NASA TREETOPS

FINAL REPORT

Prepared by:
Oakwood College/Dynamic Concepts, Inc.
June 1996

CONTRACT NAS8-40194

Table of Contents

Project Summary

Under the provisions of contract number NAS8-40194, which was entitled "TREETOPS Structural Dynamics and Controls Simulation System Upgrade", Oakwood College contracted to produce an upgrade to the existing TREETOPS suite of analysis tools. This suite includes the main simulation program, TREETOPS, two interactive preprocessors, TREESET and TREEFLX, an interactive post processor, TREEPLOT, and an adjunct program, TREESEL.

The capability of the TREETOPS simulation package was extended by providing the ability to handle geometrically nonlinear problems through the use of a specially designed "User Controller". Software for computing the modal integrals from any finite element code (e.g., NASTRAN, ANSYS, etc.) was developed in two forms—Fortran source code and a MATLAB script file.

A "Software Design Document", which provides descriptions of the argument lists and internal variables for each subroutine in the TREETOPS suite, was established. Additionally, installation guides for both DOS and UNIX platforms were developed. Finally, updated User's Manuals, as well as a Theory Manual, were generated under this contract.

1.0 Introduction

The TREETOPS simulation package is a multibody dynamics tool capable of formulating and integrating the equations of motion for a wide variety of aerospace and mechanical systems. Special attention to implementation and integration of controllers (passive and/or active) into the state equations was paid in the initial formulation of the code—which sets it apart form other mechanical dynamics codes.

Over time, certain deficiencies and limitations in the TREETOPS code have been noted in the user community. Among these deficiencies was a lack of software control documents, limitations to geometrically linear response regimes, and a lack of user friendly modal integral tools. NASA contract number NAS8-40194 was established to address these concerns.

This report will detail the activities undertaken in pursuit of the goals established for contract NAS8-40194. The Software Design Document, not included here for the sake of brevity, will be discussed in Section 2 of this report. The updates and additions to the TREETOPS Manuals are presented in Section 3. Section 4 details the simulation enhancements—which include the addition of geometric stiffening to the code as well as the development of user-friendly modal integral software tools. Section 5 will conclude this report.

2.0 Software Design Documentation

The TREETOPS family of tools is composed of the following major software modules or programs: TREETOPS, TREEFLX, TREESET, TREESEL, TREEPLOT. TREETOPS is a time history simulation tool for multibody systems with active control elements. TREETOPS considers the total structure as in interconnected set of individual rigid or flexible bodies capable of undergoing large translations and rotations relative to one another. In addition to large rigid body configuration changes, the components of the system may simultaneously experience small elastic deformations. TREETOPS uses the assumed modes method to model the flexibility of the bodies. In this technique, the elastic deformation of each body is approximated through a linear combination of the products of shape functions and generalized time coordinates. The shape functions are typically a subset of normal modes derived from a finite element model of the component.

TREEFLX is a program designed to compute flexible body input data for the components of TREETOPS from both COSMIC and MSC NASTRAN finite element packages. This data consists of the component shape functions, generalized mass and stiffness matrices, and modal integrals. The output of TREEFLX is the 'problem.FLN' file.

TREESET is an interactive preprocessor program to assist the user in entering data for the various TREETOPS programs. TREESET acts as an editor for adding new data to a file, deleting old data, or modifying existing data. The output of TREESET is the 'problem.INT' file.

TREESEL is also an interactive program to aid the user in the selection of a subset component modes which model the flexible elements of the system. Fewer modes reduces simulation run time and facilitates control system design. TREESEL is based on the modified component cost analysis method of model reduction.

The TREEPLOT program reads the output data file 'problem.OUT' written by TREETOPS and plots or prints those variables selected by the user. It is a menu based post-processor based on keywords. Other versions of TREETOPS have been modified to output data in a MATLAB compatible file for plotting and analysis of simulation results,

thereby minimizing the use of TREEPLOT.

Oakwood College constructed a functional hierarchical organization of these code modules with call and caller graphs. A detailed flow chart of the entire code was also developed. For each of the files which make up the TREETOPS family of programs, Oakwood College generated documentation which consists: filename, purpose, calling routine, called routines, common blocks, input parameters, output parameters, unused parameters, input global variables, output global variables, local variables.

For the TREETOPS program, 296 files were documented. The documented main routine is shown below.

```
     1.131   MAIN

INTERFACE :
     Called by :

     None


     Calls subprograms

     OPNTOP     INPUT      INITZE     DYNAMI     LINRZE
     OUTPUT     RESAVE

DATA :
     Commons included :
     DBP.F, DBSC.F, BUFFER.F, DBC.F

     Input Parameter Name List    :     None

     Output Parameter Name List   :     None

     Unused Parameter Name List   :     None

     Input Global Variables List  :
       matun    integer*4              MAT file unit number
       dtout    real*4                 Data inerval
       tend     real*4                 Simulation stop time
       time     real*4                 Simulation time (sec)
```

```
       tlin    real*4                          Time used in the
linearization option,
          =0 if L is chosen, =TEND + 1 for Z and N
       toend   real*4                  End of the time interval for
data
          plotting, =TEND
       tostrt  real*4                  Start time for data plotting
(is
          usually equal to 0)

     Output Global Variables List :     None

DESCRIPTION :

     Found_in_file :
          maintops.for

     Purpose :
     This is the main program of treetops and it calls seven other
major
     subprograms to perform the task of TREETOP software. The
subprograms
     are:
     -- OPNTOP  : Open necessary files to support TREETOPS input
and
                output data.
     -- INPUT   : Drive the data input function for TREETOPS by
calling
                the appropriate input routines.
     -- INITZE  : Initialize the simulation and perform initial
                computation of simulation data.
     -- DYNAMI  : Execute the simulation dynamics by performing
                 integration and propagation of the system state
vector.
     -- LINRZE  : Generate the linearized system dynamics.
     -- OUTPUT  : Store simulation output data in two data files.
Both
               files contain identical information, but one file is
                formated and the other is unformated.
     -- RESAVE   : Outputs event information at the end of the
simulation.
```

For the TREEFLX program, 119 files were documented. The documentation for the main routine of TREEFLX is as follows.

```
     3.118 TREFLX

INTERFACE :
     Called by :

     None

     Calls subprograms :

     DEDAMP     ERRWRT     EXTDAT     MEMCHK     NODCHK     OPNDAF
OPNFACE
     RDAGMT     RDNAST     RDRTMD     TMCALC     WRTFLX

DATA :
     Commons included           :
     PAR1.F   PAR2.F

     Input Parameter Name List    :     None

     Output Parameter Name List   :     None

     Unused Parameter Name List   :     None

     Input Global Variables List  :
       augy     integer*4(pnbb)
       bbid     integer*4
       bbnu     integer*4
       flxbnm   integer*4(pnbb)
       flxid    integer*4(pnbb)
       nmodbj   integer*4(pnbb)
       nnodbj   integer*4(pnbb)
       nuflbd   integer*4
       dammth   character*2(pnbb)
       redmth   character*1(pnbb)
       error    logical*4
       doesex   logical*4

     Output Global Variables List :
```

```
        bbid    integer*4
        bbnu    integer*4
        nmde    integer*4
        nnde    integer*4
        error   logical*4

DESCRIPTION :

     Found in file :
          mainpr.for

     Purpose :
     This is the main calling program for the TREETOPS
pre-processor
     TREEFLX. Major inputs are the TREESET interactive setup file
     problem.INT and the NASTRAN output data for each body. The
     major output is a problem.FLN file for TREETOPS.
```

