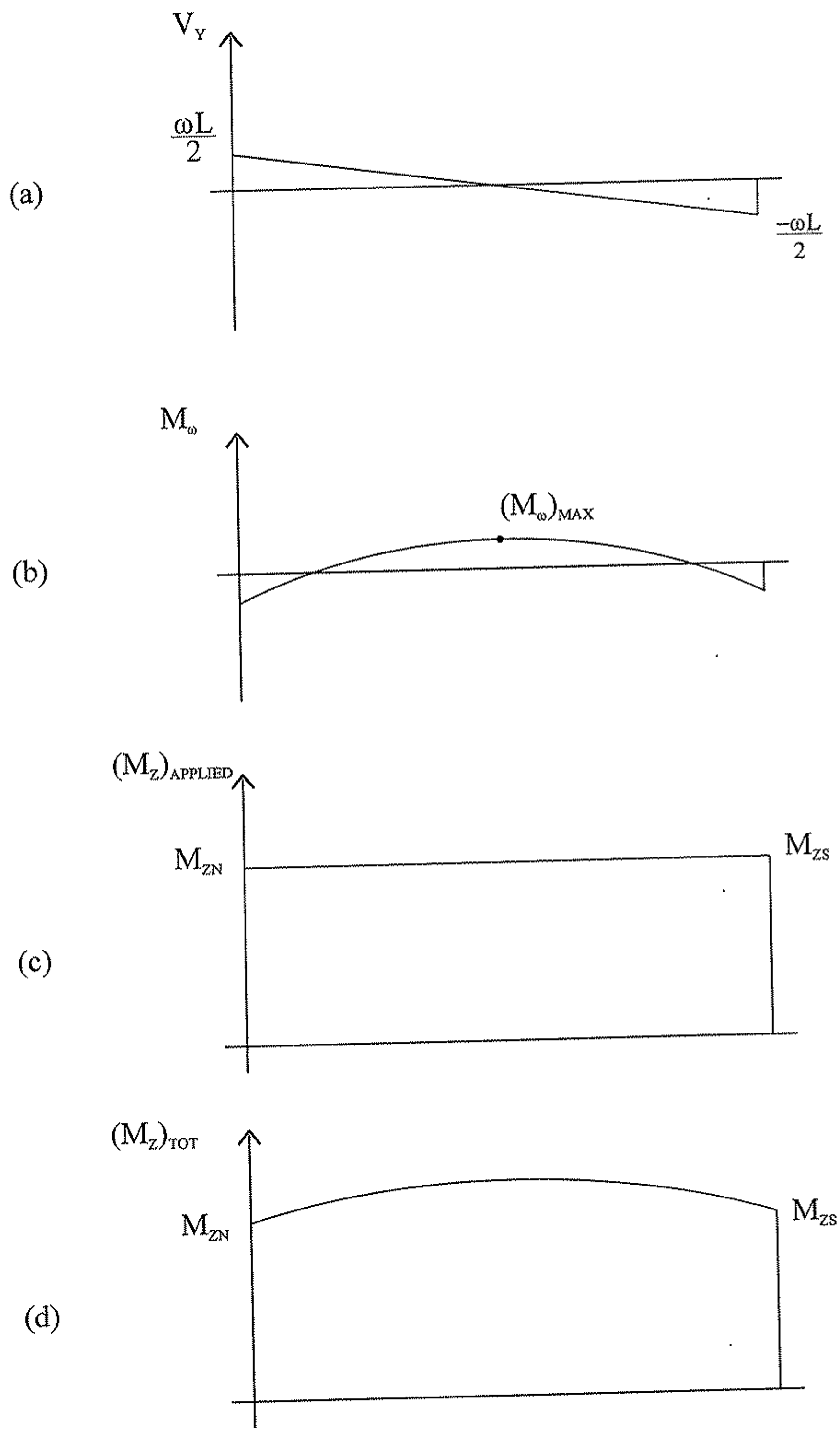
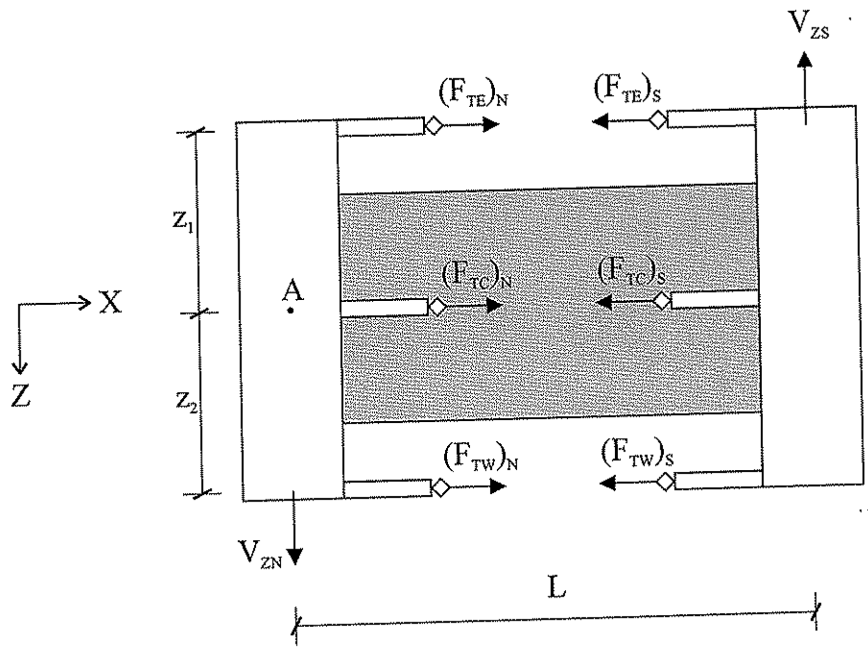
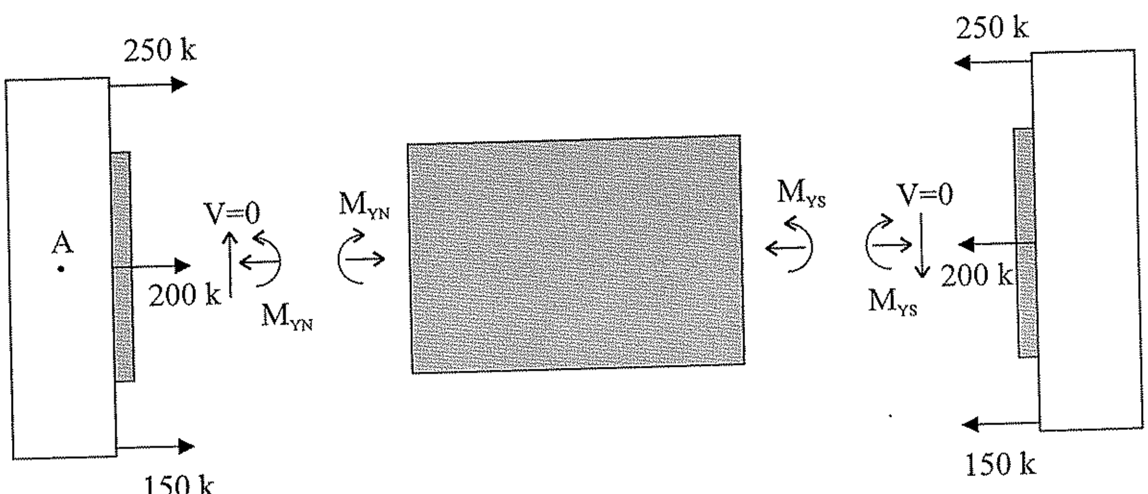


Figure 2.10 XY plane shear and moment diagrams



(a)



(b)

Figure 2.11 XZ plane equilibrium

CHAPTER 3
EXPERIMENTAL CONTROL THEORY

3.1 INTRODUCTION

This chapter presents the concepts that provide the basis for the control program. The main parameters involved in controlling the loading of the hull structure are the applied primary bending moment (M_Z) and the axial force. Control of a test is achieved by determining the magnitude and directions of the force in each actuator and link in the test fixture, and writing the equations of equilibrium to determine the forces applied to the hull structure. If the total tension force applied by the top actuators is larger in magnitude than the total compression force applied by the bottom actuators, then a net compressive force develops in the hull structure, as shown in Figure 3.1(a). However, it is desired that the axial force be equal to zero, as explained in Section 2.3.1. Therefore, to increase the applied moment but reduce the net axial compression force, the bottom actuators are extended until the sum of their forces is equal and opposite to the sum of the forces in the top actuators (Figure 3.1(b)). It is noted here that as the bottom actuators extend, the forces in both the bottom and top actuators are affected.

In order to write the equilibrium equations that determine the axial force and moments applied to the hull structure at any given step during the test, the actuator forces in all directions must be known. As explained in Section 2.3.2 and shown in Figure 2.3, if the north and south grillages have different rotations about the Z axis, the actuators will have both X and Y components of force. Figure 2.4 similarly showed that when the grillages rotate about the Y axis, a Z component of force develops as well.

The total force in each actuator is measured by a force transducer. Determining the components of this force can be accomplished by representing each actuator by a vector **A**, as illustrated in Figure 3.2. Note that these are position vectors, not force vectors, whose coordinates simply represent the length of the actuator in each direction. Given the total force in the actuator and the position vector **A**, simple calculations can then be performed to find the three components of the force.

First, in Section 3.2, an explanation of the grillage kinematics is given. Then, in Section 3.3, the method for determining the actuator vectors and force components is described. From these force components, the equilibrium equations which determine the net axial force and moments in the hull structure are found (Section 3.4). The actuator force components and net axial force and moments in the hull structure are necessary for the control logic algorithm, which is explained in Section 3.5.

3.2 SYSTEM KINEMATICS

3.2.1 Coordinate Systems

The first step in determining the actuator vectors is to establish a global coordinate system with axes X, Y, and Z. The origin r of this global system, which is called the global reference point, is located on a fixed reference plane in space behind the north grillage as shown in Figure 3.3.

Each grillage also has its own local coordinate system. The north grillage has axes labeled x'_N, y'_N, and z'_N and has an origin r'_N, while the south grillage has axes x'_S, y'_S, and z'_S and origin r'_S.

3.2.2 Grillage Kinematics

The next step in determining the actuator vectors is to determine the orientation of each grillage in space with respect to the global reference point. With each loading step, the grillages can change position. The grillages are assumed to be rigid bodies, and as such they each have 6 possible displacements that are defined with respect to their local coordinate axes. The north grillage, for example, has 3 translations, $\Delta x'_N$, $\Delta y'_N$ and $\Delta z'_N$, and 3 rotations, $\theta x'_N$, $\theta y'_N$, and $\theta z'_N$.

Because grillage deformations are neglected, each grillage can be represented as a plane. Three non-collinear points define a plane; therefore, if the location of three points on the grillage are known the orientation of the plane in space is known. Let these points be called P1, P2 and P3. At any point during the test, the coordinates of P1, P2 and P3 with respect to the global origin, r, are given by:

$$\begin{array}{llll} P1: & P1X = P1X_o + \Delta P1X; & P1Y = P1Y_o + \Delta P1Y; & P1Z = P1Z_o + \Delta P1Z \\ P2: & P2X = P2X_o + \Delta P2X; & P2Y = P2Y_o + \Delta P2Y; & P2Z = P2Z_o + \Delta P2Z \\ P3: & P3X = P3X_o + \Delta P3X; & P3Y = P3Y_o + \Delta P3Y; & P3Z = P3Z_o + \Delta P3Z \end{array}$$

where the X_o,Y_o, and Z_o terms are the known original coordinates and the ΔX, ΔY, and ΔZ terms are the unknown changes in position.

As a grillage undergoes some movement during a test, each point has three unknown Δ's, thus nine total measurements are needed to define the new coordinates of P1, P2, and P3. However, as explained below, the grillages are restrained against some movements, reducing the number of measurements needed. Also, the location of points P1, P2, and P3 on the grillage are chosen so that additional measurements are eliminated. Refer to Figure 3.4. Let P2 coincide with r'_N and P1 be collinear with P2 along the y′ axis. Likewise, let P3 be collinear with P2 along the z′ axis. These points define the plane of the grillage. Note that in the following discussion, the grillage displacements are defined in the local coordinate system, but the changes in position of points P1, P2, and P3 due to the grillage displacements are defined in the global system because we are interested in the global coordinates of these points.

Case 1 First consider Case 1, in which a grillage translates in its local x′ direction. This translation, shown in Figure 3.5, occurs due to axial shortening of the hull structure. The x′ translation gives rise to a y′ translation of the grillage because the vertical links are of fixed length and rotate on a radius. As mentioned above, grillage deformations are neglected, so the grillage moves as a rigid body. Therefore, given the movement shown in Figure 3.5,

$$\Delta P1X = \Delta P2X = \Delta P3X$$

and

$$\Delta P1Y = \Delta P2Y = \Delta P3Y.$$

The grillage does not move in its z′ direction, therefore,

$$\Delta P1Z = \Delta P2Z = \Delta P3Z = 0.$$

Case 2   Now consider Case 2, in which a grillage rotates in its x′y′ plane due to a primary bending moment, M_Z, in the hull structure. This rotation is called $\theta z'$, but the rotation may not occur about the actual z′ axis, rather about some axis parallel to z′. In Figure 3.6, the rotation is shown to occur about the point where the vertical links attach to the grillage. The figure shows that as a result of this condition,

$$\Delta P2X = \Delta P3X$$

and

$$\Delta P2Y = \Delta P3Y.$$

In addition, since a rotation about z′ does not cause a displacement of any point in the Z direction,

$$\Delta P1Z = \Delta P2Z = \Delta P3Z = 0.$$

The above equations are true regardless of the location of the axis of rotation.

Case 3   Finally consider Case 3, in which a grillage rotates about its y′ axis due to a secondary bending moment, M_Y, in the hull structure (Figure 3.7(a)). The rotation is shown to occur about the local y′ axis of the grillage, where points P1 and P2 lie. The test fixture is constructed to restrain points P1 and P2 from displacement in the Z direction, which forces the rotation to occur about this point. (Rotation about any other point would cause points P1 and P2 to move in the Z direction). As a result,

$$\Delta P1Z = \Delta P2Z = 0$$

and

$$\Delta P1X = \Delta P2X = 0.$$

Restraining points P1 and P2 from Z displacement disallows rigid body displacement of the grillage in its local z′ direction. This restraint is imposed in order to eliminate transverse shearing forces in the hull structure. Figure 3.7(b) illustrates how the grillage is restrained against z′ translation. The horizontal and vertical W sections that make up the grillage are represented as line elements in the same plane. Restraints are placed on either side of the center vertical section as shown and anchored by the testing frame. As a result, points P1 and P2 cannot translate in the Z direction. In addition to being restrained against rigid body displacement in the z′ direction, the grillages are also restrained against θx.

Point P3 does translate in the Z direction due to y′ rotation, but this translation is small and can therefore be neglected. This assumption can be made because the largest possible $\theta y'$ permitted by the spherical bearings at the actuator clevises is plus or minus 5 degrees and for a Z distance of 150 inches between points P2 and P3, the resulting $\Delta P3Z$ is less than 0.5 inches. As a comparison, $\Delta P3X$ due to this rotation is greater than 10 inches. It is then concluded that

$$\Delta P3Z = 0.$$

When y′ rotation of a grillage occurs, it must also translate in the y′ direction in order to accommodate the fixed-length vertical links. The grillage translates as a rigid body, such that

$$\Delta P1Y = \Delta P2Y = \Delta P3Y.$$

If all of the allowable grillage displacements ($\Delta x'$, $\Delta y'$, $\theta y'$, and $\theta z'$) are permitted simultaneously, the consequences will be those that Cases 1, 2, and 3 have in common. Therefore, given any permissible grillage motion,

$$\Delta P1Z = \Delta P2Z = \Delta P2Z = 0$$

and

$$\Delta P2Y = \Delta P3Y.$$

Now, the coordinates of P1, P2 and P3 are given by:

$$\begin{array}{llll} P1: & P1X = P1X_o + \Delta P1X; & P1Y = P1Y_o + \Delta P1Y; & P1Z = P1Z_o \\ P2: & P2X = P2X_o + \Delta P2X; & P2Y = P2Y_o + \Delta P3Y; & P2Z = P2Z_o \\ P3: & P3X = P3X_o + \Delta P3X; & P3Y = P3Y_o + \Delta P3Y; & P3Z = P3Z_o \end{array}$$

The number of unknowns has been reduced from nine to five. P1Z, P2Z, and P3Z are always equal to their original values and $\Delta P2Y$ is always equal to $\Delta P3Y$. Therefore there are now five unknowns and only five measurements need to be made to determine the coordinates of P1, P2, and P3. Figure 3.8 shows the configuration of the displacement transducers needed to make the necessary measurements for the north grillage. The transducers are labeled $X1_N$, $X2_N$, $X3_N$, $Y1_N$, and $Y1_N$. (Note that P1, P2 and P3 and their coordinates have now been given subscripts of N on the north grillage and S on the south grillage). Transducer $X1_N$ measures $\Delta P1_NX$, transducer $X2_N$ measures $\Delta P2_NX$, and so on.

Consider that the north grillage undergoes x'_N and y'_N translations and a rotation about z'_N. As shown in Figure 3.9, transducer $X1_N$ rotates through an angle β and as a result does not measure the true change in X position of point P1, $\Delta P1_NX$. However, the larger we make $P1_NX_o$ (or the greater the distance between the grillage and the fixed reference frame), the smaller β is and $\Delta P1_NX$ can be approximated by the $X1_N$ transducer measurement. The same holds true for the Y transducers. Given this approximation, the equations that determine the five unknown coordinates of $P1_N$, $P2_N$, and $P3_N$ are:

$$\begin{array}{ll} P1_NX = P1_NX_o + \Delta P1_NX & P1_NY = P1_NY_o + \Delta P1_NY \\ P2_NX = P2_NX_o + \Delta P2_NX & \\ P3_NX = P3_NX_o + \Delta P3_NX & P3_NY = P3_NY_o + \Delta P3_NY. \end{array}$$

Figure 3.10 shows the configuration of the transducers that make the necessary measurements for the south grillage. The unknown coordinates of $P1_S$, $P2_S$, and $P3_S$ are then given by

$$P1_SX = P1_SX_o + \Delta P1_SX \qquad P1_SY = P1_SY_o + \Delta P1_SY$$
$$P2_SX = P2_SX_o + \Delta P2_SX$$
$$P3_SX = P3_SX_o + \Delta P3_SX \qquad P3_SY = P3_SY_o + \Delta P3_SY$$

3.2.3 Coordinate System Transformation

Refer to Figure 3.11. Let $\mathbf{C'}_N$ be a vector defining the coordinates of the north end of an actuator with respect to the local reference point r'_N. Assuming that deformations of the grillage are negligible, this vector remains constant in the local coordinate system. However, the global coordinates of the actuator end change due to translations and rotations of the grillage. These global coordinates can be determined if the vector transformation between the local and global systems is known.

Let **i**, **j**, and **k** be the mutually orthogonal unit vectors that form the basis of the X, Y, Z global coordinate system. Likewise let $\mathbf{i'}_N$, $\mathbf{j'}_N$, and $\mathbf{k'}_N$ be the basis vectors of the x'_N, y'_N, z'_N local system for the north grillage (see Figure 3.12(a)). Just as any vector **v** in the **i**, **j**, **k** basis can be written in the form $\mathbf{v} = t_1\mathbf{i} + t_2\mathbf{j} + t_3\mathbf{k}$, the unit vectors $\mathbf{i'}_N$, $\mathbf{j'}_N$, and $\mathbf{k'}_N$ can be expressed in the form

$$\begin{aligned} \mathbf{i}'_N &= t_{11}\mathbf{i} + t_{12}\mathbf{j} + t_{13}\mathbf{k} \\ \mathbf{j}'_N &= t_{21}\mathbf{i} + t_{22}\mathbf{j} + t_{23}\mathbf{k} \\ \mathbf{k}'_N &= t_{31}\mathbf{i} + t_{32}\mathbf{j} + t_{33}\mathbf{k} \end{aligned} \qquad (3.1)$$

Writing these equations in matrix form, the vector transformation from the global to local basis is given by

$$\begin{Bmatrix} \mathbf{i}'_N \\ \mathbf{j}'_N \\ \mathbf{k}'_N \end{Bmatrix} = \mathbf{T}_N \cdot \begin{Bmatrix} \mathbf{i} \\ \mathbf{j} \\ \mathbf{k} \end{Bmatrix}$$

where the transformation matrix $\mathbf{T}_N$ is defined as

$$\mathbf{T}_N = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$$

Since the vector $\mathbf{C'}_N$ is known in the local system, it can be expressed

$$\mathbf{C}'_N = (c'_1)\mathbf{i}'_N + (c'_2)\mathbf{j}'_N + (c'_3)\mathbf{k}'_N \qquad (3.2)$$

where c'_1, c'_2, and c'_3 are known constants. $\mathbf{C'_N}$ can also be written using the global basis in the form

$$\mathbf{C}'_N = (c_1)\mathbf{i} + (c_2)\mathbf{j} + (c_3)\mathbf{k} \tag{3.3}$$

where c_1, c_2, and c_3 are unknowns that vary with grillage movement. Substituting Equations (3.1) into Equation (3.2) gives

$$\mathbf{C}'_N = (c'_1)[t_{11}\mathbf{i} + t_{12}\mathbf{j} + t_{13}\mathbf{k}] + (c'_2)[t_{21}\mathbf{i} + t_{22}\mathbf{j} + t_{23}\mathbf{k}] + (c'_3)[t_{31}\mathbf{i} + t_{32}\mathbf{j} + t_{33}\mathbf{k}]$$

Rearranging,

$$\mathbf{C}'_N = [c'_1 t_{11} + c'_2 t_{21} + c'_3 t_{31}]\mathbf{i} + [c'_1 t_{12} + c'_2 t_{22} + c'_3 t_{32}]\mathbf{j} + [c'_1 t_{13} + c'_2 t_{23} + c'_3 t_{33}]\mathbf{k}. \tag{3.4}$$

Equating the c_1, c_2, and c_3 terms of Equation 3 with the bracketed terms of Equation 3.4 and arranging into matrix form, we have

$$\begin{Bmatrix} c_1 \\ c_2 \\ c_3 \end{Bmatrix} = \begin{bmatrix} t_{11} & t_{21} & t_{31} \\ t_{12} & t_{22} & t_{32} \\ t_{13} & t_{23} & t_{33} \end{bmatrix} \cdot \begin{Bmatrix} c'_1 \\ c'_2 \\ c'_3 \end{Bmatrix},$$

and the vector $\mathbf{C'_N}$ can now be expressed in the global basis using the transpose of the transformation matrix $\mathbf{T_N}$.

The selection of points P1, P2, and P3 to be along the local axes allows us to directly determine the transformation matrix $\mathbf{T_N}$. As shown in Figure 3.12(b), let $\mathbf{v'_N}$ be a vector from point $P1_N$ to point $P2_N$ and $\mathbf{w'_N}$ a vector from $P2_N$ to $P3_N$. These vectors are given by:

$$\mathbf{v}'_N = (P1_N X - P2_N X)\mathbf{i} + (P1_N Y - P2_N Y)\mathbf{j} + (P1_N Z - P2_N Z)\mathbf{k}$$

or

$$\mathbf{v}'_N = (v_1)\mathbf{i} + (v_2)\mathbf{j} + (v_3)\mathbf{k}$$

and

$$\mathbf{w}'_N = (P3_N X - P2_N X)\mathbf{i} + (P3_N Y - P2_N Y)\mathbf{j} + (P3_N Z - P2_N Z)\mathbf{k}$$

or

$$\mathbf{w}'_N = (w_1)\mathbf{i} + (w_2)\mathbf{j} + (w_3)\mathbf{k}.$$

The vector normal to the plane, called $\mathbf{u'_N}$, is then calculated by taking the cross product of $\mathbf{v'_N}$ and $\mathbf{w'_N}$ as follows:

$$\mathbf{u}'_N = [(v_2 \cdot w_3) - (w_2 \cdot v_3)]\mathbf{i} + [-(v_1 \cdot w_3) + (w_1 \cdot v_3)]\mathbf{j} + [(v_1 \cdot w_2) - (w_1 \cdot v_2)]\mathbf{k}$$

or

$$\mathbf{u}'_N = (u_1)\mathbf{i} + (u_2)\mathbf{j} + (u_3)\mathbf{k}.$$

The vectors $\mathbf{u}'_N$, $\mathbf{v}'_N$, and $\mathbf{w}'_N$ are defined in the global basis because the coordinates of $P1_N$, $P2_N$, and $P3_N$ are global coordinates. Figure 3.12(b) shows that the unit vectors $\mathbf{i}'_N$, $\mathbf{j}'_N$, and $\mathbf{k}'_N$ that define the local basis for the north grillage are the unit vectors associated with $\mathbf{u}'_N$, $\mathbf{v}'_N$, and $\mathbf{w}'_N$. If we define

$$|\mathbf{u}'_N| = \sqrt{(u_1^2 + u_2^2 + u_3^2)},$$
$$|\mathbf{v}'_N| = \sqrt{(v_1^2 + v_2^2 + v_3^2)},$$
$$|\mathbf{w}'_N| = \sqrt{(w_1^2 + w_2^2 + w_3^2)},$$

then

$$\mathbf{i}'_N = \frac{u_1}{|\mathbf{u}'_N|}\mathbf{i} + \frac{u_2}{|\mathbf{u}'_N|}\mathbf{j} + \frac{u_3}{|\mathbf{u}'_N|}\mathbf{k} = t_{11}\mathbf{i} + t_{12}\mathbf{j} + t_{13}\mathbf{k},$$

$$\mathbf{j}'_N = \frac{v_1}{|\mathbf{v}'_N|}\mathbf{i} + \frac{v_2}{|\mathbf{v}'_N|}\mathbf{j} + \frac{v_3}{|\mathbf{v}'_N|}\mathbf{k} = t_{21}\mathbf{i} + t_{22}\mathbf{j} + t_{23}\mathbf{k},$$

$$\mathbf{k}'_N = \frac{w_1}{|\mathbf{w}'_N|}\mathbf{i} + \frac{w_2}{|\mathbf{w}'_N|}\mathbf{j} + \frac{w_3}{|\mathbf{w}'_N|}\mathbf{k} = t_{31}\mathbf{i} + t_{32}\mathbf{j} + t_{33}\mathbf{k}.$$

Thus, the transformation matrix is defined and $\mathbf{C}'_N$ can be found.

This same method can be applied to the south grillage. Let $\mathbf{C}'_S$ be the vector that goes from r'_S to the south end of the actuator. As shown in Figure 3.13(a), the south grillage has its own local basis $\mathbf{i}'_S$, $\mathbf{j}'_S$, $\mathbf{k}'_S$. $\mathbf{C}'_S$ is known in the local basis but can be transformed to the global basis through the calculation of $\mathbf{u}'_S$, $\mathbf{v}'_S$, and $\mathbf{w}'_S$ (Figure 3.13(b)) and their corresponding unit vectors. As for the north grillage, the final result is a transformation matrix $\mathbf{T}_S$. $\mathbf{C}'_S$ can be determined in global coordinates using the transpose of $\mathbf{T}_S$.

3.3 ACTUATOR FORCE COMPONENT CALCULATIONS

3.3.1 Actuator Vectors

Top Actuators Now that the coordinate transformations have been defined, the actuator vectors can be determined. Let $\mathbf{R}_N$ be the vector from the global reference point r to the local reference point r'_N on the north grillage (see Figure 3.14). Point r'_N coincides with point $P2_N$ thus,

$$\mathbf{R}_N = (P2_N X)\mathbf{i} + (P2_N Y)\mathbf{j} + (P2_N Z)\mathbf{k}.$$

Also let $\mathbf{E}_N$ be defined as the vector from point r to the north end of an actuator. Since $\mathbf{R}_N$ is known in global coordinates and $\mathbf{C}'_N$ has been transformed to global coordinates, $\mathbf{E}_N$ can be determined from the vector addition

$$\mathbf{E}_N = \mathbf{C}'_N + \mathbf{R}_N$$

Figure 3.15 shows that similar vectors can be established for the south grillage. Point r'_S coincides with point $P2_S$ so the vector $\mathbf{R_S}$ from r to r'_S is given by

$$\mathbf{R_S} = (P2_S X)\mathbf{i} + (P2_S Y)\mathbf{j} + (P2_S Z)\mathbf{k}$$

$\mathbf{C'_S}$ is known in global coordinates therefore the vector $\mathbf{E_S}$ from point r to the south end of the actuator can be determined by the equation

$$\mathbf{E_S} = \mathbf{C'_S} + \mathbf{R_S}.$$

Now, knowing $\mathbf{E_N}$ and $\mathbf{E_S}$, the actuator vector **A** can be found. As shown in Figure 3.16,

$$\mathbf{A} = \mathbf{E_S} - \mathbf{E_N}.$$

If **A** is written in the form

$$\mathbf{A} = (a_X)\mathbf{i} + (a_Y)\mathbf{j} + (a_Z)\mathbf{k},$$

then a_X, a_Y, and a_Z represent the components of the length of the actuator in the X, Y, and Z directions, respectively.

<u>Bottom Actuators</u> The calculation of the **A** vectors for the bottom actuators is similar to that for the top actuators. The only difference is in the calculation of $\mathbf{E_N}$, which is shown in Figure 3.17. As described in Section 2.3.2, point C, which represents the north end of a bottom actuator, is restrained against translation in the Y and Z directions. The only permissible motion is translation in the X direction. If the magnitude of this translation is measured directly at point C, the $\mathbf{E_N}$ vector can be determined without the use of $\mathbf{C'_N}$ and $\mathbf{R_N}$ vectors. Let $\mathbf{E_{No}}$ be a vector which represents the original coordinates of the north end of the actuator with respect to point r and is defined as

$$\mathbf{E_{No}} = (e_1)\mathbf{i} + (e_2)\mathbf{j} + (e_3)\mathbf{k}.$$

If the X translation of point C at a given loading step is called X_C then $\mathbf{E_N}$ at that step is given by

$$\mathbf{E_N} = (e_1 + X_C)\mathbf{i} + (e_2)\mathbf{j} + (e_3)\mathbf{k}.$$

$\mathbf{E_S}$ is determined in the same manner as it was for the top actuators, as is **A**. See Figures 3.18 and 3.19 for illustrations of these vectors.

<u>Bottom Links</u> The forces in the bottom links that are attached to the north grillage are also measured by force transducers. The components of these forces are needed in order to write the equilibrium equations for the north grillage. Therefore, **A** vectors are found for these links as well. $\mathbf{E_N}$ is determined in the same manner as it was for the top actuators. However, point B (see

Figure 3.17), which represents the south end of these links, is subjected to the same constraints as point C. The X translation of point B is equal to the X translation of point C, while its Y and Z translations are zero. If a vector $\mathbf{E}_{So}$ representing the original location of the south end of the actuator is given by

$$\mathbf{E}_{So} = (e_4)\mathbf{i} + (e_5)\mathbf{j} + (e_6)\mathbf{k},$$

then

$$\mathbf{E}_S = (e_4 + X_C)\mathbf{i} + (e_5)\mathbf{j} + (e_6)\mathbf{k}$$

and **A** can be determined for the bottom links.

3.3.2 Actuator Forces

The actual force components can now be determined using the actuator vectors. Refer to Figure 3.20, which shows the top actuators acting in tension. At the north grillage, the X component of the actuator force is acting in the positive X direction, while the Y component acts in the negative Y direction. At the south grillage, these forces act in the opposite directions. This is also true for the Z force component. The forces F_{XN}, F_{YN}, and F_{ZN} will first be determined for use in the north grillage equilibrium equations. Then F_{XS}, F_{YS}, and F_{ZS}, which are equal in magnitude but opposite in sign to F_{XN}, F_{YN}, and F_{ZN}, will be found for use in the south grillage equilibrium equations. The bottom link forces at the north grillage and the bottom actuator forces at the south grillage are found independently of each other.

North Grillage Again consider the loading shown in Figure 3.20, where the top actuators act in tension and the bottom actuators act in compression. Also consider that θ_Z is larger for the south grillage than for the north grillage and that θy is zero for both. Figure 3.21 shows the actuator and link position vectors that would result from this configuration. As evident in Figure 3.21, the X direction component of the **A** vector, or a_X, is always positive because **A** originates at the north end of the actuator or link and terminates at the south end. Therefore the sense of the force, which is measured by the force transducer, is what determines the direction of its X component. For example, if the force in the top actuator is measured as positive, we can conclude that the force is tensile and that the X component of the force at the north grillage acts in the positive X direction, as shown in Figure 3.20. Likewise, if the force in the bottom link is negative, the force is compressive and its X component acts in the negative X direction. Thus the magnitude and direction of the X component of force for any actuator or link is given by

$$F_{XN} = F_{TOT} \cdot \frac{a_X}{|\mathbf{A}|}.$$

The equation for $|\mathbf{A}|$, which represents the total length of the actuator, is

$$|\mathbf{A}| = \sqrt{(a_X^2 + a_Y^2 + a_Z^2)}.$$

Unlike a_X, a_Y can be either positive or negative. The direction of the Y component of force is then dependent on both the sense of the total force and the actuator vector direction. Refer to Figure 3.22, which again shows the force in the top actuators is tensile and thus is positive. As shown in Figure 3.22(a), if θ_Z is larger for the south grillage than for the north grillage, then the Y component of force at the north grillage is negative. However, Figure 3.22(c) shows that if θ_Z is larger for the north grillage, the Y component of force is positive. Therefore, the sign of F_Y is dependent on direction of the actuator vector. Figure 3.22(b) shows that a_Y is negative when the situation in 3.22(a) occurs. Although the total force is positive, whèn multiplied by the negative value of a_Y, the resulting F_Y is negative. The equation for F_Y is then

$$F_{YN} = F_{TOT} \cdot \frac{a_Y}{|\mathbf{A}|}. \quad (3.5)$$

Clearly this equation is valid for the case illustrated in Figure 3.22(c) and (d). It can also be shown that the equation works when the actuator is acting in compression. Figure 3.23 summarizes the four possible cases that can occur for the top actuators. Since the last two columns of this figure agree, Equation 3.5 is valid for all cases. Equation 3.5 also applies to the bottom links at the north grillage.

The Z component of force can be determined by the same method. Figure 3.24(a) and (b) illustrate that when the top actuators are in tension and a_Z is negative, the Z component of force is negative at the north grillage. When the top actuators are in tension and a_Z is positive, the Z component of force is in the positive Z direction (Figure 3.24(c) and (d)). It can be shown that the equation

$$F_{ZN} = F_{TOT} \cdot \frac{a_z}{|\mathbf{A}|} \quad (3.6)$$

applies given any sense of force and any direction of actuator vector. Equation 3.6 is also valid for the bottom links at the north grillage.

<u>South Grillage</u> The X, Y, and Z components of force in the top actuators at the south grillage are equal in magnitude but opposite in sign to those at the north grillage. Therefore,

$$F_{XS} = -F_{XN} = -F_{TOT} \cdot \frac{a_x}{|\mathbf{A}|},$$

$$F_{YS} = -F_{YN} = -F_{TOT} \cdot \frac{a_Y}{|\mathbf{A}|},$$

and

$$F_{ZS} = -F_{ZN} = -F_{TOT} \cdot \frac{a_Z}{|\mathbf{A}|}$$

for the top actuators.

The X components of force in the bottom actuators are determined as follows. Since a_X is always positive (Figure 3.25(a)), the sign of F_X depends on the sense of the force. Figures 3.25(b) and (c) show that if F_{TOT} is compressive (negative) F_X acts in the positive X direction, while if F_{TOT} is tensile (positive) F_X is negative. Therefore,

$$F_X = -F_{TOT} \cdot \frac{a_X}{|\mathbf{A}|}$$

for the bottom actuators.

The Y and Z force components in the bottom actuators are determined by the same method used for the top actuators and bottom links at the north grillage. The resulting equations are

$$F_Y = -F_{TOT} \cdot \frac{a_Y}{|\mathbf{A}|}$$

and

$$F_Z = -F_{TOT} \cdot \frac{a_Z}{|\mathbf{A}|}.$$

3.3.3 Vertical Link Vectors and Forces

The components of the forces in the vertical links are also needed in order to write the equilibrium equations for each grillage. These forces can be found using a procedure like that used for the actuators. Each link will first be represented by a position vector **L**, then its force components will be calculated. This vector is analogous to the position vector **A** used for the actuators.

As shown in Figure 3.26, let **L1** be the vector from point r to the point where the vertical link attaches to the north grillage. Also, let **L2** be the vector from point r to the point where the link attaches to the overhead frame. **L2** is constant and is given by

$$\mathbf{L2} = (L2_X)\mathbf{i} + (L2_Y)\mathbf{j} + (L2_Z)\mathbf{k}.$$

However, vector **L1** changes as the grillage translates and rotates and its global coordinates are unknown. Just as we defined a vector $\mathbf{C'_N}$ in local coordinates for the actuators, we can define a similar vector $\mathbf{D'_N}$ that goes from the local reference point r′ to the bottom of the link (see Figure 3.27). This vector is constant in the local coordinate system and is given by

$$\mathbf{D}'_N = (d'_1)\mathbf{i}'_N + (d'_2)\mathbf{j}'_N + (d'_3)\mathbf{k}'_N,$$

where d'_1, d'_2, and d'_3 are known. $\mathbf{D'}_N$ can be expressed in the global basis as

$$\mathbf{D}'_N = (d_1)\mathbf{i} + (d_2)\mathbf{j} + (d_3)\mathbf{k}'.$$

The values d_1, d_2, and d_3 are determined by the transpose of the transformation matrix $\mathbf{T}_N$ given earlier:

$$\begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{bmatrix} t_{11} & t_{21} & t_{31} \\ t_{12} & t_{22} & t_{32} \\ t_{13} & t_{23} & t_{33} \end{bmatrix} \cdot \begin{Bmatrix} d'_1 \\ d'_2 \\ d'_3 \end{Bmatrix}.$$

L1 can now be determined from

$$\mathbf{L1} = \mathbf{R}_N + \mathbf{D}'_N$$

and **L** is then

$$\mathbf{L} = \mathbf{L2} - \mathbf{L1} = (l_X)\mathbf{i} + (l_Y)\mathbf{j} + (l_Z)\mathbf{k}.$$

The total forces in the vertical links, which are always tensile forces, are measured by force transducers. Since the algebraic sign of F_{TOT} is always positive, the sign of the force components are dependent on the direction of the **L** vector. Given the grillage movement illustrated in Figure 3.28(a), F_X is negative. The value of l_X is negative, as evident in Figure 3.28(b), therefore

$$F_X = F_{TOT} \cdot \frac{l_X}{|\mathbf{L}|}$$

where

$$|\mathbf{L}| = \sqrt{(l_X^2 + l_Y^2 + l_Z^2)}.$$

Figure 3.28 also shows that F_Y is positive, as is the value of l_Y, therefore

$$F_Y = F_{TOT} \cdot \frac{l_Y}{|\mathbf{L}|}.$$

These equations apply for the links on both the north and south grillages. Because the translation of the bottom link point in the Z direction is assumed to be very small, the Z component of force in all vertical links is neglected.

3.4 EQUILIBRIUM EQUATIONS

3.4.1 XY Plane Equilibrium

The X, Y and Z forces for all actuators and links are now known and equilibrium equations can be written for each grillage the XY plane. The equilibrium equations will be used to determine the net axial force and moments in the hull structure, which are parameters needed for the control algorithm.

North Grillage Refer to Figure 3.29, which shows the north grillage and all forces acting upon it. All force components have been given subscripts corresponding to their actuator or link name assigned in Section 2.2 (see Figure 2.2). For example, actuator TE (or Top East) has forces $(F_X)_{TE}$, $(F_Y)_{TE}$, and $(F_Z)_{TE}$. The axial force P_N, shear V_{YN}, and moment M_{ZN} have also been defined. All actuator and link forces are shown in the positive sense. However, if a given force is negative (as determined from the equations in Section 3.2.2), its direction and sign will be opposite from that shown. From $\Sigma F_X = 0$,

$$P_N + (F_{XN})_{TE} + (F_{XN})_{TC} + (F_{XN})_{TW} + (F_X)_{BNE} + (F_X)_{BNW} + (F_X)_{NE} + (F_X)_{NW} = 0.$$

Solving for the axial force, we get

$$P_N = -(F_{XN})_{TE} - (F_{XN})_{TC} - (F_{XN})_{TW} - (F_X)_{BNE} - (F_X)_{BNW} - (F_X)_{NE} - (F_X)_{NW}.$$

From $\Sigma F_Y = 0$,

$$V_{YN} + (F_{YN})_{TE} + (F_{YN})_{TC} + (F_{YN})_{TW} + (F_Y)_{BNE} + (F_Y)_{BNW} + (F_Y)_{NE} + (F_Y)_{NE} + \frac{W}{2} = 0.$$

Solving for the shear force, we get

$$V_{YN} = -(F_{YN})_{TE} - (F_{YN})_{TC} - (F_{YN})_{TW} - (F_Y)_{BNE} - (F_Y)_{BNW} - (F_Y)_{NE} - (F_Y)_{NE} - \frac{W}{2}.$$

Moments are summed about the local reference point r'_N. This point is convenient because the moment arms from r'_N are known from the transformed $\mathbf{C}'_N$ and $\mathbf{D}'_N$ vectors. For example, if

$$(\mathbf{C}'_N)_{TE} = (c_1)_{TE}\mathbf{i} + (c_2)_{TE}\mathbf{j} + (c_3)_{TE}\mathbf{k},$$

then $(c_1)_{TE}$ is the moment arm for $(F_{YN})_{TE}$ and $(c_2)_{TE}$ is the moment arm for $(F_{XN})_{TE}$ in the XY plane. A positive moment is counter-clockwise about r'_N. The shear force, V_{YN}, passes through r'_N. From $\Sigma M_Z = 0$,

$$\begin{aligned}
&M_{ZN} - (F_{XN})_{TE}(c_2)_{TE} - (F_{XN})_{TC}(c_2)_{TC} - (F_{XN})_{TW}(c_2)_{TW} \\
&- (F_X)_{BNE}(c_2)_{BNE} - (F_X)_{BNW}(c_2)_{BNW} - (F_X)_{NE}(d_2)_{NE} - (F_X)_{NW}(d_2)_{NW} \\
&+ (F_{YN})_{TE}(c_1)_{TE} + (F_{YN})_{TC}(c_1)_{TC} + (F_{YN})_{TW}(c_1)_{TW} \\
&+ (F_Y)_{BNE}(c_1)_{BNE} + (F_Y)_{BNW}(c_1)_{BNW} + (F_Y)_{NE}(d_1)_{NE} + (F_Y)_{NW}(d_1)_{NW} = 0.
\end{aligned} \tag{3.7}$$

The signs of the terms in the Equation 3.7 are determined as follows. The (F_{XN}) terms for the top actuators, for example, are shown as positive in Figure 3.29. Their moment arms, or the (c_1) terms, are also positive. However, the moment of these forces about point r'_N are negative (or counter-clockwise). Therefore these terms need a negative sign in the moment equation. If an (F_{XN}) force is calculated as negative, then the moment of this force about point r'_N is positive (or clockwise). The product of the negative sign in Equation 3.7 and the negative sign of the force results in the necessary positive sign for the moment term.

Upon inspection, it is seen that certain forces may cause positive or negative moments depending on the position of the grillage. For the rotation shown in Figure 3.29, the moment caused by the Y component of the vertical link force is positive (counterclockwise). For the case shown, both the force and the moment arm are positive, so the sign of the term in Equation 3.7 is positive. If the grillage is rotated counterclockwise until the bottom link point is to the left of point r'_N, the moment of this force about r'_N is negative (clockwise). For this configuration, the Y component of the force is positive and the moment arm (d_1) is negative, so the product of the two is negative, as needed. Solving Equation 3.7 for M_{ZN},

$$\begin{aligned}
&M_{ZN} = (F_{XN})_{TE}(c_2)_{TE} + (F_{XN})_{TC}(c_2)_{TC} + (F_{XN})_{TW}(c_2)_{TW} \\
&+ (F_X)_{BNE}(c_2)_{BNE} + (F_X)_{BNW}(c_2)_{BNW} + (F_X)_{NE}(d_2)_{NE} + (F_X)_{NW}(d_2)_{NW} \\
&- (F_{YN})_{TE}(c_1)_{TE} - (F_{YN})_{TC}(c_1)_{TC} - (F_{YN})_{TW}(c_1)_{TW} \\
&- (F_Y)_{BNE}(c_1)_{BNE} - (F_Y)_{BNW}(c_1)_{BNW} - (F_Y)_{NE}(d_1)_{NE} - (F_Y)_{NW}(d_1)_{NW}.
\end{aligned}$$

<u>South Grillage</u> Figure 3.30 shows the south grillage and all forces acting on it. Using the same method as for the north grillage, the equilibrium equations for axial force P_S, shear V_{YS}, and moment M_{ZS} are as follows:

$$P_S = -(F_{XS})_{TE} - (F_{XS})_{TC} - (F_{XS})_{TW} - (F_X)_{BSE} - (F_X)_{BSW} - (F_X)_{SE} - (F_X)_{SE},$$

$$V_{YS} = -(F_{YS})_{TE} - (F_{YS})_{TC} - (F_{YS})_{TW} - (F_Y)_{BSE} - (F_Y)_{BSW} - (F_Y)_{SE} - (F_Y)_{SE}\frac{W}{2},$$

and

$$M_{ZS} = -(F_{XS})_{TE}(c_2)_{TE} - (F_{XS})_{TC}(c_2)_{TC} - (F_{XS})_{TW}(c_2)_{TW}$$
$$- (F_X)_{BSE}(c_2)_{BSE} - (F_X)_{BSW}(c_2)_{BSW} - (F_X)_{SE}(d_2)_{SE} - (F_X)_{SW}(d_2)_{SW}$$
$$+ (F_{YS})_{TE}(c_1)_{TE} + (F_{YS})_{TC}(c_1)_{TC} + (F_{YS})_{TW}(c_1)_{TW}$$
$$+ (F_Y)_{BSE}(c_1)_{BSE} + (F_Y)_{BSW}(c_1)_{BSW} + (F_Y)_{SE}(d_1)_{SE} + (F_Y)_{SW}(d_1)_{SW}.$$

where a positive M_{ZS} is clockwise at the south grillage. The shear force in the Y direction should be approximately equal to zero therefore M_{ZN} should equal M_{ZS} (see Section 2.2).