For the program TREESET, 59 files were reviewed and documented. The description of the main routine for TREESET is as follows.

```
     2.42 MAIN

INTERFACE :
     Called by :

     None

     Calls subprograms :

     ADD        CONVRT     DELETE     FCHECK     GETF       HELPS
     MODIFY     OPNSET     PRINT      RBUF       SAVE       STOP

DATA :
     Commons included            :
     VARS.FOR  BUFFER.FOR

     Input Parameter Name List    :     None
```

```
      Output Parameter Name List   :      None

      Unused Parameter Name List   :      None

      Input Global Variables List  :
        revno   integer*4
        revdat  character*8
        ifunct  integer*4

      Output Global Variables List :
        revno   integer*4
        revdat  character*8

DESCRIPTION :

      Found in file :
           main.for

      Purpose :
      This is the main routine for TREETOPS pre-processor TREESET.
      The output of TREESET is the  problem.INT file which is read
      by TREETOPS.
```

The TREESEL program consists of 81 files. Again, the documented main routine of TREESEL is shown below.

```
      4.78 TRESEL

INTERFACE :
      Called by :

      None

      Calls subprograms :

      ACTUA      CCOST      DECMAT     MEZER      MEZERS     MLTS2
      MPRT1      MTEQS2     SGET

DATA:
      Commons included             :
      PAR.F    SEL.F    TCA.F
```

```
Input Parameter Name List    :     None

Output Parameter Name List   :     None

Unused Parameter Name List   :     None

Input Global Variables List  :
  gmf     double precision(nacma,nacma)
  gdf     double precision(nacma,nacma)
  gkf     double precision(nacma,nacma)
  ba double precision(nacma,numax)
  pa double precision(numax,nacma)
  ra double precision(numax,nacma)
  idev    integer*4
  iout    integer*4
  iu integer*4
  nac     integer*4
  nxva    integer*4
  nxa     integer*4
  inpop   integer*4
  nx integer*4
  nr integer*4
  nu integer*4
  nxv     integer*4
  nbody   integer*4
  ad double precision(nxmax,nxmax)
  bd double precision(nxmax,numax)
  cd double precision(nrmax,nxmax)
  dd double precision(nrmax,numax)

Output Global Variables List :
  gm double precision(nxvma,nxvma)
  gd double precision(nxvma,nxvma)
  gk double precision(nxvma,nxvma)
  gmf     double precision(nacma,nacma)
  gdf     double precision(nacma,nacma)
  gkf     double precision(nacma,nacma)
  ba double precision(nacma,numax)
  pa double precision(numax,nacma)
  ra double precision(numax,nacma)
  nac     integer*4
  nxva    integer*4
  nxa     integer*4
```

```
      inpop   integer*4
      nr integer*4
      nxv     integer*4
      nms     integer*4(nbmax)
      nme     integer*4(nbmax)
      nrs     integer*4(nbmax)
      nte     integer*4(nbmax)
      nre     integer*4(nbmax)
      nts     integer*4(nbmax)
      nbody   integer*4
      idbod   integer*4(nbmax)
      rst     logical*4
      nct     logical*4
      ad double precision(nxmax,nxmax)
      bd double precision(nxmax,numax)
      cd double precision(nrmax,nxmax)
      dd double precision(nrmax,numax)
      dataun  integer*4

DESCRIPTION :

     Found in file :
          treesel.f

     Purpose :
     This is the main routine for the pre-processor TREESEL
     designed to aid the user in the modal selection / model
     reduction process.
```

The TREEPLOT program consists of 48 files which were documented by Oakwood College. The main routine for TREEPLOT is displayed below.

```
     5.23 MAIN

INTERFACE :
     Called by :

     None

     Calls subprograms :
```

DEFALT PLOT PRINT RESET SAVECM SETUP

DATA :

Commons included :

FRAMES.FOR

Input Parameter Name List : None

Output Parameter Name List : None

Unused Parameter Name List : None

Input Global Variables List :

buf character*50(1000)

count integer*4

Output Global Variables List : None

DESCRIPTION :

Found in file :

mainp.f

Purpose :

This is the main routine of Treeplot software. It resets data, display a menu and call the proper routine according to the user choice from the menu.

3.0 TREETOPS Manuals

Revised TREETOPS theoretical and user manuals were developed by Oakwood College under this effort. The User's Manual was prepared in a format approved by the NASA COTR to allow the user to exercise the TREETOPS package. Additionally, the User's Manual modified to correct current deficiencies and changes. The manual presents detailed information necessary to use all options available in the program. Various examples which cover the main features were included as well as trouble shooting information and software compilation and installation notes. Example problems were worked in the manual and supplied on disk to validate the operation of TREETOPS.

The current TREETOPS Theory Manual was reviewed and revised. Parts of the TREETOPS and CONTOPS User's Manuals were included in the revised Theory Manual to provide background on how TREETOPS works. The notation within the Theory Manual was changed to be consistent. A section on geometric non-linearities was added to the manual.

4.0 Simulation Enhancements

Under this contract, a general procedure for incorporating geometric stiffening into the TREETOPS simulation package was developed. An algorithm, FORTRAN program, and MATLAB M file were also developed to compute the modal integrals and other data required for the 'problem.FLX' file. These accomplishments are discussed in Sections 4.1 and 4.2 through documents previously distributed to NASA and reprinted here in entirety.

4.1 Incorporation of Geometric Stiffening into TREETOPS

4.1.1 Introduction.

The purpose of this task was to develop a procedure for incorporating geometric stiffening into the TREETOPS simulation package. Although TREETOPS had a rudimentary capability of handling geometrically nonlinear beams that were spin-stiffened in previous versions, no general procedure applicable to arbitrary bodies was made available.[1] The procedure developed herein is capable of addressing geometrically nonlinear problems that arise in a typical multibody system setting—a setting in which the inter-body forces, as well as external forces, must be considered in the development of the geometric stiffness terms associated with each body of the system.

A TREETOPS "user controller" is used to actually augment the underlying equations of motion of each body with the appropriate geometric stiffening terms. The modeling techniques (i.e., additions to the standard TREETOPS model of a structure) required to facilitate the geometric stiffening will be illustrated in several example problems. A brief introduction to the theoretical basis of the procedure, followed by some general guidelines for implementing the technique into a standard TREETOPS model will precede the example problems.

As an introductory note of caution, it must be mentioned that the inclusion of geometric stiffness into a TREETOPS dynamic model (or any other dynamic simulation

---

[1]It does appear that in Version 10, the subroutine GENMOD attempts to mimic the procedure of Banerjee and Dickens [1]; however, there are two problems with this implementation. First, it is not at all clear how to use the GENMOD code—it was never documented. Second, the method of Banerjee and Dickens is not sufficiently general to solve the general problem presented by multibody systems with externally applied loads [2].

package) is generally a task that should be undertaken only by individuals with a relatively high degree of experience in both TREETOPS modeling and the finite element method. Whereas the required additions to the TREETOPS model (<...>.int and <...>.flx files) are straightforward, the preparation of the required generalized geometric stiffness matrices is somewhat laborious and typically requires the use of a finite element package (e.g., NASTRAN) capable of solving geometrically nonlinear problems.

4.1.2 Brief Theoretical Background.

This section presents a synopsis of the underlying mechanics associated with the inclusion of geometric stiffness into TREETOPS. A complete derivation of the associated nonlinear equations can be found in Reference [1]. For the sake of clarity, the inclusion of geometric stiffness in the "standard" structural dynamics problem, i.e.,

$$M\ddot{x} + Kx = F \tag{1}$$

will be used to develop the salient features of the new TREETOPS procedures.[2] The stiffness matrix in the above equation can be broken down into two parts,

$$K = K_L + K_G \tag{2}$$

where K_L is the usual linear stiffness matrix and K_G is the geometric stiffness matrix.

The geometric stiffness matrix is a function of the internal (stress) loads acting in the body (i.e., $K_G = K_G(\sigma)$). For linear structures, the internal loads are a linear function of the applied loads. In other words, $\sigma = \sigma(f_{inert}, f_{ext})$. If we make the assumption that the inertial loads' contribution to K_G is due to the rigid body motions of the body in question, then the inertial loads can be developed directly from Newton's 2nd Law and Euler's equation for the lumped mass case, e.g.,

---

[2] Reference [1] contains the analogous development for the fully nonlinear multibody equations of motion; however, the simpler development presented herein should allow the user sufficient depth of knowledge to address a wide range of geometrically nonlinear problems.

$$f_I^k = -m^k a^k \quad \text{and} \quad t_I^k = -\mathbb{I}_k \alpha - \omega \times \mathbb{I}_k \cdot \omega \tag{3}$$

where f_I^k is the inertial force, t_I^k represents the inertial torque, m^k and $\mathbb{I}_k$ are the mass and mass moment of inertia matrix, respectively, a^k is the translational acceleration, and finally α and ω are the angular acceleration and rate vectors. The "k" that is tagged to the terms in Eq.(3) indicates an association with the k^{th} nodal body (or in simple configurations, the k^{th} particle) of the body.

Since the acceleration term a^k is being computed as if the body in question is rigid, then we can write

$$a^k = a^o + \alpha \times \rho^k + \omega \times \omega \times \rho^k \tag{4}$$

where ρ^k is the position vector from the body's origin ("o") to point k. This is illustrated in Figure 1.

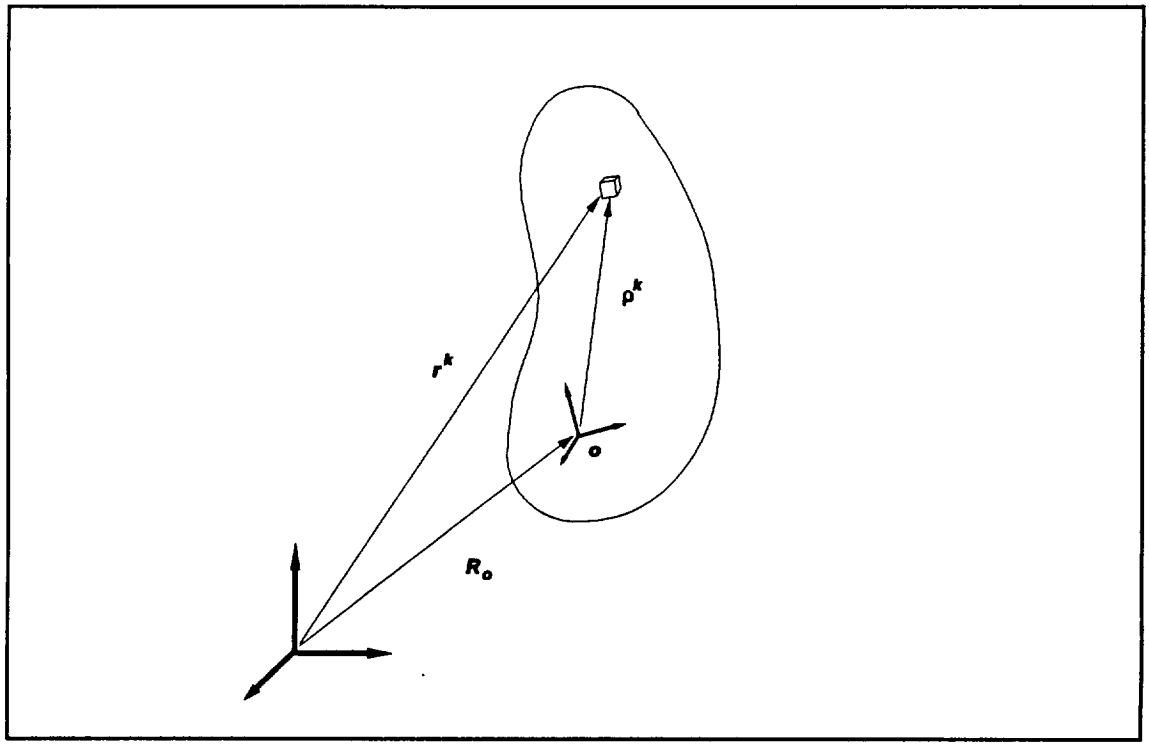


Figure 1. Singe Body Position Vectors.

Equations (3) and (4) can be manipulated to give the following forms of the inertial

forces and moments acting on node k.

$$f_I^k = -m^k \begin{bmatrix} 1 & 0 & 0 & 0 & \rho_z & -\rho_y & 0 & -\rho_x & -\rho_x & \rho_y & \rho_z & 0 \\ 0 & 1 & 0 & -\rho_z & 0 & \rho_x & -\rho_y & 0 & -\rho_y & \rho_x & 0 & \rho_z \\ 0 & 0 & 1 & \rho_y & -\rho_x & 0 & -\rho_z & -\rho_z & 0 & 0 & \rho_x & \rho_y \end{bmatrix} \begin{pmatrix} a_x^o \\ a_y^o \\ a_z^o \\ \alpha_x \\ \alpha_y \\ \alpha_z \\ \omega_x^2 \\ \omega_y^2 \\ \omega_z^2 \\ \omega_x \omega_y \\ \omega_x \omega_z \\ \omega \ \omega \end{pmatrix} \tag{5}$$

and

(6)

Several things should be noted about the form of the inertial loads given in Eqs.(5) and (6). First of all, the above equations are restricted to the lumped mass case (i.e., we have relied upon the fact that the mass and mass moment of inertia for nodal body k is explicitly known). This restriction shall be overcome shortly. The most important thing to note about Eqs.(5) and (6) is that the 3×12 coefficient matrix is a function of only the body's geometry and inertial properties. All the required kinematic information necessary to compute the inertial loads is isolated into the trailing 12×1 vector.