The axial force in the specimen, which should be equal to or close to zero, is determined in the following manner. If a net tensile force develops in the specimen, as shown in Figure 3.31, P_N will be calculated as positive and P_S will be calculated as negative. Their magnitude should be equal because the axial force in the specimen is constant. However, P_N and P_S may differ slightly for a variety of reasons. First, the moments in all clevises are assumed to be zero, but, because of friction, this may not be true. Secondly, the grillages are not rigid as assumed, so the lines of action of forces may change. Finally, error can be introduced by force transducer measurements. As a result, the average of P_N and P_S is taken. The net axial force in the specimen is thus,

$$P = \frac{(P_N - P_S)}{2}.$$

If the net force is positive it is tensile; if negative, it is compressive.

3.4.2 XZ Plane Equilibrium

North Grillage Equilibrium equations are written for the XZ plane in order to find the bending moment M_Y and shear V_Z at each grillage. Figure 3.32 shows the XZ plane of the north grillage and all forces acting on it. All forces are shown in the positive direction. Shear in the Z direction is given by

$$V_{ZN} = -(F_{ZN})_{TE} - (F_{ZN})_{TC} - (F_{ZN})_{TW} - (F_Z)_{BNE} - (F_Z)_{BNW}.$$

From $\Sigma M_Y = 0$,

$$M_{YN} = -(F_{XN})_{TE}(c_3)_{TE} - (F_{XN})_{TC}(c_3)_{TC} - (F_{XN})_{TW}(c_3)_{TW}$$
$$- (F_X)_{BNE}(c_3)_{BNE} - (F_X)_{BNW}(c_3)_{BNW} - (F_X)_{NE}(d_3)_{NE} - (F_X)_{NW}(c_3)_{NW}$$
$$+ (F_{ZN})_{TE}(c_1)_{TE} + (F_{ZN})_{TC}(c_1)_{TC} + (F_{ZN})_{TW}(c_1)_{TW}$$
$$+ (F_Z)_{BNE}(c_1)_{BNE} + (F_Z)_{BNW}(c_1)_{BNW}.$$

A positive M_{YN} is counter-clockwise. The signs of the terms in the above equation are determined in the same manner as for the XY plane moment equations.

South Grillage Figure 3.33 illustrates the XZ plane of the south grillage and all forces acting upon it. From $\Sigma F_Z = 0$,

$$V_{ZS} = -(F_{ZS})_{TE} - (F_{ZS})_{TC} - (F_{ZS})_{TW} - (F_Z)_{BSE} - (F_Z)_{BSW}.$$

From $\Sigma M_Y = 0$,

$$\begin{aligned} M_{YS} &= (F_{XS})_{TE}(c_3)_{TE} + (F_{XS})_{TC}(c_3)_{TC} + (F_{XS})_{TW}(c_3)_{TW} \\ &+ (F_X)_{BSE}(c_3)_{BSE} + (F_X)_{BSW}(c_3)_{BSW} + (F_X)_{SE}(d_3)_{SE} + (F_X)_{SW}(c_3)_{SW} \\ &- (F_{ZS})_{TE}(c_1)_{TE} - (F_{ZS})_{TC}(c_1)_{TC} - (F_{ZS})_{TW}(c_1)_{TW} \\ &- (F_Z)_{BSE}(c_1)_{BSE} - (F_Z)_{BSW}(c_1)_{BSW}, \end{aligned}$$

where a positive M_{YS} is clockwise. The shear force in the Z direction should be approximately equal to zero therefore M_{YN} should equal M_{YS} (see Section 2.2).

3.5 CONTROL LOGIC

3.5.1 Objectives

The loading objectives, which were presented in Section 2.3.1 and need to be considered in the control logic, are as follows:

1. Apply primary bending moment M_Z to the hull structure.
2. Maintain zero net axial force in the hull structure. This ensures that the elastic neutral axis remains at the centroid of the cross section. The axial force is kept at or near zero by selective movement of the actuators.

Additional objectives to be considered in the control logic are:

3. Keep the Y rotation of the north grillage equal to the Y rotation of the south grillage ($\theta_{YN} - \theta_{YS} = 0$). This is achieved by selective movement of the top actuators and by always keeping the strokes of the bottom actuators equal to each other. This is explained further below.
4. Keep the shear forces in the Y and Z directions equal to zero ($V_{YN} = V_{YS} = 0$ and $V_{ZN} = V_{ZS} = 0$). It follows from these equalities that the moment about Z at the north grillage is equal to the moment about Z at the south grillage and the moment about Y at the north grillage is equal to the moment about Y at the south grillage ($M_{ZN} - M_{ZS} = 0$ and $M_{YN} - M_{YS} = 0$). This objective is accomplished by the actuator setup, as explained in Section 2.2.

3.5.2 The Control Algorithm

The control algorithm decides which actuators to move in order to simultaneously reduce the net compression force or net tension force in the hull structure and increase the rotation (if the hull structure is being loaded) or decrease the rotation (if the hull structure is being unloaded). The following scenarios explain how this is done.

Consider the situation shown in Figure 3.34(a), where θ_Z is positive and the hull structure is in net compression. Reduction of the compressive axial force can be accomplished in two ways. Figure 3.34(b) shows that extending the top actuators reduces the net compression while also reducing the rotation θ_Z. Extending the bottom actuators, as illustrated in Figure 3.34(c), also reduces the axial force but increases θ_Z.

In Figure 3.35(a), θ_Z is again positive and the axial force is tensile. The axial force can be eliminated by either retracting the bottom actuators, which decreases θ_Z, or by retracting the top actuators, which increases θ_Z (see Figures 3.35(b) and 3.35(c)).

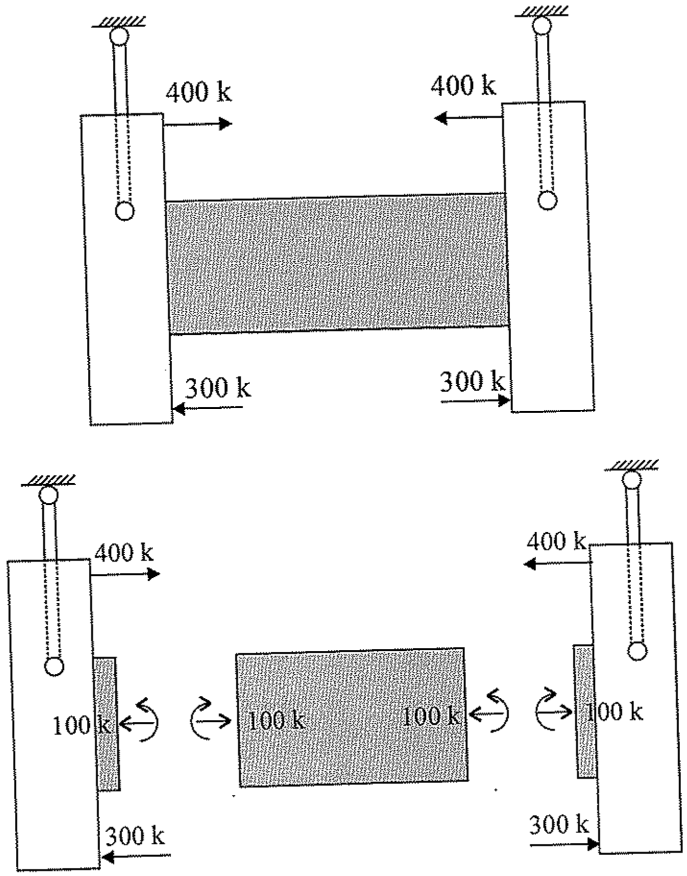
In the situations described in Figures 3.34 and 3.35, it is assumed that the forces in the top actuators are equal to each other. In other words, there is no imbalance of forces across the width of the grillage. If this is true, then when the top actuators are to be moved, they are all moved the same amount. However, when an imbalance of forces does occur, it is possible in some instances to selectively move certain actuators to reduce the imbalance. This is explained below in greater detail. If the forces in the bottom actuators are not equal to each other, neither actuator can be moved on its own to eliminate the imbalance. Doing so would cause $\theta_{YN} \neq \theta_{YS}$. Anytime the bottom actuators are to be moved, they must be moved together.

It is unlikely that the force in the top actuators will ever be exactly equal to each other. Therefore, a tolerance is computed using one of the outer actuators, say actuator TE. This tolerance is a user-specified value in the software, as explained in Chapter 4. If the tolerance factor is 0.1 and $(F_{XN})_{TE} = 100$ kips, the tolerance is $(0.1)\cdot(100 \text{ kips}) = 10$ kips. Then if $(F_{XN})_{TC} = (100 \pm 10)$ kips and $(F_{XN})_{TW} = (100 \pm 10)$ kips, the three forces are assumed to be approximately equal to each other. As a result, if the top actuators are to be moved they are all extended or retracted the same amount.

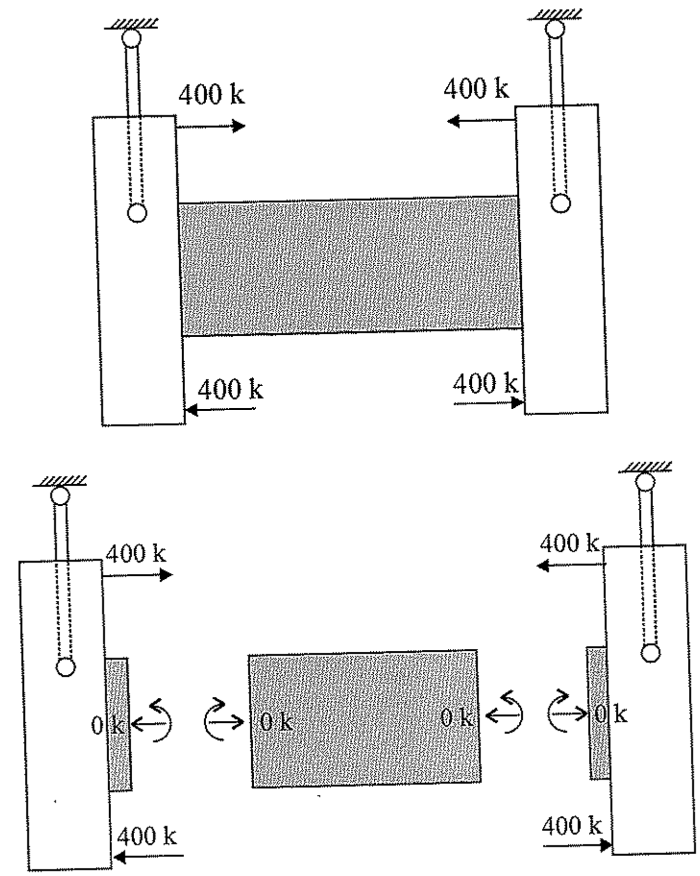
If the forces in the top actuators are not equal to each other within the tolerance, then the forces are considered imbalanced. This may occur due to localized softening, or a reduction in stiffness at some location in the test specimen. The result is moment and rotation about the Y axis. Figure 3.36 shows two types of imbalances that may occur. In Figure 3.36(a), the forces in the outer actuators are approximately equal to each other, while the force in the center actuator is larger. If the goal is to increase the rotation then the outer actuators should be retracted, thus increasing their forces. As long as actuators TE and TW are moved the same amount, the requirement that $\theta_{YN} - \theta_{YS} = 0$ is still satisfied. If the goal is to decrease the rotation the center actuator should be extended, this decreasing its force. Again, this will satisfy $\theta_{YN} - \theta_{YS} = 0$. The same process can be used if the force in the center actuator is smaller than those in the outer two.

In Figure 3.36(b), the forces in actuators TE and TC are equal, but the force in actuator TW is larger. In order to eliminate the imbalance of forces, actuator TW must be moved on its own or actuators TE and TC must be moved simultaneously. However, this type of movement causes a violation of the condition that $\theta_{YN} - \theta_{YS} = 0$. In this situation, all actuators must be moved. If the forces are different in all three actuators, the same problem occurs and all actuators should be extended or retracted. The imbalance of forces cannot be corrected unless the forces in the outer actuators are equal. Note that moving the outer actuators only or the center actuator only violates the assumption that the grillages are rigid, because doing so forces them to deform. However, it is recognized that the grillages are going to deform in any event, and that these deformations are small.

The decision algorithm is shown in Figure 3.37. First, it is determined from user input whether the primary rotation θ_Z is increasing (the hull structure is being loaded) or decreasing (the hull structure is being unloaded). Then, it is determined whether the axial force is compressive or tensile. As shown in the figure, if the rotation is increasing and the axial force is compressive, the are actuators are extended. If the axial force is compressive, the action depends on whether the user has specified that top actuator force imbalances should be corrected, or that the top actuators should all be moved the same amount, regardless of the existence of an imbalance of forces. If the user specified that the imbalance should be corrected, then at this point the forces in the outer actuators are compared. If they are not equal (within the tolerance), all actuators are retracted. If they are equal within the tolerance, then the force in the center actuator is checked. Depending on its value, various decisions are made, as shown in Figure 3.37.



(a) Hull structure under net axial compression



(b) No net axial force

Figure 3.1 Illustration of axial force correction

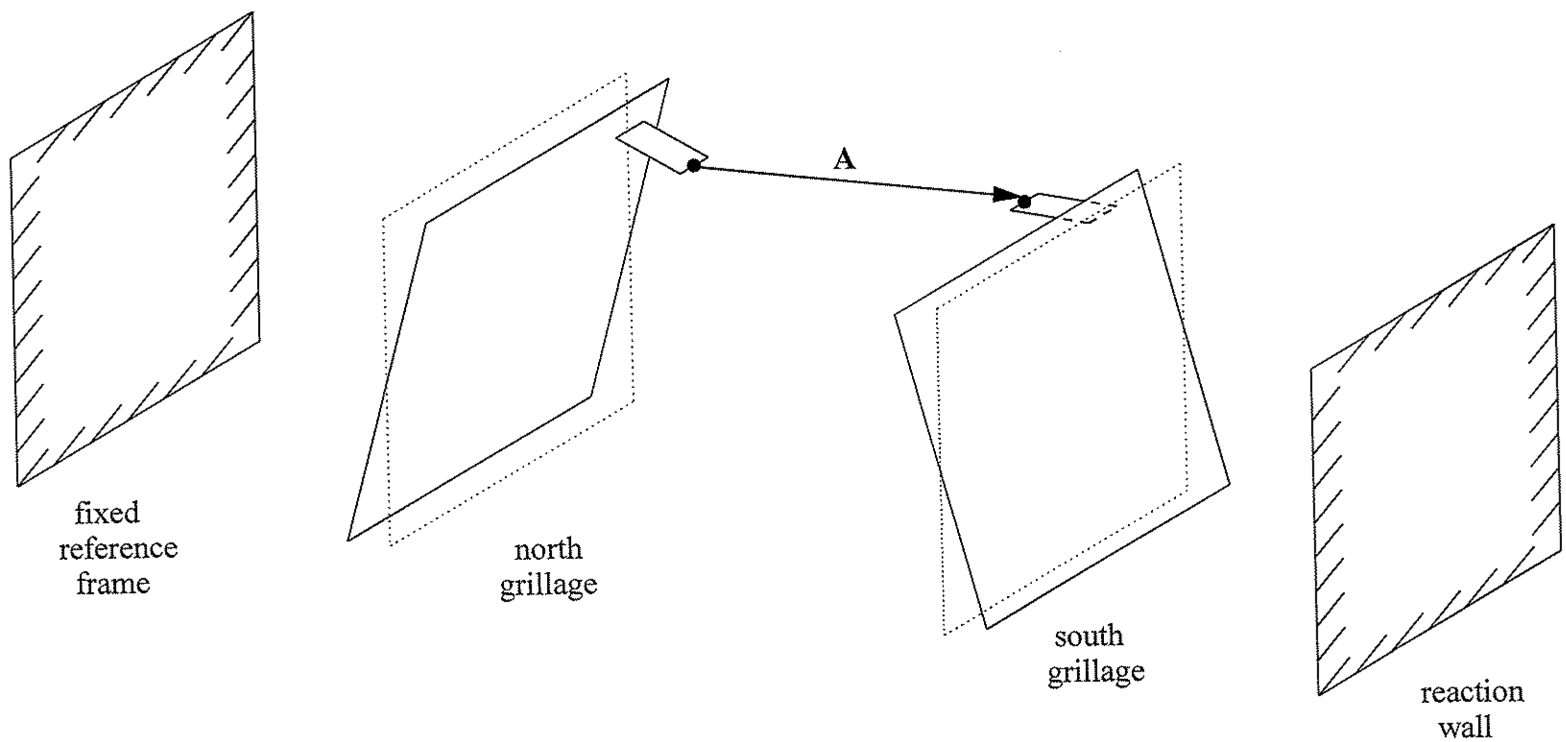


Figure 3.2 Representation of an actuator by a position vector **A**

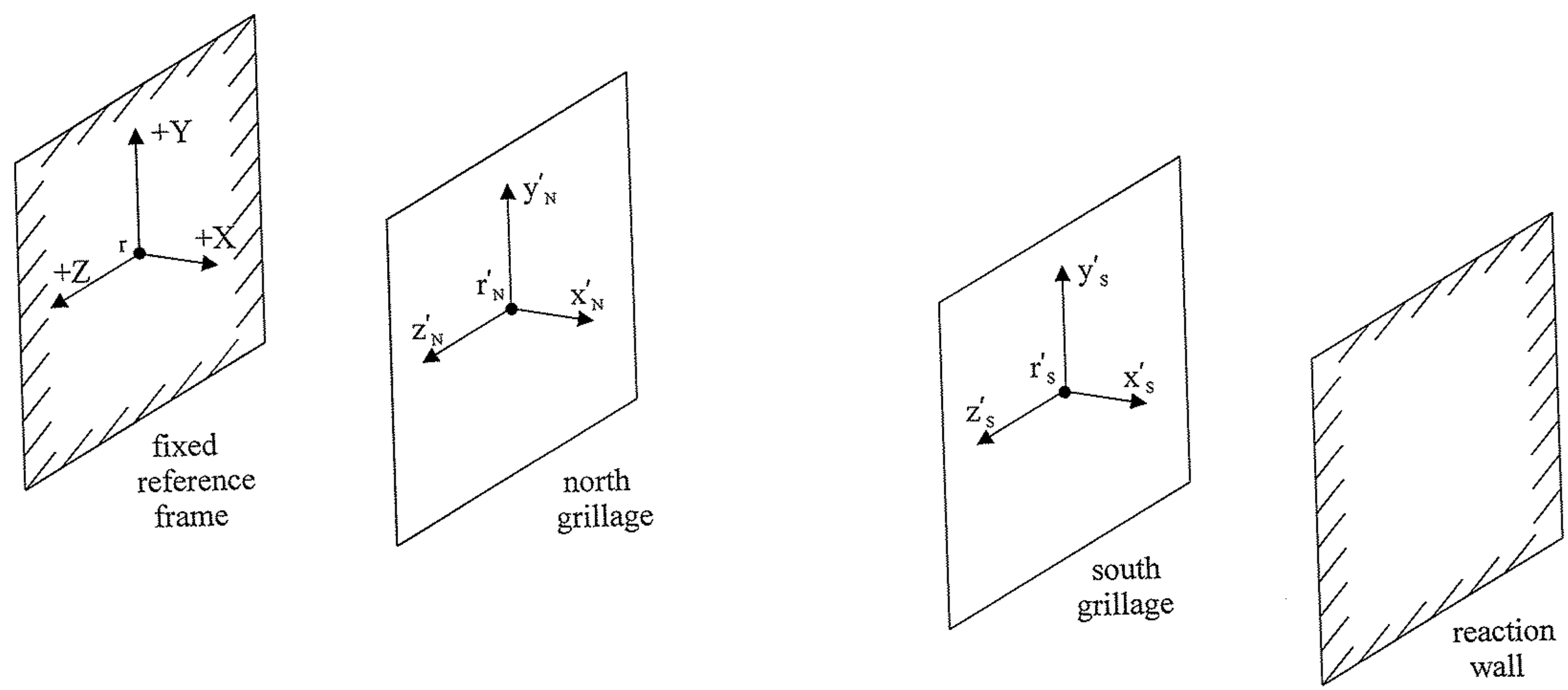


Figure 3.3 Coordinate systems

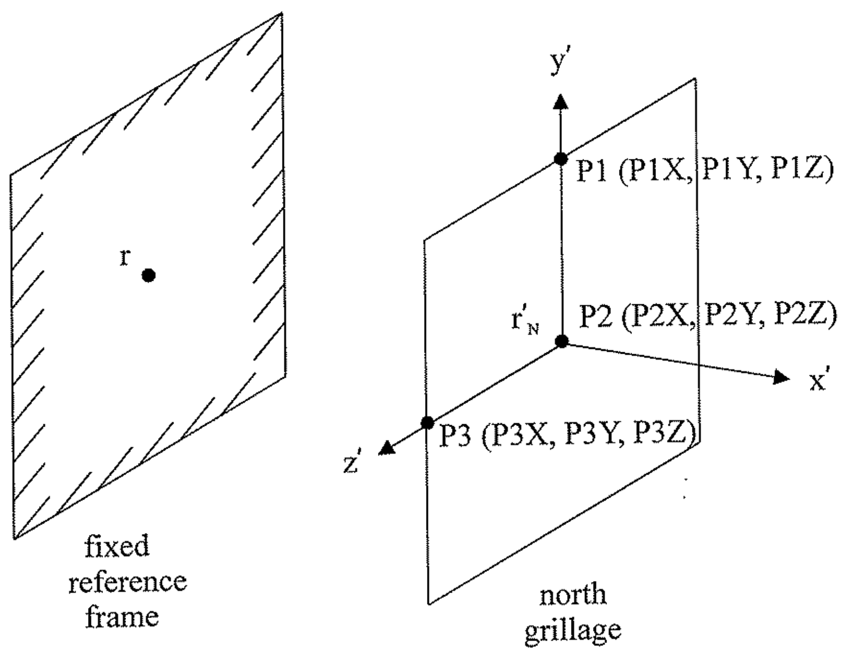


Figure 3.4 Location of points P1, P2, and P3

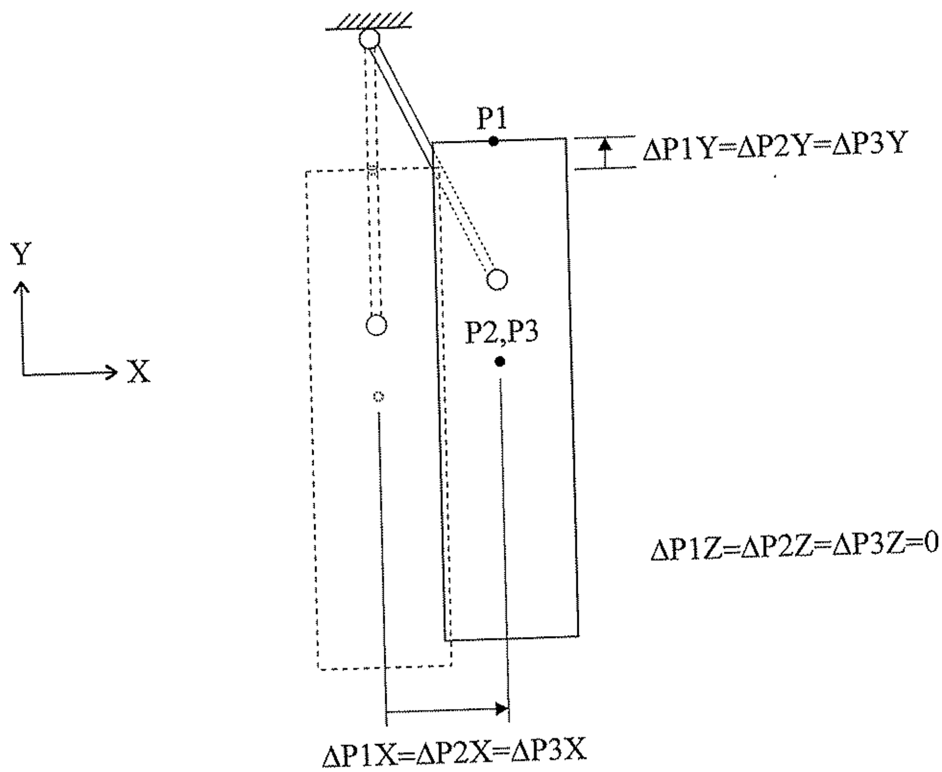


Figure 3.5 Case 1: Grillage translations in local x′ and y′ directions

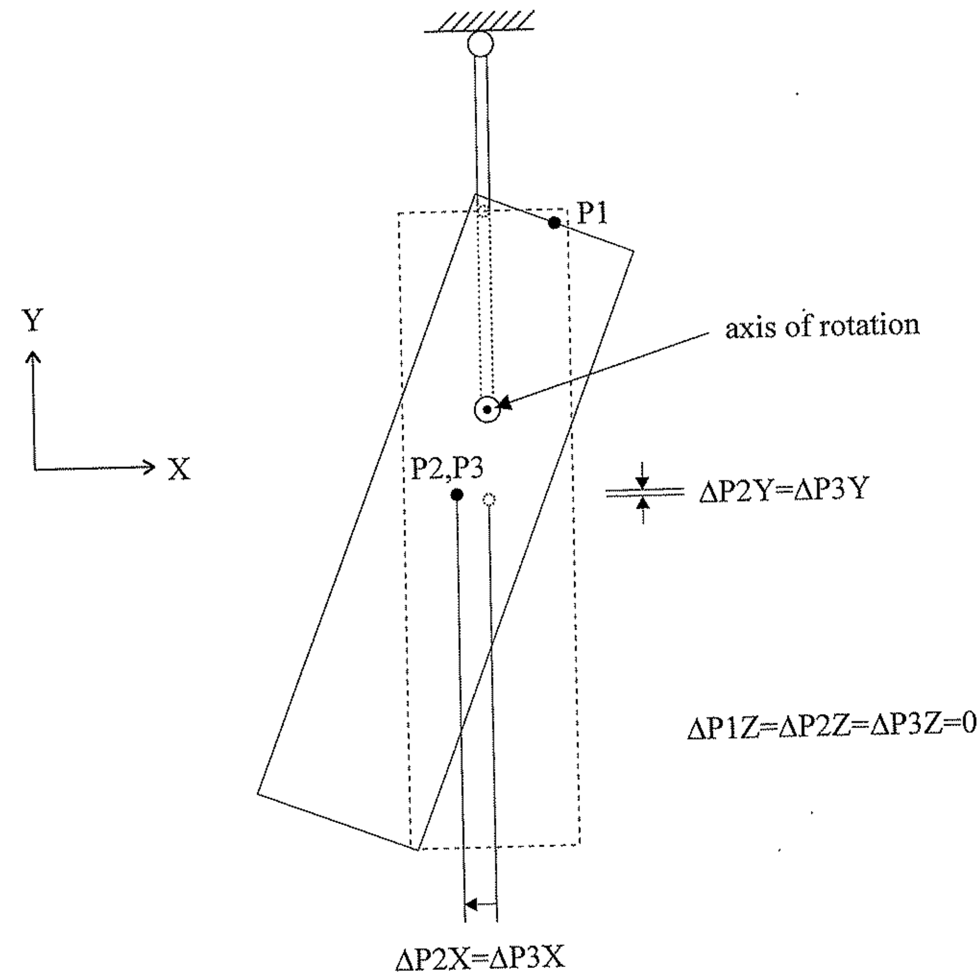
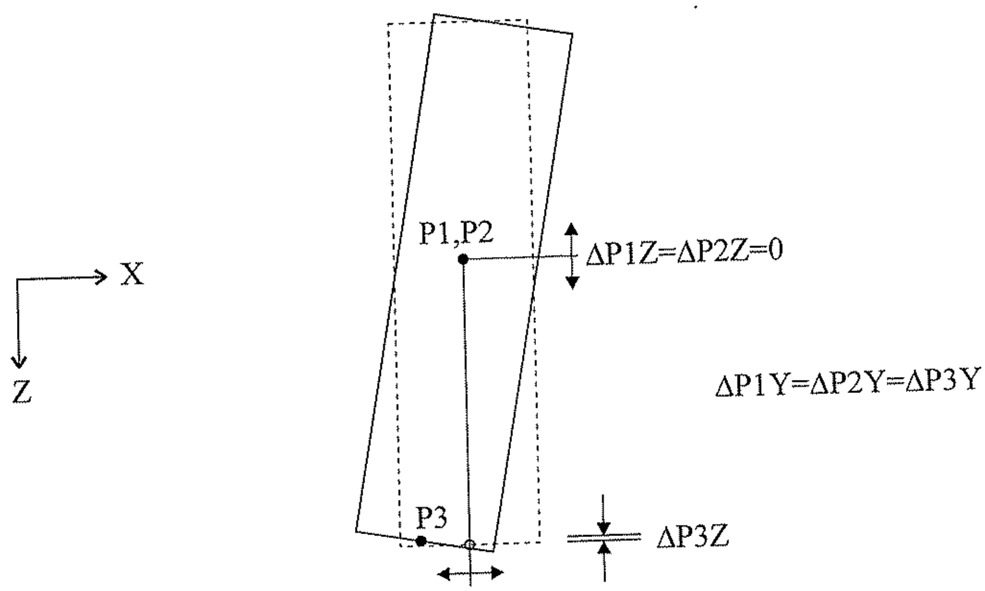
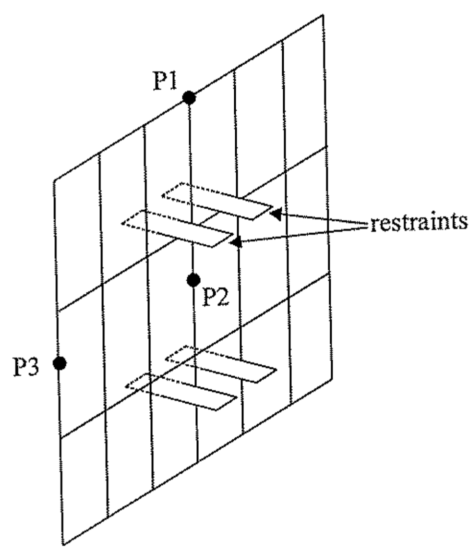


Figure 3.6 Case 2: Rotation of grillage in x'y' plane



(a) Rotation of grillage about its local y′ axis



(b) Restraints that disallow Δz′

Figure 3.7 Case 3

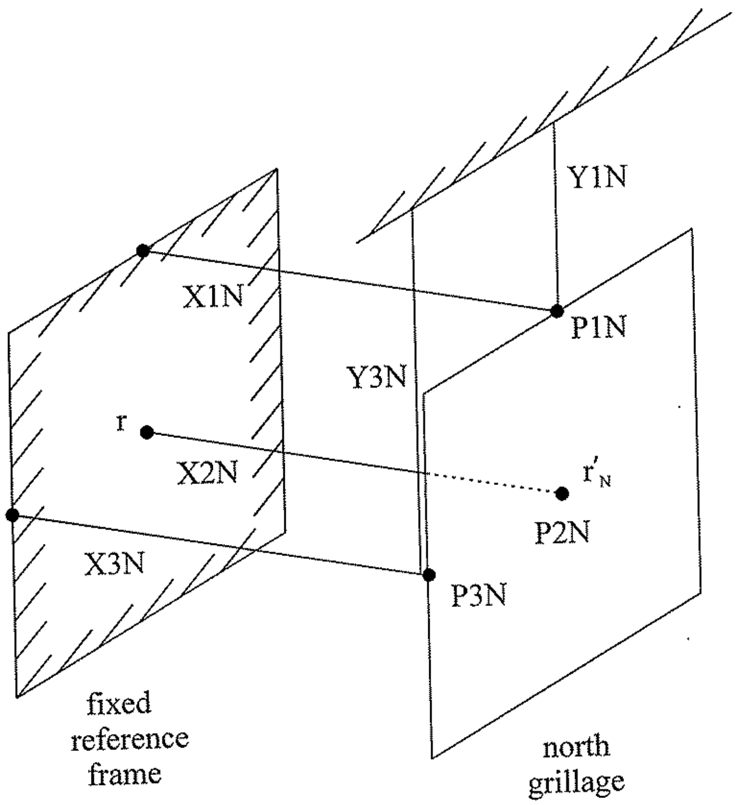


Figure 3.8 Configuration of north grillage transducers

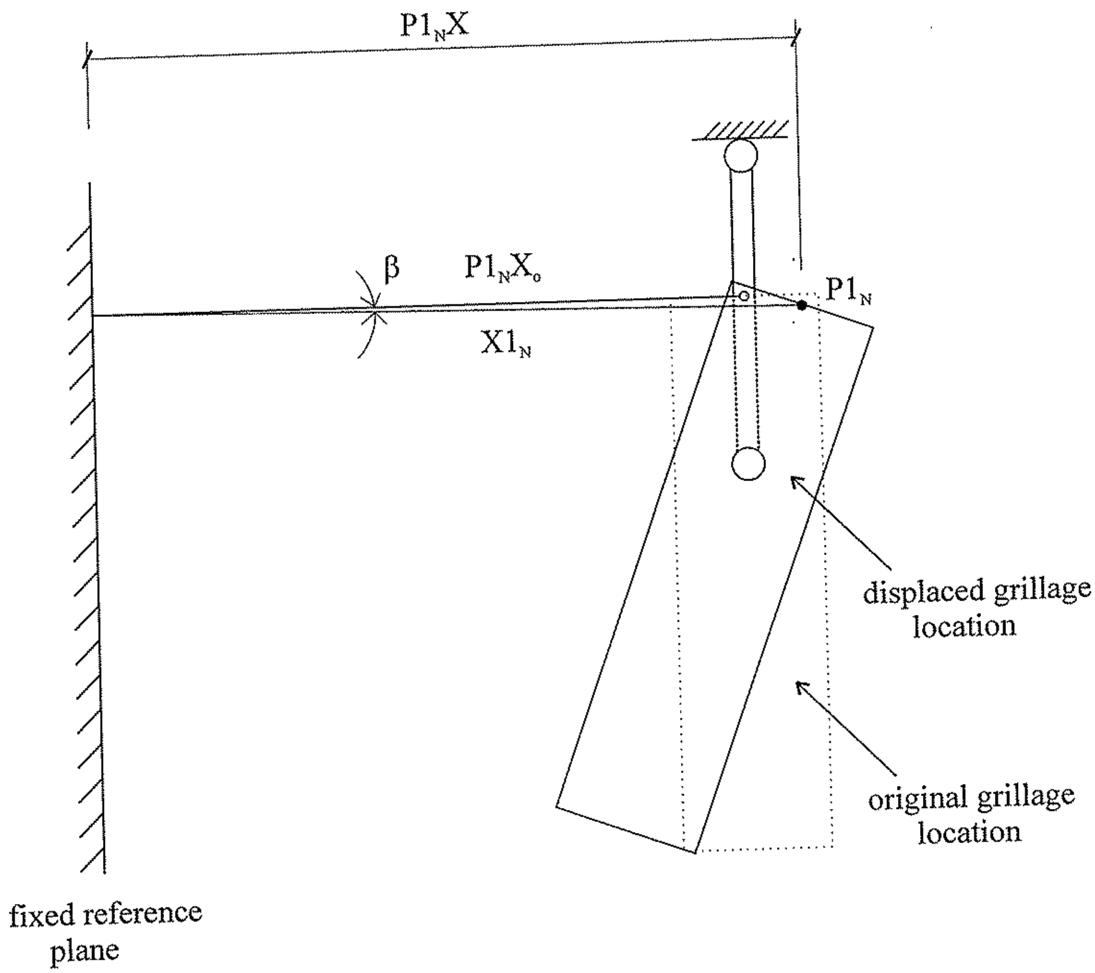


Figure 3.9 Approximation for $P1_NX$

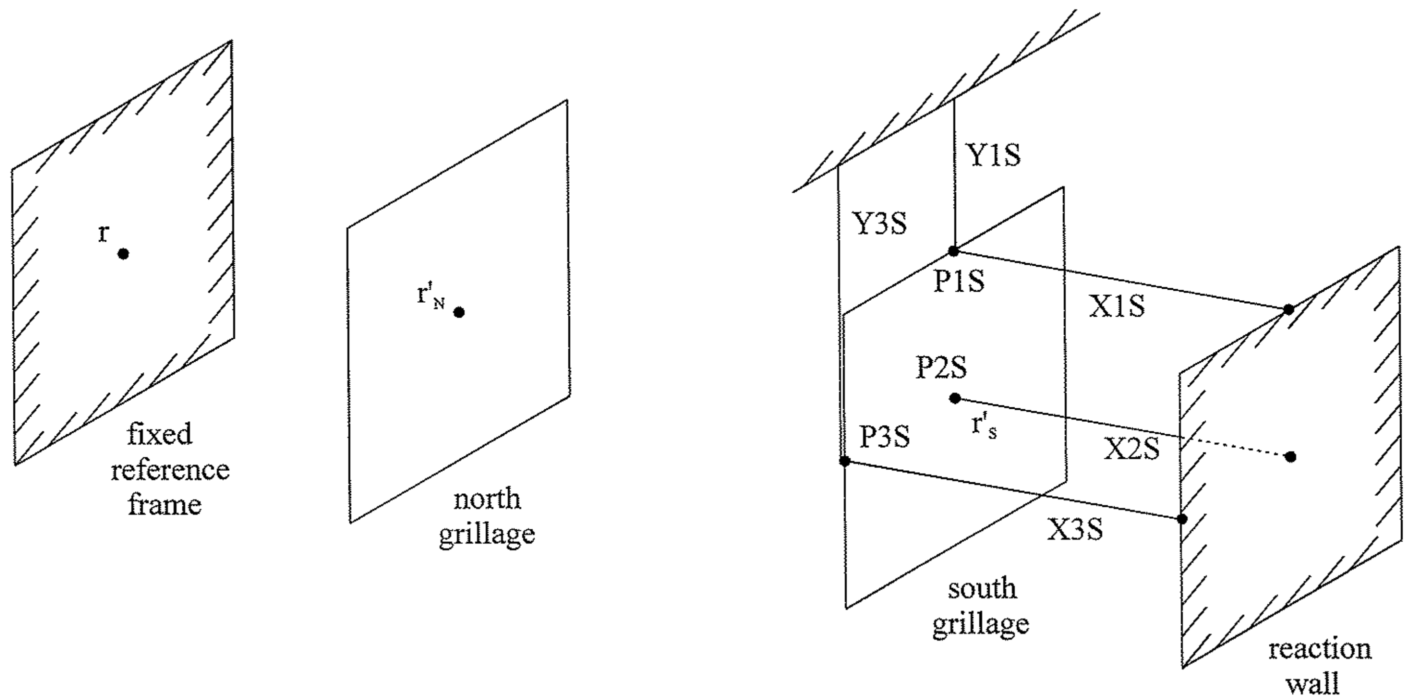


Figure 3.10 Configuration of south grillage transducers

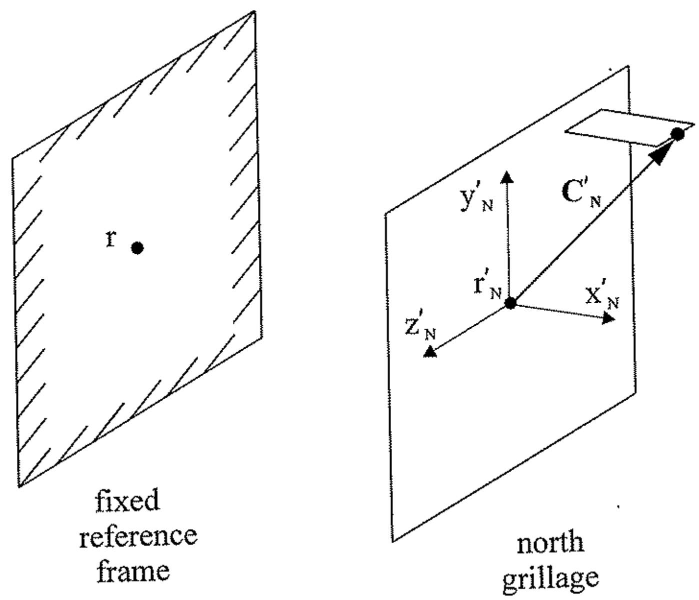
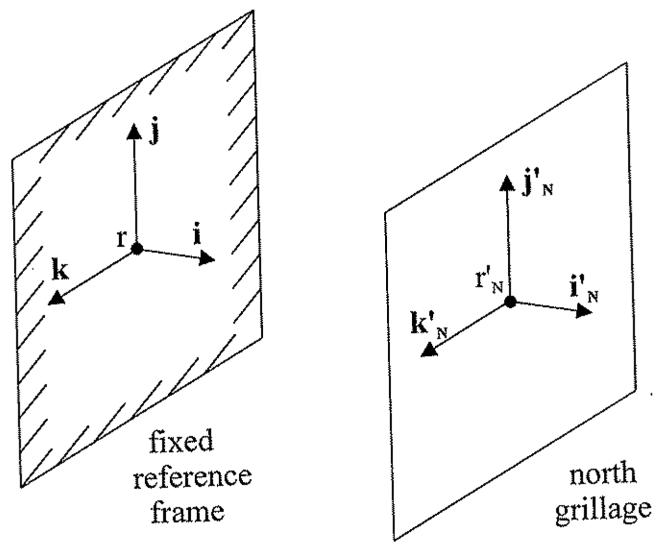
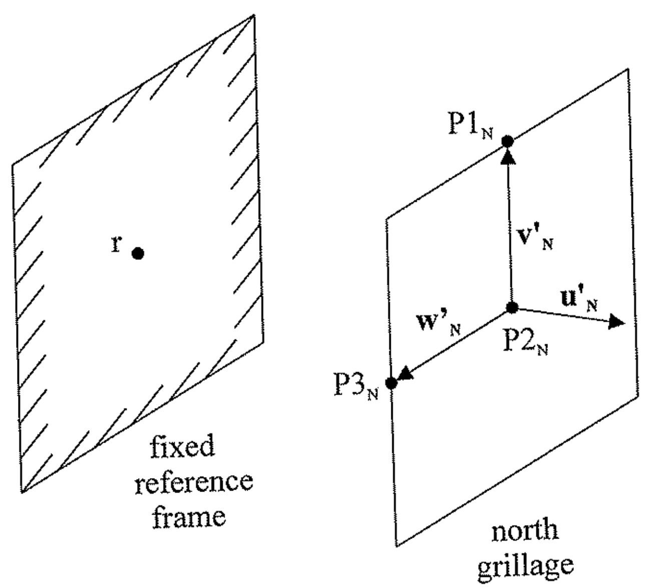


Figure 3.11 $\mathbf{C'_N}$ vector

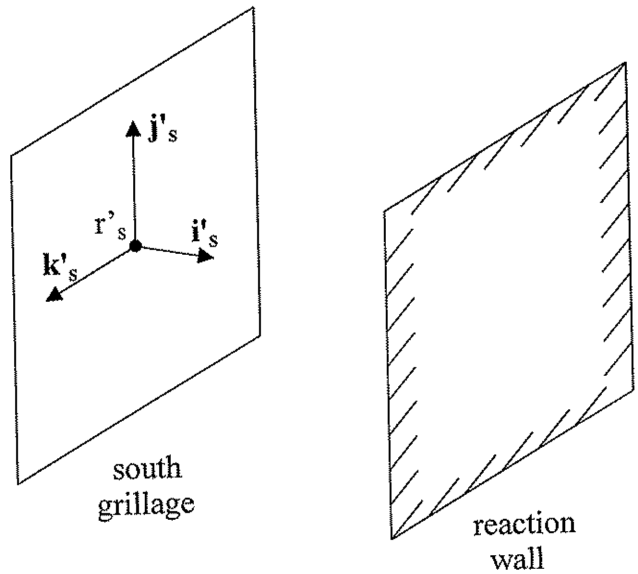


(a) North grillage basis

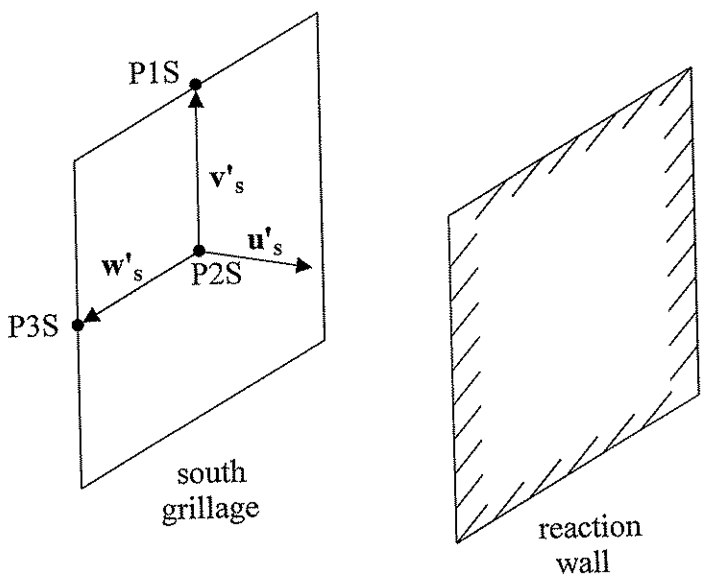


(b) North grillage transformation vectors

Figure 3.12 North grillage vectors



(a) South grillage basis



(b) South grillage transformation vectors

Figure 3.13 South grillage vectors

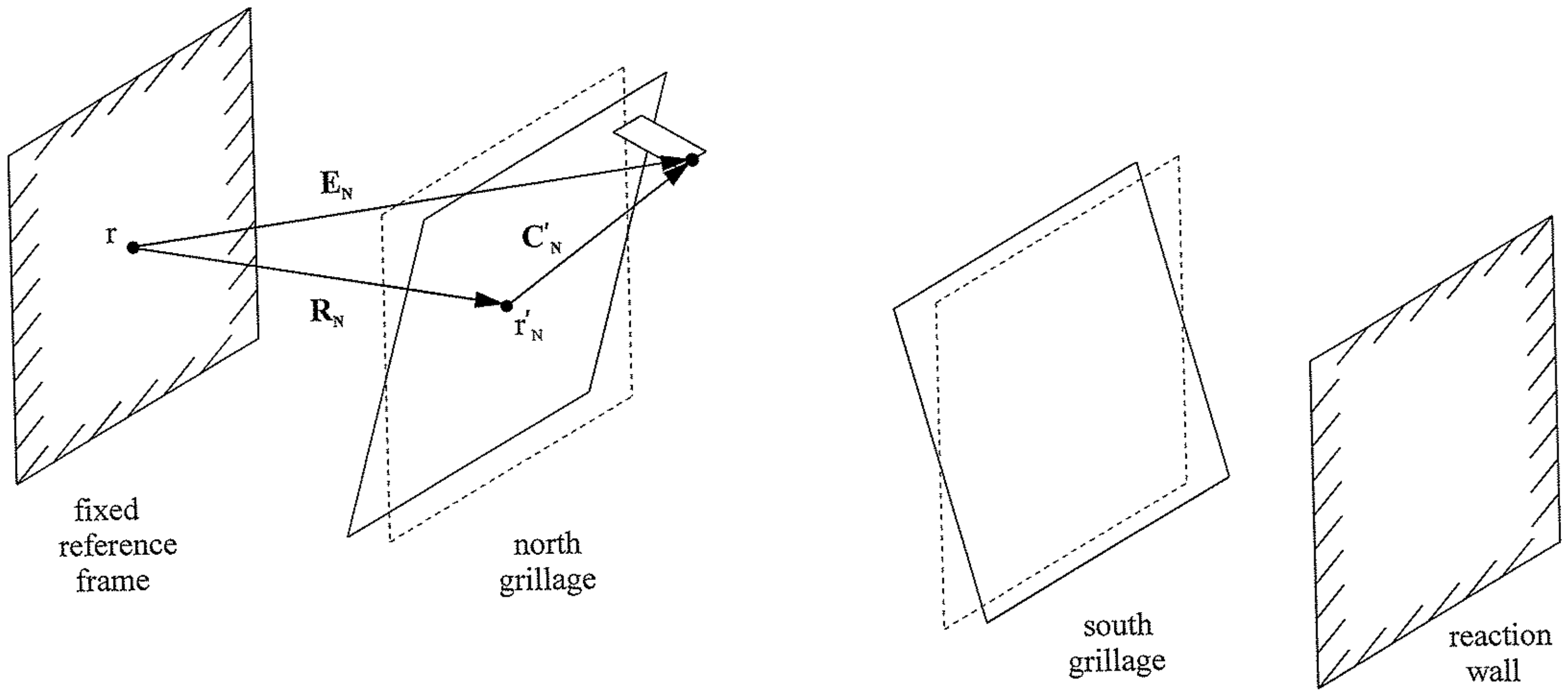


Figure 3.14 Representation of $\mathbf{E}_N$ vector for top actuators

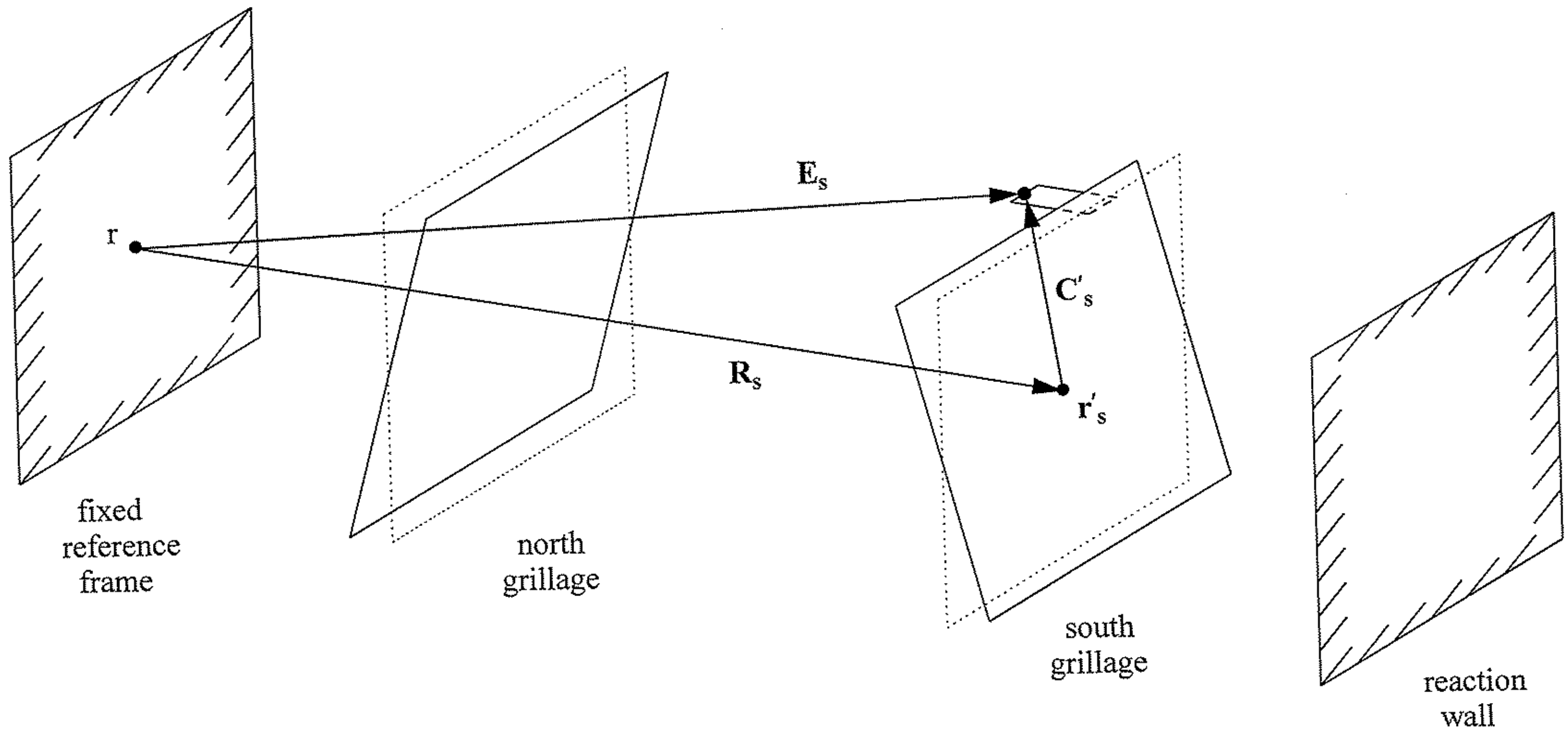


Figure 3.15 Representation of $\mathbf{E}_s$ vector for top actuators

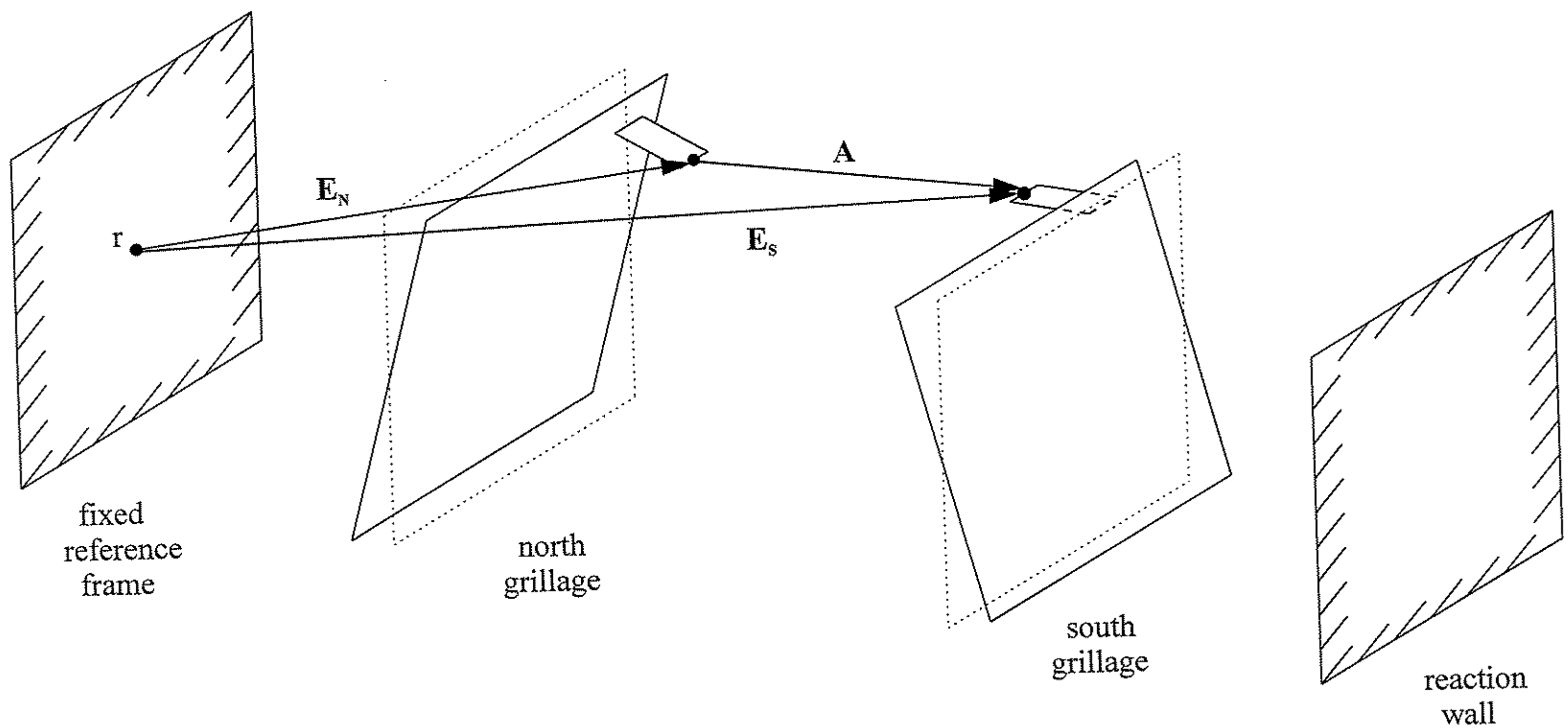


Figure 3.16 Representation of **A** vector for top actuators

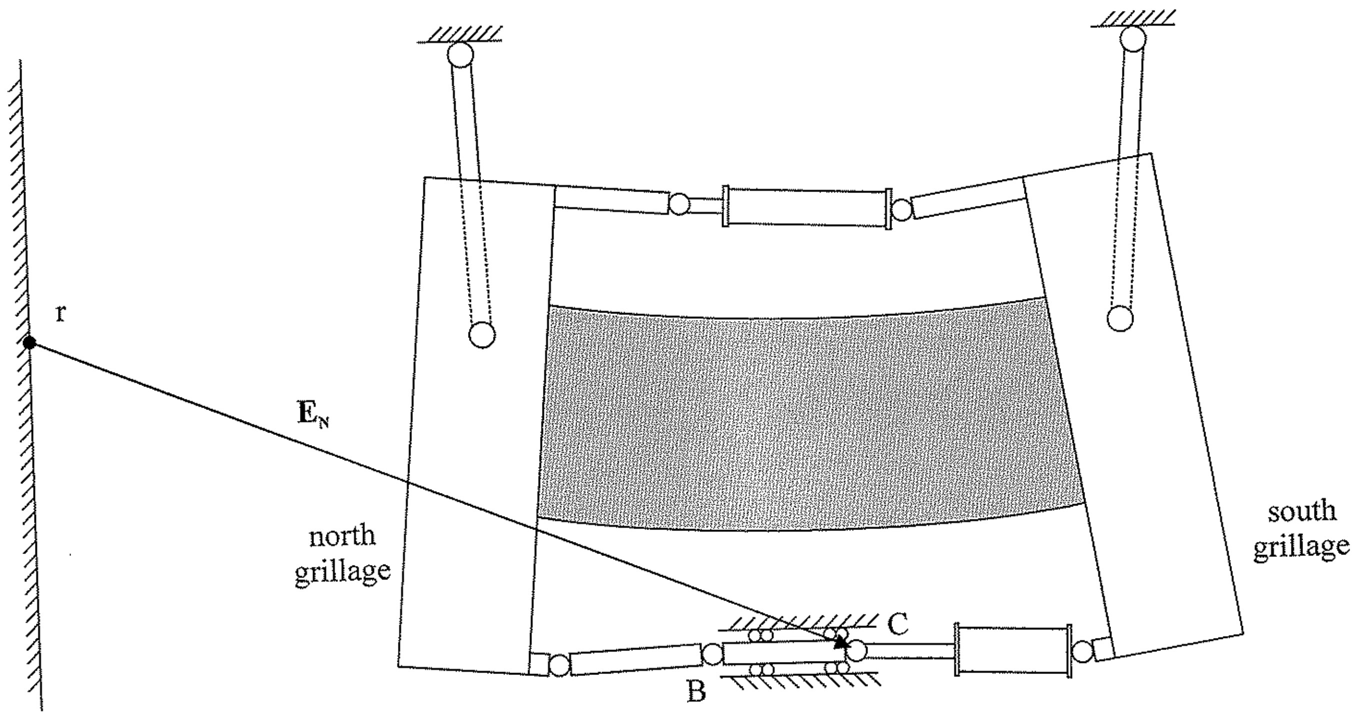


Figure 3.17 Representation of $\mathbf{E}_N$ vector for bottom actuators

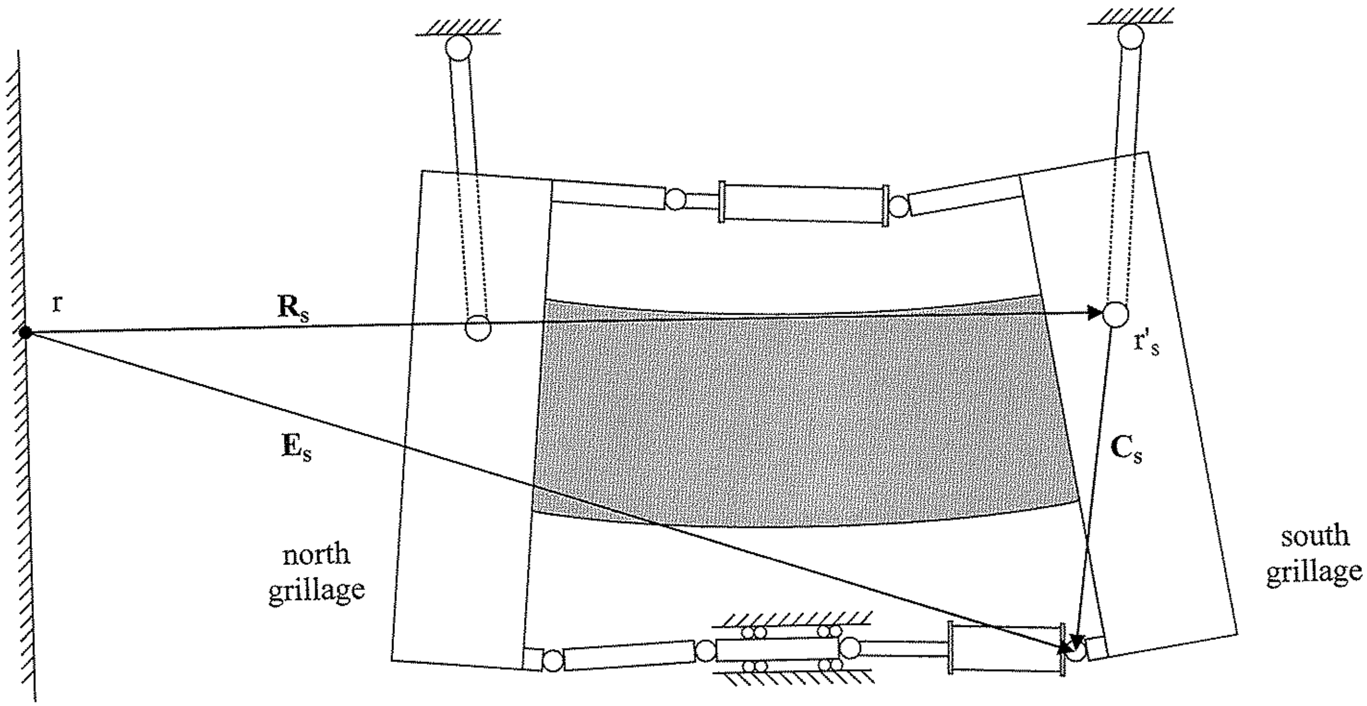


Figure 3.18 Representation of $\mathbf{E}_s$ vector for bottom actuators

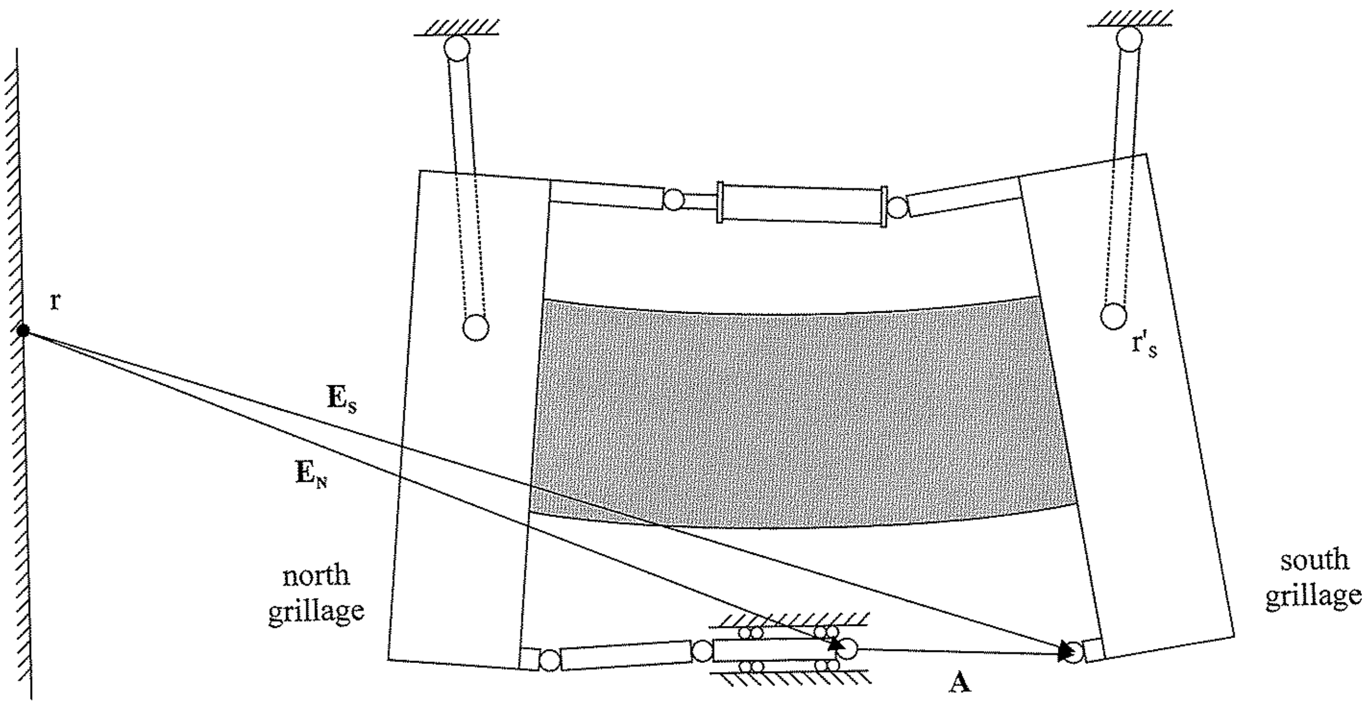


Figure 3.19 Representation of **A** vector for bottom actuators

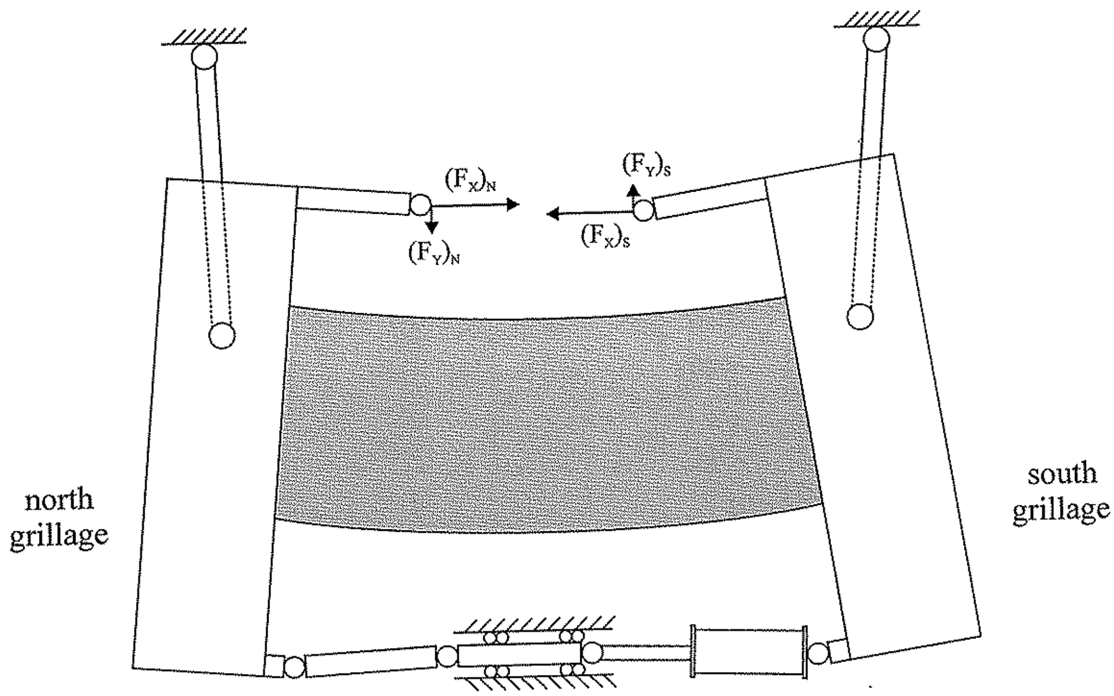


Figure 3.20 Top actuator force components

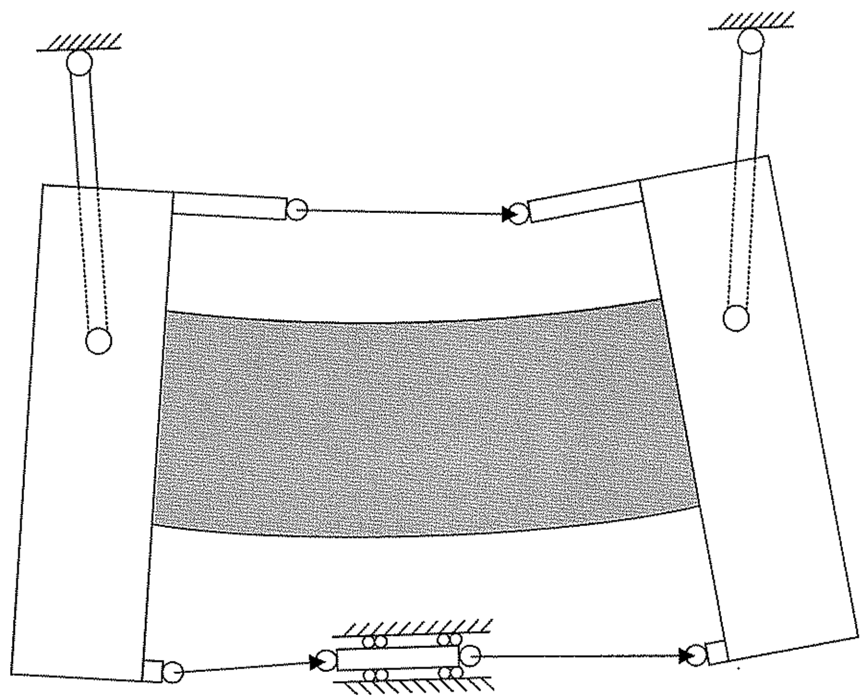
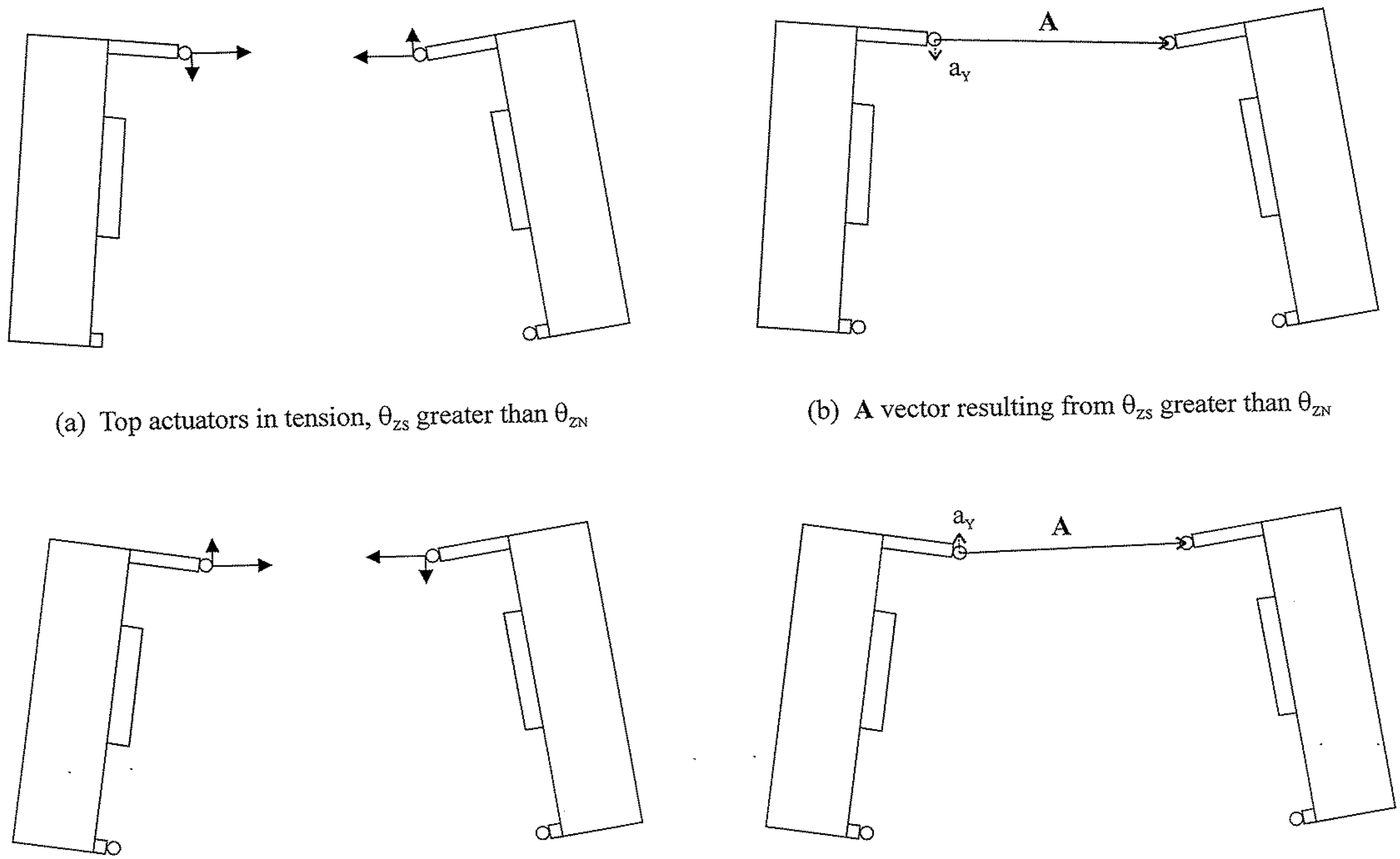


Figure 3.21 **A** vectors resulting from $\theta_{ZS} > \theta_{ZN}$



(a) Top actuators in tension, θ_{ZS} greater than θ_{ZN}

(b) **A** vector resulting from θ_{ZS} greater than θ_{ZN}

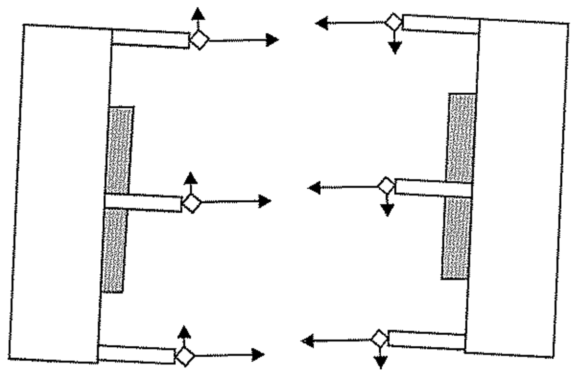
(b) Top actuators in tension, θ_{ZN} greater than θ_{ZS}

(d) **A** vector resulting from θ_{ZN} greater than θ_{ZS}

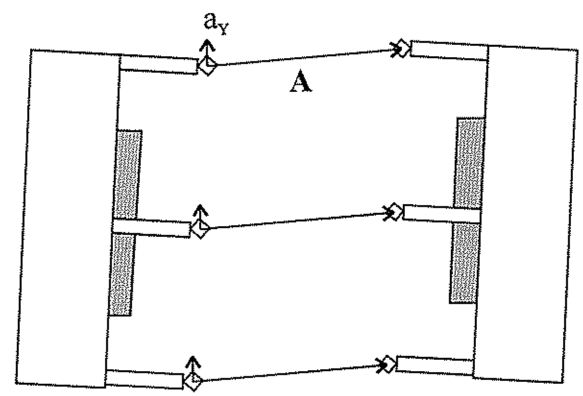
Figure 3.22 Determining direction of Y component of force for top actuators

Total Force	Sign of F_{TOT}	Geometric Condition	Sign of a_Y	Actual Sign of F_Y	Sign of F_Y using Eq.(5)
Tension	+	$\theta_{ZN}>\theta_{ZS}$	+	+	+
	+	$\theta_{ZS}>\theta_{ZN}$	-	-	-
Compression	-	$\theta_{ZN}>\theta_{ZS}$	+	-	-
	-	$\theta_{ZS}>\theta_{ZN}$	-	+	+

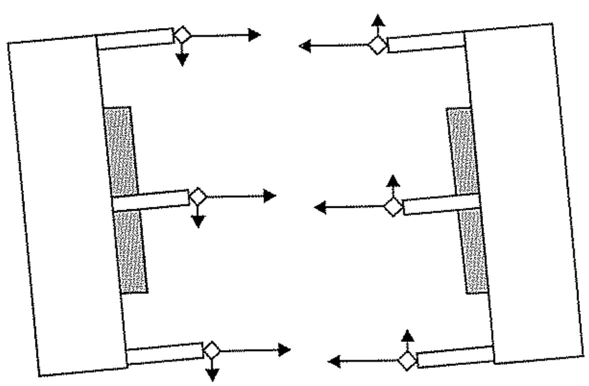
Figure 3.23 Determining sign of Y component of force for top actuators



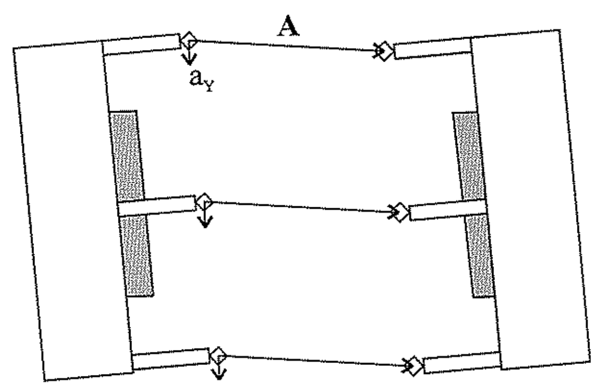
(a) Top actuators in tension, F_Z negative



(b) a_z negative



(c) Top actuators in tension, F_Z positive



(d) a_z positive

Figure 3.24 Determining direction of Z component of force for top actuators

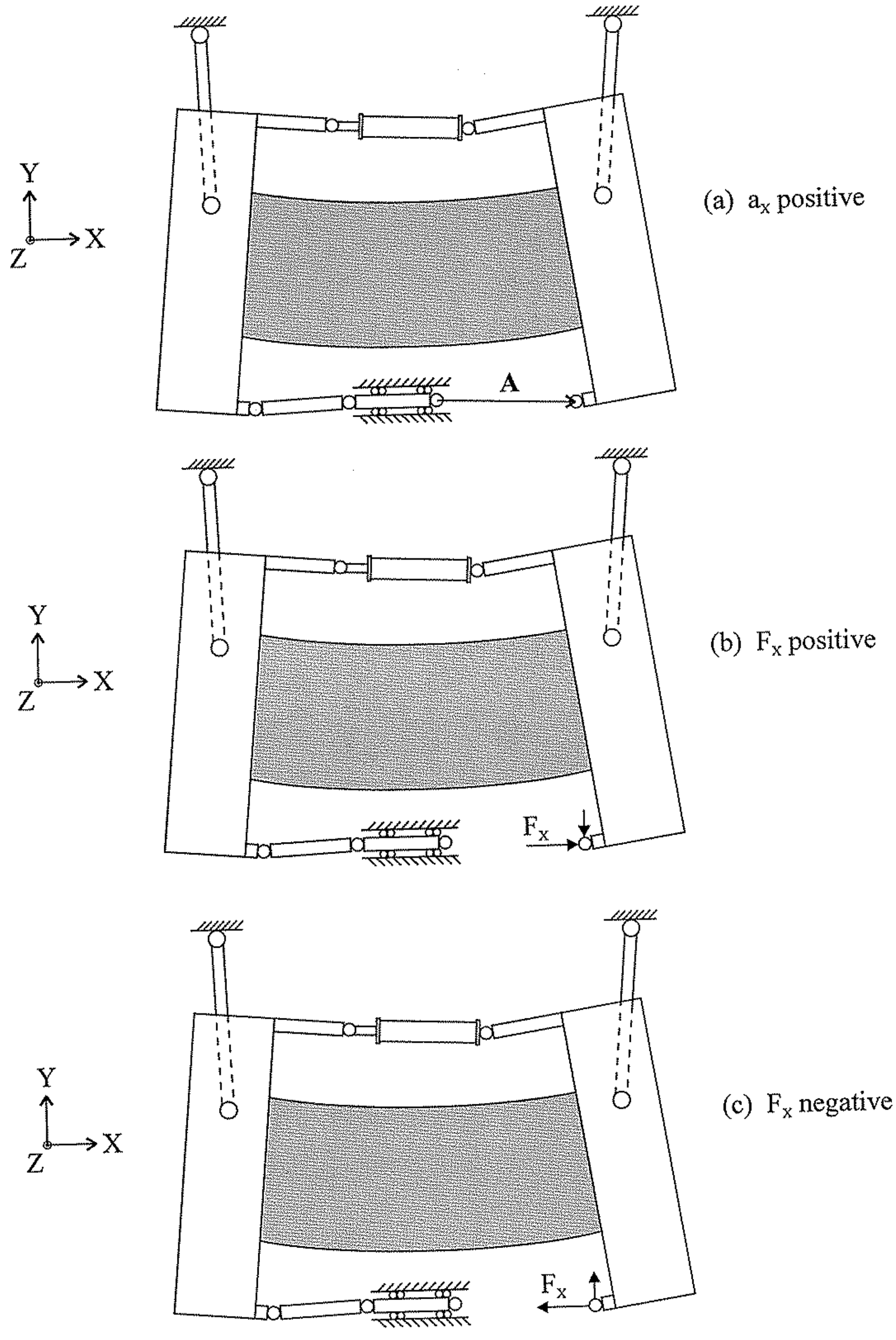


Figure 3.25 Determining direction of X component of force for bottom actuators

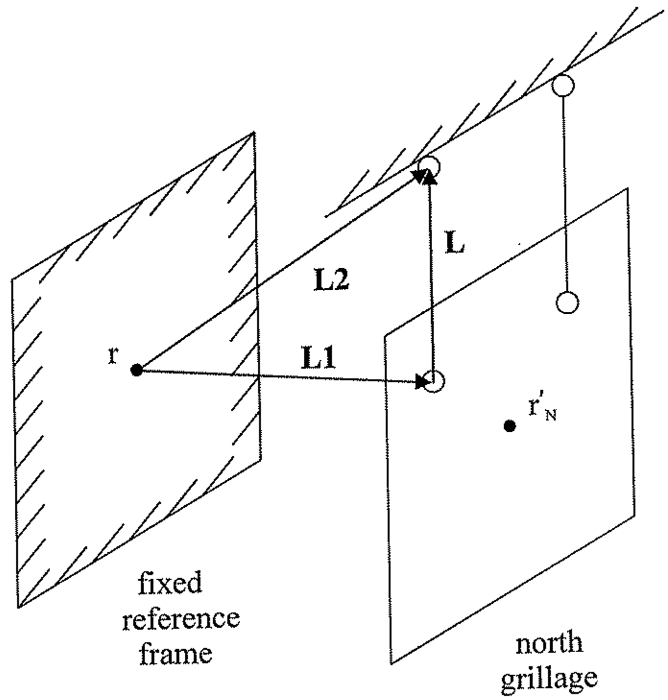


Figure 3.26 **L** vector for vertical links

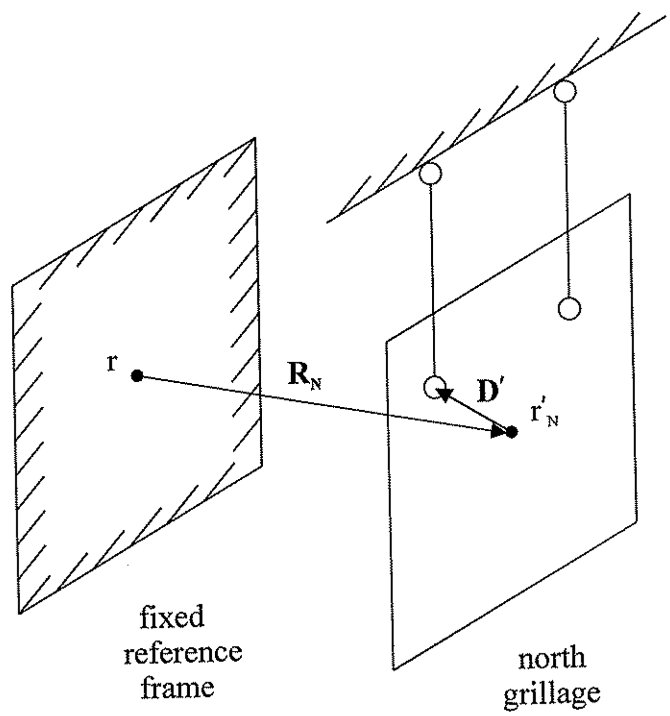
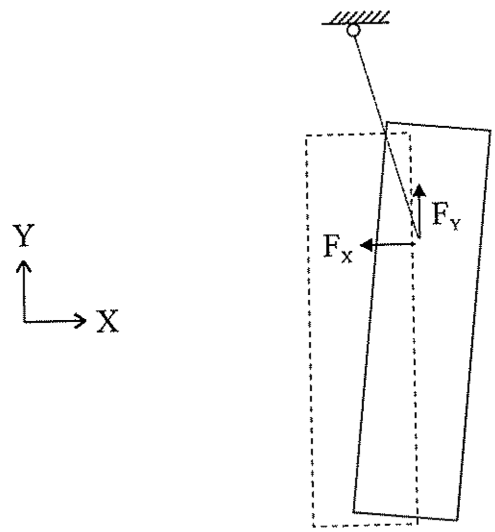
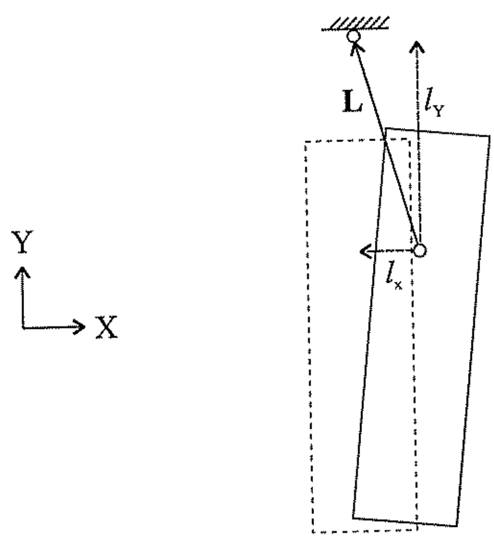