It can be shown [1] that when the body under consideration has a distributed mass matrix representation, the inertial forces and moments acting on a particular node can be

written as

$$f_I^k = -\left[m_T^k \Phi_{GRB} \; Y_T^k(\hat{i},\hat{i}) \; Y_T^k(\hat{j},\hat{j}) \; Y_T^k(\hat{k},\hat{k}) \; Y_{T1}^k(\hat{i},\hat{j}) \; Y_{T1}^k(\hat{i},\hat{k}) \; Y_{T1}^k(\hat{j},\hat{k}) \right] \begin{pmatrix} a_x^o \\ a_y^o \\ a_z^o \\ \alpha_x \\ \alpha_y \\ \alpha_z \\ \omega_x^2 \\ \omega_y^2 \\ \omega_z^2 \\ \omega_x \omega_y \\ \omega_x \omega_z \\ \omega_y \omega_z \end{pmatrix} \tag{7}$$

and

$$t_I^k = -\left[m_R^k \Phi_{GRB} \; Y_R^k(\hat{i},\hat{i}) \; Y_R^k(\hat{j},\hat{j}) \; Y_R^k(\hat{k},\hat{k}) \; Y_{R1}^k(\hat{i},\hat{j}) \; Y_{R1}^k(\hat{i},\hat{k}) \; Y_{R1}^k(\hat{j},\hat{k}) \right] \begin{pmatrix} a_x^o \\ a_y^o \\ a_z^o \\ \alpha_x \\ \alpha_y \\ \alpha_z \\ \omega_x^2 \\ \omega_y^2 \\ \omega_z^2 \\ \omega_x \omega_y \\ \omega_x \omega_z \\ \omega_y \omega_z \end{pmatrix} \tag{8}$$

where Φ_{GRB} is the matrix of "geometric rigid body modes" and the following definitions are employed:

$$Y_T^k(s,r) \triangleq s \times m_T^k \Phi_{GRB} \begin{pmatrix} 0 \\ r \end{pmatrix} \tag{9}$$

and

$$Y_{T1}^k(s,r) \triangleq Y_T^k(s,r) + Y_T^k(r,s) \tag{10}$$

The $Y_R^k(s,r)$ are $Y_{R1}^k(s,r)$ terms are defined analogously.

The form of Eqs.(7) and (8) appear somewhat unwieldy, but in fact the procedures followed to compute the inertial loads are straightforward. It should also be noted that Eqs.(7) and (8), which are sufficiently general to address the distributed mass case, collapse to the simpler forms of Eqs.(5) and (6) when a lumped mass representation is assumed.

For brevity, Eqs.(7) and (8) can be stacked and written as

$$f_{inert}^k = \begin{pmatrix} f_I^k \\ t_I^k \end{pmatrix} = F_I^k \cdot A_I \tag{11}$$

where f_{inert}^k is a 6×1 vector, F_I^k is a 6×12 coefficient matrix, and A_I is the 12×1 vector of kinematic variables. The above equation can be generalized to compute the inertial load vectors for the entire body (i.e., all nodes, not just node k) by essentially "stacking up" all of the nodal inertial loads. This process leads to the following equation,

$$\underset{n\times 1}{f_{inert}} = \underset{n\times 12}{F_I} \cdot \underset{12\times 1}{A_I} \tag{12}$$

where f_{inert} is the n×1 vector of inertial loads acting on the body, F_I is a n×12 coefficient matrix, and A_I is the usual 12×1 vector of kinematic variables. At this point, it is useful to think of the F_I matrix as simply the container for 12 n×1 inertial load vectors—each associated with a particular kinematic quantity contained in the A_I vector.

We are now in a position to expand the geometric stiffness matrix into a form that will be appropriate for introduction into the dynamic simulation. Since the K_σ matrix is function of the loads acting on the body (inertial and external), it can be expanded into the following form

$$K_\sigma = K_\sigma(f_{inert} + f_{ext}) = K_\sigma(f_{inert}) + K_\sigma(f_{ext}) \tag{13}$$

By substituting Eq.(12) into Eq.(13), the geometric stiffness matrix that accounts for the inertial loading (i.e., the "motion stiffness") can be written as

$$K_\sigma(f_{inert}) = K_\sigma\left(\sum_{I=1}^{12} F_I(i)\,A_I(i)\right) = \sum_{I=1}^{12} K_\sigma(F_I(i))\,A_I \tag{14}$$

where the $F_I(i)$ vectors are simply columns of the F_I matrix. Note that there are 12 geometric stiffness matrices associated with the inertial loads, namely $K_G(F_I(i))$, $I = 1,12$.

The geometric stiffening due to external loading can be expanded in a manner similar to the inertial loading, i.e.,

$$K_G(f_{ext}) = K_G\left(\sum_{I=1}^{ne} F_{ext}(i) \cdot f_{ext}(i)\right) = \sum_{I=1}^{ne} K_G(F_{ext}(i)) \cdot f_{ext}(i) \qquad (15)$$

The external loads in the above equation have been expanded into *ne* "locator" vectors ($F_{ext}(i)$) and associated magnitudes ($f_{ext}(i)$). It should be noted that there are *ne* geometric stiffness matrices, each associated with one of the *ne* external loads. For the multibody case, all inter-body forces acting through joints or hinges are considered externally applied forces as far as the geometric stiffening is concerned.

By expanding the K_G matrix into the forms represented by Eqs.(14) and (15), the time-varying nature of K_G is isolated to the $A_I(i) = A_I(i,t)$ and $f_{ext}(i) = f_{ext}(i,t)$ terms. This allows the 12 $K_G(F_I(i))$ and *ne* $K_G(F_{ext}(i))$ matrices to be computed in advance of any simulation effort. Any finite element code capable of producing geometric stiffness matrices should be able to provide the required data.

As is typical in the consideration of flexibility of multibody systems, we will consider the flexibility of any particular body to be expressed as a linear combination of shape vectors (e.g., modeshapes). Hence, a need arises for the calculation of generalized or modal stiffness matrices. For TREETOPS, the linear modal stiffness matrix is input through the <...>.flx file. The modal geometric stiffness terms, which are defined below, are read into the user controller wherein they are multiplied by the appropriate modal coordinates. Elements of the modal geometric stiffness matrices are defined as:

$$K^I_{G1\_qj} \triangleq \phi_q^T K_G(F_I(i))\,\phi_j \qquad (16)$$

and

$$K^I_{G2\_qj} \triangleq \phi_q^T K_G(F_{ext}(i))\,\phi_j \qquad (17)$$

where Φ_j is the j^{th} modeshape vector. The modal force associated with the geometric stiffening terms can now be written as

$$\mathcal{F} = \left[\sum_{I=1}^{12} K'_{G1} \cdot A_I(i,t) + \sum_{I=1}^{ne} K'_{G2} \cdot f_{ext}(i,t) \right] \eta \qquad (18)$$

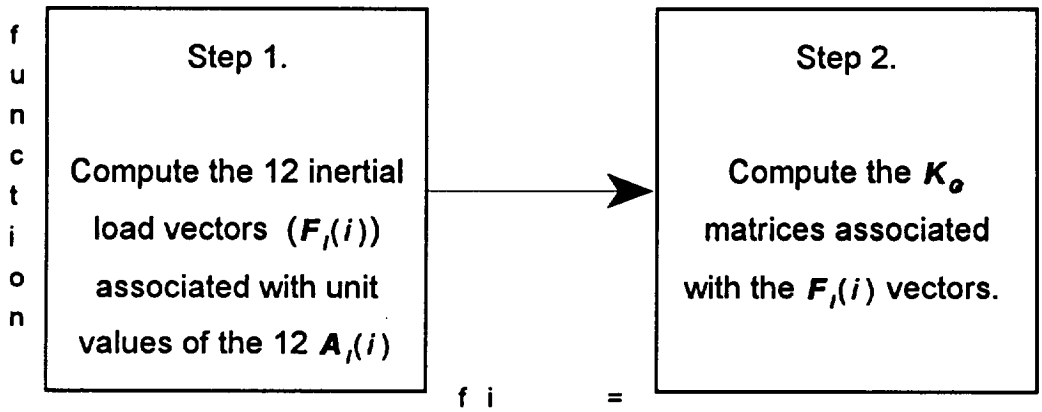
where η is the vector of modal coordinates. The calculation of the modal force terms given in Eq.(18) is performed in the user controller. This will be demonstrated in the example problems presented later in this report.