Figure 3.27 **D′** vector for vertical links



(a) Force components for vertical links



(b) **L** vector and its components

Figure 3.28 Vertical link vectors and force components

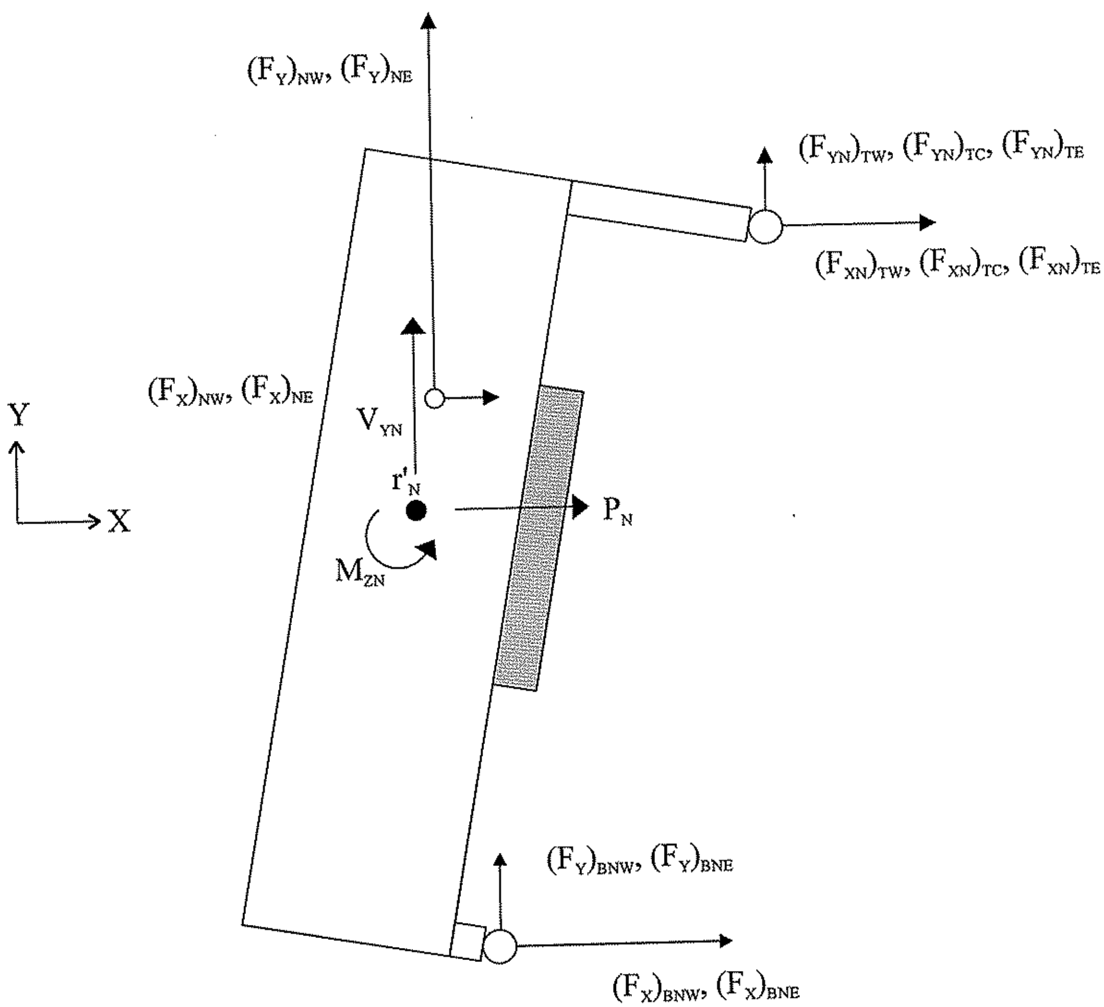


Figure 3.29 XY plane equilibrium at north grillage

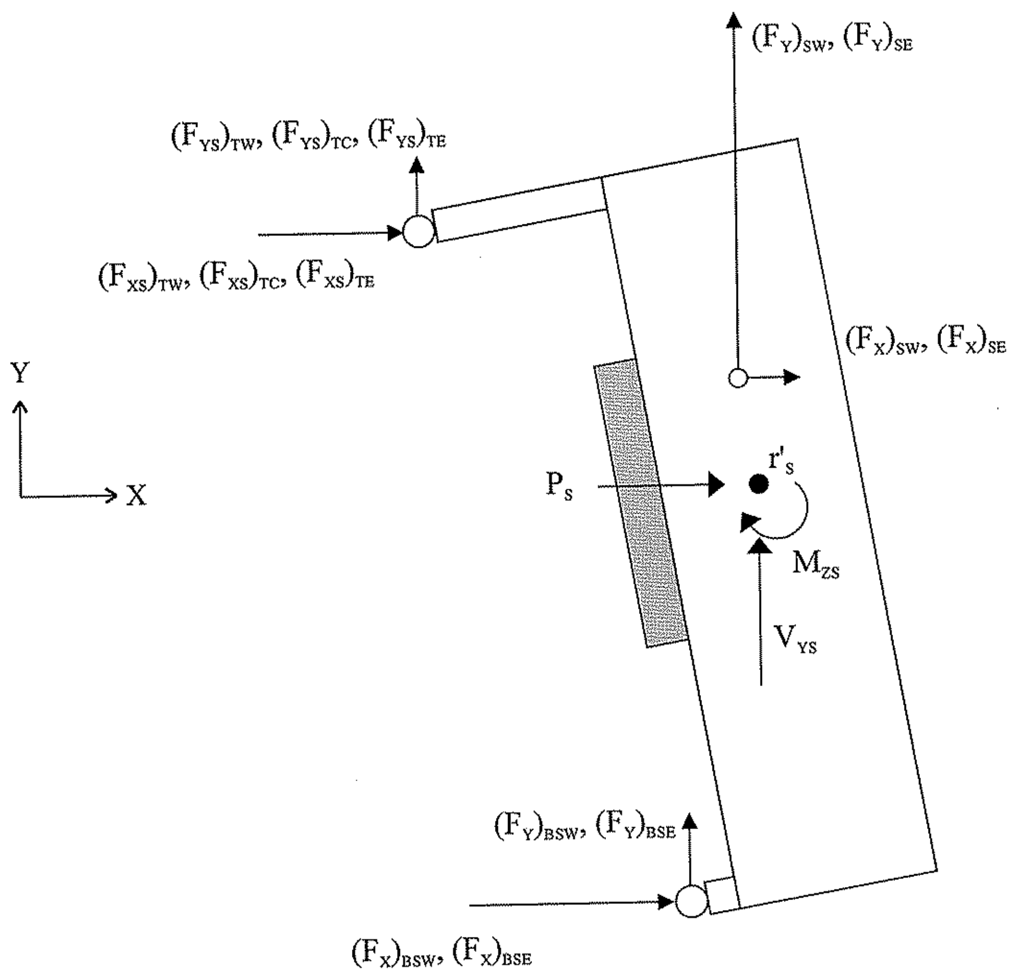


Figure 3.30 XY plane equilibrium at south grillage

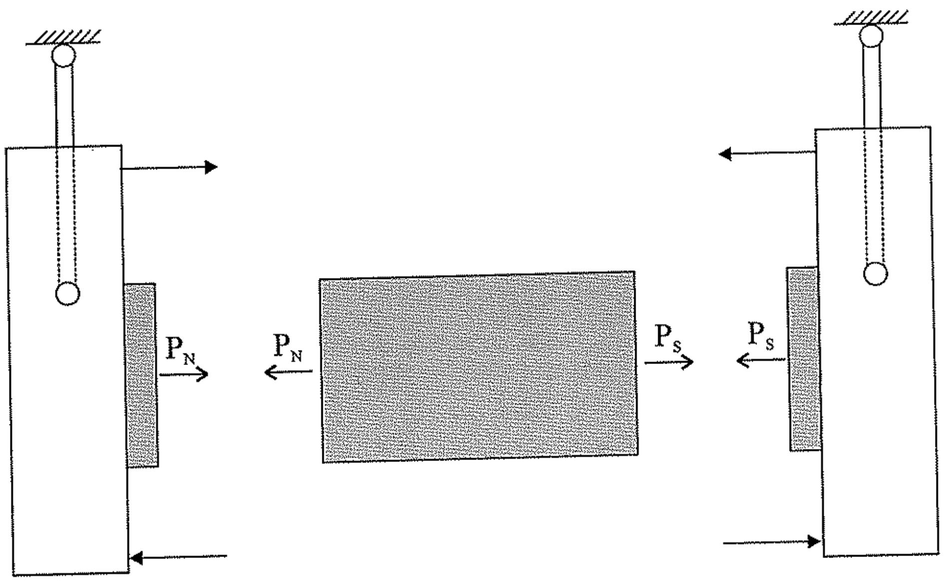


Figure 3.31 P_N and P_S when hull structure is in net tension

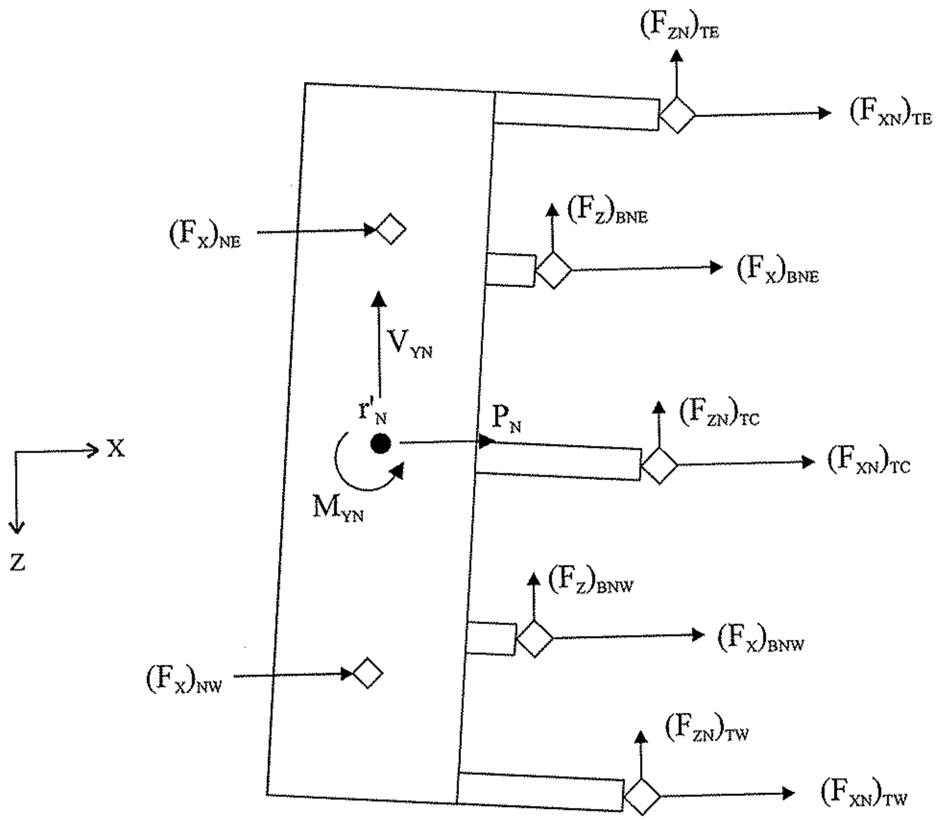


Figure 3.32 XZ plane equilibrium at north grillage

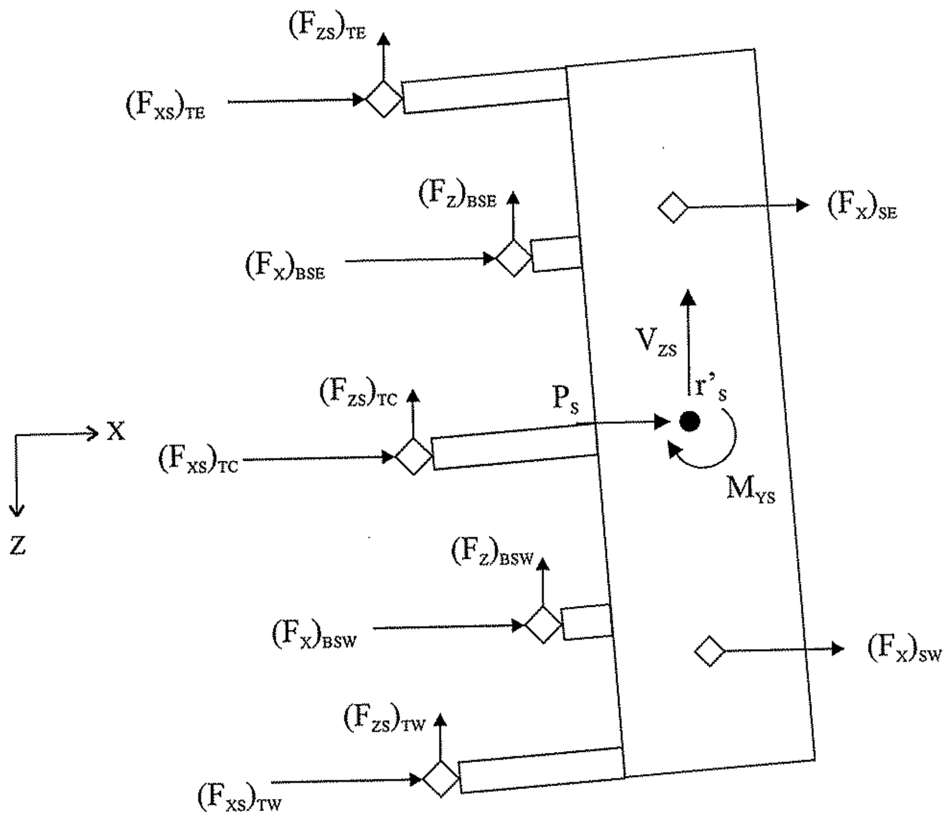
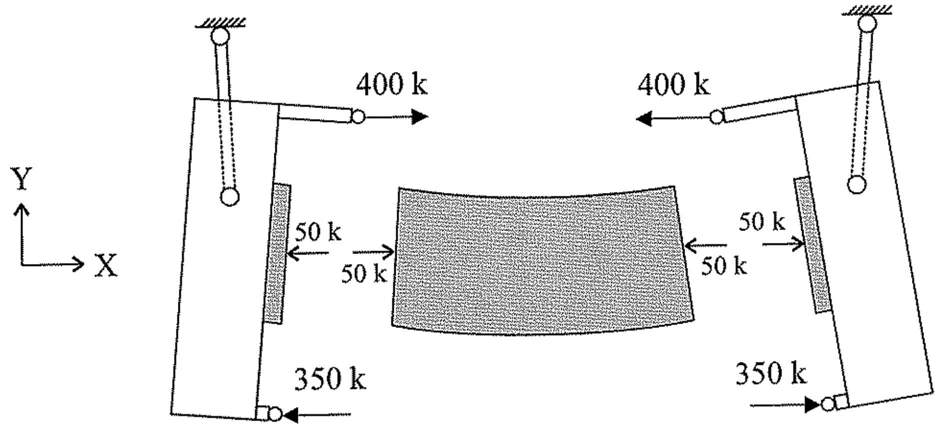
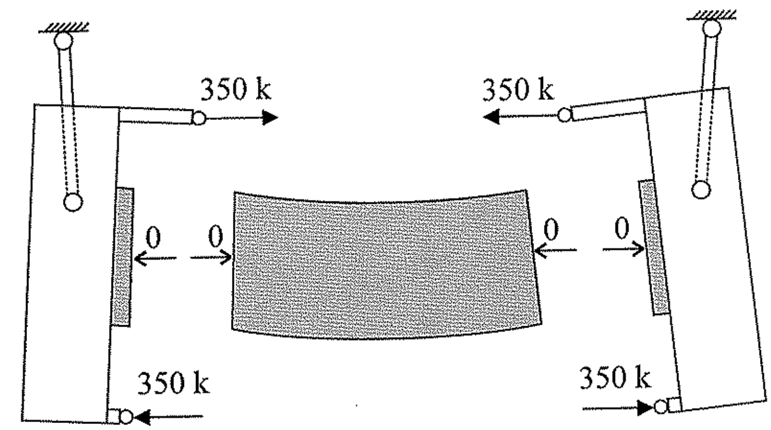


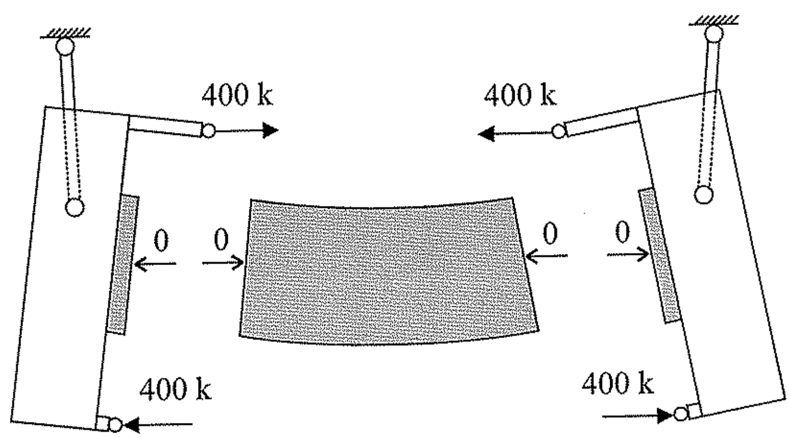
Figure 3.33 XZ plane equilibrium at south grillage



(a) Hull structure under axial compression

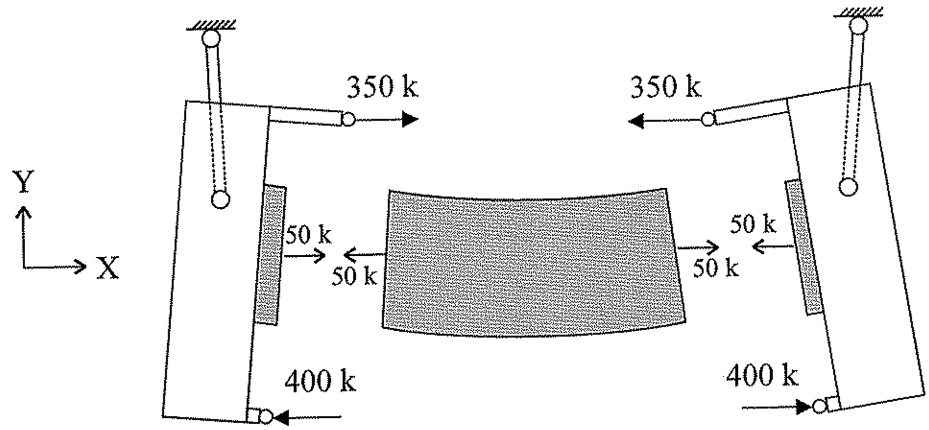


(b) Top actuators extended; rotation θ_z decreased

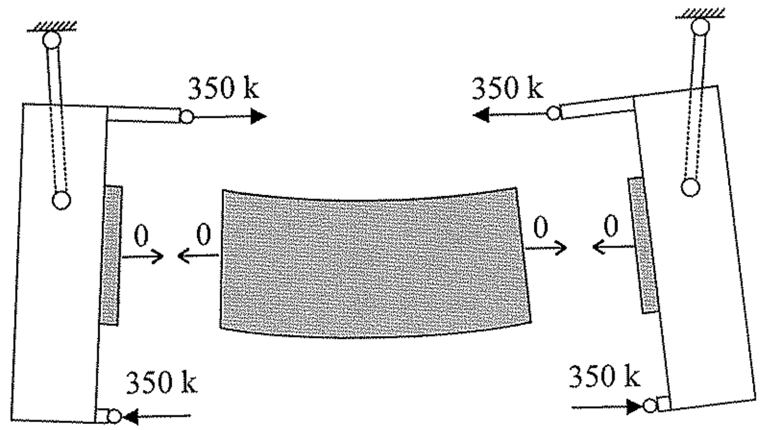


(c) Bottom actuators extended; rotation θ_z increased

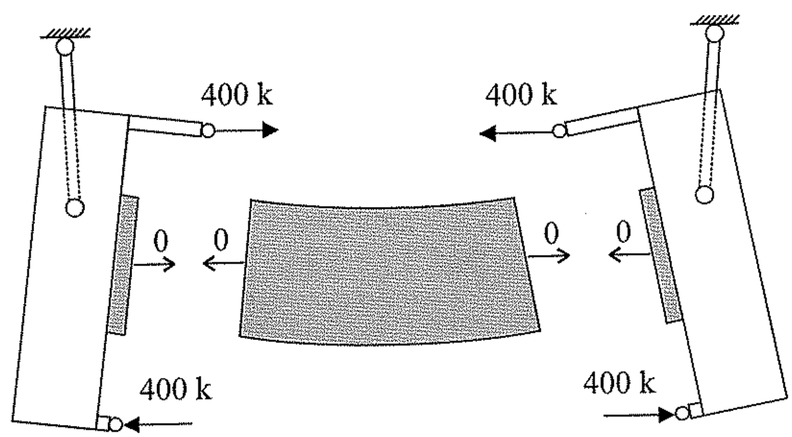
Figure 3.34 Methods for eliminating net compression force in hull



(a) Hull structure under axial tension

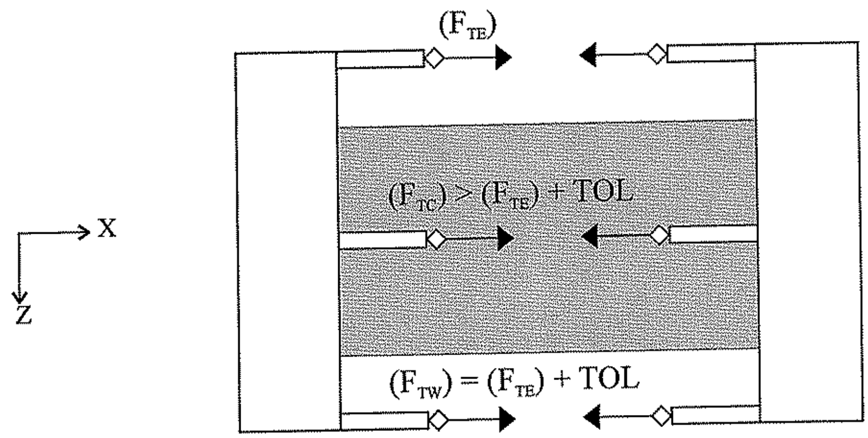


(b) Bottom actuators retracted; rotation θ_z decreased

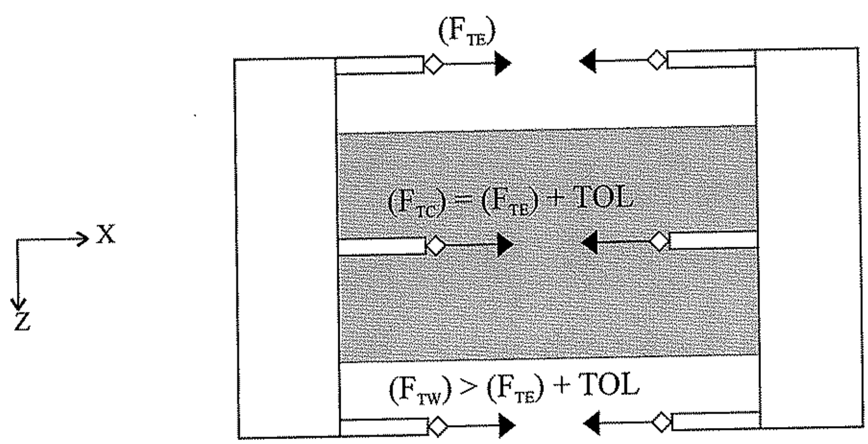


(c) Top actuators retracted; rotation θ_z increased

Figure 3.35 Methods for eliminating net tension force in hull



(a) Forces in outer actuators approximately equal



(b) Forces in actuators TE and TC approximately equal

Figure 3.36 Imbalance of forces between top actuators

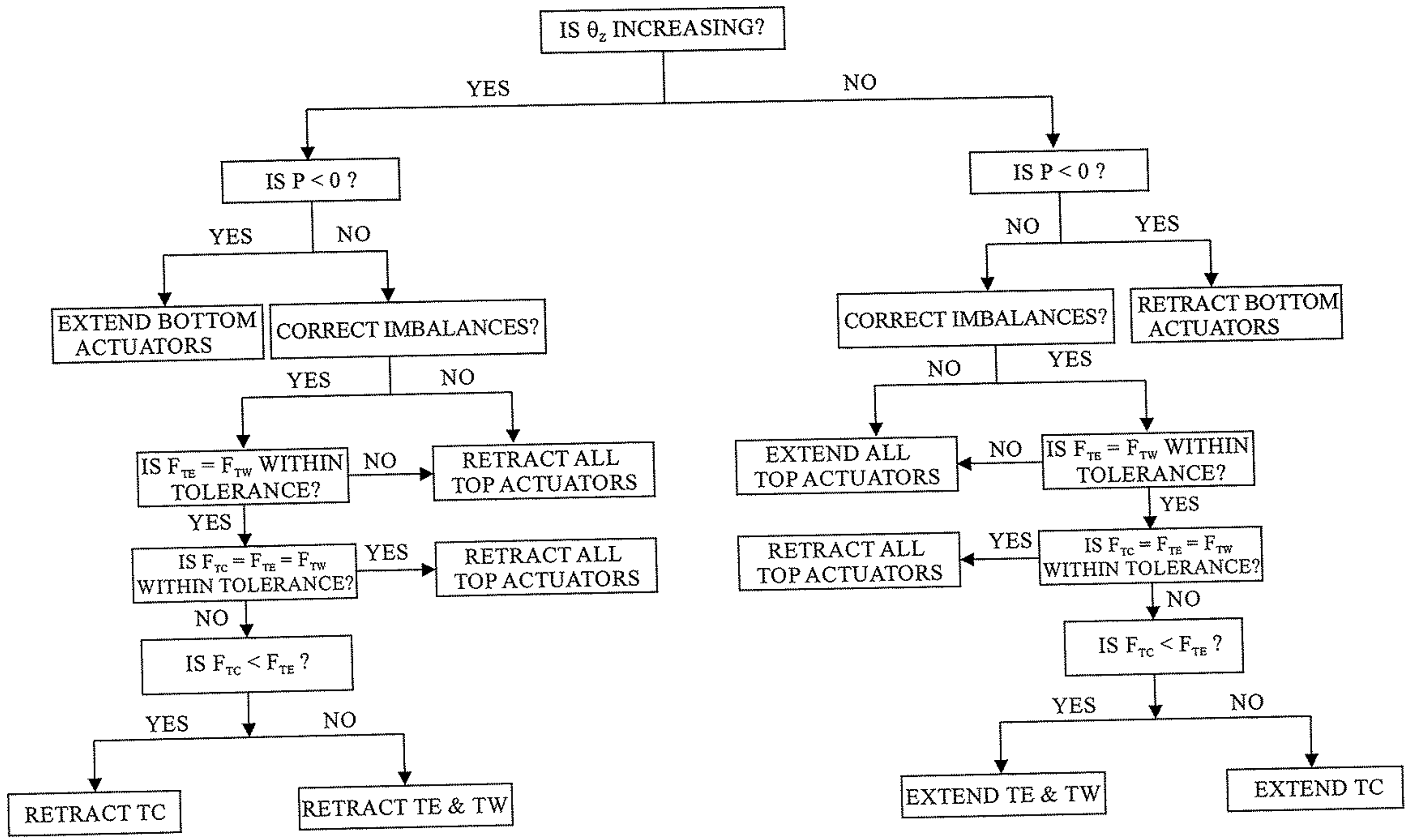


Figure 3.37 Decision algorithm

CHAPTER 4
CONTROL SYSTEM

4.1 INTRODUCTION

As described in Chapter 1 and Figure 1.2, the *test system* is comprised of the physical hardware used to provide forces and reactions to the hull structures (*test fixture*) and of the electronic hardware and software used to orchestrate the tasks of load application and data acquisition during the tests (*control system*). The entire test system was developed to accomplish the loading objectives outlined in Section 2.3.1.

During testing, the hardware acquires data from various transducers. The software then uses this data in decision-making algorithms to determine commands that specify actuator movements. The hardware then implements these commands and the cycle repeats.

In this chapter, the hardware is first presented. The software and its functions are then discussed in detail.

4.2 CONTROL SYSTEM HARDWARE

Figure 4.1 is a schematic drawing of all major components of the control system hardware. The control system includes five servo-controlled hydraulic actuators operating in closed-loop displacement control. For clarity, only one of the five actuators is included in the figure. Additional components of the control system hardware that are not shown in the figure include various power supplies and two hydraulic service manifolds.

As shown in Figure 4.2, three different types of transducers are included in the control system hardware:

1. *Feedback transducers* provide displacement feedback directly to the controller for each actuator. This signal is used in the closed-loop displacement control of the actuator. There are a total of 5 feedback transducers in the control system.
2. *Control transducers* provide information about the current state of actuator forces, link forces, and grillage movements. There are a total of 21 control transducers in the control system. This includes 5 force transducers for the actuators, 4 force transducers for the vertical links, 2 force transducers for the horizontal links, and 5 displacement transducers for each grillage.
3. *Data transducers* are not directly a part of the control system. These transducers provide additional information about the behavior of the test specimen. Many of the data transducers are strain gages attached to the hull structure. Signals from these transducers are acquired by an auxiliary data acquisition system which is triggered by the control system. When the auxiliary data acquisition system is triggered to sample and save information from the data transducers, the control transducers are also sampled and their values are saved to the control computer. This assures simultaneous readings of the data transducers and control transducers for later processing.

As shown in Figure 4.1, the analog signals from feedback transducers are sent directly to the controllers. The analog signals from control transducers are first conditioned if necessary, then converted to digital signals by an analog-to-digital (A/D) converter, and finally received by the computer software. Here, the command signals are determined. They are then converted to analog signals by a digital-to-analog (D/A) converter and sent to the controller. The controllers compute error signals and send them to the servovalves, causing the actuators to move.

Up to 32 channels of differential voltage input from control transducers can be sampled by the 12 bit A/D converter. The 12-bit converter divides the ±10 volt (20 volts total) range of the input into 2^{12} = 4096 parts. Thus the theoretical voltage resolution of the A/D converter is 20 volts/4096 parts = 0.005 volts. Up to 16 channels of analog voltage output are provided by the 12-bit D/A converter. Five of these are reserved for outgoing actuator command signals. A sixth channel is used for sending a voltage signal which triggers the data acquisition system to sample and save data. The 12-bit D/A conversion divides the ±10 volt (20 volt total) range of the output into 2^{12} = 4096 parts, providing an output voltage resolution of 0.005 volts. The theoretical resolution of each actuator is computed as the total calibrated displacement range of the actuator divided by 4096 parts. The top actuators are calibrated for a displacement range of 36 inches, and thus they have a theoretical positioning resolution of 0.0088 inches. The bottom actuators are calibrated for a displacement range of 24 inches, and thus they have a theoretical positioning resolution of 0.0059 inches.

4.3 CONTROL SYSTEM SOFTWARE

4.3.1 Overview

Figure 4.2 shows a simplified block diagram that outlines the basic flow of the control system software during the execution of a test. First, the control transducers are sampled and their values are used by the control software to compute desired parameters such as the positions of the grillages and the forces applied to the hull structure. These computations are made using the approach described in Chapter 3. These parameters are then used in decision-making algorithms (aided by user input) to compute new commands. The decision algorithm was discussed in Section 3.5.2 and presented in Figure 3.37. New commands (actuator positions) are then issued by the computer to the controllers and the controllers compute error signals. The error signal for each actuator is the difference between the feedback signal (its current position) and the command signal (its desired position). Finally, the servovalves receive the error signals and implement the new desired actuator positions. This figure represents one loading step.

The control software is designed to perform multiple loading steps autonomously in order to expedite testing. However, the user still has ultimate control over the entire loading process because of her ability to specify (and modify during the test) many of the critical parameters that govern the test. For example, the user can specify the number of loading steps to be performed in one program loop (one *loading step* is given in Figure 4.2 and a *program loop* consists of a user-specified number of loading steps). The user can also specify the number of loading steps to execute before saving data. The force tolerance, defined in Section 3.5.2, is also changeable. These parameters, and others, are further described in Section 4.3.4.

When the program is first executed, the user is presented with a main menu. It is from this menu that the major functions of the program are chosen. Figure 4.3 is a diagram of the structure of the control program. Shown in the figure are the main menu options. The available options are as follows:

1. Load Channel Setup from File
2. Check Currently Loaded Values
3. Initialize Actuators
4. Run Main Program Loop
5. Display Raw Voltages
6. Exit Program

Options 1 and 2, collectively called the *Channel Setup*, are described in Section 4.3.2. Option 3, or *Initialize Actuators*, is discussed in Section 4.3.3. The *Main Program Loop*, accessible through Option 4, is described in Section 4.3.4. It is through the main program loop that the loading and unloading functions are performed and program parameters are changed. Figure 4.3 shows the functions accessed through the main program loop which will be discussed later. Option 5 simply allows the user an opportunity to view the A/D channel raw voltages for the purpose of verifying that all channels and transducers are working properly. This screen is shown in Figure 4.4. Through Option 6, the user can exit the program.

4.3.2 Channel Setup

When the program is first executed, the user must load the channel setup from a previously created input file by choosing Option 1. This file contains information regarding the number of A/D and D/A channels to be used and the calibration constant for each control transducer. This step must be done first in order to continue with other program options. After loading the data, the user can then view, verify, and modify it if desired by choosing Option 2.

4.3.3 Actuator Initialization

After loading the channel setup, the next step is actuator initialization. The initialization process ensures that the system will remain at rest when hydraulic pressure is first applied. The user is prompted to enter an initial command signal for each actuator. The value entered for each actuator should be equal to its current feedback signal, which can be selected for display on the controller console. The operator is then presented with another screen, shown in Figure 4.5, which permits her to make adjustments to the command signals until they are equal to the feedback signals as nearly as possible. Since the command signal is made equal to the feedback signal for each actuator, the error signals computed by the controllers all equal zero. At this point, hydraulic pressure is applied and the actuators do not move.

4.3.4 Main Program Loop

After test setup and initialization are complete, the user can choose Option 4 from the main menu to run the main program loop. This is where the loading and unloading processes, as well as other program functions, are performed. Upon running the main program loop, the program begins to sample the control transducers, determine the grillage locations, compute actuator and link vectors, and finally calculate the forces and moments applied to the hull structure. However, the program does not make decisions (or compute new actuator commands). In this stage, the program is said to be *pausing*. In the pausing mode, the program repeatedly makes all of the calculations just described, allowing the user to monitor the state of the system. Figure 4.6 shows the parameters that are displayed. As shown in the figure, the main graphics screen has four viewports. In the upper right portion of the screen is a graphic drawing of the hull structure with actuator and link forces displayed at appropriate locations on the drawing. At the bottom right is a moment versus rotation graph which plots both primary moment (M_Z) and secondary moment (M_Y). The upper left portion of the screen is reserved for the display of the numerical values of key reduced data. In addition to displaying pertinent force and displacement values here, parameters important to the loading process are displayed. A few of these are the loading step number, the total number of loading steps to be executed in one program loop, and the number of steps to execute before data is saved (this parameter is called the save data step size).

When the user specifies for loading of the hull structure to begin, the program enters the *execution* mode. Now, the program samples transducers, calculates the grillage positions and forces applied to the structure, and makes decisions. The program will go through one loading loop, executing the number of loading steps that the user specified. During the execution phase of the test, the user interacts with the program through the viewport in the lower left part of the screen. This viewport is used to present two menus that are accessed via function keys. These are the Program Functions menu (F5 key) and the Program Parameters menu (F3 key). The Program Parameters menu and its options are described first.

Program Parameters Through the Program Parameters menu, the user has the option to change six test parameters. Parameters that can be modified include the scale used for the moment versus rotation graph, the number of loading steps in a program loop, (called loop steps) and the number of loading steps executed before data is saved (called the save data step size). For example, if there are 20 loading steps in a program loop and the save data step size is 5, then data will be saved 4 times, or every 5 loading steps, of the program loop. A parameter called the DAC step size can also be altered through the screen shown in Figure 4.6. A DAC step size of 1 corresponds to the smallest actuator movement that can be made (DAC is shorthand for Digital-to-Analog Conversion). As discussed in Section 4.2, this is 0.0088 inches for the top actuators and 0.0059 inches for the bottom actuators. The user can make the DAC step sizes for each actuator any integer multiple of its resolution.

A parameter called the tolerance is changeable through the Program Parameters menu as well. At the end of each loading step, the program pauses until the control transducer values settle and the system comes into equilibrium. To determine whether this has happened, data is sampled twice and actuator forces are calculated at each sampling. If the difference in force values for all actuators between the two samplings is equal to or smaller than the tolerance, loading can continue. If not, data is again sampled twice and the process is repeated until equilibrium occurs. This tolerance, which has units of kips, can have any user-defined value, and, as noted above, can be modified during execution of the test.

As described in Section 3.5.2, the user has the ability to specify whether an imbalance of force in the top actuators should be corrected, or if all of the top actuators should be moved the same amount. This specification can be made through the Program Parameters menu.

All six parameters are given initial default values. During testing, they may be modified between loading loops, or by exiting a loading loop in progress through a function key and accessing the Program Parameters menu.

Program Functions The Program Functions (see Figure 4.3) menu contains options that allow the user to initiate loading or unloading, as well as to execute other program functions. In order to begin the application of a primary sagging bending moment, *Increase Rotation* is chosen from this menu. At this point, the program will execute the program loop comprised of a preset number of loading steps and a preset DAC step size. In other words, if the loading loop has 10 steps, the program will acquire data, make decisions, and send commands ten times without user interaction. Data is saved at the interval specified by the save data step size.

While looping, two function keys remain active for use. One function key (F9), permits the user to exit the program loop. This may be done so certain parameters that govern the test may be modified, or this may be done simply to pause the test during execution. The second function key (F7) causes data to be saved to disk when it otherwise would not be saved automatically. Function key F7 also triggers the auxiliary data acquisition system to sample and save data, thus preserving the simultaneous sampling and storage of the control transducers and data transducers.

Decrease Rotation is another choice that can be made from the Program Functions menu. The same loading loop and parameters are used as for increasing rotation, but the decision algorithms are modified to cause the loads and rotation to be decreased. Once the specimen has been loaded to the desired extent, *Decrease Rotation* can be selected to unload the hull structure.

Also available through the Program Functions menu is the ability to monitor the A/D channel input voltages and to manually specify any user-defined adjustments to actuator command signals outside of the main program loop. The user can also choose to view a full screen version of the moment versus rotation plot. Sufficient data is written to arrays to allow this plot to be recreated in its entirety as often as desired.

Appendix A of this report contains the source code for the control program. Appendix B contains definitions for the variables used in the program.

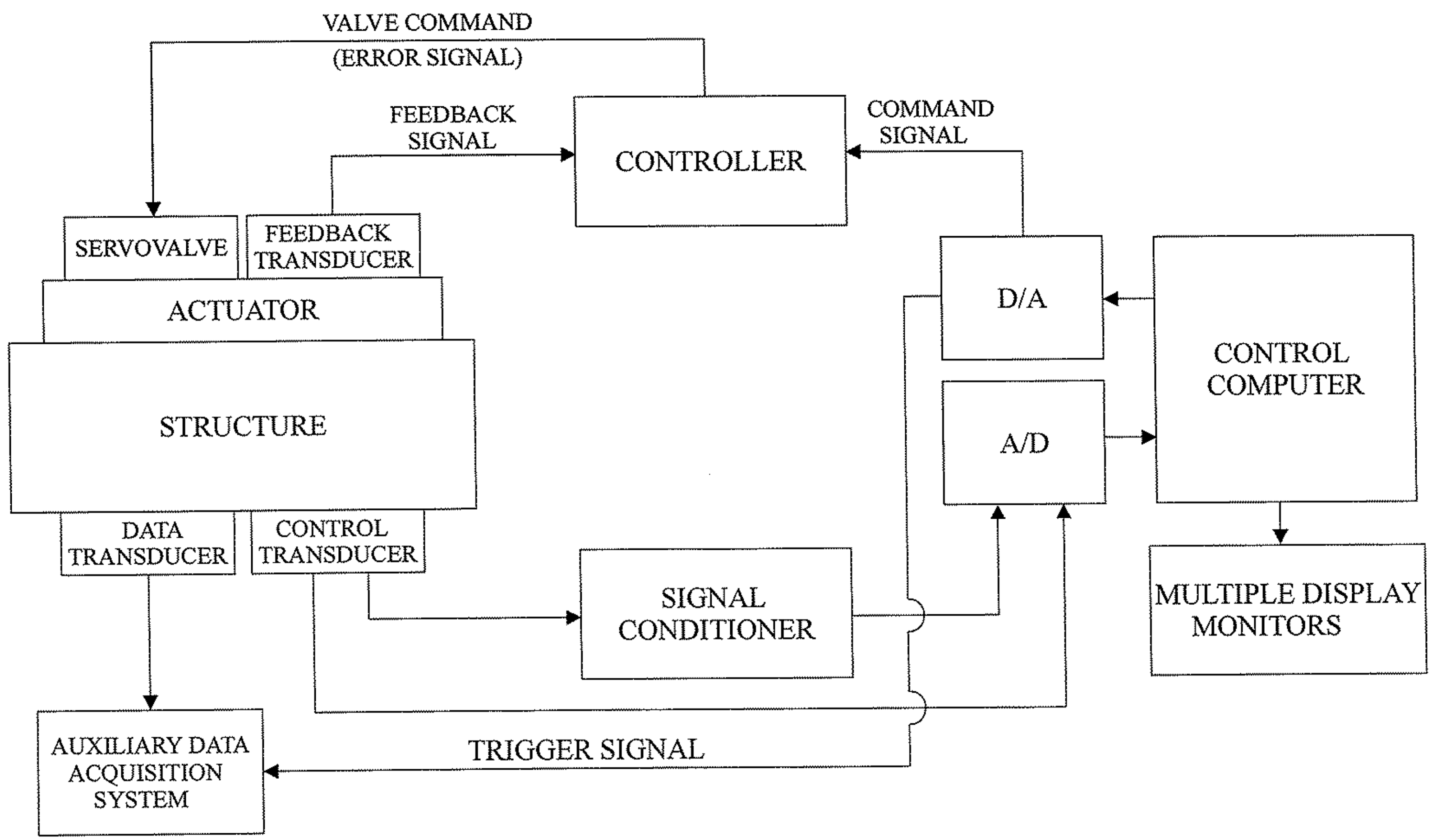


Figure 4.1 Control system hardware

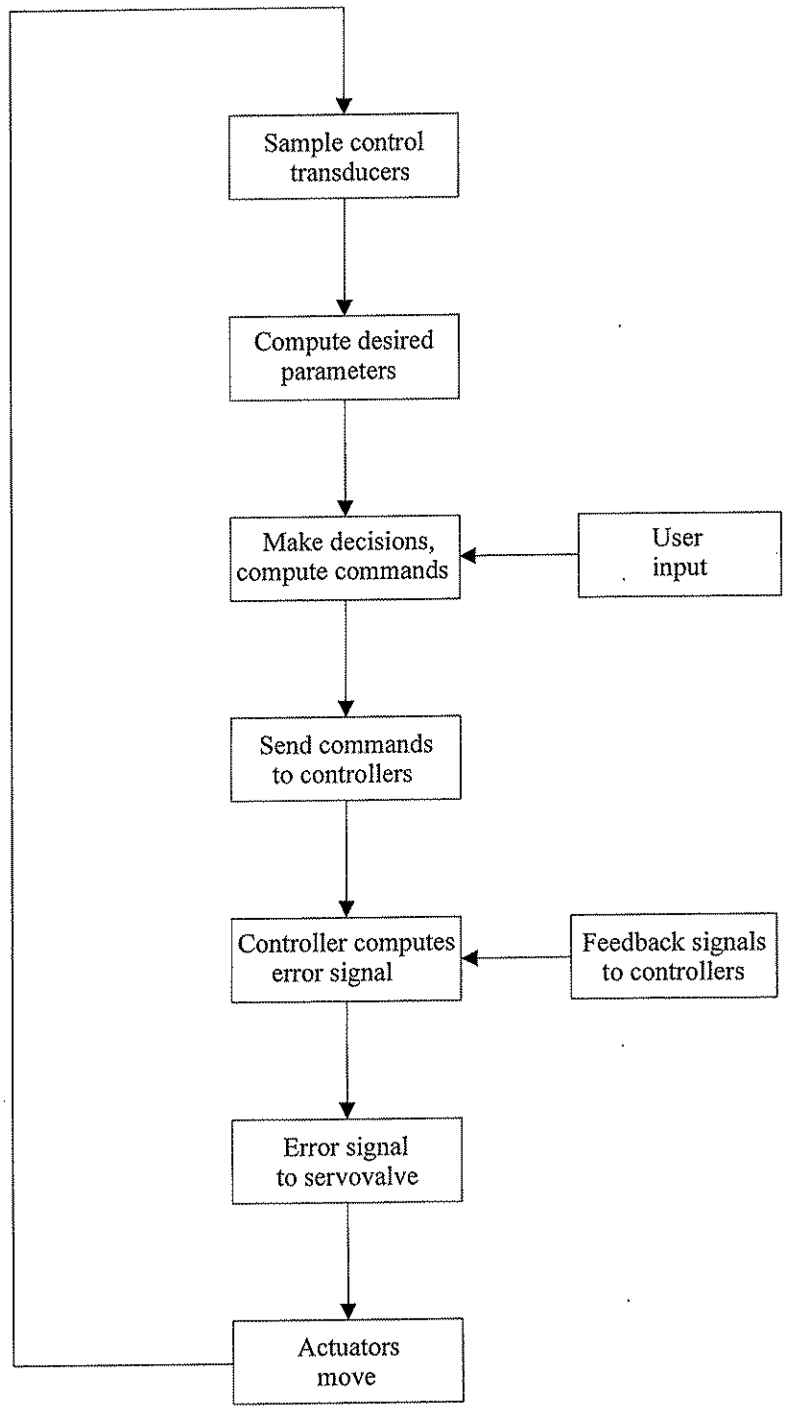


Figure 4.2 Simplified block diagram of control system software

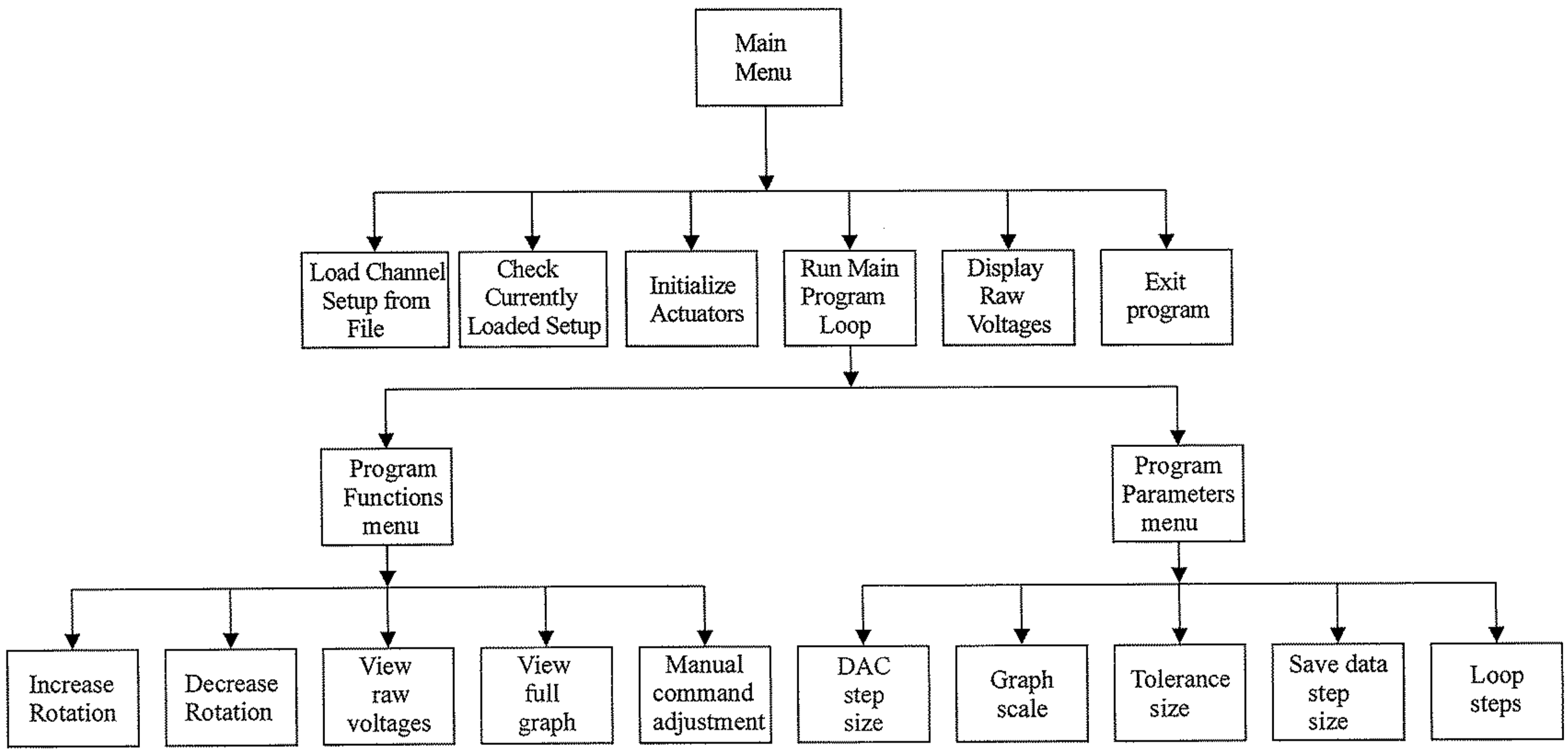
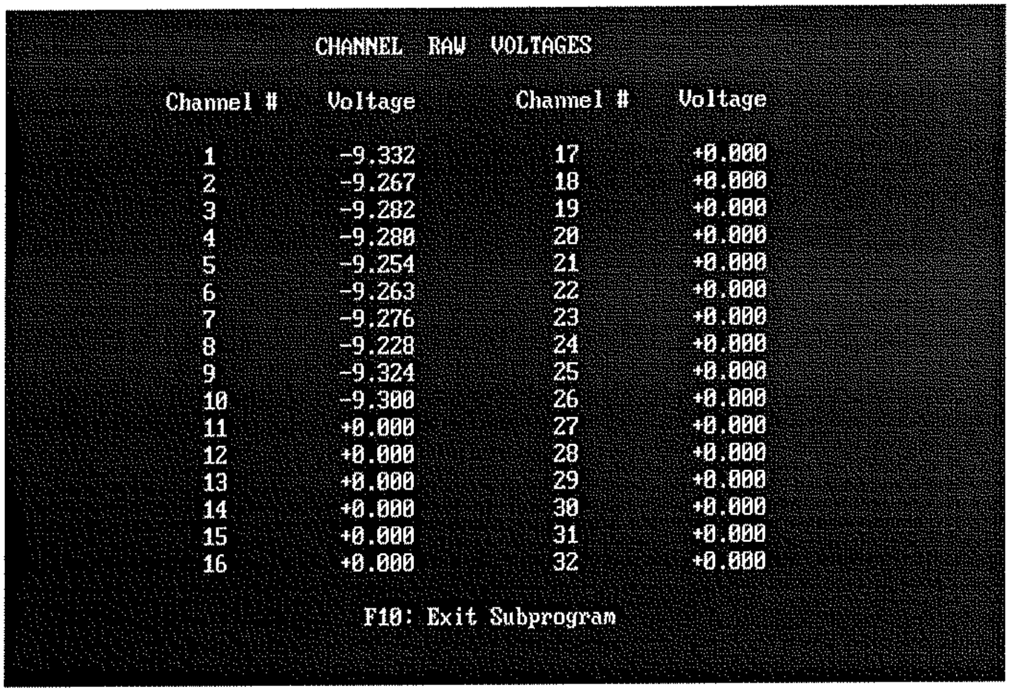