4.1.3 General Implementation Discussion.

As a prerequisite to actually incorporating the geometrically nonlinear capabilities into TREETOPS for a particular dynamic system, the analyst should go through a mental screening process to determine the necessity of the nonlinear analysis. The essence of this screening test is the question, "Is a geometrically nonlinear response likely from this system?" In general, this question is very difficult to answer. For relatively stiff structures, geometrically nonlinear responses are uncommon. However, there are several situations in which geometric stiffening may play a significant role in the system response. For rotating structures, if the spin rate approaches (or exceeds) the first natural frequency of the structure, then the geometric stiffness terms are necessary in the analysis. For structures constructed from beams and plates, compressive loading in the body can significantly modify the stiffness of the structure. When the user finds himself in the unfortunate position of not knowing whether or not to expect a geometrically nonlinear response, the inclusion of the nonlinear stiffening terms is the safest course to follow despite the not insigficant effort involved.

The initial step in building the geometrically nonlinear TREETOPS model is to decide which of the bodies in the system require the geometric stiffening terms. Consider a satellite/antenna system during a slewing manuever. Since the core body (satellite) is likely to be much stiffer than the antenna, only the antenna body would be considered likely to produce a geometrically nonlinear response. Hence, the geometric stiffening terms utilized in Eq.(18) need only be computed for the antenna body, not the core body.

The overall procedure for integrating the geometric stiffening terms into TREETOPS is presented in Figure 2. For each body requiring geometric stiffening terms, the first step in the analysis is to compute the $F_I(i)$ inertial load vectors. A Matlab m-file (fimake5.m) has been created which will produce these inertial load vectors. The initial portion of this Matlab function is shown below.




```
f i          =
fimake5(idof_
tab,nnode,phir,mass)
        % function fi = fimake5(idof_tab,nnode,phir,mass)
        % this  m file function constructs the fi array (ndof X 12) from the
        % idof_tab , mass matrix ,
        % and also the phir matrix.
```

The inputs to the function are an *idof_tab* array, which is an ndof × 2 array that maps each degree of freedom of the FEM model. The *idof_tab* contains, for each DOF, the associated node number and coordinate direction (x = 1, y = 2, ... , θ_z = 6). For example, if the jth DOF in the FEM model represents the y direction deflection of the 205th node then *idof_tab*(j,1) = 205 and *idof_tab*(j,2) = 2. The scalar *nnode* is simply the highest node number in the model. Typically this is also the number of nodes in the model. The *phir* input to the fimake5 function is the ndof × 6 array of geometric rigid body modes. *Mass* is the ndof × ndof mass matrix.

The next step in the analysis is to compute the individual $K_g(F_l(i))$ and $K_g(F_{ext}(i))$ matrices. Note that there will be *ne* $K_g(F_{ext}(i))$ matrices, and up to 12 of the $K_g(F_l(i))$ matrices that need to be computed. A typical procedure for obtaining these geometric stiffness matrices is to apply (in the FEM model) the $F_{ext}(i)$ or $F_l(i)$ load vectors to the structure and then let the FEM compute the resulting K_g matrix. In MSC NASTRAN, a buckling solution or a nonlinear static solution (SOL66) can be run which forces the code to assemble and store the K_g matrix in the model database. If one chooses to run SOL66, then the following DMAP alter can be added to a SOL63 restart run to obtain the K_g matrix. Note that the DMAP given here is valid for MSC Version 65, and will probably need adjustments to be valid in other releases. (Note: MSC's technical support actually provided the DMAP, and will surely have the updated version available upon request.)

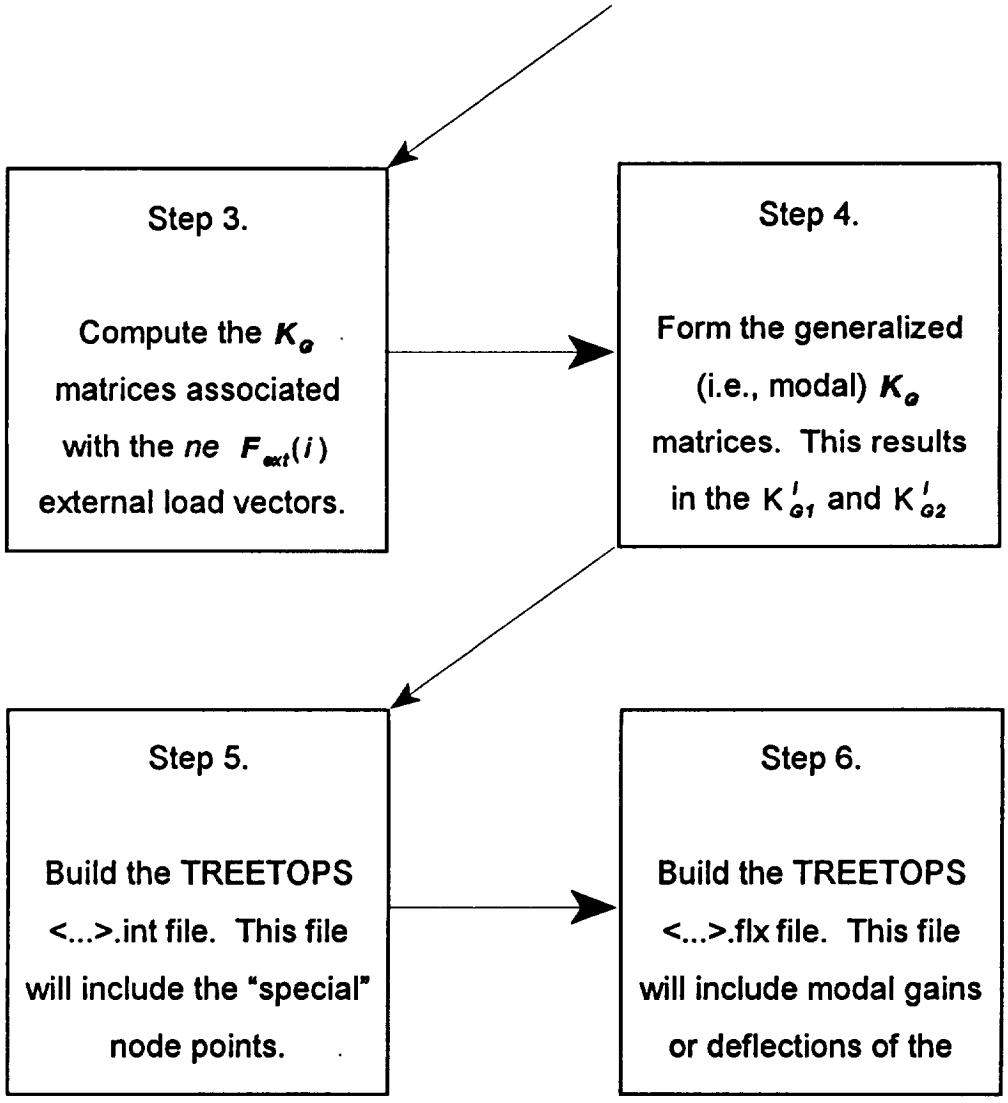


Figure 2.