Figure 4.3 Structure of the control program

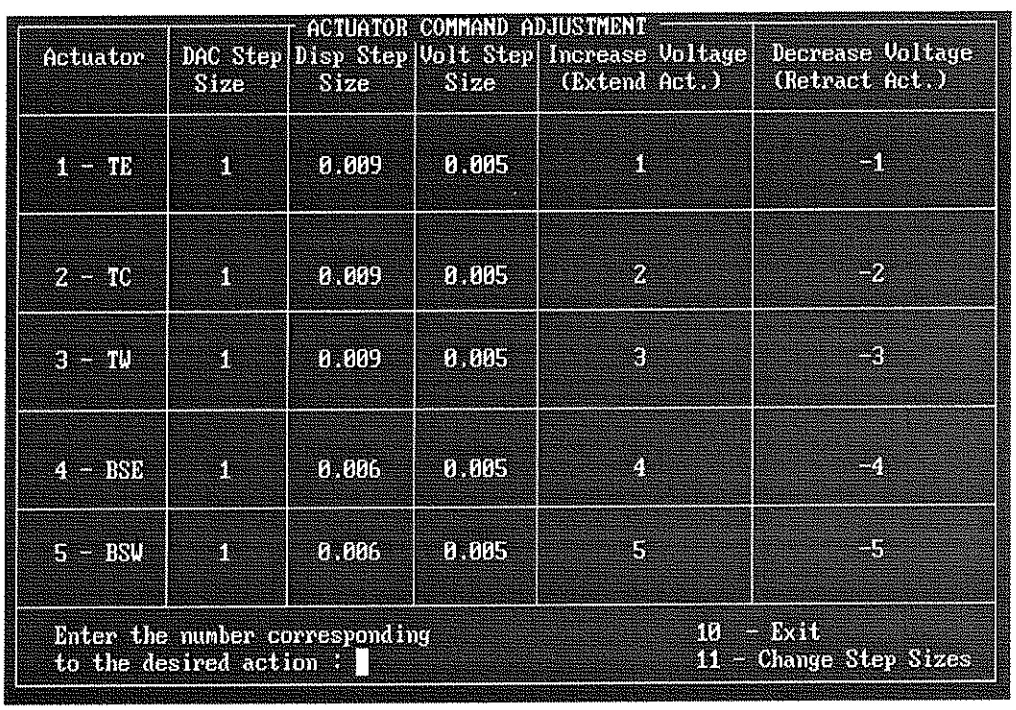


CHANNEL RAW VOLTAGES

Channel #	Voltage	Channel #	Voltage
1	-9.332	17	+0.000
2	-9.267	18	+0.000
3	-9.282	19	+0.000
4	-9.280	20	+0.000
5	-9.254	21	+0.000
6	-9.263	22	+0.000
7	-9.276	23	+0.000
8	-9.228	24	+0.000
9	-9.324	25	+0.000
10	-9.300	26	+0.000
11	+0.000	27	+0.000
12	+0.000	28	+0.000
13	+0.000	29	+0.000
14	+0.000	30	+0.000
15	+0.000	31	+0.000
16	+0.000	32	+0.000

F10: Exit Subprogram

Figure 4.4 Channel input voltage screen



ACTUATOR COMMAND ADJUSTMENT

Actuator	DAC Step Size	Disp Step Size	Volt Step Size	Increase Voltage (Extend Act.)	Decrease Voltage (Retract Act.)
1 - TE	1	0.009	0.005	1	-1
2 - TC	1	0.009	0.005	2	-2
3 - TW	1	0.009	0.005	3	-3
4 - BSE	1	0.006	0.005	4	-4
5 - BSW	1	0.006	0.005	5	-5

Enter the number corresponding to the desired action :

10 - Exit
11 - Change Step Sizes

Figure 4.5 Manual command adjustment screen

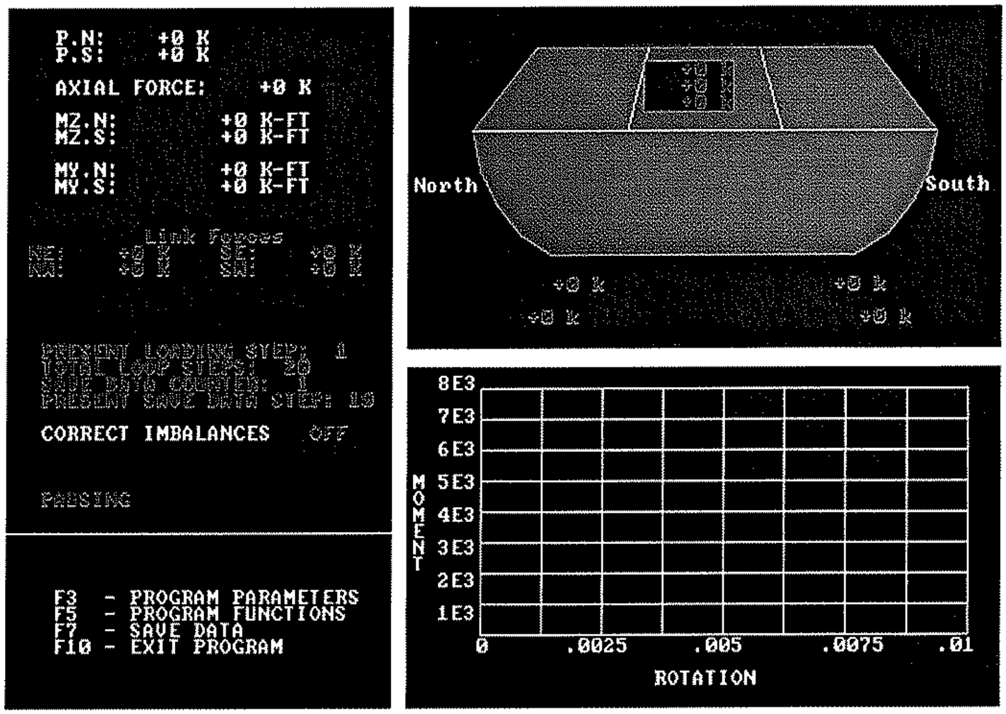
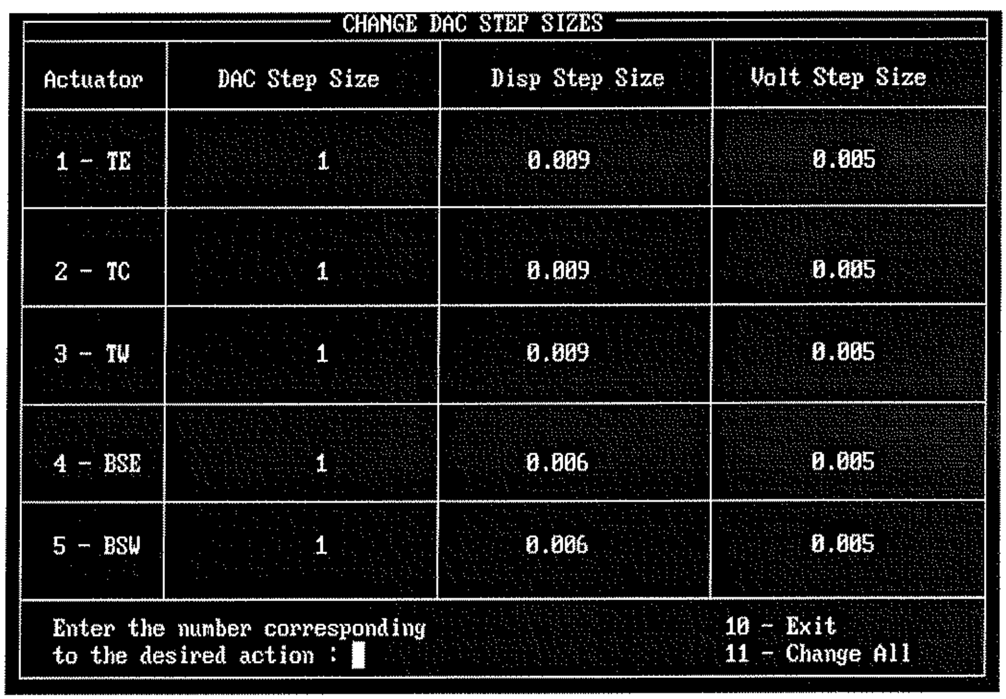


Figure 4.6 Main graphics screen



CHANGE DAC STEP SIZES

Actuator	DAC Step Size	Disp Step Size	Volt Step Size
1 - TE	1	0.009	0.005
2 - TC	1	0.009	0.005
3 - TW	1	0.009	0.005
4 - BSE	1	0.006	0.005
5 - BSW	1	0.006	0.005

Enter the number corresponding
to the desired action :

10 - Exit
11 - Change All

Figure 4.7 DAC step size screen

CHAPTER 5
SUMMARY AND FUTURE WORK

5.1 SUMMARY

This report presented research focusing on the development of the test system for a test program being conducted by Lehigh University on two composite ship hull and deck structures. The objectives of the test program are to perform low-level load tests to determine the elastic flexibility of the hull structures and collapse-level tests to determine the ultimate strength and failure mode in primary bending. These test objectives were outlined in Chapter 1, along with an overview of the test system.

The test system is comprised of the test fixture and the control system. The control system is comprised of hardware and software used to coordinate the load application and data acquisition during the tests.

In Chapter 2, the components of the test system, and more specifically the test fixture, were described. Also included in Chapter 2 was a description of the loading objectives and how they are achieved. The loading objectives are to maintain zero net axial force in the hull structure, while applying a primary bending moment (M_Z) to the hull structure. A secondary bending moment (M_Y) can also occur in the hull structure. The statics of the test fixture were explained, with a description of how the actuator configuration accomplishes the loading objectives.

Chapter 3 outlined the theory developed for the control program. A method was developed for determining the components of all forces applied to the hull structure. Due to the configuration of the test fixture, each actuator and link is permitted to develop three components of force. From these forces, equilibrium equations were written to find the net axial force in the hull structure and the moments applied to the hull structure. All of these parameters (actuator forces, net axial force, and moments) are then used by the software in a decision algorithm to control the loading of the hull structure.

In Chapter 4, the control system was described. The control system incorporates closed-loop displacement control of five actuators into an external loop that allows for user input in order to control the test. This chapter included detailed descriptions of the hardware and software. The control software, or control program, consists of three main parts, the channel setup, actuator initialization, and the main program loop. Through the main program loop, the user is permitted to interact with the program, giving her ultimate control over the testing.

5.2 FUTURE WORK

The test program being executed by Lehigh University on the composite ship hull and deck structures is currently in progress. Upon completion of the erection of the test fixture, testing of the hull structures will begin. The LTC-Prepreg hull structure will be tested first as a calibration specimen in order to ensure that the test fixture and control system can successfully accomplish the loading objectives. The UV-Prepreg hull will then be tested. As described in Section 1.2, both low-level load tests and collapse-level tests will be performed.

REFERENCES

1. Nguyen, L.B., Critchfield, M.O., "Feasibility Study and Fabrication Demonstration of FRP Hull Structures for Naval Surface Combatants," *Proceedings, International Conference on Advances in Marine Structures III*, Dunfermline, Scotland, 20-23 May, 1997.

2. Pessiki, S.P., Sause, R., "Composite Hull and Deck Structure Test Program," *Proposal to Naval Surface Warfare Center*, Carderock Division, July 1997.