Overview of the Geometric Stiffening Data Preparation.

```
$ Restart from SOL 66 run
READ 9
ALTER 287 $
$ Insert DMAP Alters
COND     LNSTIF, ACON $ SKIP THE ALTERS FOR SUPERELEMENTS
PARAM    //C,N,LT/V,N,NOLOOP/V,Y,LOOPID=0/1 $ READ LOOPID CARD
COND     LNSTIF, NOLOOP $ NO LOOPID, GO TO LINEAR STIFFNESS
DBFETCH  /UGV,ESTNL,,,/SOLID/LOOPID//DBSET3  $ DATA FROM SOL 66 RUN
PARAML   ESTNL//C,N,PRES////V,N,NONLK $ CHECK PRESENCE OF ESTNL
COND     LNSTIF, NONLK $ NO ESTNL, GO TO LINEAR STIFFNESS
TA1,     MPT,ECTS,EPT,BGPDTS,SILS,ETT,CSTMS,DIT/
         EST,DESTNL,GEI,GPECT,ESTL/V,N,LUSETS/S,N,NOESTL/
         S,N,NP/2/S,N,NOGENL/SEID/S,Y,LGDISP=-1/
         V,N,NLAYERS=5  $ GENERATE TABLE FOR LINEAR ELEMENT
COND     JMPKGG,NOKGGX $
EMG      ESTL,CSTMS,MPT,DIT,GEOM2S,,,/KELM,KDICT,,,,/
         S,N,NOKGXX/0/0/0/////////////K6ROT  $ STIFFNESS FOR LINEAR ELEMENTS
EMA      GPECT,KDICT,KELM,BGPDTS,SILS,CSTMS/KBJJZ,/  $
EMG      ESTL,CSTMS,MPT,DIT,,UGV,ETT,EDT/KDLEL,KDLDI,,,,/
         S,Y,NOD=1/0/0//NP  $ DIFFERENTIAL STIFFNESS FOR LINEAR ELEMENTS
EMA      GPECT,KDLDI,KDLEL,BGPDTS,SILS,CSTMS/KDLGG,/-1/$
ADD      KBJJZ,KDLGG/KLTOT/ $
EMG      ESTNL,CSTMS,MPT,DIT,GEOM2S,,,/KELMNL,KDICTNL,,,,/
         1/0/0/////////////V,Y,K6ROT  $ STIFFNESS FOR NONLINEAR ELEMENTS
EMA      GPECT,KDICTNL,KELMNL,BGPDTS,SILS,CSTMS/KBJJZNL,/$
ADD      KLTOT,KBJJZNL/KB1/  $
EMG      ESTNL,CSTMS,MPT,DIT,,UGV,ETT,EDT/KDELM,KDDICT,,,,/
         1/0/0//NP/  $ DIFFERENTIAL STIFFNESS FOR NONLINEAR ELEMENTS
EMA      GPECT,KDDICT,KDELM,BGPDTS,SILS,CSTMS/KBDJJ,/-1/$
$
$  PRINT GEOMETRIC STIFFNESS MATRIX
MATPRT   KBDJJ// $
OUTPUT4  KBDJJ//-1/11/1  $
$
ADD      KB1,KBDJJ/KJJZ/  $
JUMP     JMPKGG $ SKIP  LINEAR STIFFNESS GENERATION
LABEL    LNSTIF $ LABEL FOR LINEAR STIFFNESS GENERATION
ENDALTER $
```

If NASTRAN is not employed, then the user should consult the User's Manual of the particular FEM package in question to determine the procedure for extracting the individual K_G matrices. Note that K_G also is known as the stress-stiffening matrix.

Before leaving the discussion of the generation of the K_G matrices, a brief comment on the construction of the $F_{ext}(i)$ vector is in order. As an example, let us assume that there is an externally applied load acting in the z direction of node number 10 in the FEM model of the body. In this case, $F_{ext}(i)$ would simply be an ndof × 1 vector of zeros, with the exception of a 1 in the row corresponding to the z direction of node 10. In general, the $F_{ext}(i)$ vector simply identifies (with the unit value 1) the location of the i^{th} external load.

It should be recognized that often it is not necessary to compute all 12 of the $\boldsymbol{K}_g(\boldsymbol{F}_l(i))$ matrices due to the natural constraints of the system. For example, if the problem at hand is a planar problem (e.g., x-y plane) then a_z^o, ω_x, ω_y, α_x, and α_y are all zero, as well as any term that contains them in the $\boldsymbol{A}_l$ vector. In the case of a planar problem, since only 4 of the $\boldsymbol{A}_l$ elements can be non-zero, only the 4 corresponding $\boldsymbol{K}_g(\boldsymbol{F}_l(i))$ matrices need be computed. Recognition of the system constraints reduces (in the planar case) the computation burden by 66%.

Step 4 in Figure 2 is carried out by simply pre- and post-multiplying the individual $\boldsymbol{K}_g$ matrices by the matrix of modal vectors. In other words, the modal geometric stiffness matrices are simply computed as indicated in Eqs.(16) and (17). The modal geometric stiffness matrices should be saved to disk since they will be required in the user controller employed by the TREETOPS model.

The next step in the modeling process of geometrically nonlinear problems is to build the <...>.int file. Only a few additions to the normal <...>.int file have to be made to accommodate the geometric stiffening procedure. The first is that nmode+1 "special" nodes must be added to the body, where nmode is the number of flexible modes used to model the flexibility of the body. A "jet" actuator must be located at each special node. The modal deflections of the special nodes (contained in the <...>.flx file) are designed such that the first special node can observe (and effect) only the rigid body modes, the second special node can observe only the 1st flexible mode—and no others, and so on. This concept is illustrated in the example problems. A set of translational accelerometers, rate gyros, and rotational accelerometers are attached to the first special node to measure the rigid body accelerations and angular rates of the body. A Matlab function called modal1.m has been produced which will aid in the development of the <...>.flx file for the geometrically nonlinear problem. Provisions have been made in this function to incorporate the special nodes into the model.

Before presenting the example problems, a brief discussion of the "user controller" which actually computes the geometrically nonlinear terms for the body is in order. Essentially, the purpose of the user controller is to carry out the operations indicated in Eq.(18). In order to do this, the user controller must be implemented to accept the a_o, α, and ω sensors as inputs and drive the "jet" modal actuators with the geometric modal forces as outputs. Internally, the a_o, α, and ω can be combined into the required $\boldsymbol{A}_l$ components inside the user controller. For example, the input to the user controller from a sensor may be ω_z, but the $\boldsymbol{A}_l(9) = \omega_z^2$. These ideas are consolidated in the following

example problems.

4.1.4 Example Problems

The following examples of geometrically stiffened problems illustrate the techniques discussed in this document. All supporting models and software are included in the accompanying disk.