3. Chisholm, J.S.R., Vectors in three-dimensional space, Cambridge University Press, 1978.

APPENDIX A
CONTROL PROGRAM SOURCE CODE

```
DECLARE SUB Graph ()
DECLARE SUB Decision.Algorithm ()
DECLARE SUB Pause ()
DECLARE SUB Correct.Imbalances ()
DECLARE SUB Save.Data ()
DECLARE SUB Save.Data.Step.Size ()
DECLARE SUB Change.Loop.Steps ()
DECLARE SUB Equilibrium.Eqns ()
DECLARE SUB Save.Data.Check ()
DECLARE SUB Tolerance.Size ()
DECLARE SUB F5Flag.Check ()
DECLARE SUB F3Flag.Check ()
DECLARE SUB Clear.Window ()
DECLARE SUB Force.Comp.Calcs ()
DECLARE SUB Actuator.Vectors ()
DECLARE SUB Calc.Engr.Values ()
DECLARE SUB Initialize.Vars ()
DECLARE SUB Run.Program ()
DECLARE SUB Check.New.Commands ()
DECLARE SUB Compute.New.Commands ()
DECLARE SUB Move.Actuators ()
DECLARE SUB DAC.Step.Size ()
DECLARE SUB Volt.Command.Adj ()
DECLARE SUB Initialize.Acts ()
DECLARE SUB Main.Display ()
DECLARE SUB Zero ()
DECLARE SUB Main.Menu ()
DECLARE SUB Load.Setup ()
DECLARE SUB Constants ()
DECLARE SUB Initialize.Boards ()
DECLARE SUB Sample.AD ()
DECLARE SUB Raw.Voltages ()
DECLARE SUB Large.Graph ()
DECLARE SUB Choose.Graph.Scale ()
'**************************************************************************
'                control3.BAS
'          QuickBasic Test Control Program
'
' This program utilizes a 64 channel Keithley-Metrabyte DAS1802HC A/D board and a 16 channel
' Keithley-Metrabyte DDA16 D/A board to acquire and transmit voltages for closed loop
' displacement control. This is the main module of the program. A second
' module contains additional subroutines and is a part of the library
' LIBRARY.QLB. The program must be linked with this library before running
' the program. The source code of the second module is called control4.bas.
'**************************************************************************
' The following variables are declared as common because they are common
' to one or more subroutines. If a certain COMMON variable is used in
' various subroutines, it must be listed in the SHARED statements at the
' beginning of each subroutine.
' The common group /navy2/ are those variables which are used in both this
' module and the second module in the library. The common group /navy/ are
```

```
' those variables used only in the subroutines in this module.
'
COMMON /navy2/ Loop.Steps, Tolerance, scale, Save.Data.Step, Correction
COMMON /navy2/ CommandError, Command.TE%, Command.TC%
COMMON /navy2/ Command.TW%, Command.BSE%, Command.BSW%
COMMON /navy2/ Command.16%, Command.Trigger AS INTEGER
COMMON /navy2/ Delta.TE%, Delta.TC%, Delta.TW%, Delta.BSE%, Delta.BSW%
COMMON /navy2/ DACStep.TE%, DACStep.TC%, DACStep.TW%, DACStep.BSE%, DACStep.BSW%
COMMON /navy/ Ident$, OutFile1$, OutFile2$, OutFile3$
COMMON /navy/ F5Flag, F10Flag, F3Flag, F7Flag, F4Flag, nPass, F9Flag, nData
COMMON /navy/ NChannel, Pause.Execution
COMMON /navy/ nStep, Direction, nPlot, nSave
COMMON /navy/ LAMBDA.N, LAMBDA.S, Save.Data.Counter
COMMON /navy/ CommandV.TE, CommandV.TW, CommandV.BSE, CommandV.BSW, CommandV.TC
COMMON /navy/ FORCE.X.BNW, FORCE.X.BNE, FORCE.X.BSW, FORCE.X.BSE
COMMON /navy/ FORCE.Y.BNW, FORCE.Y.BNE, FORCE.Y.BSW, FORCE.Y.BSE
COMMON /navy/ FORCE.Z.BNW, FORCE.Z.BNE, FORCE.Z.BSW, FORCE.Z.BSE
COMMON /navy/ FORCE.XN.TW, FORCE.XS.TW, FORCE.XN.TC, FORCE.XS.TC
COMMON /navy/ FORCE.XN.TE, FORCE.XS.TE, FORCE.YN.TW, FORCE.YS.TW
COMMON /navy/ FORCE.YN.TC, FORCE.YS.TC, FORCE.YN.TE, FORCE.YS.TE
COMMON /navy/ FORCE.ZN.TW, FORCE.ZS.TW, FORCE.ZN.TC, FORCE.ZS.TC
COMMON /navy/ FORCE.ZN.TE, FORCE.ZS.TE
COMMON /navy/ FORCE.TE, FORCE.TW, FORCE.BSE, FORCE.BSW, FORCE.BNE
COMMON /navy/ FORCE.BNW, FORCE.TC
COMMON /navy/ FLINK.NE, FLINK.NW, FLINK.SE, FLINK.SW
COMMON /navy/ FLINK.X.NE, FLINK.X.NW, FLINK.X.SW, FLINK.X.SE, FLINK.Y.NE
COMMON /navy/ FLINK.Y.NW, FLINK.Y.SW, FLINK.Y.SE
COMMON /navy/ TEMPO.1NX, TEMPO.2NX, TEMPO.3NX, TEMPO.1NY, TEMPO.3NY
COMMON /navy/ TEMPO.1SX, TEMPO.2SX, TEMPO.3SX, TEMPO.1SY, TEMPO.3SY
COMMON /navy/ DELTA.X.BSE, DELTA.X.BSW
COMMON /navy/ CN.X.TE, CN.Y.TE, CN.X.TC, CN.Y.TC, CN.X.TW, CN.Y.TW
COMMON /navy/ CS.X.TE, CS.Y.TE, CS.X.TC, CS.Y.TC, CS.X.TW, CS.Y.TW
COMMON /navy/ CN.X.BNE, CN.Y.BNE, CN.X.BNW, CN.Y.BNW
COMMON /navy/ CS.X.BSE, CS.Y.BSE, CS.X.BSW, CS.Y.BSW
COMMON /navy/ D.X.NE, D.Y.NE, D.X.NW, D.Y.NW, D.X.SE, D.Y.SE, D.X.SW, D.Y.SW
COMMON /navy/ P.N, P.S, MZ.N, MZ.S, MY.N, MY.S, P
COMMON /navy/ THETA.Z.N, THETA.Z.S, THETA.Y.N, THETA.Y.S
'************************************************************************
' The following include statements are necessary, and the files listed must
' be in the directory in which you are operating.
'
' $INCLUDE: 'QB4DECL.BI'
' $INCLUDE: 'DASDECL.BI'
' $INCLUDE: 'DAS1800.BI'
' $INCLUDE: 'DDA16.BI'
'
' The following statements are necessary for allocating memory.
'
CLEAR
' FarHeapSize& = SETMEM(0)
' NewFarHeapSize& = SETMEM(-FarHeapSize& / 2)
' $DYNAMIC
'************************************************************************
' The varibles declared below can be used by all subroutines. Arrays are
```

```
' declared here, as are the variables used by the D/A and A/D boards.
'
DIM SHARED DataBuf(32) AS INTEGER
DIM SHARED nDasErr AS INTEGER
DIM SHARED szCfgName AS STRING
DIM SHARED hDev AS LONG
DIM SHARED hAD AS LONG
DIM SHARED wStatus AS INTEGER
DIM SHARED dwCount AS LONG
DIM SHARED dwFactor AS LONG
DIM SHARED ACommand(16) AS INTEGER
DIM SHARED DERR AS INTEGER
DIM SHARED hDDA16 AS LONG
DIM SHARED hDA AS LONG
DIM SHARED ChData(0 TO 20, 1 TO 32) AS LONG
DIM SHARED ChVolt(1 TO 32) AS SINGLE
DIM SHARED SensCon(1 TO 32) AS SINGLE
DIM SHARED M(100) AS SINGLE
DIM SHARED GTHETA(100) AS SINGLE
'****************************************************************************
' The following values are constants used throughout the program.  They can
' be used by any subroutine.
'
CONST CPN.X.TW = 100, CPN.X.TC = 100, CPN.X.TE = 100
CONST CPS.X.TW = -100, CPS.X.TC = -100, CPS.X.TE = -100
CONST CPN.Y.TW = 70, CPN.Y.TC = 70, CPN.Y.TE = 70
CONST CPS.Y.TW = 70, CPS.Y.TC = 70, CPS.Y.TE = 70
CONST CPN.Z.TW = 100, CPN.Z.TC = 0, CPN.Z.TE = -100
CONST CPS.Z.TW = 100, CPS.Z.TC = 0, CPS.Z.TE = -100
CONST CPN.X.BNW = 50, CPN.X.BNE = -50
CONST CPN.Y.BNW = -240, CPN.Y.BNE = -240
CONST CPN.Z.BNW = 80, CPN.Z.BNE = -80
CONST CPS.X.BSW = -50, CPS.X.BSE = -50
CONST CPS.Y.BSW = -240, CPS.Y.BSE = -240
CONST CPS.Z.BSW = 80, CPS.Z.BSE = -80
CONST LINKLENGTH = 150
CONST ENO.X.BSE = 300, ENO.Y.BSE = -200, ENO.Z.BSE = -50
CONST ENO.X.BSW = 300, ENO.Y.BSW = -200, ENO.Z.BSW = 50
CONST ESO.X.BNE = 230, ESO.Y.BNE = -200, ESO.Z.BNE = -50
CONST ESO.X.BNW = 230, ENO.Y.BNW = -200, ENO.Z.BNW = 50
CONST L2.X.NE = 100, L2.Y.NE = 150, L2.Z.NE = 40
CONST L2.X.NW = 100, L2.Y.NW = 150, L2.Z.NW = -40
CONST L2.X.SE = 400, L2.Y.SE = 150, L2.Z.SE = 40
CONST L2.X.SW = 400, L2.Y.SW = 150, L2.Z.SW = -40
CONST P1N.X.O = 60, P1N.Y.O = 80, P1N.Z.O = 0
CONST P2N.X.O = 60, P2N.Y.O = 0, P2N.Z.O = 0
CONST P3N.X.O = 60, P3N.Y.O = 0, P3N.Z.O = 100
CONST P1S.X.O = 360, P1S.Y.O = 80, P1S.Z.O = 0
CONST P2S.X.O = 360, P2S.Y.O = 0, P2S.Z.O = 0
CONST P3S.X.O = 360, P3S.Y.O = 0, P3S.Z.O = 100
'****************************************************************************
' Here is where the program begins.  The following statements enable the
' function keys used in the program.
'
```

```
   KEY(10) ON
   ON KEY(10) GOSUB 515
   KEY(9) ON
   ON KEY(9) GOSUB 516
   KEY(7) ON
   ON KEY(7) GOSUB 517
   KEY(5) ON
   ON KEY(5) GOSUB 520
   KEY(3) ON
   ON KEY(3) GOSUB 525
   KEY(4) ON
   ON KEY(4) GOSUB 530
   KEY(6) ON
   ON KEY(6) GOSUB 535
   SCREEN 9
   WIDTH 80, 43
   VIEW PRINT

   CALL Initialize.Vars
   CALL Main.Menu

' This is the end of the main program.  All of the functions of the program
' are accessed by the main menu subroutine.

510  END
'****************************************************************************
515  F10Flag = 1
RETURN

516  F9Flag = 1
RETURN

517  F7Flag = 1
RETURN

520  F5Flag = 1
RETURN

525  F3Flag = 1
RETURN

530  F4Flag = 1
RETURN

535  F6Flag = 1
RETURN

REM $STATIC
```

```
SUB Actuator.Vectors
'***********************************************************************
SHARED TEMPO.1NX, TEMPO.2NX, TEMPO.3NX, TEMPO.1NY, TEMPO.3NY, TEMPO.1SX
SHARED TEMPO.2SX, TEMPO.3SX, TEMPO.1SY, TEMPO.3SY, FORCE.X.BSW
SHARED FORCE.Y.BSW, FORCE.Z.BSW, FORCE.X.BSE, FORCE.Y.BSE, FORCE.Z.BSE
SHARED FORCE.X.BNW, FORCE.Y.BNW, FORCE.Z.BNW, FORCE.X.BNE, FORCE.Y.BNE
SHARED FORCE.Z.BNE
SHARED FORCE.XN.TE, FORCE.YN.TE, FORCE.ZN.TE, FORCE.XS.TE, FORCE.YS.TE, FORCE.ZS.TE
SHARED FORCE.XN.TC, FORCE.YN.TC, FORCE.ZN.TC, FORCE.XS.TC, FORCE.YS.TC,
FORCE.ZS.TC
SHARED FORCE.XN.TW, FORCE.YN.TW, FORCE.ZN.TW, FORCE.XS.TW, FORCE.YS.TW,
FORCE.ZS.TW
SHARED DELTA.X.BSW, DELTA.X.BSE
SHARED FORCE.X.NE, FORCE.Y.NE, FORCE.Z.NE, FORCE.X.NW, FORCE.Y.NW, FORCE.Z.NW
SHARED FORCE.X.SE, FORCE.Y.SE, FORCE.Z.SE, FORCE.X.SW, FORCE.Y.SW, FORCE.Z.SW
SHARED FORCE.NE, FORCE.NW, FORCE.SE, FORCE.SW
SHARED CN.X.TE, CN.Y.TE, CN.X.TC, CN.Y.TC, CN.X.TW, CN.Y.TW
SHARED CS.X.TE, CS.Y.TE, CS.X.TC, CS.Y.TC, CS.X.TW, CS.Y.TW
SHARED CN.X.BNE, CN.Y.BNE, CN.X.BNW, CN.Y.BNW
SHARED CS.X.BSE, CS.Y.BSE, CS.X.BSW, CS.Y.BSW
SHARED D.X.NE, D.Y.NE, D.X.NW, D.Y.NW, D.X.SE, D.Y.SE, D.X.SW, D.Y.SW
'***********************************************************************
' This subroutine calculates the coordinates of points P1, P2, and P3 on
' the north and south grillages, calculates the vectors U, V, and W, and then
' finds the actuator vectors and force components for all actuators and links.
' First, the coordinates of points P1, P2, and P3 are found. They are equal
' to their original positions (constants) plus their changes in position, as
' measured by the grillage transducers. (Note that some coordinates are simply
' equal to their original distances).
'
    P1N.X = P1N.X.O + TEMPO.1NX
    P2N.X = P2N.X.O + TEMPO.2NX
    P3N.X = P3N.X.O + TEMPO.3NX
    P1N.Y = P1N.Y.O + TEMPO.1NY
    P2N.Y = P2N.Y.O
    P3N.Y = P3N.Y.O + TEMPO.3NY
    P1N.Z = P1N.Z.O
    P2N.Z = P2N.Z.O
    P3N.Z = P3N.Z.O

    P1S.X = P1S.X.O + TEMPO.1SX
    P2S.X = P2S.X.O + TEMPO.2SX
    P3S.X = P3S.X.O + TEMPO.3SX
    P1S.Y = P1S.Y.O + TEMPO.1SY
    P2S.Y = P2S.Y.O
    P3S.Y = P3S.Y.O + TEMPO.3SY
    P1S.Z = P1S.Z.O
    P2S.Z = P2S.Z.O
    P3S.Z = P3S.Z.0

' Now, the U, V, and W vectors are found for each grillage using the coordinates
' of points P1, P2, and P3.

    VN.X = P1N.X - P2N.X
```

```
    VN.Y = P1N.Y - P2N.Y
    VN.Z = P1N.Z - P2N.Z
    WN.X = P3N.X - P2N.X
    WN.Y = P3N.Y - P2N.Y
    WN.Z = P3N.Z - P2N.Z

    VS.X = P1S.X - P2S.X
    VS.Y = P1S.Y - P2S.Y
    VS.Z = P1S.Z - P2S.Z
    WS.X = P3S.X - P2S.X
    WS.Y = P3S.Y - P2S.Y
    WS.Z = P3S.Z - P2S.Z

    UN.X = (VN.Y * WN.Z) - (WN.Y * VN.Z)
    UN.Y = -((VN.X * WN.Z) - (WN.X * VN.Z))
    UN.Z = (VN.X * WN.Y) - (WN.X * VN.Y)
    US.X = (VS.Y * WS.Z) - (WS.Y * VS.Z)
    US.Y = -((VS.X * WS.Z) - (WS.X * VS.Z))
    US.Z = (VS.X * WS.Y) - (WS.X * VS.Y)

' Here, the length of each U, V, and W  vector is found.

    VN.L = (VN.X ^ 2 + VN.Y ^ 2 + VN.Z ^ 2) ^ .5
    WN.L = (WN.X ^ 2 + WN.Y ^ 2 + WN.Z ^ 2) ^ .5
    UN.L = (UN.X ^ 2 + UN.Y ^ 2 + UN.Z ^ 2) ^ .5
    VS.L = (VS.X ^ 2 + VS.Y ^ 2 + VS.Z ^ 2) ^ .5
    WS.L = (WS.X ^ 2 + WS.Y ^ 2 + WS.Z ^ 2) ^ .5
    US.L = (US.X ^ 2 + US.Y ^ 2 + US.Z ^ 2) ^ .5

' Calculate the components of the transformation matrices for both grillages.

    tN.11 = UN.X / UN.L
    tN.12 = UN.Y / UN.L
    tN.13 = UN.Z / UN.L
    tN.21 = VN.X / VN.L
    tN.22 = VN.Y / VN.L
    tN.23 = VN.Z / VN.L
    tN.31 = WN.X / WN.L
    tN.32 = WN.Y / WN.L
    tN.33 = WN.Z / WN.L

    tS.11 = US.X / US.L
    tS.12 = US.Y / US.L
    tS.13 = US.Z / US.L
    tS.21 = VS.X / VS.L
    tS.22 = VS.Y / VS.L
    tS.23 = VS.Z / VS.L
    tS.31 = WS.X / WS.L
    tS.32 = WS.Y / WS.L
    tS.33 = WS.Z / WS.L

' Calculate the RN and RS vectors.  The components of these vectors are
' equal to the coordinates of points P2N and P2S, respectively.
```

```
RN.X = P2N.X
RN.Y = P2N.Y
RN.Z = P2N.Z
RS.X = P2S.X
RS.Y = P2S.Y
RS.Z = P2S.Z

' The CN and CS vectors are found for the top actuators by multiplying the
' transformation matrix by the CPN and CPS vectors (these are constants).

CN.X.TW = tN.11 * CPN.X.TW + tN.21 * CPN.Y.TW + tN.31 * CPN.Z.TW
CN.Y.TW = tN.12 * CPN.X.TW + tN.22 * CPN.Y.TW + tN.32 * CPN.Z.TW
CN.Z.TW = tN.13 * CPN.X.TW + tN.23 * CPN.Y.TW + tN.33 * CPN.Z.TW

CN.X.TC = tN.11 * CPN.X.TC + tN.21 * CPN.Y.TC + tN.31 * CPN.Z.TC
CN.Y.TC = tN.12 * CPN.X.TC + tN.22 * CPN.Y.TC + tN.32 * CPN.Z.TC
CN.Z.TC = tN.13 * CPN.X.TC + tN.23 * CPN.Y.TC + tN.33 * CPN.Z.TC

CN.X.TE = tN.11 * CPN.X.TE + tN.21 * CPN.Y.TE + tN.31 * CPN.Z.TE
CN.Y.TE = tN.12 * CPN.X.TE + tN.22 * CPN.Y.TE + tN.32 * CPN.Z.TE
CN.Z.TE = tN.13 * CPN.X.TE + tN.23 * CPN.Y.TE + tN.33 * CPN.Z.TE

CS.X.TW = tS.11 * CPS.X.TW + tS.21 * CPS.Y.TW + tS.31 * CPS.Z.TW
CS.Y.TW = tS.12 * CPS.X.TW + tS.22 * CPS.Y.TW + tS.32 * CPS.Z.TW
CS.Z.TW = tS.13 * CPS.X.TW + tS.23 * CPS.Y.TW + tS.33 * CPS.Z.TW

CS.X.TC = tS.11 * CPS.X.TC + tS.21 * CPS.Y.TC + tS.31 * CPS.Z.TC
CS.Y.TC = tS.12 * CPS.X.TC + tS.22 * CPS.Y.TC + tS.32 * CPS.Z.TC
CS.Z.TC = tS.13 * CPS.X.TC + tS.23 * CPS.Y.TC + tS.33 * CPS.Z.TC

CS.X.TE = tS.11 * CPS.X.TE + tS.21 * CPS.Y.TE + tS.31 * CPS.Z.TE
CS.Y.TE = tS.12 * CPS.X.TE + tS.22 * CPS.Y.TE + tS.32 * CPS.Z.TE
CS.Z.TE = tS.13 * CPS.X.TE + tS.23 * CPS.Y.TE + tS.33 * CPS.Z.TE

' Calculate the CN vectors for the bottom links.

CN.X.BNW = tN.11 * CPN.X.BNW + tN.21 * CPN.Y.BNW + tN.31 * CPN.Z.BNW
CN.Y.BNW = tN.12 * CPN.X.BNW + tN.22 * CPN.Y.BNW + tN.32 * CPN.Z.BNW
CN.Z.BNW = tN.13 * CPN.X.BNW + tN.23 * CPN.Y.BNW + tN.33 * CPN.Z.BNW

CN.X.BNE = tN.11 * CPN.X.BNE + tN.21 * CPN.Y.BNE + tN.31 * CPN.Z.BNE
CN.Y.BNE = tN.12 * CPN.X.BNE + tN.22 * CPN.Y.BNE + tN.32 * CPN.Z.BNE
CN.Z.BNE = tN.13 * CPN.X.BNE + tN.23 * CPN.Y.BNE + tN.33 * CPN.Z.BNE

' Calculate the CS vectors for the bottom actuators.

CS.X.BSW = tS.11 * CPS.X.BSW + tS.21 * CPS.Y.BSW + tS.31 * CPS.Z.BSW
CS.Y.BSW = tS.12 * CPS.X.BSW + tS.22 * CPS.Y.BSW + tS.32 * CPS.Z.BSW
CS.Z.BSW = tS.13 * CPS.X.BSW + tS.23 * CPS.Y.BSW + tS.33 * CPS.Z.BSW

CS.X.BSE = tS.11 * CPS.X.BSE + tS.21 * CPS.Y.BSE + tS.31 * CPS.Z.BSE
CS.Y.BSE = tS.12 * CPS.X.BSE + tS.22 * CPS.Y.BSE + tS.32 * CPS.Z.BSE
CS.Z.BSE = tS.13 * CPS.X.BSE + tS.23 * CPS.Y.BSE + tS.33 * CPS.Z.BSE
```

```
' Calculate the D vectors for the vertical links.

    D.X.NE = tN.11 * DP.X.NE + tN.21 * DP.Y.NE + tN.31 * DP.Z.NE
    D.Y.NE = tN.12 * DP.X.NE + tN.22 * DP.Y.NE + tN.32 * DP.Z.NE
    D.Z.NE = tN.13 * DP.X.NE + tN.23 * DP.Y.NE + tN.33 * DP.Z.NE

    D.X.NW = tN.11 * DP.X.NW + tN.21 * DP.Y.NW + tN.31 * DP.Z.NW
    D.Y.NW = tN.12 * DP.X.NW + tN.22 * DP.Y.NW + tN.32 * DP.Z.NW
    D.Z.NW = tN.13 * DP.X.NW + tN.23 * DP.Y.NW + tN.33 * DP.Z.NW

    D.X.SE = tS.11 * DP.X.SE + tS.21 * DP.Y.SE + tS.31 * DP.Z.SE
    D.Y.SE = tS.12 * DP.X.SE + tS.22 * DP.Y.SE + tS.32 * DP.Z.SE
    D.Z.SE = tS.13 * DP.X.SE + tS.23 * DP.Y.SE + tS.33 * DP.Z.SE

    D.X.SW = tS.11 * DP.X.SW + tS.21 * DP.Y.SW + tS.31 * DP.Z.SW
    D.Y.SW = tS.12 * DP.X.SW + tS.22 * DP.Y.SW + tS.32 * DP.Z.SW
    D.Z.SW = tS.13 * DP.X.SW + tS.23 * DP.Y.SW + tS.33 * DP.Z.SW

' The EN (north end) and ES (south end) vectors are calculated for all
' actuators and bottom links.

    EN.X.TW = RN.X + CN.X.TW
    EN.Y.TW = RN.X + CN.Y.TW
    EN.Z.TW = RN.Z + CN.Z.TW
    ES.X.TW = RS.X + CS.X.TW
    ES.Y.TW = RS.Y + CS.Y.TW
    ES.Z.TW = RS.Y + CS.Z.TW

    EN.X.TC = RN.X + CN.X.TC
    EN.Y.TC = RN.X + CN.Y.TC
    EN.Z.TC = RN.Z + CN.Z.TC
    ES.X.TC = RS.X + CS.X.TC
    ES.Y.TC = RS.Y + CS.Y.TC
    ES.Z.TC = RS.Y + CS.Z.TC

    EN.X.TE = RN.X + CN.X.TE
    EN.Y.TE = RN.X + CN.Y.TE
    EN.Z.TE = RN.Z + CN.Z.TE
    ES.X.TE = RS.X + CS.X.TE
    ES.Y.TE = RS.Y + CS.Y.TE
    ES.Z.TE = RS.Y + CS.Z.TE

    EN.X.BSW = ENO.X.BSW + DELTA.X.BSW
    EN.Y.BSW = ENO.Y.BSW
    EN.Z.BSW = ENO.Z.BSW
    ES.X.BSW = RS.X + CS.X.BSW
    ES.Y.BSW = RS.Y + CS.Y.BSW
    ES.Z.BSW = RS.Y + CS.Z.BSW

    EN.X.BSE = ENO.X.BSE + DELTA.X.BSE
    EN.Y.BSE = ENO.Y.BSE
    EN.Z.BSE = ENO.Z.BSE
    ES.X.BSE = RS.X + CS.X.BSE
    ES.Y.BSE = RS.Y + CS.Y.BSE
```

```
    ES.Z.BSE = RS.Y + CS.Z.BSE

    EN.X.BNE = RN.X + CN.X.BNE
    EN.Y.BNE = RN.Y + CN.Y.BNE
    EN.Z.BNE = RN.Z + CN.Z.BNE
    ES.X.BNE = ESO.X.BNE + DELTA.X.BSE
    ES.Y.BNE = ESO.Y.BNE
    ES.Z.BNE = ESO.Z.BNE

    EN.X.BNW = RN.X + CN.X.BNW
    EN.Y.BNW = RN.Y + CN.Y.BNW
    EN.Z.BNW = RN.Z + CN.Z.BNW
    ES.X.BNW = ESO.X.BNW + DELTA.X.BSW
    ES.Y.BNW = ESO.Y.BNW
    ES.Z.BNW = ESO.Z.BNW

' Calculate the A vectors for all actuators and bottom links.

    A.X.TW = ES.X.TW - EN.X.TW
    A.Y.TW = ES.Y.TW - EN.Y.TW
    A.Z.TW = ES.Z.TW - EN.Z.TW

    A.X.TC = ES.X.TC - EN.X.TC
    A.Y.TC = ES.Y.TC - EN.Y.TC
    A.Z.TC = ES.Z.TC - EN.Z.TC

    A.X.TE = ES.X.TE - EN.X.TE
    A.Y.TE = ES.Y.TE - EN.Y.TE
    A.Z.TE = ES.Z.TE - EN.Z.TE

    A.X.BSW = ES.X.BNW - EN.X.BNW
    A.Y.BSW = ES.Y.BNW - EN.Y.BNW
    A.Z.BSW = ES.Z.BNW - EN.Z.BNW

    A.X.BSE = ES.X.BNE - EN.X.BNE
    A.Y.BSE = ES.Y.BNE - EN.Y.BNE
    A.Z.BSE = ES.Z.BNE - EN.Z.BNE

    A.X.BNW = ES.X.BNW - EN.X.BNW
    A.Y.BNW = ES.Y.BNW - EN.Y.BNW
    A.Z.BNW = ES.Z.BNW - EN.Z.BNW

    A.X.BNE = ES.X.BNE - EN.X.BNE
    A.Y.BNE = ES.Y.BNE - EN.Y.BNE
    A.Z.BNE = ES.Z.BNE - EN.Z.BNE

    A.TW = (A.X.TW ^ 2 + A.Y.TW ^ 2 + A.Z.TW ^ 2) ^ .5
    A.TC = (A.X.TC ^ 2 + A.Y.TC ^ 2 + A.Z.TC ^ 2) ^ .5
    A.TE = (A.X.TE ^ 2 + A.Y.TE ^ 2 + A.Z.TE ^ 2) ^ .5
    A.BSE = (A.X.BSE ^ 2 + A.Y.BSE ^ 2 + A.Z.BSE ^ 2) ^ .5
    A.BSW = (A.X.BSW ^ 2 + A.Y.BSW ^ 2 + A.Z.BSW ^ 2) ^ .5
    A.BNE = (A.X.BNE ^ 2 + A.Y.BNE ^ 2 + A.Z.BNE ^ 2) ^ .5
    A.BNW = (A.Z.BNW ^ 2 + A.Y.BNW ^ 2 + A.Z.BNW ^ 2) ^ .5
```

```
' Calculate the L1 and L vectors for vertical links.

    L1.X.NE = RN.X + D.X.NE
    L1.Y.NE = RN.Y + D.Y.NE
    L1.Z.NE = RN.Z + D.Z.NE
    L1.X.NW = RN.X + D.X.NW
    L1.Y.NW = RN.Y + D.Y.NW
    L1.Z.NW = RN.Z + D.Z.NW
    L1.X.SE = RS.X + D.X.SE
    L1.Y.SE = RS.Y + D.Y.SE
    L1.Z.SE = RS.Z + D.Z.SE
    L1.X.SW = RS.X + D.X.SW
    L1.Y.SW = RS.Y + D.Y.SW
    L1.Z.SW = RS.Z + D.Z.SW

    L.X.NE = L2.X.NE - L1.X.NE
    L.Y.NE = L2.Y.NE - L1.Y.NE
    L.Z.NE = L2.Z.NE - L1.Z.NE
    L.X.NW = L2.X.NW - L1.X.NW
    L.Y.NW = L2.Y.NW - L1.Y.NW
    L.Z.NW = L2.Z.NW - L1.Z.NW
    L.X.SE = L2.X.SE - L1.X.SE
    L.Y.SE = L2.Y.SE - L1.Y.SE
    L.Z.SE = L2.Z.SE - L1.Z.SE
    L.X.SW = L2.X.SW - L1.X.SW
    L.Y.SW = L2.Y.SW - L1.Y.SW
    L.Z.SW = L2.Z.SW - L1.Z.SW

' Calculate total length of the L vectors for vertical links.

    L.NE = (L.X.NE ^ 2 + L.Y.NE ^ 2 + L.Z.NE ^ 2) ^ .5
    L.NW = (L.X.NW ^ 2 + L.Y.NW ^ 2 + L.Z.NW ^ 2) ^ .5
    L.SE = (L.X.SE ^ 2 + L.Y.SE ^ 2 + L.Z.SE ^ 2) ^ .5
    L.SW = (L.X.SW ^ 2 + L.Y.SW ^ 2 + L.Z.SW ^ 2) ^ .5

' Force component calculations for actuators and bottom links.

    FORCE.XN.TW = FORCE.TW * A.X.TW / A.TW
    FORCE.XS.TW = -FORCE.TW * A.X.TW / A.TW
    FORCE.YN.TW = FORCE.TW * A.Y.TW / A.TW
    FORCE.YS.TW = -FORCE.TW * A.Y.TW / A.TW
    FORCE.ZN.TW = FORCE.TW * A.Z.TW / A.TW
    FORCE.ZS.TW = -FORCE.TW * A.Z.TW / A.TW
    FORCE.XN.TC = FORCE.TC * A.X.TC / A.TC
    FORCE.XS.TC = -FORCE.TC * A.X.TC / A.TC
    FORCE.YN.TC = FORCE.TC * A.Y.TC / A.TC
    FORCE.YS.TC = -FORCE.TC * A.Y.TC / A.TC
    FORCE.ZN.TC = FORCE.TC * A.Z.TC / A.TC
    FORCE.ZS.TC = -FORCE.TC * A.Z.TC / A.TC
    FORCE.XN.TE = FORCE.TE * A.X.TE / A.TW
    FORCE.XS.TE = -FORCE.TE * A.X.TE / A.TW
    FORCE.YN.TE = FORCE.TE * A.Y.TE / A.TW
    FORCE.YS.TE = -FORCE.TE * A.Y.TE / A.TW
    FORCE.ZN.TE = FORCE.TE * A.Z.TE / A.TW
```

```
FORCE.ZS.TE = -FORCE.TE * A.Z.TE / A.TW

FORCE.X.BSW = FORCE.BSW * A.X.BSW / A.BSW
FORCE.Y.BSW = FORCE.BSW * A.Y.BSW / A.BSW
FORCE.Z.BSW = FORCE.BSW * A.Z.BSW / A.BSW
FORCE.X.BSE = FORCE.BSE * A.X.BSE / A.BSE
FORCE.Y.BSE = FORCE.BSE * A.Y.BSE / A.BSE
FORCE.Z.BSE = FORCE.BSE * A.Z.BSE / A.BSE

FORCE.X.BNW = FORCE.BNW * A.X.BNW / A.BNW
FORCE.Y.BNW = FORCE.BNW * A.Y.BNW / A.BNW
FORCE.Z.BNW = FORCE.BNW * A.Z.BNW / A.BNW
FORCE.X.BNE = FORCE.BNE * A.X.BNE / A.BNE
FORCE.Y.BNE = FORCE.BNE * A.Y.BNE / A.BNE
FORCE.Z.BNE = FORCE.BNE * A.Z.BNE / A.BNE

' Calculations for force components in vertical links.

FORCE.X.NE = FORCE.NE * L.X.NE / L.NE
FORCE.Y.NE = FORCE.NE * L.Y.NE / L.NE
FORCE.Z.NE = FORCE.NE * L.Z.NE / L.NE
FORCE.X.NW = FORCE.NW * L.X.NW / L.NW
FORCE.Y.NW = FORCE.NW * L.Y.NW / L.NW
FORCE.Z.NW = FORCE.NW * L.Z.NW / L.NW

FORCE.X.SE = FORCE.SE * L.X.SE / L.SE
FORCE.Y.SE = FORCE.SE * L.Y.SE / L.SE
FORCE.Z.SE = FORCE.SE * L.Z.SE / L.SE
FORCE.X.SW = FORCE.SW * L.X.SW / L.SW
FORCE.Y.SW = FORCE.SW * L.Y.SW / L.SW
FORCE.Z.SW = FORCE.SW * L.Z.SW / L.SW

END SUB
```

```
SUB Calc.Engr.Values
'****************************************************************************
SHARED FORCE.TE, FORCE.TC, FORCE.TW, FORCE.BNE, FORCE.BNW, FORCE.BSE, FORCE.BSW
SHARED FORCE.NE, FORCE.NW, FORCE.SE, FORCE.SW, DELTA.X.BSW, DELTA.X.BSE
SHARED XDUCER.1NX, XDUCER.2NX, XDUCER.3NX, XDUCER.1NY, XDUCER.3NY
SHARED XDUCER.1SX, XDUCER.2SX, XDUCER.3SX, XDUCER.1SY, XDUCER.3SY, THETA.Z.N
'****************************************************************************
' This subroutine takes the channel voltages from sub Sample.AD and
' multiples each by its calibration constant, which gives the engineering
' values for that channel.  The constants are in the form of engineering
' unit/volt.
'
   FORCE.TE = ChVolt(1) * XducerCon(1)
   FORCE.TC = ChVolt(2) * XducerCon(2)
   FORCE.TW = ChVolt(3) * XducerCon(3)
   FORCE.BSE = ChVolt(4) * XducerCon(4)
   FORCE.BSW = ChVolt(5) * XducerCon(5)
   FORCE.BNE = ChVolt(6) * XducerCon(6)
   FORCE.BNW = ChVolt(7) * XducerCon(7)
'
   FORCE.NE = ChVolt(8) * XducerCon(8)
   FORCE.NW = ChVolt(9) * XducerCon(9)
   FORCE.SE = ChVolt(10) * XducerCon(10)
   FORCE.SW = ChVolt(11) * XducerCon(11)
'
   XDUCER.1NX = ChVolt(16) * XducerCon(16)
   XDUCER.2NX = ChVolt(17) * XducerCon(17)
   XDUCER.3NX = ChVolt(18) * XducerCon(18)
   XDUCER.1NY = ChVolt(19) * XducerCon(19)
   XDUCER.3NY = ChVolt(20) * XducerCon(20)
   XDUCER.1SX = ChVolt(21) * XducerCon(21)
   XDUCER.2SX = ChVolt(22) * XducerCon(22)
   XDUCER.3SX = ChVolt(23) * XducerCon(23)
   XDUCER.1SY = ChVolt(24) * XducerCon(24)
   XDUCER.3SY = ChVolt(25) * XducerCon(25)

   DELTA.X.BSW = ChVolt(26) * XducerCon(26)
   DELTA.X.BSE = ChVolt(27) * XducerCon(27)

   THETA.Z.N = ChVolt(28) * XducerCon(28)

END SUB
```

```
SUB Constants
'*************************************************************************
SHARED NChannel
'*************************************************************************
' This subroutine allows the user to view the calibration constants which
' were read from the input file and to correct them if necessary.  The
' constants must be in the form of engineering unit/volt.
'
   IF NChannel < 19 THEN
550     CLS 0
        PRINT "A/D Board Calibration Constants"
        PRINT "CH #    Constant"
        FOR j = 1 TO NChannel
          PRINT USING "##"; j;
          PRINT "       "; SensCon(j)
        NEXT j
        LOCATE NChannel + 4, 1
        INPUT "Modify channel constants(Y or N)"; rsvp$
        IF (UCASE$(rsvp$) = "Y") THEN
          LOCATE NChannel + 5, 1
          INPUT "Enter channel # to modify: ", ch%
          LOCATE NChannel + 6, 1
          INPUT "Enter calibration constant: ", SensCon(ch%)
          LOCATE NChannel + 5, 1
          PRINT "                              "
          PRINT "                              "
          GOTO 550
        ELSEIF (UCASE$(rsvp$) = "N") THEN
          GOTO 551
        ELSE
          GOTO 550
551     END IF
   ELSEIF NChannel > 17 AND NChannel < 33 THEN
560     CLS 0
        PRINT "A/D Board Calibration Constants"
        PRINT "CH #  Constant        CH #  Constant"
        FOR j = 1 TO 18
          PRINT USING "##"; j;
          PRINT "       "; SensCon(j)
        NEXT j
        FOR j = 19 TO NChannel
          LOCATE 3 + j - 19, 25
          PRINT USING "##"; j;
          PRINT "       "; SensCon(j)
        NEXT j
        LOCATE 22, 1
        INPUT "Modify channel constants(Y or N)"; rsvp$
        IF (UCASE$(rsvp$) = "Y") THEN
562         LOCATE 23, 1
          INPUT "Enter channel # to modify: ", ch%
          IF ch% > NChannel OR ch% < 1 THEN
             LOCATE 23, 1
             PRINT "                              "
             GOTO 562
```

```
        END IF
        INPUT "Enter calibration constant: ", SensCon(ch%)
        LOCATE 23, 1
        PRINT "                              "
        PRINT "                              "
        GOTO 560
      ELSEIF (UCASE$(rsvp$) = "N") THEN
        GOTO 561
      ELSE
        GOTO 560
561     END IF
   END IF

END SUB
```

```
SUB Decision.Algorithm
'*************************************************************************
SHARED Delta.TE, Delta.TC, Delta.TW, Delta.BSE, Delta.BSW
SHARED Direction, Correction
SHARED FORCE.TE, FORCE.TC, FORCE.TW, FORCE.BSE, FORCE.BSW
'*************************************************************************
' This subroutine takes the computed axial force and some user input to
' make decisions about which actuators to move.  First, the force tolerance
' is computed.  This factor is used if the force imbalances in the top
' actuators are to be corrected.  Then the program looks at whether the
' rotation is being increased or decreased, and goes to the appropriate
' branch of the if-then statement.  Then, the axial force is analyzed and
' the program decides which actuator to move.  If an actuator is to be
' extended, its Delta factor is given a value of 1, if it is to be
' retracted, its Delta factor is given a value of -1.  These are used in
' the calculations for new actuator commands.
' The following factor that is computed is used to determine whether the
' forces in the top actuators are equal to each other or not.  The factor
' is a tolerance.
 Factor = .05 * FORCE.TE
  IF Direction > .5 AND Direction < 1.5 THEN
    IF P < 0 THEN
      Delta.TE = 0
      Delta.TC = 0
      Delta.TW = 0
      Delta.BSE = 1
      Delta.BSW = 1
    ELSEIF P > 0 AND Correction = -1 THEN
      Delta.TE = -1
      Delta.TC = -1
      Delta.TW = -1
      Delta.BSE = 0
      Delta.BSW = 0
    ELSEIF P > 0 AND Correction = 1 THEN
      IF ABS(FORCE.TW) < ABS(FORCE.TE + Factor) AND ABS(FORCE.TW) > ABS(FORCE.TE -
Factor) THEN
        IF ABS(FORCE.TC) < ABS(FORCE.TE + Factor) AND ABS(FORCE.TC) > ABS(FORCE.TE -
Factor) THEN
          Delta.TE = -1
          Delta.TC = -1
          Delta.TW = -1
          Delta.BSE = 0
          Delta.BSW = 0
        ELSEIF ABS(FORCE.TC) < ABS(FORCE.TE - Factor) THEN
          Delta.TE = 0
          Delta.TC = -1
          Delta.TW = 0
          Delta.BSE = 0
          Delta.BSW = 0
        ELSEIF ABS(FORCE.TC) > ABS(FORCE.TE + Factor) THEN
          Delta.TE = -1
          Delta.TW = -1
          Delta.TC = 0
          Delta.BSE = 0
```

```
          Delta.BSW = 0
        END IF
      ELSEIF ABS(FORCE.TW) < ABS(FORCE.TE - Factor) OR ABS(FORCE.TW) > ABS(FORCE.TE +
Factor) THEN
        Delta.TE = -1
        Delta.TC = -1
        Delta.TW = -1
        Delta.BSW = 0
        Delta.BSE = 0
      END IF
    END IF
  ELSEIF Direction < 0 THEN
    IF P < 0 THEN
      Delta.TE = 0
      Delta.TC = 0
      Delta.TW = 0
      Delta.BSE = -1
      Delta.BSW = -1
    ELSEIF P > 0 AND Correction = -1 THEN
      Delta.TE = 1
      Delta.TC = 1
      Delta.TW = 1
      Delta.BSE = 0
      Delta.BSW = 0
    ELSEIF P > 0 AND Correction = 1 THEN
      IF ABS(FORCE.TW) < ABS(FORCE.TE + Factor) AND ABS(FORCE.TW) > ABS(FORCE.TE -
Factor) THEN
        IF ABS(FORCE.TC) < ABS(FORCE.TE + Factor) AND ABS(FORCE.TC) > ABS(FORCE.TE -
Factor) THEN
          Delta.TE = 1
          Delta.TC = 1
          Delta.TW = 1
          Delta.BSW = 0
          Delta.BSE = 0
        ELSEIF ABS(FORCE.TC) < ABS(FORCE.TE - Factor) THEN
          Delta.TE = 1
          Delta.TC = 0
          Delta.TW = 1
          Delta.BSW = 0
          Delta.BSE = 0
        ELSEIF ABS(FORCE.TC) > ABS(FORCE.TE + Factor) THEN
          Delta.TE = 0
          Delta.TC = 1
          Delta.TW = 0
          Delta.BSW = 0
          Delta.BSE = 0
        END IF
      ELSEIF ABS(FORCE.TW) < ABS(FORCE.TE - Factor) OR ABS(FORCE.TW) > ABS(FORCE.TE +
Factor) THEN
        Delta.TE = 1
        Delta.TC = 1
        Delta.TW = 1
        Delta.BSW = 0
        Delta.BSE = 0
```

```
        END IF
      END IF
    END IF

END SUB
```

```
SUB Equilibrium.Eqns
'*****************************************************************************
SHARED FORCE.XN.TE, FORCE.XN.TC, FORCE.XN.TW, FORCE.X.BNE, FORCE.X.BNW
SHARED FORCE.X.BSE, FORCE.X.BSW
SHARED FORCE.XS.TE, FORCE.XS.TC, FORCE.XS.TW
SHARED FORCE.X.NE, FORCE.X.SE, FORCE.X.NW, FORCE.X.SW
SHARED FORCE.YN.TE, FORCE.YN.TC, FORCE.YN.TW, FORCE.Y.BNE, FORCE.Y.BNW
SHARED FORCE.Y.BSE, FORCE.Y.BSW
SHARED FORCE.Y.NE, FORCE.Y.SE, FORCE.Y.NW, FORCE.Y.SW
SHARED FORCE.YS.TE, FORCE.YS.TC, FORCE.YS.TW
SHARED CN.X.TE, CN.Y.TE, CN.X.TC, CN.Y.TC, CN.X.TW, CN.Y.TW
SHARED CS.X.TE, CS.Y.TE, CS.X.TC, CS.Y.TC, CS.X.TW, CS.Y.TW
SHARED CN.X.BNE, CN.Y.BNE, CN.X.BNW, CN.Y.BNW
SHARED CS.X.BSE, CS.Y.BSE, CS.X.BSW, CS.Y.BSW
SHARED D.X.NE, D.Y.NE, D.X.NW, D.Y.NW, D.X.SE, D.Y.SE, D.X.SW, D.Y.SW
SHARED P.N, P.S, P, MZ.N, MZ.S, MY.N, MY.S
'*****************************************************************************
' This subroutine takes all of the actuator and link force components and
' computes the axial force and moments applied to the structure using
' various equilibrium equations.
' First, the axial force at the north and south grillages are computed, then
' averaged.
'
   P.N = -FORCE.XN.TE - FORCE.XN.TC - FORCE.XN.TW - FORCE.X.BNE - FORCE.X.BNW -
FORCE.X.NE - FORCE.X.NW

   P.S = -FORCE.XS.TE - FORCE.XS.TC - FORCE.XS.TW - FORCE.X.BSE - FORCE.X.BSW -
FORCE.X.SE - FORCE.X.SW

   P = (P.N - P.S) / 2

' Next, the moment about the Z axis at each grillage is computed. Note that
' the equation for each of these is broken up into two equations, simply
' because of their length. The final values for moment are converted to
' kip-ft.

   MZ.N.1 = FORCE.XN.TW * CN.Y.TW + FORCE.XN.TE * CN.Y.TE + FORCE.XN.TC * CN.Y.TC +
FORCE.X.BNE * CN.Y.BNE + FORCE.X.BNW * CN.Y.BNW + FORCE.X.NE * D.Y.NE + FORCE.X.NW
* D.Y.NW

   MZ.N.2 = FORCE.YN.TW * CN.X.TW + FORCE.YN.TE * CN.X.TE + FORCE.YN.TC * CN.X.TC +
FORCE.Y.BNE * CN.X.BNE + FORCE.Y.BNW * CN.X.BNW + FORCE.Y.NE * D.X.NE + FORCE.Y.NW
* D.X.NW

   MZ.N = (MZ.N.1 - MZ.N.2) / 12
'
'
   MZ.S.1 = FORCE.XS.TW * CS.Y.TW + FORCE.XS.TE * CS.Y.TE + FORCE.XS.TC * CS.Y.TC +
FORCE.X.BSE * CS.Y.BSE + FORCE.X.BSW * CS.Y.BSW + FORCE.X.SE * D.Y.SE + FORCE.X.SW *
D.Y.SW

   MZ.S.2 = FORCE.YS.TW * CS.X.TW + FORCE.YS.TE * CS.X.TE + FORCE.YS.TC * CS.X.TC +
FORCE.Y.BSE * CS.X.BSE + FORCE.Y.BSW * CS.X.BSW + FORCE.Y.SE * D.X.SE + FORCE.Y.SW *
D.X.SW
```

```
MZ.S = (MZ.S.2 - MZ.S.1) / 12

' Now the moments are found about the Y axis at the north and south
' grillages.  Note again that the moment equations are split into two
' equations each because of their length.

   MY.N.1 = FORCE.XN.TW * CN.Z.TW + FORCE.XN.TC * CN.Z.TC + FORCE.XN.TE * CN.Z.TE +
FORCE.X.BNE * CN.Z.BNE + FORCE.X.BNW * CN.Z.BNW

   MY.N.2 = FORCE.ZN.TW * CN.X.TW + FORCE.ZN.TC * CN.X.TC + FORCE.ZN.TE * CN.X.TC +
FORCE.Z.BNE * CN.X.BNE + FORCE.Z.BNW * CN.X.BNW

   MY.N = (MY.N.2 - MY.N.1) / 12
'
'
   MY.S.1 = FORCE.XS.TW * CS.Z.TW + FORCE.XS.TC * CS.Z.TC + FORCE.XS.TE * CS.Z.TE +
FORCE.X.BSE * CS.Z.BSE + FORCE.X.BSW * CS.Z.BSW

   MY.S.2 = FORCE.ZS.TW * CS.X.TW + FORCE.ZS.TC * CS.X.TC + FORCE.ZS.TE * CS.X.TE +
FORCE.Z.BSE * CS.X.BSE + FORCE.Z.BSW * CS.X.BSW

   MY.S = (MY.S.1 - MY.S.2) / 12

END SUB
```

```
SUB F3Flag.Check
'***************************************************************************
SHARED nPass, F3Flag
'***************************************************************************
' This subroutine checks to see whether the F3 key has been pressed, and if
' so, the Program Parameters menu is displayed in the menu box.  This allows
' the user to change several of the program's key parameters.
'
  IF F3Flag = 1 THEN
    COLOR 15, 0
150   LOCATE 35, 3
    PRINT "    PROGRAM PARAMETERS    "
    LOCATE 36, 3
    PRINT "  1 - DAC STEP SIZE       "
    LOCATE 37, 3
    PRINT "  2 - SAVE DATA STEP SIZE  "
    LOCATE 38, 3
    PRINT "  3 - NUMBER OF LOOP STEPS  "
    LOCATE 39, 3
    PRINT "  4 - MORE PARAMETERS      "
    LOCATE 40, 3
    PRINT "  5 - EXIT MENU           "
    LOCATE 42, 3
    INPUT "   DESIRED ACTION: ", resp
    CALL Clear.Window
    LOCATE 37, 5
    IF resp > .5 AND resp < 1.5 THEN
      PRINT "   CHANGE DAC STEP "
    ELSEIF resp > 1.5 AND resp < 2.5 THEN
      PRINT "CHANGE SAVE DATA STEP SIZE"
    ELSEIF resp > 2.5 AND resp < 3.5 THEN
      PRINT "CHANGE NUMBER OF LOOP STEPS"
    ELSEIF resp > 3.5 AND resp < 4.5 THEN
      PRINT "   MORE PARAMETERS      "
    ELSEIF resp > 4.5 AND resp < 5.5 THEN
      PRINT "     EXIT MENU "
    ELSE
      CALL Clear.Window
      GOTO 150
    END IF
    LOCATE 38, 5
    PRINT "   HAS BEEN CHOSEN."
    LOCATE 39, 3
    INPUT "IS THIS CORRECT (Y OR N)?: ", ans$
'
    IF UCASE$(ans$) = "Y" THEN
      GOTO 152
    ELSE
      CALL Clear.Window
      GOTO 150
    END IF
'
152   CALL Clear.Window
    IF resp > .5 AND resp < 1.5 THEN
```

```
      CALL DAC.Step.Size
      nPass = 0
      CALL Main.Display
    ELSEIF resp > 1.5 AND resp < 2.5 THEN
      CALL Save.Data.Step.Size
    ELSEIF resp > 2.5 AND resp < 3.5 THEN
      CALL Change.Loop.Steps
    ELSEIF resp > 3.5 AND resp < 4.5 THEN
157     CALL Clear.Window
        LOCATE 36, 6
        PRINT "   MORE PARAMETERS    "
        LOCATE 37, 6
        PRINT " 1 - TOLERANCE SIZE    "
        LOCATE 38, 6
        PRINT " 2 - GRAPH SCALE       "
        LOCATE 39, 6
        PRINT " 3 - CORRECT IMBALANCES "
        LOCATE 40, 6
        PRINT " 4 - EXIT MENU         "
        LOCATE 41, 6
        INPUT "  DESIRED ACTION: ", resp2
        CALL Clear.Window
        LOCATE 37, 5
        IF resp2 = 1 THEN
            PRINT " CHANGE TOLERANCE SIZE "
        ELSEIF resp2 = 2 THEN
          PRINT "   CHANGE GRAPH SCALE  "
        ELSEIF resp2 = 3 THEN
          PRINT "   CORRECT IMBALANCES  "
        ELSEIF resp2 = 4 THEN
          PRINT "       EXIT MENU "
        ELSE
          GOTO 157
        END IF
        LOCATE 38, 5
        PRINT "    HAS BEEN CHOSEN."
        LOCATE 39, 3
        INPUT "IS THIS CORRECT (Y OR N)?: ", ans$
        IF UCASE$(ans$) = "Y" THEN
          CALL Clear.Window
          GOTO 156
        ELSE
          CALL Clear.Window
          GOTO 157
        END IF
156     IF resp2 = 1 THEN
          CALL Tolerance.Size
        ELSEIF resp2 = 2 THEN
          CALL Choose.Graph.Scale
        ELSEIF resp2 = 3 THEN
          CALL Correct.Imbalances
        ELSEIF resp2 = 4 THEN
          GOTO 155
        END IF
```

```
    ELSEIF resp > 4.5 AND resp < 5.5 THEN
      GOTO 155
    END IF

155 CALL Clear.Window
  nPass = 0
  F3Flag = -1
  END IF

END SUB
```

```
SUB F5Flag.Check
'************************************************************************
SHARED Direction, nPass, F5Flag
'************************************************************************
' This subroutine checks to see if F5 was pressed, and if so, the Program
' Functions menu is brought up in the menu box.  The user is given several
' options and based on her choice, a separate subroutine is called to
' execute it.
'
  IF F5Flag = 1 THEN
    COLOR 15, 0
160   LOCATE 35, 3
    PRINT "    PROGRAM FUNCTIONS  "
    LOCATE 36, 3
    PRINT "1 - INCREASE ROTATION     "
    LOCATE 37, 3
    PRINT "2 - DECREASE ROTATION     "
    LOCATE 38, 3
    PRINT "3 - VIEW RAW VOLTAGES     "
    LOCATE 39, 3
    PRINT "4 - VIEW FULL GRAPH       "
    LOCATE 40, 3
    PRINT "5 - MANUAL COMMAND ADJUSTMENT"
    LOCATE 41, 3
    PRINT "6 - EXIT MENU             "
    LOCATE 42, 3
    INPUT "    DESIRED ACTION:", resp
    CALL Clear.Window
    LOCATE 37, 5
    IF resp > .5 AND resp < 1.5 THEN
      PRINT "   INCREASE ROTATION "
    ELSEIF resp > 1.5 AND resp < 2.5 THEN
      PRINT "   DECREASE ROTATION "
    ELSEIF resp > 2.5 AND resp < 3.5 THEN
      PRINT "   VIEW RAW VOLTAGES "
    ELSEIF resp > 3.5 AND resp < 4.5 THEN
      PRINT "    VIEW FULL GRAPH "
    ELSEIF resp > 4.5 AND resp < 5.5 THEN
      PRINT "MANUAL COMMAND ADJUSTMENT"
    ELSEIF resp > 5.5 AND resp < 6.5 THEN
      PRINT "      EXIT MENU "
    ELSE
      CALL Clear.Window
      GOTO 160
    END IF
    LOCATE 38, 5
    PRINT "    HAS BEEN CHOSEN."
    LOCATE 39, 3
    INPUT "IS THIS CORRECT (Y OR N)?: ", ans$

    IF UCASE$(ans$) = "Y" THEN
      GOTO 162
    ELSE
      CALL Clear.Window
```

```
      GOTO 160
    END IF

162   CALL Clear.Window
    IF resp > .5 AND resp < 1.5 THEN
      Direction = 1
    ELSEIF resp > 1.5 AND resp < 2.5 THEN
      Direction = -1
      nPass = 0
    ELSEIF resp > 2.5 AND resp < 3.5 THEN
      CALL Zero
      nPass = 0
      CALL Main.Display
    ELSEIF resp > 3.5 AND resp < 4.5 THEN
      nPlot = 0
      CALL Large.Graph
      nPass = 0
      CALL Main.Display
    ELSEIF resp > 4.5 AND resp < 5.5 THEN
      CALL Volt.Command.Adj
      nPass = 0
      CALL Main.Display
    ELSEIF resp > 5.5 AND resp < 6.5 THEN
      nPass = 0
      GOTO 165
    END IF
165 F5Flag = -1
  END IF

END SUB
```

```
SUB Graph
'*******************************************************************************
SHARED nData, THETA.Z.N, MZ.N, nPass, scale, Direction
'*******************************************************************************
' This subroutine sets up the graphics for the moment vs. rotation plot and
' plots the small graph on the main display.
' First, allocate the area for the plot and redefine the coordinates.

    VIEW (260, 175)-(630, 345), 0, 15
    WINDOW (0, 105)-(100, 0)

' Plot the main axes lines.

    LINE (13, 22)-(13, 102), 15
    LINE (13, 22)-(93, 22), 15

' Draw the vertical lines.

    LINE (23, 22)-(23, 102), 15
    LINE (33, 22)-(33, 102), 15
    LINE (43, 22)-(43, 102), 15
    LINE (53, 22)-(53, 102), 15
    LINE (63, 22)-(63, 102), 15
    LINE (73, 22)-(73, 102), 15
    LINE (83, 22)-(83, 102), 15
    LINE (93, 22)-(93, 102), 15

' Draw the horizontal lines.

    LINE (13, 32)-(93, 32), 15
    LINE (13, 42)-(93, 42), 15
    LINE (13, 52)-(93, 52), 15
    LINE (13, 62)-(93, 62), 15
    LINE (13, 72)-(93, 72), 15
    LINE (13, 82)-(93, 82), 15
    LINE (13, 92)-(93, 92), 15
    LINE (13, 102)-(93, 102), 15

' Print the text.

    COLOR 15
    LOCATE 30, 34
    PRINT "M"
    LOCATE 31, 34
    PRINT "O"
    LOCATE 32, 34
    PRINT "M"
    LOCATE 33, 34
    PRINT "E"
    LOCATE 34, 34
    PRINT "N"
    LOCATE 35, 34
    PRINT "T"
    LOCATE 42, 53
```

```
    PRINT "ROTATION"
    COLOR 15, 0
    LOCATE 40, 39
    PRINT "0"

' Print the values on the axes, depending on the scale chosen.

    IF scale = 1 THEN
        LOCATE 38, 36
        PRINT "1E3"
        LOCATE 36, 36
        PRINT "2E3"
        LOCATE 34, 36
        PRINT "3E3"
        LOCATE 32, 36
        PRINT "4E3"
        LOCATE 30, 36
        PRINT "5E3"
        LOCATE 28, 36
        PRINT "6E3"
        LOCATE 26, 36
        PRINT "7E3"
        LOCATE 24, 36
        PRINT "8E3"
        LOCATE 40, 46
        PRINT ".0025"
        LOCATE 40, 56
        PRINT ".005"
        LOCATE 40, 65
        PRINT ".0075"
        LOCATE 40, 75
        PRINT ".01"
    ELSEIF scale = 2 THEN
        LOCATE 38, 36
        PRINT "2E3"
        LOCATE 36, 36
        PRINT "4E3"
        LOCATE 34, 36
        PRINT "6E3"
        LOCATE 32, 36
        PRINT "8E3"
        LOCATE 30, 35
        PRINT "10E4"
        LOCATE 28, 35
        PRINT "12E3"
        LOCATE 26, 35
        PRINT "14E3"
        LOCATE 24, 35
        PRINT "16E3"
        LOCATE 40, 47
        PRINT ".005"
        LOCATE 40, 57
        PRINT ".01"
        LOCATE 40, 65
```

```
    PRINT ".015"
    LOCATE 40, 75
    PRINT ".02"
ELSEIF scale = 3 THEN
    LOCATE 38, 36
    PRINT "3E3"
    LOCATE 36, 36
    PRINT "6E3"
    LOCATE 34, 36
    PRINT "9E3"
    LOCATE 32, 35
    PRINT "12E3"
    LOCATE 30, 35
    PRINT "15E3"
    LOCATE 28, 35
    PRINT "18E3"
    LOCATE 26, 35
    PRINT "21E3"
    LOCATE 24, 35
    PRINT "24E3"
    LOCATE 40, 47
    PRINT ".0075"
    LOCATE 40, 56
    PRINT ".015"
    LOCATE 40, 65
    PRINT ".0225"
    LOCATE 40, 75
    PRINT ".03"
ELSEIF scale = 4 THEN
    LOCATE 38, 36
    PRINT "4E3"
    LOCATE 36, 36
    PRINT "8E3"
    LOCATE 34, 35
    PRINT "12E3"
    LOCATE 32, 35
    PRINT "16E3"
    LOCATE 30, 35
    PRINT "20E3"
    LOCATE 28, 35
    PRINT "24E3"
    LOCATE 26, 35
    PRINT "28E3"
    LOCATE 24, 35
    PRINT "32E3"
    LOCATE 40, 46
    PRINT ".0125"
    LOCATE 40, 56
    PRINT ".025"
    LOCATE 40, 65
    PRINT ".0375"
    LOCATE 40, 75
    PRINT ".05"
END IF
```

```
' The following two statements take the present MZ.N and THETA.Z.N values and
' add them to the M() and Theta() arrays so that these values can be
' plotted.

    GTHETA(nData) = THETA.Z.N
    M(nData) = MZ.N
    COLOR 2

' The following if-then block plots the graph.  Different equations are
' needed, depending on the scale chosen and depending on whether the main
' display was just called (if nPass = 0, the entire graph needs to be re-
' plotted) or if we are just plotting the present values of moment and
' rotation.
' The first part of the if-then plots the moment range of 0-8000 k-ft and rotation
' range of 0 - 0.01 rad.
      IF scale = 1 THEN
       IF nPass = 0 THEN
         FOR i = 1 TO nData
           IF i = 1 THEN
             LINE (13, 22)-(GTHETA(1) * 8000 + 13, M(1) / 100 + 22)
           END IF
           LINE (GTHETA(i - 1) * 8000 + 13, M(i - 1) / 100 + 22)-(GTHETA(i) * 8000 + 13, M(i) / 100 + 22)
         NEXT i
       ELSEIF nPass > 0 THEN
         LINE (GTHETA(nData) * 8000 + 13, M(nData) / 100 + 22)-(GTHETA(nData - 1) * 8000 + 13,
M(nData - 1) / 100 + 22)
       END IF

' The following are the equations for the moment range of 0 – 16000 k-ft
' and rotation range of 0 – 0.02 rad.

      ELSEIF scale = 2 THEN
       IF nPass = 0 THEN
         FOR i = 1 TO nData
           IF i = 1 THEN
             LINE (13, 22)-(GTHETA(1) * 4000 + 13, M(1) / 200 + 22)
           END IF
           LINE (GTHETA(i - 1) * 4000 + 13, M(i - 1) / 200 + 22)-(GTHETA(i) * 4000 + 13, M(i) / 200 + 22)
         NEXT i
       ELSEIF nPass > 0 THEN
         LINE (GTHETA(nData) * 4000 + 13, M(nData) / 200 + 22)-(GTHETA(nData - 1) * 4000 + 13,
M(nData - 1) / 200 + 22)
       END IF

' The following are the equations for the moment range of 0 – 24000 k-ft
' and rotation range of 0 – 0.03 rad.

      ELSEIF scale = 3 THEN
       IF nPass = 0 THEN
         FOR i = 1 TO nData
           IF i = 1 THEN
             LINE (13, 22)-(GTHETA(1) * 2667 + 13, M(1) / 300 + 22)
           END IF
```

```
        LINE (GTHETA(i - 1) * 2667 + 13, M(i - 1) / 300 + 22)-(GTHETA(i) * 2667 + 13, M(i) / 300 + 22)
      NEXT i
    ELSEIF nPass > 0 THEN
      LINE (GTHETA(nData) * 2667 + 13, M(nData) / 300 + 22)-(GTHETA(nData - 1) * 2667 + 13,
M(nData - 1) / 300 + 22)
    END IF

' The following are the equations for the moment range of 0 – 32000 k-ft
' and rotation range of 0 – 0.04 rad.

   ELSEIF scale = 4 THEN
    IF nPass = 0 THEN
      FOR i = 1 TO nData
        IF i = 1 THEN
          LINE (13, 22)-(GTHETA(1) * 2000 + 13, M(1) / 400 + 22)
        END IF
        LINE (GTHETA(i - 1) * 2000 + 13, M(i - 1) / 400 + 22)-(GTHETA(i) * 2000 + 13, M(i) / 400 + 22)
      NEXT i
    ELSEIF nPass > 0 THEN
      LINE (GTHETA(nData) * 2000 + 13, M(nData) / 400 + 22)-(GTHETA(nData - 1) * 2000 + 13,
M(nData - 1) / 400 + 22)
    END IF
   END IF
230  WINDOW

END SUB
```

```
SUB Initialize.Acts
'**************************************************************************
SHARED Command.TE%, Command.TC%, Command.TW%, Command.BSE%, Command.BSW%
SHARED Command.16%, Command.Trigger AS INTEGER
'**************************************************************************
' This subroutine is used for initializing the actuator positions before
' hydraulic pressure is applied.  The voltage value from each feedback
' transducer is read from the MTS 458 MicroConsole.  These voltages are then entered
' as the initial  command voltages for the corresponding actuators.  Once
' all 5 initial voltages are entered, the subroutine calls Volt.Command.Adj,
' which will allow the user to fine tune these voltages so that the error
' between the command signals and the feedback signals are zero (as
' measured by the MicroConsole).  Then hydraulic pressure may be applied.
' The initial command limits are set as +/- 9.95 volts.
'
   CLS
   WIDTH 80, 25
   COLOR 15, 4
   LOCATE 1, 25
   PRINT "Initialize Actuator Positions"
   LOCATE 3, 1
   PRINT "         Enter the output voltage for the actuator specified "
   PRINT "            Voltage must be between -9.95 and 9.95 V"
   PRINT "                                                   "
   PRINT "         T = Top  B = Bottom       N = North   S = South  "
   PRINT "                    E = East   W = West"
   PRINT "                                                     "
460  LOCATE 10, 15
   INPUT "Use Default (0.00 V) for all Actuators? (Y or N): ", rsvp$
   IF UCASE$(rsvp$) = "Y" THEN
      CommandV.TE = 0
      CommandV.TC = 0
      CommandV.TW = 0
      CommandV.BSE = 0
      CommandV.BSW = 0
      GOTO 466
   ELSEIF UCASE$(rsvp$) = "N" THEN
      GOTO 461
   ELSE
      GOTO 460
   END IF

461  LOCATE 10, 15
   PRINT "                                       "
   PRINT "                                       "
   PRINT "                                       "
   LOCATE 10, 15
'
   INPUT "Actuator TE:  ", CommandV.TE
   LOCATE 11, 15
   PRINT USING "Actuator TE command is +##.### V "; CommandV.TE
   LOCATE 12, 15
   INPUT "Is this correct (Y or N)? ", rsvp$
   IF UCASE$(rsvp$) = "Y" THEN
```

```
    IF CommandV.TE > 9.95 THEN
      CommandV.TE = 9.95
    END IF
    IF CommandV.TE < -9.95 THEN
      CommandV.TE = -9.95
    END IF
    GOTO 462
  ELSE
    GOTO 461
  END IF
'
462 LOCATE 10, 15
  PRINT "                              "
  PRINT "                              "
  PRINT "                              "
  LOCATE 10, 15
'
  INPUT "Actuator TC:  ", CommandV.TC
  LOCATE 11, 15
  PRINT USING "Actuator TC command is +##.### V "; CommandV.TC
  LOCATE 12, 15
  INPUT "Is this correct (Y or N)? ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
    IF CommandV.TC > 9.95 THEN
      CommandV.TC = 9.95
    END IF
    IF CommandV.TC < -9.95 THEN
      CommandV.TC = -9.95
    END IF
    GOTO 463
  ELSE
    GOTO 462
  END IF
'
463 LOCATE 10, 15
  PRINT "                              "
  PRINT "                              "
  PRINT "                              "
'
  LOCATE 10, 15
  INPUT "Actuator TW:  ", CommandV.TW
  LOCATE 11, 15
  PRINT USING "Actuator TW command is +##.### V "; CommandV.TW
  LOCATE 12, 15
  INPUT "Is this correct (Y or N)? ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
    IF CommandV.TW > 9.95 THEN
      CommandV.TW = 9.95
    END IF
    IF CommandV.TW < -9.95 THEN
      CommandV.TW = -9.95
    END IF
    GOTO 464
  ELSE
```

```
    GOTO 463
  END IF
'
464 LOCATE 10, 15
  PRINT "                                        "
  PRINT "                                        "
  PRINT "                                        "
'
  LOCATE 10, 15
  INPUT "Actuator BSE:  ", CommandV.BSE
  LOCATE 11, 15
  PRINT USING "Actuator BSE command is +##.### V "; CommandV.BSE
  LOCATE 12, 15
  INPUT "Is this correct (Y or N)? ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
    IF CommandV.BSE > 9.95 THEN
      CommandV.BSE = 9.95
    END IF
    IF CommandV.BSE < -9.95 THEN
      CommandV.BSE = -9.95
    END IF
    GOTO 465
  ELSE
    GOTO 464
  END IF
'
465 LOCATE 10, 15
  PRINT "                                        "
  PRINT "                                        "
  PRINT "                                        "
'
  LOCATE 10, 15
  INPUT "Actuator BSW:  ", CommandV.BSW
  LOCATE 11, 15
  PRINT USING "Actuator BSW command is +##.### V "; CommandV.BSW
  LOCATE 12, 15
  INPUT "Is this correct (Y or N)? ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
    IF CommandV.BSW > 9.95 THEN
      CommandV.BSW = 9.95
    END IF
    IF CommandV.BSW < -9.95 THEN
      CommandV.BSW = -9.95
    END IF
    GOTO 466
  ELSE
    GOTO 465
  END IF
'
466 Command.TE% = CommandV.TE / .004883 + 2048
  Command.TC% = CommandV.TC / .004883 + 2048
  Command.TW% = CommandV.TW / .004883 + 2048
  Command.BSE% = CommandV.BSE / .004883 + 2048
  Command.BSW% = CommandV.BSW / .004883 + 2048
```

```
   Command.16% = Command.Trigger
'
   CALL Move.Actuators
   CALL Volt.Command.Adj
 END SUB

SUB Initialize.Boards
'******************************************************************************
SHARED NChannel
'******************************************************************************
' In this subroutine, the A/D and D/A boards are initialized.  It establishes
' such parameters as the base address for each, the number of A/D channels
' and their gain.  The only parameter that may need to be changed is the
' sampling rate of the A/D board.  The rest of this subroutine should not
' be changed.
'
   szCfgName = "DAS1800.CFG" + CHR$(0)
   nDasErr = DAS1800DEVOPEN%(SSEGADD(szCfgName), 1)
   IF nDasErr <> 0 THEN
     BEEP
     PRINT "ERROR "; HEX$(nDasErr); " OCCURRED DURING 'DAS1800DEVOPEN'": STOP
   END IF

' This step establishes communication with the driver through the device
' handle.

   nDasErr = DAS1800GETDEVHANDLE%(0, hDev)
   IF (nDasErr <> 0) THEN
     BEEP
     PRINT "ERROR "; HEX$(nDasErr); " OCCURRED DURING 'DAS1800GETDEVHANDLE'": STOP
   END IF

' To perform any of the A/D operations, you must first get a handle to an A/D
' frame (Data tables inside the driver pertaining to the A/D operations).

   nDasErr = KGetADFrame%(hDev, hAD)
   IF (nDasErr <> 0) THEN
     BEEP
     PRINT "ERROR "; HEX$(nDasErr); " OCCURRED DURING 'KGETADFRAME'": STOP
   END IF

' Assign the data array declared above to the frame handle.  This says that
' the data sampled by the board will be stored in the array DataBuf.  It also
' specifies how many samples to take in one interrupt step, here this is the
' number of channels. (Will take one sample per channel, then stop).  If the
' number of samples is twice the number of channels, it will sample each
' channel twice, but the first sample for each channel is overwritten.

   nDasErr = KSetBufI%(hAD, DataBuf(0), NChannel)
   IF nDasErr <> 0 THEN
     BEEP
     PRINT "ERROR "; HEX$(nDasErr); " OCCURRED DURING 'KSetBuf'": STOP
   END IF
```

```
' Set up the conversion clock rate: 5000000/dwFactor (Hz). This controls the
' speed at which the board will sample.  Smaller dwFactors mean a faster
' sampling rate.

   dwFactor = 10000
   nDasErr = KSetClkRate%(hAD, dwFactor)
   IF nDasErr <> 0 THEN
      BEEP
      PRINT "ERROR "; HEX$(nDasErr); " OCCURRED DURING 'KSetClkRate'": STOP
   END IF

' Set the channels at which to start and stop and their gain.  The gain is
' set for one (gain code is 0) but when the voltage is calculated in sub
' Sample.AD, the "gain", or voltage range, that was entered by the user is
' part of the equation.

   nDasErr = KSetStartStopG%(hAD, 0, NChannel - 1, 0)
   IF nDasErr <> 0 THEN
      BEEP
      PRINT "ERROR "; HEX$(nDasErr); " OCCURRED DURING 'KSetStartStopG'": STOP
   END IF

' Now, initialize the D/A board.  This step initializes the internal data
' tables according to the information contained in the configuration file
' DDA16.CFG. Make sure that the configuration of the board matches that of
' the file, that the settings are correct for the application, and that the
' file is in the directory in which you are running the program.

   A$ = "DDA16.CFG" + CHR$(0)
   DERR = DDA16DEVOPEN%(SSEGADD(A$), 1)
   IF DERR <> 0 THEN
     BEEP
     PRINT "ERROR "; HEX$(DERR); " OCCURRED DURING '..DEVOPEN'"
     STOP
   END IF

' This step establishes communication with the driver through the Device
' Handle for board 0.

   DERR = DDA16GETDEVHANDLE%(0, hDDA16)
   IF (DERR <> 0) THEN
     BEEP
     PRINT "ERROR "; HEX$(DERR); " OCCURRED DURING '..GETDEVHANDLE'"
     STOP
   END IF

' To perform any D/A operations, you must first get a Handle to a D/A frame
' (the data tables inside the driver pertaining to D/A operations).

   DERR = KGetDAFrame%(hDDA16, hDA)
   IF (DERR <> 0) THEN
     BEEP
     PRINT "ERROR "; HEX$(DERR); " OCCURRED DURING 'KGETADFRAME'"
     STOP
```

```
  END IF

' Assign the data array to the Frame Handle.  This tells the board what
' count values to assign to the channels.

  DERR = KSetBufI%(hDA, ACommand(1), 16)
  IF DERR <> 0 THEN
    BEEP
    PRINT "ERROR "; HEX$(DERR); " OCCURRED DURING 'KSetBufI'"
    STOP
  END IF

' Set the desired channels for D/A operation.  The first number is the first
' channel in the group and the second number is the last channel to be
' sampled.  Note that the board calls the first channel 0 and not 1.

  DERR = KSetStartStopChn%(hDA, 0, 15)
  IF DERR <> 0 THEN
    BEEP
    PRINT "ERROR "; HEX$(DERR); " OCCURRED DURING 'KSetStartStopChn'"
    STOP
  END IF

' Specify the internal clock rate: 1000 tics at 1MHz/tic is a 1 KHz rate.
' The number of tics is how many clock tics will be counted until the
' channels are updated with new values (smaller number means faster
' execution).

  DERR = KSetClkRate%(hDA, 1000)
  IF DERR <> 0 THEN
    BEEP
    PRINT "ERROR "; HEX$(DERR); " OCCURRED DURING 'KSetClkRate'"
    STOP
  END IF

' The following loop initializes the array ChData to zero.  These are the
' count values that are sampled by the A/D board before they are converted
' to voltages.

  FOR N = 1 TO NChannel
    ChData(0, N) = 0
  NEXT N
  FOR N = 0 TO NChannel - 1
    DataBuf(N) = 0
  NEXT N

END SUB
```

```
SUB Initialize.Vars
'*************************************************************************
SHARED Direction, F10Flag, F3Flag, F9Flag, F5Flag, nData, nPass
SHARED scale, Loop.Steps, nStep, Save.Data.Step, Tolerance, F6Flag, F4Flag
SHARED DACStep.TE%, DACStep.TC%, DACStep.TW%, DACStep.BSE%, DACStep.BSW%
SHARED Delta.TE%, Delta.TC%, Delta.TW%, Delta.BSW%, Delta.BSE%, Correction
SHARED Command.Trigger AS INTEGER, Save.Data.Counter, Pause.Execution
SHARED F7Flag, nPlot, nSave
'*************************************************************************
' This subroutine is called at the beginning of the program. It simply
' initializes many of the program's variables giving them default values.
'
  Direction = 0
  Correction = -1
  F10Flag = -1
  F5Flag = -1
  F4Flag = -1
  F3Flag = -1
  F9Flag = -1
  F7Flag = -1
  F6Flag = -1
  DACStep.TE% = 1
  DACStep.TC% = 1
  DACStep.TW% = 1
  DACStep.BSE% = 1
  DACStep.BSW% = 1
  Delta.TE% = 0
  Delta.TC% = 0
  Delta.TW% = 0
  Delta.BSE% = 0
  Delta.BSW% = 0
  nData = 1
  nSave = 0
  nPlot = 0
  nPass = 0
  scale = 1
  Pause.Execution = 0
  Loop.Steps = 20
  Save.Data.Step = 10
  Save.Data.Counter = 1
  nStep = 1
  Tolerance = 1!
  Command.Trigger = 3072

END SUB
```

```
SUB Large.Graph
'*****************************************************************************
SHARED nPlot, scale, nData, F10Flag, nPass
'*****************************************************************************
' This subroutine sets up the graphics for the large moment vs. rotation
' plot and graphs the values.
' If the subroutine is called for the first time, nPlot = 0 and the graphics
' must be set up.  If not, nPlot>0 and the graphics setup can be skipped.
' Only one scale can be plotted on this graph, the entire moment range of
' 0 – 32,000 k-ft and entire rotation range of 0 – 0.04 k-ft.
'
206 IF nPlot > 0 THEN
      GOTO 200
   END IF
   CLS 0
   WIDTH 80, 43
   VIEW (0, 0)-(639, 349), 0, 0
   VIEW PRINT
   WINDOW (0, 240)-(600, 0)
   COLOR 15, 4

' Plot Border.

   LINE (55, 20)-(55, 220)
   LINE (55, 20)-(575, 20)
   LINE (575, 20)-(575, 220)
   LINE (55, 220)-(575, 220)

' Plot Gridlines.

   FOR i = 30 TO 210 STEP 10
      LINE (55, i)-(575, i)
   NEXT i
   FOR i = 120 TO 510 STEP 65
      LINE (i, 20)-(i, 220)
   NEXT i
   LOCATE 2, 24
   PRINT "      F10: Exit Subprogram"

' X - Axis Label.

   LOCATE 16, 2
   PRINT "M"
   LOCATE 19, 2
   PRINT "o"
   LOCATE 22, 2
   PRINT "m"
   LOCATE 25, 2
   PRINT "e"
   LOCATE 28, 2
   PRINT "n"
   LOCATE 31, 2
   PRINT "t"
   LOCATE 43, 30
```

```
PRINT " R o t a t i o n "

LOCATE 3, 3
PRINT "32000"
LOCATE 7, 3
PRINT "28800"
LOCATE 10, 3
PRINT "25600"
LOCATE 14, 3
PRINT "22400"
LOCATE 18, 3
PRINT "19200"
LOCATE 21, 3
PRINT "16000"
LOCATE 25, 3
PRINT "12800"
LOCATE 29, 4
PRINT "9600"
LOCATE 32, 4
PRINT "6400"
LOCATE 36, 4
PRINT "3200"
LOCATE 40, 4
PRINT "0"

LOCATE 41, 8
PRINT "0"
LOCATE 41, 13
PRINT "0.005"
LOCATE 41, 23
PRINT "0.01"
LOCATE 41, 31
PRINT "0.015"
LOCATE 41, 41
PRINT "0.02"
LOCATE 41, 48
PRINT "0.025"
LOCATE 41, 58
PRINT "0.03"
LOCATE 41, 65
PRINT "0.035"
LOCATE 41, 76
PRINT ".04"

200 COLOR 1

  IF nPlot = 0 THEN
    FOR i = 1 TO nData
      IF i = 1 THEN
        LINE (55, 22)-(GTHETA(1) * 13000 + 55, M(1) / 220 + 20)
      END IF
      LINE (GTHETA(i - 1) * 13000 + 55, M(i - 1) / 220 + 20)-(GTHETA(i) * 13000 + 55, M(i) / 220 + 20)
    NEXT i
  ELSEIF nPlot > 0 THEN
```

```
     LINE (GTHETA(nData) * 13000 + 55, M(nData) / 220 + 20)-(GTHETA(nData - 1) * 13000 + 55,
M(nData - 1) / 220 + 20)
   END IF

   KEY(10) ON

   IF F10Flag = 1 THEN
205    LOCATE 2, 24
     PRINT "                    "
     LOCATE 2, 24
     COLOR 15
     INPUT "Exit Program (Y or N):", ans$
     ans$ = UCASE$(ans$)
     IF ans$ = "Y" THEN
       F10Flag = -1
       GOTO 210
     ELSEIF ans$ = "N" THEN
       LOCATE 2, 24
       PRINT "                    "
       F10Flag = -1
     ELSE
       GOTO 205
     END IF
   ELSEIF F10Flag = -1 THEN
     nPlot = nPlot + 1
     GOTO 206
   END IF

210 WINDOW

END SUB
```

```
SUB Load.Setup
'***************************************************************************
SHARED Ident$, NChannel, OutFile1$, OutFile2$, OutFile3$
'***************************************************************************
' This subroutine allows the user to load a file already containing the
' channel setup data into the program.
'
    CLS
    COLOR 15, 8
    LOCATE 4, 10
    PRINT " Load Channel Setup From File "
    LOCATE 6, 10
    INPUT "Enter name of file to retrieve Channel Information from: ", filein$
    LOCATE 7, 10
    PRINT "Information will be loaded from file: "; filein$
    LOCATE 8, 10
    INPUT "Modify Filename (Y or N)"; rsvp$
    IF UCASE$(rsvp$) = "Y" THEN
       LOCATE 9, 10
       INPUT "Enter Correct Drive, Directory and Filename: ", filein$
    END IF

    OPEN filein$ FOR INPUT AS #4
    INPUT #4, Ident$
    INPUT #4, NChannel
    FOR j = 1 TO NChannel
       INPUT #4, SensCon(j)
    NEXT j
    INPUT #4, OutFile1$
    INPUT #4, OutFile2$
    INPUT #4, OutFile3$
    CLOSE #4

END SUB
```

```
SUB Main.Display
'****************************************************************************
SHARED nPass, scale, nStep, Loop.Steps, Save.Data.Step, nData, Save.Data.Counter
SHARED FORCE.TE, FORCE.TC, FORCE.TW, FORCE.BNE, FORCE.BNW, FORCE.BSE, FORCE.BSW
SHARED FLINK.NE, FLINK.NW, FLINK.SE, FLINK.SW, Correction
SHARED P, P.N, P.S, MZ.N, MZ.S, MY.N, MY.S
SHARED THETA.Z.N, THETA.Z.S, THETA.Y.N, THETA.Y.S
'****************************************************************************
' This subroutine sets up the graphics of the main screen.
'
   KEY(10) OFF
   KEY(5) OFF
   KEY(3) OFF
   IF nPass > 0 THEN
      GOTO 430
   END IF
421  CLS 0

' Set up graphics viewport.

   WIDTH 80, 43
   VIEW (0, 0)-(639, 349), 15
   VIEW PRINT
   VIEW (260, 5)-(630, 170), 0, 15

' Draw main straight lines of hull.

   LINE (80, 20)-(290, 20), 2
   LINE (80, 20)-(40, 60), 2
   LINE (290, 20)-(330, 60), 2
   LINE (40, 60)-(330, 60), 2
   PAINT (100, 40), 2
   LINE (90, 120)-(280, 120), 2

' Draw curved ends of hull.

   LINE (40, 60)-(45, 80), 2
   LINE (45, 80)-(55, 95), 2
   LINE (55, 95)-(70, 110), 2
   LINE (70, 110)-(90, 120), 2
   LINE (330, 60)-(325, 80), 2
   LINE (325, 80)-(315, 95), 2
   LINE (315, 95)-(300, 110), 2
   LINE (300, 110)-(280, 120), 2
   PAINT (100, 100), 2
   LINE (40, 60)-(330, 60), 15

' Draw top of bulkheads.

   LINE (150, 20)-(137, 60), 15
   LINE (220, 20)-(234, 60), 15

' Designate area to left for text viewport and menu viewport.
```

```
   VIEW (10, 5)-(250, 345), 0, 15
   LINE (0, 255)-(240, 255), 15
'
430 CALL Graph
   COLOR 4, 0

' Print transducer values to screen.

   LOCATE 5, 52
   PRINT USING "+#### k"; FORCE.TE
   LOCATE 6, 52
   PRINT USING "+#### k"; FORCE.TC
   LOCATE 7, 52
   PRINT USING "+#### k"; FORCE.TW
   LOCATE 20, 40
   PRINT USING "+#### k"; FORCE.BNW
   LOCATE 18, 42
   PRINT USING "+#### k"; FORCE.BNE
   LOCATE 20, 66
   PRINT USING "+#### k"; FORCE.BSW
   LOCATE 18, 64
   PRINT USING "+#### k"; FORCE.BSE
'
   COLOR 15, 0
   LOCATE 12, 74
   PRINT "South"
   LOCATE 12, 34
   PRINT "North"

' Print values to data viewport.

   COLOR 15, 0
   LOCATE 3, 5
   PRINT USING " P.N: +#### K"; P.N
   LOCATE 4, 5
   PRINT USING " P.S: +#### K"; P.S
   LOCATE 5, 5
   PRINT USING " AXIAL FORCE: +#### K"; P
   LOCATE 7, 5
   PRINT USING " MZ.N:   +##### K-FT"; MZ.N
   LOCATE 8, 5
   PRINT USING " MZ.S:   +##### K-FT"; MZ.S
   LOCATE 9, 5
   PRINT USING " MY.N:   +##### K-FT"; MY.N
   LOCATE 10, 5
   PRINT USING " MY.S:   +##### K-FT"; MY.S
   COLOR 4, 0
   LOCATE 12, 13
   PRINT "Link Forces"
   LOCATE 13, 4
   PRINT USING "NE: +#### K"; FLINK.NE
   LOCATE 13, 19
   PRINT USING "SE: +#### K"; FLINK.SE
   LOCATE 14, 4
```

```
PRINT USING "NW: +#### K"; FLINK.NW
LOCATE 14, 19
PRINT USING "SW: +#### K"; FLINK.SW

COLOR 15, 0
LOCATE 16, 6
PRINT USING "THETA.Z.N: +#.### RAD"; THETA.Z.N
LOCATE 17, 6
PRINT USING "THETA.Z.S: +#.### RAD"; THETA.Z.S
LOCATE 18, 6
PRINT USING "THETA.Y.N: +#.### RAD"; THETA.Y.N
LOCATE 19, 6
PRINT USING "THETA.Y.S: +#.### RAD"; THETA.Y.S

COLOR 8, 0
LOCATE 21, 5
PRINT USING "nData: #####"; nData

LOCATE 22, 5
PRINT USING "PRESENT LOADING STEP: ##"; nStep
LOCATE 23, 5
PRINT USING "TOTAL LOOP STEPS:  ##"; Loop.Steps
LOCATE 24, 5
PRINT USING "SAVE DATA COUNTER: ##"; Save.Data.Counter
LOCATE 25, 5
PRINT USING "PRESENT SAVE DATA STEP: ##"; Save.Data.Step
COLOR 15, 0
LOCATE 27, 5
PRINT "CORRECT IMBALANCES"
LOCATE 27, 26
IF Correction = 1 THEN
  COLOR 2, 0
  PRINT "ON"
ELSEIF Correction = -1 THEN
  COLOR 4, 0
  PRINT "OFF"
END IF
'
KEY(10) ON
KEY(5) ON
KEY(3) ON
nPass = nPass + 1

439 END SUB
```

```
SUB Main.Menu
'*************************************************************************
SHARED nPass
'*************************************************************************
' This subroutine allows the user to view the channel data, run the program,
' initialize the actuators, view the raw voltages, or exit the program.

401 CLS
   WIDTH 80, 25
   COLOR 15, 8
   LOCATE 4, 21
   PRINT "   M A I N   M E N U   "
   LOCATE 6, 13
   PRINT " 1 - Load Channel Setup from File "
   PRINT TAB(13); " 2 - Check Currently Loaded Values "
   PRINT TAB(13); " 3 - Initialize Actuators   "
   PRINT TAB(13); " 4 - Run Program           "
   PRINT TAB(13); " 5 - Display Raw Voltages   "
   PRINT TAB(13); " 6 - Exit Program          "
   LOCATE 15, 13
   INPUT " Enter Selection: ", choice$
   SELECT CASE choice$
     CASE IS = "1"
       CALL Load.Setup
       CALL Initialize.Boards
       GOTO 401
     CASE IS = "2"
       CALL Constants
       GOTO 401
     CASE IS = "3"
       CALL Initialize.Acts
       GOTO 401
     CASE IS = "4"
       nPass = 0
       CALL Run.Program
       GOTO 401
     CASE IS = "5"
       LOCATE 18, 13
       CALL Zero
       GOTO 401
     CASE IS = "6"
       GOTO 402
     END SELECT

402 END SUB
```

```
SUB Move.Actuators
'*****************************************************************************
SHARED Command.TE%, Command.TC%, Command.TW%, Command.BSE%, Command.BSW%
SHARED Delta.TE%, Delta.TC%, Delta.TW%, Delta.BSE%, Delta.BSW%
SHARED Command.16%
'*****************************************************************************
' This subroutine takes the command signals previously determined in sub
' Compute.New.Commands and sends them to the D/A board.  Channels 1 through
' 5 are the actuator commands and channel 16 is the data acquisition command
' trigger.
' Here, the commands are written to the array that can be sent to the D/A board.
'
  ACommand(1) = Command.TE%
  ACommand(2) = Command.TC%
  ACommand(3) = Command.TW%
  ACommand(4) = Command.BSE%
  ACommand(5) = Command.BSW%
  ACommand(6) = 0
  ACommand(7) = 0
  ACommand(8) = 0
  ACommand(9) = 0
  ACommand(10) = 0
  ACommand(11) = 0
  ACommand(12) = 0
  ACommand(13) = 0
  ACommand(14) = 0
  ACommand(15) = 0
  ACommand(16) = Command.16%

' The following step is the communication with the D/A board and sends the
' commands.

  DERR = KSyncStart%(hDA)
  IF DERR <> 0 THEN
    BEEP
    PRINT "ERROR "; HEX$(DERR); " OCCURRED DURING 'KSyncStart'"
    STOP
  END IF

' Here, the Delta factors are reinitialized to zero.

  Delta.TE% = 0
  Delta.TC% = 0
  Delta.TW% = 0
  Delta.BSE% = 0
  Delta.BSW% = 0

END SUB
```

```
SUB Pause
'*************************************************************************
SHARED FORCE.TE, FORCE.TC, FORCE.TW, FORCE.BSE, FORCE.BSW, Tolerance
'*************************************************************************
' The purpose of this subroutine is to provide a pause in the program to
' allow for equilibrium to be reached after the actuators are moved.
' The A/D board is sampled and the forces in the actuators are found.  The
' A/D board is sampled again and the forces found.  These two consecutive
' values are then compared and if their difference is larger than a
' specified tolerance the loop is repeated.
' If this process is repeated 10 times and a tolerance could not be reached,
' a message prints to the screen and the subroutine is exited so that the
' program does not get stuck here.  The execution loop continues.
'
   nPause = 0
300  CALL Sample.AD
   CALL Calc.Engr.Values
   FORCE.1.TE = FORCE.TE
   FORCE.1.TC = FORCE.TC
   FORCE.1.TW = FORCE.TW
   FORCE.1.BSE = FORCE.BSE
   FORCE.1.BSW = FORCE.BSW

   CALL Sample.AD
   CALL Calc.Engr.Values
   FORCE.2.TE = FORCE.TE
   FORCE.2.TC = FORCE.TC
   FORCE.2.TW = FORCE.TW
   FORCE.2.BSW = FORCE.BSW
   FORCE.2.BSE = FORCE.BSE

   nPause = nPause + 1
'
   IF ABS(FORCE.1.TE - FORCE2.TE) > Tolerance THEN
     GOTO 305
   ELSEIF ABS(FORCE.1.TC - FORCE2.TC) > Tolerance THEN
     GOTO 305
   ELSEIF ABS(FORCE.1.TW - FORCE2.TW) > Tolerance THEN
     GOTO 305
   ELSEIF ABS(FORCE.1.BSE - FORCE2.BSE) > Tolerance THEN
     GOTO 305
   ELSEIF ABS(FORCE.1.BSW - FORCE2.BSW) > Tolerance THEN
     GOTO 305
   ELSE
     GOTO 304
   END IF

 305  IF nPause > 9 THEN
      COLOR 4
      LOCATE 32, 4
      PRINT "EQUILIBRUIM NOT REACHED"
      GOTO 304
   ELSEIF nPause < 10 THEN
      LOCATE 32, 4
```

```
      PRINT "                    "
      GOTO 300
    END IF

304 END SUB
```

```
SUB Raw.Voltages
'*****************************************************************************
SHARED nPass
'*****************************************************************************
' This subroutine displays the raw voltages of the A/D channels.
'
   KEY(10) OFF
   KEY(5) OFF
   IF nPass > 0 THEN
      GOTO 600
   END IF
   CLS
   WIDTH 80, 25
   COLOR 15, 1
   LOCATE 24, 22
   PRINT "      F10: Exit Subprogram"
   LOCATE 2, 17
   PRINT "       CHANNEL  RAW  VOLTAGES"
   LOCATE 4, 14: PRINT "Channel #    Voltage"
   LOCATE 4, 42: PRINT "Channel #    Voltage"
600  FOR j = 1 TO 16
      LOCATE 5 + j, 16
      PRINT j
      LOCATE 5 + j, 27
      PRINT USING "+##.###"; ChVolt(j)
   NEXT j
   FOR j = 17 TO 32
      LOCATE 5 + (j - 16), 44
      PRINT j
      LOCATE 5 + (j - 16), 55
      PRINT USING "+##.###"; ChVolt(j)
   NEXT j
   KEY(10) ON
   KEY(5) ON
   nPass = nPass + 1

END SUB
```

```
SUB Run.Program
'***************************************************************************
SHARED F9Flag, nStep, Direction, nData, Loop.Steps, F10Flag, Save.Data.Counter
SHARED F4Flag, Pause.Execution, F6Flag, F7Flag
'***************************************************************************
' This is the subroutine that runs the main program loop. There are two
' separate loops in this subroutine, the Pausing loop and the Executing loop.
' When this subroutine is first called, the Pausing loop begins. In this
' loop, data is sampled and all calculations are made, but the decision
' algorithm is not called. During the Pausing loop, two menus are displayed
' in the menu box, the Program Parameters menu and the Program Functions
' menu. Through the Program Functions menu, the user can choose to
' increase or decrease the rotation, which will cause the program to enter
' the Executing loop. In the Executing loop, data is sampled, calculations
' are made, and the subroutine Decision.Algorithm is called. The Executing
' loop executes a user defined number of times (Loop.Steps) or stops upon
' a function key(F9) being pressed.
'
' Here is the beginning of the Pausing loop. If the subroutine is called
' for the first time (nData = 0) or the screen changed and the main display
' needs to be redisplayed (nPass = 0), then Main.Display is called.
'
   IF nData = 0 OR nPass = 0 THEN
      CALL Main.Display
   END IF

810 KEY(3) ON
   KEY(5) ON
   KEY(7) ON
   KEY(10) ON
   KEY(6) ON
   KEY(9) ON
   KEY(4) ON

' Write the menu options for the Pausing loop in the menu box.

800 CALL Clear.Window
   COLOR 15, 0
   LOCATE 37, 5
   PRINT " F3  - PROGRAM PARAMETERS  "
   LOCATE 38, 5
   PRINT " F5  - PROGRAM FUNCTIONS   "
   LOCATE 39, 5
   PRINT " F7  - SAVE DATA           "
   LOCATE 40, 5
   PRINT " F10 - EXIT PROGRAM        "

' If the Executing loop was paused, then the variable Pause.Execution was
' given a value of 100. In this case, the additional menu option of F6
' needs to be displayed in the menu box. If the Executing loop stopped
' because it finished executing the number of Loop Steps prescribed, this
' option is not displayed (Pause.Execution = 0).

811 IF Pause.Execution = 100 THEN
```

```
    LOCATE 41, 5
    PRINT " F6  - RESUME EXECUTION   "
  ELSEIF Pause.Execution = 0 THEN
    LOCATE 41, 5
    PRINT "                         "
  END IF

' Call the subroutines that sample data, make calculations, and update
' the screen.

  CALL Sample.AD
  CALL Calc.Engr.Values
  CALL Actuator.Vectors
  CALL Equilibrium.Eqns
  CALL Main.Display

  LOCATE 31, 21
  PRINT "        "
  LOCATE 32, 4
  PRINT "                    "

' Call the subroutines that check to see if the user wants to access the
' Program Parameters or Program Functions menu.  If the user chooses from
' the Program Functions menu to start loading or unloading (increase or
' decrease the rotation), the parameter Direction will be given a value of
' 1 or -1.  This will cause the program to enter the Executing loop below.

  CALL F3Flag.Check
  CALL F5Flag.Check

' Check to see if the F7 key has been pressed to save data.  If so, Save.Data
' is called, the parameter Command.Trigger is given the voltage value that
' triggers the auxiliary data acquisition system, and Move.Actuators is
' called to send this voltage (the actuators will not move because their
' command has not changed from the previous step.

  IF F7Flag = 1 THEN
     CALL Save.Data
     Command.Trigger = 1229
     CALL Move.Actuators
     COLOR 15
     LOCATE 29, 5
     PRINT "DATA IS BEING SAVED"
     F7Flag = -1
  ELSEIF F7Flag = -1 THEN
     Command.Trigger = 3072
     CALL Move.Actuators
     LOCATE 29, 5
     PRINT "                   "
  END IF

' Here, the F10Flag is checked to see if the user wants to exit the program.

  IF F10Flag = 1 THEN
```

```
890    CALL Clear.Window
       COLOR 15, 0
       LOCATE 38, 5
       INPUT "EXIT PROGRAM? (Y or N): ", ans$
       ans$ = UCASE$(ans$)
       IF ans$ = "Y" THEN
          F10Flag = -1
          GOTO 899
       ELSEIF ans$ = "N" THEN
          F10Flag = -1
       ELSE
          GOTO 890
       END IF
    END IF

    nData = nData + 1

' Check the F6 key to see if the user wants to resume execution (if the
' Executing loop was paused). If execution should resume, the parameter
' Pause.Execution is given a value of 0.

    IF F6Flag = 1 THEN
807    CALL Clear.Window
       COLOR 15, 0
       LOCATE 38, 3
       INPUT "RESUME EXECUTION?(Y or N): ", ans$
       ans$ = UCASE$(ans$)
       IF ans$ = "Y" THEN
          Pause.Execution = 0
          F6Flag = -1
          GOTO 801
       ELSEIF ans$ = "N" THEN
          GOTO 808
       ELSE
          GOTO 807
808    END IF
    END IF

' Now, check to see if we need to continue pausing or begin executing.  If
' Pause.Execution = 0 and Direction = -1 or 1, this means that the previous
' loop was completed and we want to start a new one.  The program goes to
' line 801, the beginning of the Executing loop.  If Pause.Execution = 0
' but Direction also = 0, the previous loop was completed but we don't want
' to begin another one yet.  The program goes to line 800, the beginning of
' the Pausing loop.  If Pause.Execution = 100, the previous loop was paused
' using the F4 key, and the program goes back to the beginning of the
' Pausing loop.

    IF Pause.Execution = 0 THEN
       IF Direction > 0 OR Direction < 0 THEN
          GOTO 801
       ELSEIF Direction > -.5 AND Direction < .5 THEN
          COLOR 4, 0
          LOCATE 31, 4
```

```
        PRINT " PAUSING "
        GOTO 800
      END IF
    ELSEIF Pause.Execution = 100 THEN
      COLOR 4, 0
      LOCATE 31, 4
      PRINT " PAUSING "
      GOTO 800
    END IF

' Here is the end of the Pausing loop and the beginning of the Executing
' loop.
' Print the menu options for the Executing loop in the menu box.

801 CALL Clear.Window
    COLOR 15
    LOCATE 37, 5
    PRINT " F4  - PAUSE EXECUTION   "
    LOCATE 38, 5
    PRINT " F7  - SAVE DATA         "
    LOCATE 39, 5
    PRINT " F9  - EXIT LOOP         "

' Call the subroutines that sample data, make calculations, save data,
' make decisions, and move actuators.

805 CALL Sample.AD
    CALL Calc.Engr.Values
    CALL Actuator.Vectors
    CALL Equilibrium.Eqns
    CALL Main.Display
    CALL Save.Data.Check

    COLOR 2, 0
    LOCATE 31, 21
    PRINT " EXECUTING"

    CALL Decision.Algorithm
    CALL Compute.New.Commands
    CALL Check.New.Commands
    CALL Move.Actuators
    CALL Pause

' Check the F9 key to see if the user wants to stop executing the loop.  If
' this is done, the nStep and Save.Data.Counter values are reset, which
' means that the loop cannot be reentered at its previous state.

    IF F9Flag = 1 THEN
802   CALL Clear.Window
      LOCATE 38, 7
      COLOR 15
      INPUT "EXIT LOOP? (Y or N): ", ans$
      ans$ = UCASE$(ans$)
      IF ans$ = "Y" THEN
```

```
        nStep = 1
        Save.Data.Counter = 1
        F9Flag = -1
        Direction = 0
        LOCATE 31, 21
        PRINT "        "
        GOTO 800
     ELSEIF ans$ = "N" THEN
        F9Flag = -1
     ELSE
        GOTO 802
     END IF
   END IF

   LOCATE 31, 21
   PRINT "        "

' Upate the values of nData, nStep, and Save.Data.Counter.

   nData = nData + 1
   nStep = nStep + 1
   Save.Data.Counter = Save.Data.Counter + 1

' Check to see if the Executing loop is finished (the prescribed number of
' loop steps have been executed).  If so, the program resets nStep and
' Save.Data.Counter and goes to the beginning of the Pausing loop.  If not,
' the Executing loop continues on.

   IF nStep = Loop.Steps + 1 THEN
     Direction = 0
     nStep = 1
     Save.Data.Counter = 1
     GOTO 810
   ELSEIF nStep < Loop.Steps + 1 THEN
     GOTO 804
   END IF

' Check to see if F4 has been pressed to pause execution.  If so, the
' parameter Pause.Execution is given a value of 100 and the program goes to
' the beginning of the Pausing loop.  If not, it returns to the beginning
' of the Executing loop.

804 IF F4Flag = 1 THEN
803   CALL Clear.Window
     LOCATE 38, 4
     COLOR 15
     INPUT "PAUSE EXECUTION?(Y or N): ", ans$
     ans$ = UCASE$(ans$)
     IF ans$ = "Y" THEN
        Pause.Execution = 100
        F4Flag = -1
        GOTO 810
     ELSEIF ans$ = "N" THEN
        F4Flag = -1
```

```
        ELSE
            GOTO 803
        END IF
    ELSEIF F4Flag = -1 THEN
        GOTO 801
    END IF

899 END SUB
```

```
SUB Sample.AD
'*****************************************************************************
SHARED NChannel
'*****************************************************************************
' This subroutine samples each channel 20 times and then averages the 20
' readings. The averaged values are then converted to voltages. Each block
' of code is separately commented. This subroutine should NOT be changed.
'
' The first step tells the A/D board to start sampling the data
'
  FOR K = 1 TO 20
    nDasErr = KIntStart%(hAD)
    IF nDasErr <> 0 THEN
     BEEP
     PRINT "ERROR "; HEX$(nDasErr); " OCCURRED DURING 'KIntStart'": STOP
    END IF

' This block monitors the status of the sampling rate and sample transfer
' count until done.

100   nDasErr = KIntStatus%(hAD, wStatus, dwCount)
    IF nDasErr <> 0 THEN
     BEEP
     PRINT "ERROR "; HEX$(nDasErr); " OCCURRED DURING 'KIntStatus'": GOTO 105
    END IF
    IF ((wStatus AND 1) = 1) THEN GOTO 100

' Stop Interrupt operation in case user interrupted or an error occurred.

105   nDasErr = KIntStop%(hAD, wStatus, dwCount)
    IF nDasErr <> 0 THEN
     BEEP
     PRINT "ERROR "; HEX$(nDasErr); " OCCURRED DURING 'KIntStop'": STOP
    END IF

' The sample for each channel at step 'i' throught the loop (DataBuf) is
' added to all the samples for that channel from all previous steps.

    FOR i = 1 TO NChannel
     ChData(K, i) = DataBuf(i - 1) + ChData(K - 1, i)
    NEXT i
  NEXT K

' The channel data is averaged and converted to voltages. First, dividing
' by 20 finds the average count value over the 20 samplings. Then,
' multiplying by the span of 20 volts and dividing by 4096 (the resolution
' for 12-bit boards) determines the voltage.

  FOR j = 1 TO NChannel
    ChVolt(j) = ChData(20, j) / 20! * 20 / 4096!
  NEXT j


END SUB
```

```
SUB Save.Data
'***************************************************************************
SHARED FORCE.TE, FORCE.TC, FORCE.TW, FORCE.BNE, FORCE.BNW, FORCE.BSE, FORCE.BSW
SHARED FORCE.XN.TE, FORCE.YN.TE, FORCE.ZN.TE, FORCE.XS.TE, FORCE.YS.TE, FORCE.ZS.TE
SHARED FORCE.XN.TC, FORCE.YN.TC, FORCE.ZN.TC, FORCE.XS.TC, FORCE.YS.TC,
FORCE.ZS.TC
SHARED FORCE.XN.TW, FORCE.YN.TW, FORCE.ZN.TW, FORCE.XS.TW, FORCE.YS.TW,
FORCE.ZS.TW
SHARED FORCE.X.BNE, FORCE.Y.BNE, FORCE.Z.BNE
SHARED FORCE.X.BNW, FORCE.Y.BNW, FORCE.Z.BNW, FORCE.X.BSE, FORCE.Y.BSE
SHARED FORCE.Z.BSE, FORCE.X.BSW, FORCE.Y.BSW, FORCE.Z.BSW
SHARED FORCE.NE, FORCE.NW, FORCE.SE, FORCE.SW
SHARED OutFile1$, OutFile2$, OutFile3$, nData, nSave
SHARED P.N, P.S, MZ.N, MZ.S, MY.N, MY.S, P
SHARED THETA.Z.N, THETA.Z.S, THETA.Y.N, THETA.Y.S
'***************************************************************************
' This subroutine saves data to three different files, the names of which
' are read as input from the data file loaded at the begining of the test.
' The first file saves the actuator and link forces. The second file saves
' grillage transducer values and the bottom link transducer values.  The
' third file saves the calculated axial forces and moments.
'
' The first time data is saved, the following if-then writes column headings
' for the data for each file.
'
  IF nSave = 0 THEN
    OPEN OutFile1$ FOR OUTPUT AS #2
    PRINT #2, "      ", "FORCE.TE", "FORCE.TC", "FORCE.TW", "FORCE.BNE", "FORCE.BNW",
"FORCE.BSE", "FORCE.BSW",
    PRINT #2, "FORCE.NE", "FORCE.NW", "FORCE.SE", "FORCE.SW",
    PRINT #2, "FORCE.XN.TE", "FORCE.YN.TE", "FORCE.ZN.TE",
    PRINT #2, "FORCE.XS.TE", "FORCE.YS.TE", "FORCE.ZS.TE",
    PRINT #2, "FORCE.XN.TC", "FORCE.YN.TC", "FORCE.ZN.TC",
    PRINT #2, "FORCE.XS.TC", "FORCE.YS.TC", "FORCE.ZS.TC",
    PRINT #2, "FORCE.XN.TW", "FORCE.YN.TW", "FORCE.ZN.TW",
    PRINT #2, "FORCE.XS.TW", "FORCE.YS.TW", "FORCE.ZS.TW",
    PRINT #2, "FORCE.X.BNE", "FORCE.Y.BNE", "FORCE.Z.BNE",
    PRINT #2, "FORCE.X.BNW", "FORCE.Y.BNW", "FORCE.Z.BNW",
    PRINT #2, "FORCE.X.BSE", "FORCE.Y.BSE", "FORCE.Z.BSE",
    PRINT #2, "FORCE.X.BSW", "FORCE.Y.BSW", "FORCE.Z.BSW"
    CLOSE #2
    OPEN OutFile2$ FOR OUTPUT AS #3
    PRINT #3, "      ", "XDUCER.1NX", "XDUCER.2NX", "XDUCER.3NX", "XDUCER.1NY",
"XDUCER.3NY",
    PRINT #3, "XDUCER.1SX", "XDUCER.2SX", "XDUCER.3SX", "XDUCER.1NY", "XDUCER.3NY",
    PRINT #3, "DELTA.X.BSW", "DELTA.X.BSE"
    CLOSE #3
    OPEN OutFile3$ FOR OUTPUT AS #5
    PRINT #5, "      ", "P.N", "P.S", "MZ.N", "MZ.S", "MY.N",
    PRINT #5, "MY.S", "P", "THETA.Z.N", "THETA.Z.S", "THETA.Y.N", "THETA.Y.S"
    CLOSE #5
    nSave = 1
  END IF
```

```
' Write data to the first file.

    OPEN OutFile1$ FOR APPEND AS #2
    PRINT #2, USING "nData: #####"; nData

    PRINT #2, "        ", FORCE.TE, FORCE.TC, FORCE.TW, FORCE.BNE, FORCE.BNW, FORCE.BSE,
FORCE.BSW,
    PRINT #2, FORCE.NE, FORCE.NW, FORCE.SE, FORCE.SW,
    PRINT #2, FORCE.XN.TE, FORCE.YN.TE, FORCE.ZN.TE,
    PRINT #2, FORCE.XS.TE, FORCE.YS.TE, FORCE.ZS.TE,
    PRINT #2, FORCE.XN.TC, FORCE.YN.TC, FORCE.ZN.TC,
    PRINT #2, FORCE.XS.TC, FORCE.YS.TC, FORCE.ZS.TC,
    PRINT #2, FORCE.XN.TW, FORCE.YN.TW, FORCE.ZN.TW,
    PRINT #2, FORCE.XS.TW, FORCE.YS.TW, FORCE.ZS.TW,
    PRINT #2, FORCE.X.BNE, FORCE.Y.BNE, FORCE.Z.BNE,
    PRINT #2, FORCE.X.BNW, FORCE.Y.BNW, FORCE.Z.BNW,
    PRINT #2, FORCE.X.BSE, FORCE.Y.BSE, FORCE.Z.BSE,
    PRINT #2, FORCE.X.BSW, FORCE.Y.BSW, FORCE.Z.BSW
    CLOSE #2

' Write data to the second file.

    OPEN OutFile2$ FOR APPEND AS #3
    PRINT #3, USING "nData: #####"; nData

    PRINT #3, "        ", XDUCER.1NX, XDUCER.2NX, XDUCER.3NX, XDUCER.1NY, XDUCER.3NY,
    PRINT #3, XDUCER.1SX, XDUCER.2SX, XDUCER.3SX, XDUCER.1NY, XDUCER.3NY,
    PRINT #3, DELTA.X.BSW, DELTA.X.BSE
    CLOSE #3

' Write data to the third file

    OPEN OutFile3$ FOR APPEND AS #5
    PRINT #5, USING "nData: #####"; nData

    PRINT #5, "        ", P.N, P.S, MZ.N, MZ.S, MY.N,
    PRINT #5, MY.S, P, THETA.Z.N, THETA.Z.S, THETA.Y.N, THETA.Y.S
    CLOSE #5

END SUB
```

```
SUB Save.Data.Check
'***************************************************************************
SHARED nData, Save.Data.Step, F7Flag, Loop.Steps
SHARED Command.Trigger AS INTEGER, Save.Data.Counter
'***************************************************************************
' This subroutine is called from Run.Program after all calculations are done
' and the main display is updated. Its purpose is to check whether data
' should be saved for either of two reasons. First, data is automatically
' saved at an interval called Save.Data.Step. If nData (the # of times
' that the loop has been executed) is a multiple of Save.Data.Step, then
' data is saved automatically. If data is not to be saved for this reason,
' then the F7 flag is checked to see if the user wants data to be saved
' anyway. Whenever data is to be saved, the parameter Command.Trigger is
' set to the correct voltage for triggering the auxiliary data acquisition
' system.
'
' First check to see if the Save.Data.Counter is a multiple of Save.Data.Step.
'
    FOR i = 1 TO Loop.Steps
    Multiple = i * Save.Data.Step
       IF Save.Data.Counter = Multiple THEN
         Save.Data.Flag = 1
         GOTO 355
       ELSE
         Save.Data.Flag = -1
         GOTO 354
       END IF
354  NEXT i

' If data is to be saved, give Command.Trigger the correct voltage and
' call Move.Actuators and Save.Data. If not, give Command.Trigger the
' value that will not cause the auxiliary system to be triggered.

355  IF Save.Data.Flag = 1 THEN
       CALL Save.Data
       COLOR 15, 0
       LOCATE 29, 5
       PRINT "DATA IS BEING SAVED"
       Command.Trigger = 1229
       CALL Move.Actuators
       Save.Data.Flag = -1
       GOTO 360
    ELSEIF Save.Data.Flag = -1 THEN
       Command.Trigger = 3072
       LOCATE 29, 5
       PRINT "                    "
       GOTO 350
    END IF

' Check the F7 key to see if the user wants to save data.

350  IF F7Flag = 1 THEN
'      CALL Save.Data
       Save.Data.Counter = 0
```

```
        Command.Trigger = 1229
        CALL Move.Actuators
        COLOR 15, 0
        LOCATE 29, 5
        PRINT "DATA IS BEING SAVED"
        F7Flag = -1
    ELSE
        Command.Trigger = 3072
        LOCATE 29, 5
        PRINT "                    "
    END IF

360 END SUB
```

```
SUB Volt.Command.Adj
'*************************************************************************
SHARED DACStep.TE%, DACStep.TC%, DACStep.TW%, DACStep.BSE%, DACStep.BSW%
SHARED Delta.TE%, Delta.TC%, Delta.TW%, Delta.BSE%, Delta.BSW%
SHARED CommandError
'*************************************************************************
' This subroutine allows the command signals to be modified once the
' initial voltages are entered in sub Initialize.Acts or to perform
' manual command adjustment during the test. The DAC step sizes can be
' changed by entering a command that will call sub DAC.Step.Size.
'

470 CLS
  WIDTH 80, 25
  COLOR 15, 4
  VIEW (0, 0)-(639, 349)
  VIEW PRINT

' Square border around screen.

  LINE (7, 5)-(630, 5), 15
  LINE (7, 5)-(7, 340), 15
  LINE (630, 5)-(630, 340), 15
  LINE (7, 340)-(630, 340), 15

' Horizontal lines.

  LINE (7, 50)-(630, 50), 15
  LINE (7, 100)-(630, 100)
  LINE (7, 150)-(630, 150)
  LINE (7, 200)-(630, 200)
  LINE (7, 250)-(630, 250)
  LINE (7, 300)-(630, 300)

' Vertical lines.

  LINE (102, 5)-(102, 300)
  LINE (179, 5)-(179, 300)
  LINE (260, 5)-(260, 300)
  LINE (338, 5)-(338, 300)
  LINE (475, 5)-(475, 300)

' Write text to screen.

  LOCATE 2, 4
  PRINT "Actuator"
  LOCATE 2, 15
  PRINT "DAC Step"
  LOCATE 3, 16
  PRINT "Size "
  LOCATE 2, 24
  PRINT "Disp Step"
  LOCATE 3, 26
  PRINT "Size"
  LOCATE 2, 34
```

```
   PRINT "Volt Step"
   LOCATE 3, 36
   PRINT "Size "
   LOCATE 2, 44
   PRINT "Increase Voltage"
   LOCATE 3, 45
   PRINT "(Extend Act.) "
   LOCATE 2, 62
   PRINT "Decrease Voltage"
   LOCATE 3, 62
   PRINT "(Retract Act.)"
'
   LOCATE 1, 24
   PRINT " ACTUATOR COMMAND ADJUSTMENT "
   LOCATE 6, 5
   PRINT "1 - TE"
   LOCATE 10, 5
   PRINT "2 - TC"
   LOCATE 13, 5
   PRINT "3 - TW"
   LOCATE 17, 5
   PRINT "4 - BSE"
   LOCATE 20, 5
   PRINT "5 - BSW"

' Print codes for changing commands.

   LOCATE 6, 51
   PRINT "1"
   LOCATE 6, 69
   PRINT "-1"
   LOCATE 10, 51
   PRINT "2"
   LOCATE 10, 69
   PRINT "-2"
   LOCATE 13, 51
   PRINT "3"
   LOCATE 13, 69
   PRINT "-3"
   LOCATE 17, 51
   PRINT "4"
   LOCATE 17, 69
   PRINT "-4"
   LOCATE 20, 51
   PRINT "5"
   LOCATE 20, 69
   PRINT "-5"

' Show the present DAC step size.

471 LOCATE 6, 17
   PRINT DACStep.TE%
   LOCATE 10, 17
   PRINT DACStep.TC%
```

```
   LOCATE 13, 17
   PRINT DACStep.TW%
   LOCATE 17, 17
   PRINT DACStep.BSE%
   LOCATE 20, 17
   PRINT DACStep.BSW%

' Show the corresponding voltage step size.

   LOCATE 6, 36
   PRINT USING "#.###"; DACStep.TE% * .004883
   LOCATE 10, 36
   PRINT USING "#.###"; DACStep.TC% * .004883
   LOCATE 13, 36
   PRINT USING "#.###"; DACStep.TW% * .004883
   LOCATE 17, 36
   PRINT USING "#.###"; DACStep.BSE% * .004883
   LOCATE 20, 36
   PRINT USING "#.###"; DACStep.BSW% * .004883

' Show the corresponding displacement step size.

   LOCATE 6, 26
   PRINT USING "#.###"; DACStep.TE% * .0088
   LOCATE 10, 26
   PRINT USING "#.###"; DACStep.TC% * .0088
   LOCATE 13, 26
   PRINT USING "#.###"; DACStep.TW% * .0088
   LOCATE 17, 26
   PRINT USING "#.###"; DACStep.BSE% * .005859
   LOCATE 20, 26
   PRINT USING "#.###"; DACStep.BSW% * .005859

' Print text that prompts user input.

472 LOCATE 23, 5
   PRINT "                                        "
   LINE (7, 5)-(7, 340), 15
   LOCATE 23, 56
   PRINT "10  - Exit"
   LOCATE 24, 56
   PRINT "11 - Change Step Sizes"
   LOCATE 23, 5
   PRINT "Enter the number corresponding  "
   LOCATE 24, 5
   INPUT "to the desired action : ", mvmt
   LOCATE 23, 5
   PRINT "                                        "
   LOCATE 24, 5
   PRINT "                                        "
   LOCATE 23, 5

' Use user response from the prompt to determine the Delta factor for
' whichever actuator is to be moved.
```

```
IF mvmt < 1.5 AND mvmt > .5 THEN
  INPUT "Increase Voltage of Actuator TE? (Y or N): ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
    Delta.TE% = 1
  ELSEIF UCASE$(rsvp$) = "N" THEN
    GOTO 472
  END IF
ELSEIF mvmt < -.5 AND mvmt > -1.5 THEN
  INPUT "Decrease Voltage of Actuator TE? (Y or N): ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
    Delta.TE% = -1
  ELSEIF UCASE$(rsvp$) = "N" THEN
    GOTO 472
  END IF
'
ELSEIF mvmt < 2.5 AND mvmt > 1.5 THEN
  INPUT "Increase Voltage of Actuator TC? (Y or N): ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
    Delta.TC% = 1
  ELSEIF UCASE$(rsvp$) = "N" THEN
    GOTO 472
  END IF
ELSEIF mvmt < -1.5 AND mvmt > -2.5 THEN
  INPUT "Decrease Voltage of Actuator TC? (Y or N): ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
    Delta.TC% = -1
  ELSEIF UCASE$(rsvp$) = "N" THEN
    GOTO 472
  END IF
'
ELSEIF mvmt < 3.5 AND mvmt > 2.5 THEN
  INPUT "Increase Voltage of Actuator TW? (Y or N): ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
    Delta.TW% = 1
  ELSEIF UCASE$(rsvp$) = "N" THEN
    GOTO 472
  END IF
ELSEIF mvmt < -2.5 AND mvmt > -3.5 THEN
  INPUT "Decrease Voltage of Actuator TW? (Y or N): ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
    Delta.TW% = -1
  ELSEIF UCASE$(rsvp$) = "N" THEN
    GOTO 472
  END IF
'
ELSEIF mvmt < 4.5 AND mvmt > 3.5 THEN
  INPUT "Increase Voltage of Actuator BSE? (Y or N): ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
    Delta.BSE% = 1
  ELSEIF UCASE$(rsvp$) = "N" THEN
    GOTO 472
  END IF
ELSEIF mvmt < -3.5 AND mvmt > -4.5 THEN
```

```
    INPUT "Decrease Voltage of Actuator BSE? (Y or N): ", rsvp$
    IF UCASE$(rsvp$) = "Y" THEN
      Delta.BSE% = -1
    ELSEIF UCASE$(rsvp$) = "N" THEN
      GOTO 472
    END IF
'
  ELSEIF mvmt < 5.5 AND mvmt > 4.5 THEN
    INPUT "Increase Voltage of Actuator BSW? (Y or N): ", rsvp$
    IF UCASE$(rsvp$) = "Y" THEN
      Delta.BSW% = 1
    ELSEIF UCASE$(rsvp$) = "N" THEN
      GOTO 472
    END IF
  ELSEIF mvmt < -4.5 AND mvmt > -5.5 THEN
    INPUT "Decrease Voltage of Actuator BSW? (Y or N): ", rsvp$
    IF UCASE$(rsvp$) = "Y" THEN
      Delta.BSW% = -1
    ELSEIF UCASE$(rsvp$) = "N" THEN
      GOTO 472
    END IF
'
  ELSEIF mvmt < 10.5 AND mvmt > 9.5 THEN
    INPUT "Exit Subprogram? (Y or N): ", rsvp$
    IF UCASE$(rsvp$) = "Y" THEN
      GOTO 473
    ELSEIF UCASE$(rsvp$) = "N" THEN
      LOCATE 23, 5
      PRINT "                                        "
      PRINT "                                        "
      GOTO 472
    END IF
'
  ELSEIF mvmt < 11.5 AND mvmt > 10.5 THEN
    INPUT "Change DACStep Sizes? (Y or N): ", rsvp$
    IF UCASE$(rsvp$) = "Y" THEN
      CALL DAC.Step.Size
      GOTO 470
    ELSEIF UCASE$(rsvp$) = "N" THEN
      LOCATE 23, 5
      PRINT "                                        "
      PRINT "                                        "
      GOTO 472
    END IF
  ELSE
    GOTO 472
  END IF

' Call the subroutines that take the determined Delta factor and compute
' and check the new commands.

  CALL Compute.New.Commands
  CALL Check.New.Commands
```

```
' If Check.New.Commands found that a command was beyond the limits of the
' allowable voltage range, print a warning to the screen.

   IF CommandError = 1 THEN
      LOCATE 23, 5
      PRINT "                                   "
      LOCATE 23, 2
      PRINT "Invalid Command: Beyond Limits of Voltage Range."
      LOCATE 24, 2
      INPUT "Press any key to continue: ", rsvp2$
      LOCATE 23, 2
      PRINT "                                   "
      LOCATE 24, 2
      PRINT "                                   "
      CommandError = 0
      GOTO 472
   END IF

' Move the actuators.

   CALL Move.Actuators
   GOTO 472

473 END SUB
```

```
SUB Zero
'****************************************************************************
SHARED nPass, F10Flag
'****************************************************************************
' This subroutine is used for the display of raw voltages.  It calls sub
' Sample.AD and sub Raw.Voltages, which contains the graphics code.
'
  nPass = 0
417 IF F10Flag = -1 THEN
     CALL Sample.AD
     CALL Raw.Voltages
  ELSEIF F10Flag = 1 THEN
     LOCATE 24, 1
418    INPUT "Exit Subprogram (Y or N):", ans2$
     ans2$ = UCASE$(ans2$)
     IF ans2$ = "Y" THEN
        F10Flag = -1
        GOTO 420
     ELSEIF ans2$ = "N" THEN
        LOCATE 24, 1
        PRINT "                              "
        F10Flag = -1
     ELSE
        GOTO 418
     END IF
  END IF
  GOTO 417

420 END SUB
```

```
DECLARE SUB Clear.Window ()
DECLARE SUB Change.Loop.Steps ()
DECLARE SUB Tolerance.Size ()
DECLARE SUB Choose.Graph.Scale ()
DECLARE SUB Save.Data.Step.Size ()
DECLARE SUB Correct.Imbalances ()
DECLARE SUB Check.New.Commands ()
DECLARE SUB Compute.New.Commands ()
'*************************************************************************
' This is the second module that is part of the control program.  It contains
' the subroutines declared above and uses the variables in the common
' block /navy2/ declared below.  This module is part of the library
' LIBRARY.QLB, which is comprised of this module and the library COMBO.
'*************************************************************************
COMMON /navy2/ Loop.Steps, Tolerance, scale, Save.Data.Step, Correction
COMMON /navy2/ CommandError, Command.TE%, Command.TC%
COMMON /navy2/ Command.TW%, Command.BSE%, Command.BSW%
COMMON /navy2/ Command.16%, Command.Trigger AS INTEGER
COMMON /navy2/ Delta.TE%, Delta.TC%, Delta.TW%, Delta.BSE%, Delta.BSW%
COMMON /navy2/ DACStep.TE%, DACStep.TC%, DACStep.TW%, DACStep.BSE%, DACStep.BSW%
```

```
SUB Change.Loop.Steps
'***************************************************************************
SHARED Loop.Steps
'***************************************************************************
' This subroutine is called when the user chooses to change the number of
' loop steps through the Program Parameters menu.
'
   COLOR 15
170 LOCATE 37, 3
   PRINT USING "PRESENT NO. OF LOOP STEPS: ##"; Loop.Steps
   LOCATE 39, 3
   INPUT "NEW NO. OF LOOP STEPS: ", Loop.Steps2
   CALL Clear.Window
   LOCATE 37, 3
   PRINT USING "NEW NO. OF LOOP STEPS IS ##"; Loop.Steps2
   LOCATE 39, 3
   INPUT "IS THIS CORRECT (Y OR N): ", rsvp$
   IF UCASE$(rsvp$) = "Y" THEN
      Loop.Steps = Loop.Steps2
      GOTO 175
   ELSE
      CALL Clear.Window
      GOTO 170
   END IF

175 END SUB
```

```
SUB Check.New.Commands
'*****************************************************************************
SHARED CommandError, Command.TE%, Command.TC%
SHARED Command.TW%, Command.BSE%, Command.BSW%
'*****************************************************************************
' This subroutine checks the new commands to make sure they are not beyond the
' allowable voltage range of 9.95 to -9.95 volts. If the command is invalid, a
' flag called CommandError is given a value of 1 which is used in a separate
' subroutine to print an error warning to the screen. Also, if the command
' is invalid, it is recalculated as its previous value, which was in the
' correct range.
'
  CommandError = 0

  IF Command.TE% > 4086 OR Command.TE% < 10 THEN
    CommandError = 1
    Command.TE% = Command.TE% - Delta.TE% * DACStep.TE%
  END IF

  IF Command.TC% > 4086 OR Command.TC% < 10 THEN
    CommandError = 1
    Command.TC% = Command.TC% - Delta.TC% * DACStep.TC%
  END IF

  IF Command.TW% > 4086 OR Command.TW% < 10 THEN
    CommandError = 1
    Command.TW% = Command.TW% - Delta.TW% * DACStep.TW%
  END IF

  IF Command.BSE% > 4086 OR Command.BSE% < 10 THEN
    CommandError = 1
    Command.BSE% = Command.BSE% - Delta.BSE% * DACStep.BSE%
  END IF

  IF Command.BSW% > 4086 OR Command.BSW% < 10 THEN
    CommandError = 1
    Command.BSW% = Command.BSW% - Delta.BSW% * DACStep.BSW%
  END IF

END SUB
```

```
SUB Choose.Graph.Scale
'***************************************************************************
SHARED scale
'***************************************************************************
' This subroutine allows the user to choose which scale is to be used for the
' moment-rotation graph on the main display screen. It is called from the
' Program Parameters menu.
'
   COLOR 15
   LOCATE 35, 3
   PRINT "    CHOOSE GRAPH SCALE  "
   LOCATE 36, 3
   PRINT " Moment Range  Rotation Range"
   LOCATE 37, 3
   PRINT "1: 0 - 8,000     0 - 0.01 "
   LOCATE 38, 3
   PRINT "2: 0 - 16,000    0 - 0.02 "
   LOCATE 39, 3
   PRINT "3: 0 - 24,000    0 - 0.03 "
   LOCATE 40, 3
   PRINT "4: 0 - 32,000    0 - 0.04 "
704 LOCATE 41, 3
   INPUT "    DESIRED SCALE: ", scale
   LOCATE 42, 3
   PRINT USING "YOU WANT SCALE # ?"; scale
   LOCATE 42, 21
   INPUT "(Y or N):", rsvp$
   IF UCASE$(rsvp$) = "Y" THEN
      GOTO 705
   ELSEIF UCASE$(rsvp$) = "N" THEN
      LOCATE 42, 3
      PRINT "                 "
      GOTO 704
   ELSE
      LOCATE 42, 3
      PRINT "                 "
      GOTO 704
   END IF

705 END SUB
```

```
SUB Clear.Window
'******************************************************************************
' This subroutine clears the menu viewport when a new menu is to be printed.
'******************************************************************************
   LOCATE 34, 3
   PRINT "                    "
   LOCATE 35, 3
   PRINT "                    "
   LOCATE 36, 3
   PRINT "                    "
   LOCATE 37, 3
   PRINT "                    "
   LOCATE 38, 3
   PRINT "                    "
   LOCATE 39, 3
   PRINT "                    "
   LOCATE 40, 3
   PRINT "                    "
   LOCATE 41, 3
   PRINT "                    "
   LOCATE 42, 3
   PRINT "                    "
END SUB
```

```
SUB Compute.New.Commands
'***************************************************************************
SHARED Command.TE%, Command.TC%, Command.TW%
SHARED Command.BSE%, Command.BSW%, Command.16%, Command.Trigger AS INTEGER
SHARED Delta.TE%, Delta.TC%, Delta.TW%, Delta.BSE%, Delta.BSW%
SHARED DACStep.TE%, DACStep.TC%, DACStep.TW%, DACStep.BSE%, DACStep.BSW%
'***************************************************************************
' This subroutine calculates the new commands by adding the old command to the
' product of the current DAC step and the delta factor (1,-1, or 0).
' If delta is one, the command is being increased; if delta is -1 the command
' is being decreased; if delta is zero the command does not change.
' Also, the data acquisition voltage, or Command.16%, is set equal to the
' Command.Trigger value and is sent to the auxiliary data acquisition system.
'
   Command.TE% = Command.TE% + DACStep.TE% * Delta.TE%
   Command.TC% = Command.TC% + DACStep.TC% * Delta.TC%
   Command.TW% = Command.TW% + DACStep.TW% * Delta.TW%
   Command.BSE% = Command.BSE% + DACStep.BSE% * Delta.BSE%
   Command.BSW% = Command.BSW% + DACStep.BSW% * Delta.BSW%
   Command.16% = Command.Trigger

END SUB
```

```
SUB Correct.Imbalances
'***************************************************************************
SHARED Correction
'***************************************************************************
' This subroutine allows the user to activate or deactivate the ability to
' correct the force imbalance in the top acutators.  The subroutine is
' accessed by the Program Parameters menu.
'

197  CALL Clear.Window
   COLOR 15
   LOCATE 37, 3
   IF Correction = 1 THEN
      PRINT "  CORRECT IMBALANCES = ON"
   ELSEIF Correction = -1 THEN
      PRINT "  CORRECT IMBALANCES = OFF"
   END IF
   LOCATE 39, 3
   INPUT " IS THIS CORRECT? (Y or N): ", ans$
   ans$ = UCASE$(ans$)
   IF ans$ = "Y" THEN
      GOTO 199
   ELSEIF ans$ = "N" THEN
      GOTO 198
   ELSE
      GOTO 197
   END IF

198  IF Correction = 1 THEN
      Correction = -1
   ELSEIF Correction = -1 THEN
      Correction = 1
   END IF
   CALL Clear.Window
   LOCATE 37, 3
   IF Correction = 1 THEN
      PRINT "  CORRECT IMBALANCES = ON"
   ELSEIF Correction = -1 THEN
      PRINT "  CORRECT IMBALANCES = OFF"
   END IF
   LOCATE 39, 3
   INPUT " IS THIS CORRECT? (Y or N): ", ans$
   ans$ = UCASE$(ans$)
   IF ans$ = "Y" THEN
      GOTO 199
   ELSEIF ans$ = "N" THEN
      GOTO 198
   ELSE
      GOTO 197
   END IF

199 END SUB
```

```
SUB DAC.Step.Size
'**************************************************************************
SHARED DACStep.TE%, DACStep.TC%, DACStep.TW%, DACStep.BSE%, DACStep.BSW%
'**************************************************************************
' This subroutine allows the user to change the DAC step size for any or
' all of the command channels.

   CLS
   WIDTH 80, 25
   COLOR 15, 1
   VIEW (0, 0)-(639, 349)
   VIEW PRINT

' Square border around screen.

   LINE (10, 5)-(630, 5), 15
   LINE (10, 15)-(630, 15)
   LINE (10, 5)-(10, 340), 15
   LINE (630, 5)-(630, 340), 15
   LINE (10, 340)-(630, 340), 15

' Horizontal lines.

   LINE (10, 50)-(630, 50), 15
   LINE (10, 100)-(630, 100)
   LINE (10, 150)-(630, 150)
   LINE (10, 200)-(630, 200)
   LINE (10, 250)-(630, 250)
   LINE (10, 300)-(630, 300)

' Vertical lines.

   LINE (102, 15)-(102, 300)
   LINE (278, 15)-(278, 300)
   LINE (454, 15)-(454, 300)

' Print text to screen.

   LOCATE 1, 27
   PRINT " CHANGE DAC STEP SIZES "
   LOCATE 3, 4
   PRINT "Actuator"
   LOCATE 3, 18
   PRINT "DAC Step Size"
   LOCATE 3, 40
   PRINT "Disp Step Size"
   LOCATE 3, 61
   PRINT "Volt Step Size"
'
   LOCATE 6, 5
   PRINT "1 - TE"
   LOCATE 10, 5
   PRINT "2 - TC "
   LOCATE 13, 5
```

```
   PRINT "3 - TW"
   LOCATE 17, 5
   PRINT "4 - BSE"
   LOCATE 20, 5
   PRINT "5 - BSW"

' Show the present step sizes.

481 LOCATE 23, 5
   PRINT "                                        "
   PRINT "                                        "
   LINE (10, 5)-(10, 340), 15
   LOCATE 6, 23
   PRINT USING "####"; DACStep.TE%
   LOCATE 10, 23
   PRINT USING "####"; DACStep.TC%
   LOCATE 13, 23
   PRINT USING "####"; DACStep.TW%
   LOCATE 17, 23
   PRINT USING "####"; DACStep.BSE%
   LOCATE 20, 23
   PRINT USING "####"; DACStep.BSW%

' Corresponding voltage step sizes.  These are calculated by multiplying th
' DAC step size by the voltage range and dividing by 4096.

   LOCATE 6, 66
   PRINT USING "#.###"; DACStep.TE% * .004883
   LOCATE 10, 66
   PRINT USING "#.###"; DACStep.TC% * .004883
   LOCATE 13, 66
   PRINT USING "#.###"; DACStep.TW% * .004883
   LOCATE 17, 66
   PRINT USING "#.###"; DACStep.BSE% * .004883
   LOCATE 20, 66
   PRINT USING "#.###"; DACStep.BSW% * .004883

' Corresponding displacement step sizes.  These are found by multiplying the
' DAC step size by the displacement range and dividing by 4096.

   LOCATE 6, 43
   PRINT USING "#.###"; DACStep.TE% * .0088
   LOCATE 10, 43
   PRINT USING "#.###"; DACStep.TC% * .0088
   LOCATE 13, 43
   PRINT USING "#.###"; DACStep.TW% * .0088
   LOCATE 17, 43
   PRINT USING "#.###"; DACStep.BSE% * .005859
   LOCATE 20, 43
   PRINT USING "#.###"; DACStep.BSW% * .005859

' Write the text that prompts user input.

482  LOCATE 23, 5
```

```
  PRINT "                                        "
  PRINT "                                        "
  LINE (10, 5)-(10, 340), 15
  LOCATE 23, 5
  PRINT "Enter the number corresponding  "
  LOCATE 23, 59
  PRINT "10 - Exit"
  LOCATE 24, 59
  PRINT "11 - Change All "
  LOCATE 24, 5
  INPUT "to the desired action : ", mvmt
  LOCATE 23, 5
  PRINT "                                        "
  LOCATE 24, 5

' The following if-then takes the user response to the prompt and takes the
' appropriate action (changes the desired DAC step size).

  IF mvmt < 1.5 AND mvmt > .5 THEN
   INPUT "Change Step Size of Actuator TE? (Y or N): ", rsvp$
   IF UCASE$(rsvp$) = "Y" THEN
       LOCATE 24, 5
       PRINT "                                        "
       LOCATE 24, 5
       INPUT "Enter New Step Size for Actuator TE: ", DACStep.TE%
       IF DACStep.TE% < 1 THEN
         LOCATE 24, 5
         INPUT "Invalid DAC Step entered. Press any key to continue: ", rsvp2$
         DACStep.TE% = 1
       END IF
       GOTO 481
   ELSE
       GOTO 482
   END IF
'
  ELSEIF mvmt < 2.5 AND mvmt > 1.5 THEN
   INPUT "Change Step Size of Actuator TC? (Y or N): ", rsvp$
   IF UCASE$(rsvp$) = "Y" THEN
       LOCATE 24, 5
       PRINT "                                        "
       LOCATE 24, 5
       INPUT "Enter New Step Size for Actuator TC: ", DACStep.TC%
       IF DACStep.TC% < 1 THEN
         LOCATE 24, 5
         INPUT "Invalid DAC Step entered. Press any key to continue: ", rsvp2$
         DACStep.TC% = 1
       END IF
       GOTO 481
   ELSE
       GOTO 482
   END IF
'
  ELSEIF mvmt < 3.5 AND mvmt > 2.5 THEN
   INPUT "Change Step Size of Actuator TW? (Y or N): ", rsvp$
```

```
  IF UCASE$(rsvp$) = "Y" THEN
      LOCATE 24, 5
      PRINT "                                        "
      LOCATE 24, 5
      INPUT "Enter New Step Size for Actuator TW: ", DACStep.TW%
      IF DACStep.TW% < 1 THEN
        LOCATE 24, 5
        INPUT "Invalid DAC Step entered. Press any key to continue: ", rsvp2$
        DACStep.TW% = 1
      END IF
      GOTO 481
  ELSE
      GOTO 482
  END IF
'
ELSEIF mvmt < 4.5 AND mvmt > 3.5 THEN
  INPUT "Change Step Size of Actuator BSE? (Y or N): ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
      LOCATE 24, 5
      PRINT "                                        "
      LOCATE 24, 5
      INPUT "Enter New Step Size for Actuator BSE: ", DACStep.BSE%
      IF DACStep.BSE% < 1 THEN
        LOCATE 24, 5
        INPUT "Invalid DAC Step entered. Press any key to continue: ", rsvp2$
        DACStep.BSE% = 1
      END IF
      GOTO 481
  ELSE
      GOTO 482
  END IF
'
ELSEIF mvmt < 5.5 AND mvmt > 4.5 THEN
  INPUT "Change Step Size of Actuator BSW? (Y or N): ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
      LOCATE 24, 5
      PRINT "                                        "
      LOCATE 24, 5
      INPUT "Enter New Step Size for Actuator BSW: ", DACStep.BSW%
      IF DACStep.BSW% < 1 THEN
        LOCATE 24, 5
        INPUT "Invalid DAC Step entered. Press any key to continue: ", rsvp2$
        DACStep.BSW% = 1
      END IF
      GOTO 481
  ELSE
      GOTO 482
  END IF
'
ELSEIF mvmt < 10.5 AND mvmt > 9.5 THEN
  INPUT "Exit Subprogram? (Y or N): ", rsvp$
  IF UCASE$(rsvp$) = "Y" THEN
     GOTO 490
  ELSEIF UCASE$(rsvp$) = "N" THEN
```

```
      GOTO 482
    END IF
'
  ELSEIF mvmt < 11.5 AND mvmt > 10.5 THEN
    INPUT "Change Step Size of all Actuators? (Y or N): ", rsvp$
    IF UCASE$(rsvp$) = "Y" THEN
        LOCATE 24, 5
        PRINT "                                        "
        LOCATE 24, 5
        INPUT "Enter New Step Size for all Actuators: ", DACStep.TE%
        IF DACStep.TE% < 1 THEN
          LOCATE 24, 5
          INPUT "Invalid DAC Step entered. Press any key to continue: ", rsvp2$
          DACStep.TE% = 1
        END IF
        DACStep.TW% = DACStep.TE%
        DACStep.TC% = DACStep.TE%
        DACStep.BSE% = DACStep.TE%
        DACStep.BSW% = DACStep.TE%
        GOTO 481
    ELSE
        GOTO 482
    END IF
  ELSE
    GOTO 482
490 END IF
'
END SUB
```

```
SUB Save.Data.Step.Size
'*************************************************************************
SHARED Save.Data.Step
'*************************************************************************
' This subroutine allows the user to change the interval at which data is
' saved, or the Save.Data.Step.
'
   COLOR 15
190 LOCATE 37, 3
   PRINT USING "PRESENT SAVE DATA STEP: ##"; Save.Data.Step
   LOCATE 39, 3
   INPUT "NEW SAVE DATA STEP SIZE: ", Save.Data.Step2
   CALL Clear.Window
   LOCATE 37, 3
   PRINT USING "NEW SAVE DATA STEP SIZE IS ##"; Save.Data.Step2
   LOCATE 39, 3
   INPUT "IS THIS CORRECT (Y OR N): ", rsvp$
   IF UCASE$(rsvp$) = "Y" THEN
      Save.Data.Step = Save.Data.Step2
      GOTO 195
   ELSE
      CALL Clear.Window
      GOTO 190
   END IF

195 END SUB
```

```
SUB Tolerance.Size
'*************************************************************************
SHARED Tolerance
'*************************************************************************
' This subroutine allows the user to change the tolerance.  The tolerance is
' a value used by subroutine Pause, which calculates two consecutive force
' vales in each actuator.  If the two forces are not equal within the
' specified tolerance, then two force values are calculated again until they
' are equal within the specified tolerance, meaning the system has reached equilibrium.
'
   COLOR 15
180 LOCATE 37, 3
   PRINT USING "PRESENT TOLERANCE: ##.##  K"; Tolerance
   LOCATE 39, 3
   INPUT "ENTER NEW TOLERANCE SIZE: ", Tolerance2
   CALL Clear.Window
   LOCATE 37, 3
   PRINT USING "NEW TOLERANCE IS ##.## K"; Tolerance2
   LOCATE 39, 3
   INPUT "IS THIS CORRECT (Y OR N): ", rsvp$
   IF UCASE$(rsvp$) = "Y" THEN
      Tolerance = Tolerance2
      GOTO 185
   ELSE
      CALL Clear.Window
      GOTO 180
   END IF

185 END SUB
```