4.1.4.1 Beam Spin-Up.

This problem concerns itself with computing the response of a planar beam being smoothly spun up about a pinned hub. This is illustrated in Figure 3. The beam hub is constrained to follow the profile given below.

$$\theta_s = \begin{cases} \frac{\omega_s}{T_s}\left[\frac{1}{2}t^2 + \left(\frac{T_s}{2\pi}\right)^2\left(\cos\left(\frac{2\pi t}{T_s}\right) - 1\right)\right] & , \quad t < T_s \\ \omega_s\left(t - \frac{T_s}{2}\right) & , \quad t \geq T_s \end{cases} \tag{19}$$

where the maximum spin rate is $\omega_s = 4$ rad/sec and the ramp time is $T_s = 15$ seconds.

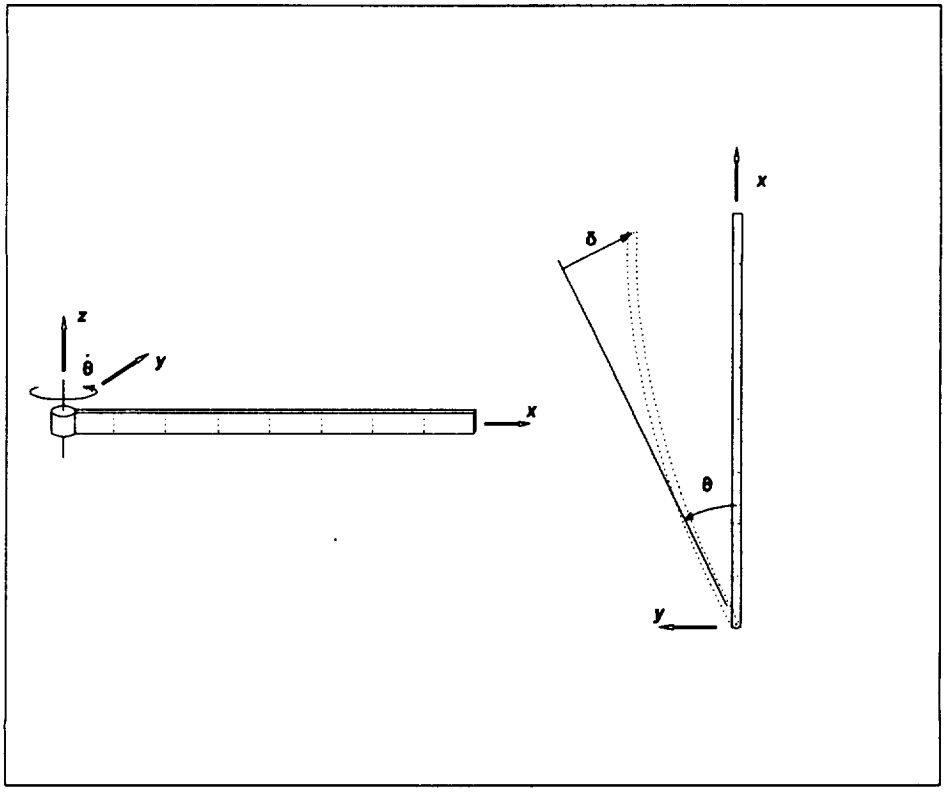


Figure 3. Beam Spin-Up

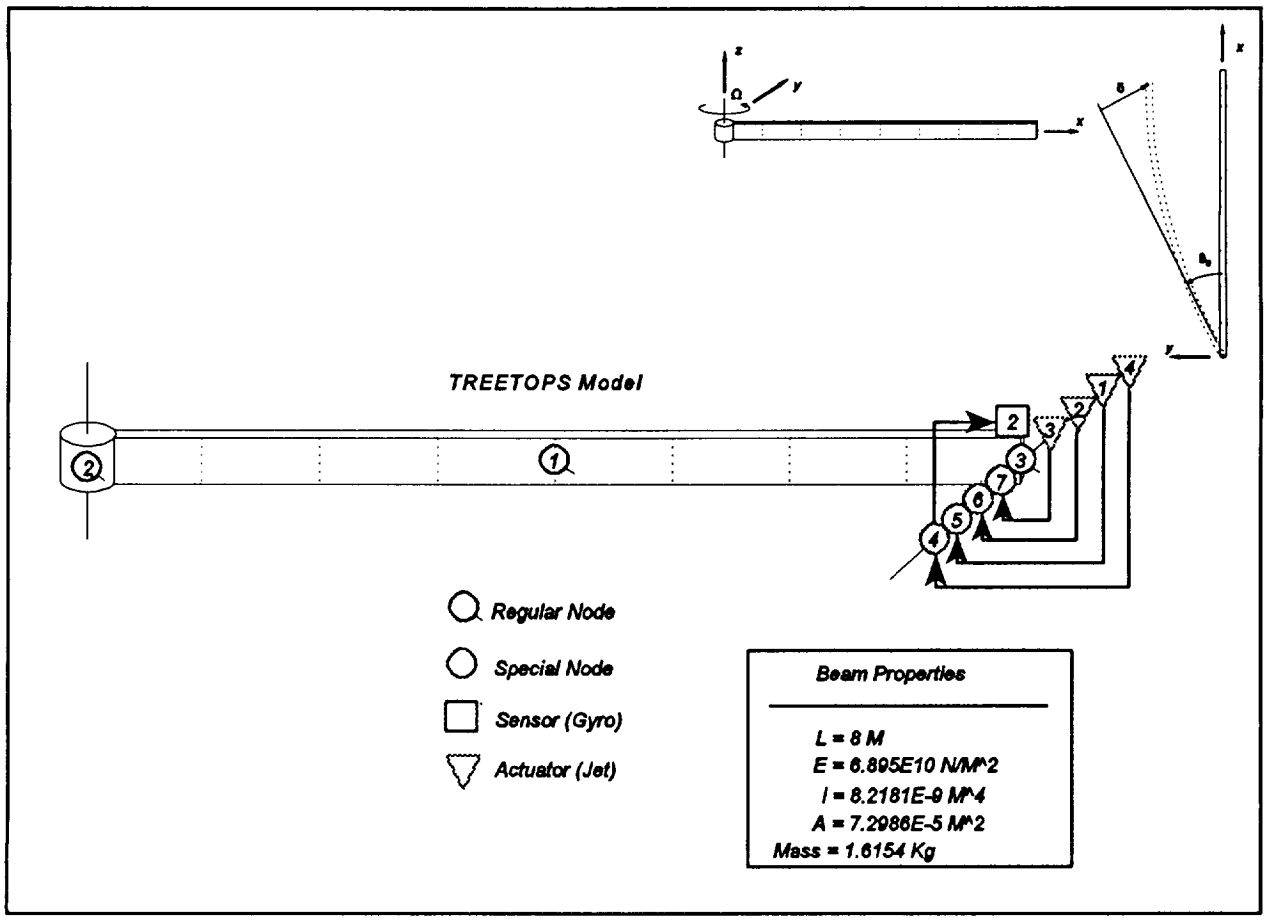


Figure 4. TREETOPS Model of Spin-Up Beam.