APPENDIX B
VARIABLE DEFINITIONS

DIM SHARED VARIABLES

DataBuf(32)	An array of 32 numbers (for the 32 A/D channels) used in Sub Sample.AD. It stores the values sampled by the board.
nDasErr	Error flag used in Sample.AD. If an error occurs (such as the computer cannot get a handle to the device), it signals the program to stop.
szCfgName	Stores the name of the configuration file for the A/D board. If the setup of the board is changed or a different application is required (different voltage range being used for example), this file must be changed.
hDev	A/D device handle.
hAD	A/D frame handle (should not change).
wStatus	Variable used for monitoring the status of A/D sampling.
dwCount	Same as above.
dwFactor	This is the factor that determines how fast the sampling will occur. the smaller the factor, the faster the sampling.
ACommand(16)	Array of values that range from 0 to 4096 that are sent to the D/A board and converted to output voltages for the actuators. These are the commands computed by the software.
DERR	Error flag used for D/A board.
hDDA16	D/A device handle.
hDA	D/A frame handle.
ChData(0 to 20, 0 to 32)	Two-dimensional array that stores the values read by the A/D board. It stores 20 count values for each channel that are then added together and averaged.
ChVolt(0 to 32)	This is the array of averaged values from ChData that are converted from count values to voltages.
XducerCon(1 to 32)	Array of calibration constants for the A/D channels.
M(1000)	Array that the moment about the Z axis gets written to in order to be plotted.
GTheta(1000)	Array that the rotation about the Z axis gets written to in order to be plotted.

COMMON VARIABLES

The first set of common variables are from the common block /navy2/, which are shared among the main module subroutines and the secondary module subroutines.

Loop.Steps	The total number of steps the program will do in one loading loop. This is changeable by the user through the step size menu.
Tolerance	Used by subroutine Pause, which ensures the system is in equilibrium before moving to the next step. Each actuator force is calculated twice, then the difference between the two is found. If the difference is larger than the tolerance, which is user defined, then the process is repeated until the difference is smaller than the tolerance and the next loading step can then be performed.
scale	Holds a number from 1 to 4 that signifies which graph scale is to be plotted for the moment-curvature plot.
Save.Data.Step	The step interval at which data is automatically saved. This variable is changeable by the user through the step size menu
Correction	This variable is used in the Decision.Algorithm subroutine. It tells subroutine whether the force imbalance in the top actuators should be corrected or not. If correction has a value of –1, the imbalance will not be corrected. If correction has a value of 1, the appropriate branch of the decision algorithm is executed so that the imbalances are corrected.
CommandError	A flag that indicates that one of the command voltages is beyond the allowable voltage range of -9.95 to 9.95 volts set by the software. When this variable is flagged, an error message prints to the screen.
Command.TE%	Command count value (converted from a voltage) that is sent to the D/A board for each actuator. The % sign indicates this variable is an integer value. (Each actuator has one of these variables).
Command.16%	Command count value that is sent to the D/A board to trigger the auxiliary data acquisition system. This value is sent to the 16th channel on the D/A board.
Command.Trigger	The variable that contains the count value for triggering the data acquisition system. When data is to be saved, this variable is given the count value of 3072, or 5.00 volts. When is is not to be saved, it is reset to the count value 1228, corresponding to a voltage of –4.00 volts that does not trigger the system. In sub Move.Actuators, Command.16% is simply set equal to the Command.Trigger.
DACStep.TE%	Size of the DAC step (from 0 to 4096). A DAC step of 1 is the smallest movement that an actuator can make in one step. This is equivalent to 20 V / 4096 = 0.004883 Volts. The DAC step can be changed by the user.
Delta.TE%	Has a value of 1,0, or -1. If the decision algorithm decides that a given actuator needs to be extended in that step, Delta is given a value of 1. If it needs to be retracted, Delta is set equal to -1. If the actuator is not to move, Delta is 0. These values are multiplied

by the DAC step size and added to the previous command to find the new command count (Command.TE%,etc.).

The second set of common variables are from the common block /navy/, which are shared only among the main module subroutines.

Ident$	String variable that stores the test name, which gets written to the output file.
OutFile1$	String variable that holds the name of the first output file, to which the actuator and link force values get written. The name of this file is input from the data file that is read at the beginning of the test.
OutFile2$	String variable that holds the name of the second output file, to which the grillage transducer values and the bottom link transducer values get written.
F5Flag	When F5 is pressed, the value of this variable changes from -1 to 1. When this flag value is checked in the second main program loop, and the value is -1 then the Program Functions menu is displayed.
F10Flag	A flag that allows the user to exit from various subprograms and from the main program.
F7Flag	A flag that tells the system to save data when F7 is pressed. This also triggers the auxiliary data acquisition system to save data.
F3Flag	A flag that causes the Program Parameters menu to be displayed.
F9Flag	A flag that allows the user to exit the main program loop.
F4Flag	A flag that allows the user to pause the execution of the main Executing loop.
nPass	Counter that counts how many times a given subroutine has been executed without being exited. When a subroutine is exited, nPass is given the value of 0 so that the next time through the graphics will be redisplayed. This helps cut down on screen flashing.
nData	Counter that holds the total number of steps performed in the test
NChannel	Number of A/D channels being used. This number is read by the input file.
nStep	Variable displayed on main screen that tells the user at which step the program is in the loading loop (how many Loop.Steps have been executed).
Direction	Tells the program whether the rotation is being increased or decreased. This value is assigned when the user chooses to increase or decrease rotation from the Program Functions menu. When Direction = 1, rotation is increasing. When Direction = -1, rotation is decreasing. These values are used in the Decision.Algorithm to decide which actuators to move.
Save.Data.Counter	Counter used for determining when data needs to be saved. When Save.Data.Counter equals a multiple of Save.Data.Step, data is

	saved automatically. If F7 is used to save data at random, Save.Data.Counter is reset to 0.
CommandV.TE	Command voltage specified by the user when the actuators are being initialized. The suffix TE designates the actuator (T=Top, B=Bottom, C=Center, N=North, S=South). Each actuator has its own CommandV variable. The voltage value entered should be equal to its initial feedback transducer voltage. These voltages are then converted to count values that can be sent to the D/A board.
FORCE.TE	Total force in an actuator or horizontal link. Calculated by multiplying the channel voltage by the load cell's calibration constant.
FORCE.XN.TE, TC, TW	The X component of force in the top actuators at the north grillage.
FORCE.XS.TE, TC, TW	The X component of force in the top actuators at the south grillage. (equal in magnitude but opposite in sign to FORCE.XN. TE, etc.).
FORCE.YN.TE, TC, TW	The Y component of force in top actuators at north grillage.
FORCE.YS.TE, TC, TW	The Y component of force in top actuators at south grillage (equal and opposite to FORCE.YN.TE, etc.).
FORCE.ZN.TE, TC, TW	The Z component of force in top actuators at north grillage.
FORCE.ZS.TE, TC, TW	The Z component of force in top actuators at south grillage (equal and opposite to FORCE.ZN.TE, etc.).
FORCE.X. BNW, BNE	The X component of force in the bottom horizontal links.
FORCE.Y.BNW, BNE	The Y component of force in the bottom horizontal links.
FORCE.Z.BNW, BNE	The Z component of force in the bottom horizontal links.
FORCE.X.BSW, BSE	The X component of force in the bottom actuators.
FORCE.Y.BSW, BSE	The Y component of force in the bottom actuators.
FORCE.Z.BSW, BSE	The Z component of force in the bottom actuators.
FLINK.NE, NW, SE, SW	The total force in each vertical link.
FLINK.X.NE, NW, SE, SW	The X component of force in each vertical link.
FLINK.Y.NE, NW, SE, SW	The Y component of force in each vertical link.
XDUCER.1NX, 2NX, 3NX	Distances measured by the north grillage X-direction transducers (temposonics). These (and the other TEMPO variables) are used for the purpose of calculating the grillage vectors and transformation matrices.
XDUCER.1NY, 3NY	Distances measured by the north grillage Y-direction transducers.
XDUCER.1SX, 2SX, 3SX	Distances measured by the south grillage X-direction transducers.
XDUCER.1SY, 3SY	Distances measured by the south grillage Y-direction transducers.
DELTA.X.BSE, BSW	Distances measured by the bottom link transducers. These values are used in the calculations in Actuator.Vectors to determine the vectors for the bottom links and actuators.
CN.X.TE, TW, TC	X component of C vector for top actuators at north grillage. These represent the distance in the global X direction from point r'_N to the top actuator end. This variable is common because these values are used as moment arms for calculating moments in

	Equilibrium.Equations.
CN.Y.TE, TW, TC	Y component of C vector for top actuators at north grillage (distance in global X direction from point r'_N to the top actuator end).
CS.X.TE, TW, TC	X component of C vector for top actuators at south grillage. These represent the distance in the global X direction from point r'_S to the top actuator end.
CS.Y.TE, TW, TC	Y component of C vector for top actuators at south grillage (distance in global X direction from point r'_S to the top actuator end).
CN.X.BNE, BNW	X component of C vector for bottom links at north grillage.
CN.Y.BNE, BNW	Y component of C vector for bottom links at north grillage.
CS.X.BSE, BSW	X component of C vector for bottom actuators at south grillage.
CS.Y.BSE, BSW	Y component of C vector for bottom actuators at south grillage.
D.X.NE, NW, SE, SW	X component of D vector for vertical links. These represent the distance in the global X direction from point r'_N to the bottom end of the link. This variable is common because the values are used as moment arms for calculating moments in Equilibrium.Equations.
D.Y.NE, NW, SE, SW	Y component of D vector for vertical links. These represent the distance in the global Y direction from point r'_N to the bottom end of the link.

<u>CONSTANTS</u>

CPN.X.TW, TC, TE	Distance in local X direction from r'_N to the north end of the top actuators. These are used for calculating the global C vectors. (X component of the C'_N vector).
CPN.Y.TW, TC, TE	Distance in local Y direction from r'_N to the north end of the top actuators. (Y component of the C'_N vector).
CPN.Z.TW, TC, TE	Distance in local Z direction from r'_N to the north end of the top actuators. (Z component of the C'_N vector).
CPS.X.TW, TC, TE	Distance in local X direction from r'_S to the south end of the top actuators. These are used for calculating the global C vectors. (X component of the C'_S vector).
CPS.Y.TW, TC, TE	Distance in local Y direction from r'_N to the south end of the top actuators. (Y component of the C'_S vector).
CPS.Z.TW, TC, TE	Distance in local Z direction from r'_N to the north end of the top actuators. (Z component of the C'_S vector).
CPN.X.BNW,BNE	Distance in local X direction from r'_N to the north end of the bottom horizontal links.
CPN.Y.BNW,BNE	Distance in local Y direction from r'_N to the north end of the

	bottom horizontal links.
CPN.Z.BNW,BNE	Distance in local Z direction from r'_N to the north end of the bottom horizontal links.
CPS.X.BSW,BSE	Distance in local X direction from r'_N to the south end of the bottom actuators.
CPS.Y.BSW,BSE	Distance in local Y direction from r'_N to the south end of the bottom actuators.
CPS.Z.BSW,BSE	Distance in local Z direction from r'_N to the south end of the bottom actuators.
ENO.X.BSE, BSW	Original distance in global X direction from pont r'_N to the north end of the bottom actuators.
ENO.Y.BSE, BSW	Original distance in global Y direction from pont r'_N to the north end of the bottom actuators.
ENO.Z.BSE, BSW	Original distance in global Z direction from pont r'_N to the north end of the bottom actuators.
ESO.X.BNE, BNW	Original distance in global X direction from pont r'_N to the south end of the bottom horizontal links.
ESO.Y.BNE, BNW	Original distance in global Y direction from pont r'_N to the south end of the bottom horizontal links.
ESO.Z.BNE, BNW	Original distance in global Z direction from pont r'_N to the south end of the bottom horizontal links.
L2.X.NE, NW, SE, SW	Distance in local X direction from point r'_N (for north grillage) or r'_S (for south grillage) to bottom end of vertical links. These values are used for calculating the same distance in global coordinates.
L2.Y.NE, NW, SE, SW	Distance in local Y direction from point r'_N (for north grillage) or r'_S (for south grillage) to bottom end of vertical links. These values are used for calculating the same distance in global coordinates.
L2.Z.NE, NW, SE, SW	Distance in local Z direction from point r'_N (for north grillage) or r'_S (for south grillage) to bottom end of vertical links. These values are used for calculating the same distance in global coordinates.
P1N.X.O, .Y.O, Z.O	Coordinates that define the original locations of point P1N.
P2N.X.O, .Y.O, Z.O	Coordinates that define the original locations of point P2N.
P3N.X.O, .Y.O, Z.O	Coordinates that define the original locations of point P3N.
P1S.X.O, .Y.O, Z.O	Coordinates that define the original locations of point P1S.
P2S.X.O, .Y.O, Z.O	Coordinates that define the original locations of point P2S.
P3S.X.O, .Y.O, Z.O	Coordinates that define the original locations of point P3S.

NON-SHARED VARIABLES - The following are non-shared variables used only by subroutine Actuator Vectors.

P1N.X, P1N.Y, P1N.Z	Global coordinates of point P1 on the north grillage - calculated

	from original coordinates plus TEMPO values (grillage transducers).
P2N.X, P2N.Y, P2N.Z	Global coordinates of point P2 on north grillage.
P3N.X, P3N.Y, P3N.Z	Global coordinates of point P3 on north grillage.
P1S.X, P1S.Y, P1S.Z	Global coordinates of point P1 on the south grillage - calculated from original coordinates plus TEMPO values (grillage transducers).
P2S.X, P2S.Y, P2S.Z	Global coordinates of point P2 on south grillage.
P3S.X, P3S.Y, P3S.Z	Global coordinates of point P3 on south grillage.
VN.X, VN.Y, VN.Z	X, Y, and Z components of the V vector on the north grillage.
VS.X, VS.Y, VS.Z	X, Y, and Z components of the V vector on the south grillage.
WN.X, WN.Y, WN.Z	X, Y, and Z components of the W vector on the north grillage.
WS.X, WS.Y, WS.Z	X, Y, and Z components of the W vector on the south grillage.
UN.X, UN.Y, UN.Z	X, Y, and Z components of the U vector on the north grillage. The U vector is the unit normal to the plane of the grillage.
US.X, US.Y, US.Z	X, Y, and Z components of the U vector on the south grillage.
tN.11 - tN.33	Values of transformation matrix for north grillage.
tS.11 - tS.33	Values of transformation matrix for south grillage.
RN.X, RN.Y, RN.Z	Components of RN vector, which goes from global reference point r to local reference point r'_N.
RS.X, RS.Y, RS.Z	Components of RS vector, which goes from global reference point r to local reference point r'_S.
EN.X.TW, TC, TE	Distance in global X direction from point r to north end of top actuator.
EN.Y.TW, TC, TE	Distance in global Y direction from point r to north end of top actuator.
EN.Z.TW, TC, TE	Distance in global Z direction from point r to north end of top actuator.
ES.X.TW, TC, TE	Distance in global X direction from point r to south end of top actuator.
ES.Y.TW, TC, TE	Distance in global Y direction from point r to south end of top actuator.
ES.Z.TW, TC, TE	Distance in global Z direction from point r to south end of top actuator.
EN.X.BNW BNE	Distance in global X direction from point r to north end of bottom horizontal link.
EN.Y.BNW, BNE	Distance in global Y direction from point r to north end of bottom horizontal link.
EN.Z.BNW, BNE	Distance in global Z direction from point r to north end of bottom horizontal link.
ES.X.BNW BNE	Distance in global X direction from point r to south end of bottom horizontal link.
ES.Y.BNW, BNE	Distance in global Y direction from point r to south end of bottom

	horizontal link.
ES.Z.BNW, BNE	Distance in global Z direction from point r to south end of bottom horizontal link.
EN.X.BSW BSE	Distance in global X direction from point r to north end of bottom actuator.
EN.Y.BSW, BSE	Distance in global Y direction from point r to north end of bottom actuator.
EN.Z.BSW, BSE	Distance in global Z direction from point r to north end of bottom actuator.
ES.X.BSW BSE	Distance in global X direction from point r to south end of bottom actuator.
ES.Y.BSW, BSE	Distance in global Y direction from point r to south end of bottom actuator.
ES.Z.BSW, BSE	Distance in global Z direction from point r to south end of bottom actuator.
A.X.TW, TC, TE	X component of the top actuator vectors.
AY.TW, TC, TE	Y component of the top actuator vectors.
A.Z.TW, TC, TE	Z component of the top actuator vectors.
A.X.BSW, BSE	X component of the bottom actuator vectors.
A.Y.BSW, BSE	Y component of the bottom actuator vectors.
A.Z.BSW, BSE	Z component of the bottom actuator vectors.
A.X.BNW, BNE	X component of the bottom horizontal link vectors.
A.Y.BNW, BNE	Y component of the bottom horizontal link vectors.
A.Z.BNW, BNE	Z component of the bottom horizontal link vectors.
A.TW, TC, TE, BSW, BSE	Total actuator length - calculated by taking the square root of the squares of A.X, A.Y, and A.Z.
A.BNW, BNE	Total horizontal link length.
L1.X.NE, NW, SE, SW	X component of L1 vector for vertical links - L1 vector goes from global reference point r to the bottom of the link.
L1.Y.NE, NW, SE, SW	Y component of L1 vector for vertical links.
L1.Z.NE, NW, SE, SW	Z component of L1 vector for vertical links.
L.X.NE, NW, SE, SW	X component of L vector for vertical links - L vector represents the location of the link.
L.Y.NE, NW, SE, SW	Y component of L vector for vertical links.
L.Z.NE, NW, SE, SW	Z component of L vector for vertical links.
L.NE, NW, SE, SW	Total length of vertical links.