Figure 4 illustrates the TREETOPS model. Since the flexibility of the beam was modeled by 3 modes, there are a total of 4 special nodes (nodes 4-7). The modal gains associated with the special nodes is given in the following table:

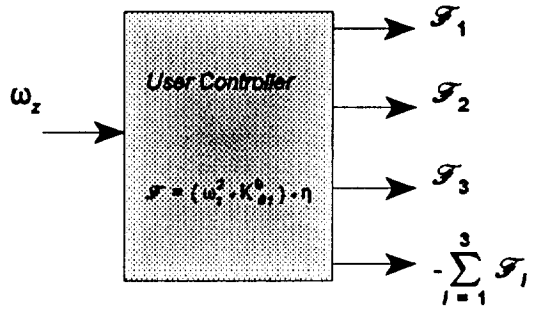
Translational Modal Gains for Special Nodes									
Node No.	Mode 1			Mode 2			Mode 3		
	δx	δy	δz	δx	δy	δz	δx	δy	δz
SN4	0	0	0	0	0	0	0	0	0
SN5	1	0	0	0	0	0	0	0	0
SN6	0	0	0	1	0	0	0	0	0
SN7	0	0	0	0	0	0	1	0	0

All rotational modal deflections (gains) are zero for the special nodes. From the table it can be seen that special node 4 (SN4) is capable of only sensing rigid body motions since it has zero gains in each of the flexible modeshapes. SN5 can sense (and effect) motions associated with mode 1 because of the unit δx gain in this modeshape. However, SN5 cannot sense modes 2 and 3. The other special nodes have similar modal sensing and effecting capabilities.

At this point, the reason for "designing" the modal gains associated with the special points in the way illustrated by the previous table can now be made apparent. Since the user controller produces the modal forces associated with the geometric stiffening terms, the model needs to have a node (i.e., a *special* node) in which only the first mode is effected by any load applied to this node. Similarly, the model should possess nodes in which only the second (or third) mode is effected (driven) by an applied load. The reasons that the first special node has no flexible modal gains is two-fold. First, the motion stiffening terms need to have access to the rigid body accelerations and angular rates, thus this special node is the ideal location to place the appropriate sensors. Secondly, a node is required which can be used to drive only the rigid body motions of the structure. This is necessary because a force acting on SN5 drives both mode 1 *and* the rigid body modes. Since the geometric stiffening terms should not effect the rigid body motions, the rigid body modes must be *un*driven. This is accomplished by simply applying the negative of the load applied to the SN5 node to the SN4 node. Similarly, the negative of the SN6 and SN7 applied geometric stiffening loads is also applied to the SN4 node.

For the spinning beam problem, if Node 2 is considered the origin for the calculation of the $\boldsymbol{F}_I$ vectors, then the only modal geometric stiffness matrix that is required is $\boldsymbol{K}^{\mathbf{0}}_{G1}$, which is associated with the $\boldsymbol{A}_I(9) = \omega_z^2$ term. The α_z term is not considered is due to the fact that the angular acceleration does not induce axial loads in the beam—hence the geometric stiffness matrix associated with α_z is zero.

The user controller subroutine, graphically illustrated below, is contained in its entirety in the accompanying software.



As a final note, it is important that the user include all five of the modal integrals that TREETOPS accepts in the <...>.flx file. These terms play an especially significant role in the system response when geometric stiffening is being considered.

The effect of including the geometric stiffness terms in the analysis of the beam spin-up problem is dramatically illustrated in the following plot.

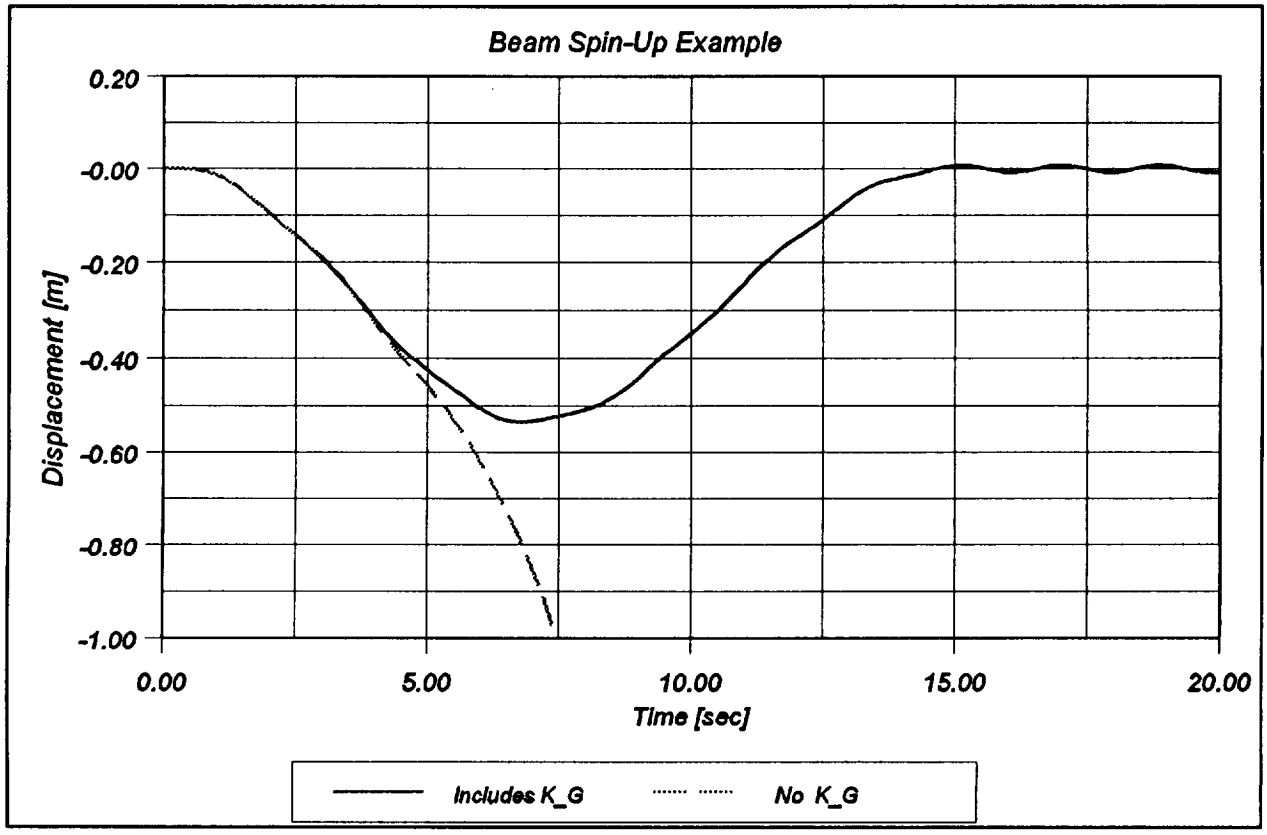


Figure 5. Effect of Geometric Stiffening on Tip Response.

The solid curve in Figure 5 represents the case in which the K_g terms are included in the TREETOPS run. This curve has been shown to be correct [1]. The dashed curve indicates that neglecting the geometric stiffening terms in this analysis leads to an unbounded response—clearly an erroneous result. In fact, this erroneous result is what is expected from *all* multibody dynamics codes which have no accommodation for geometric stiffening.

4.1.4.2 Beam Inside Rotating Hub.

In this example, a flexible beam (which is identical to the beam used in the previous example) is cantilevered from the rim of a rotating rigid hub. The goal of this problem is to determine the critical spin rate, Ω_{cr}, for which the beam becomes unstable and ultimately buckles. The centrifugal effects of the spinning hub cause the compressive loads in the beam; however, the coriolis terms in the equations of motion play a role in determining Ω_{cr}. The analytical expression for the critical spin rate results in $\Omega_{cr} = 2.186$ rad/sec.

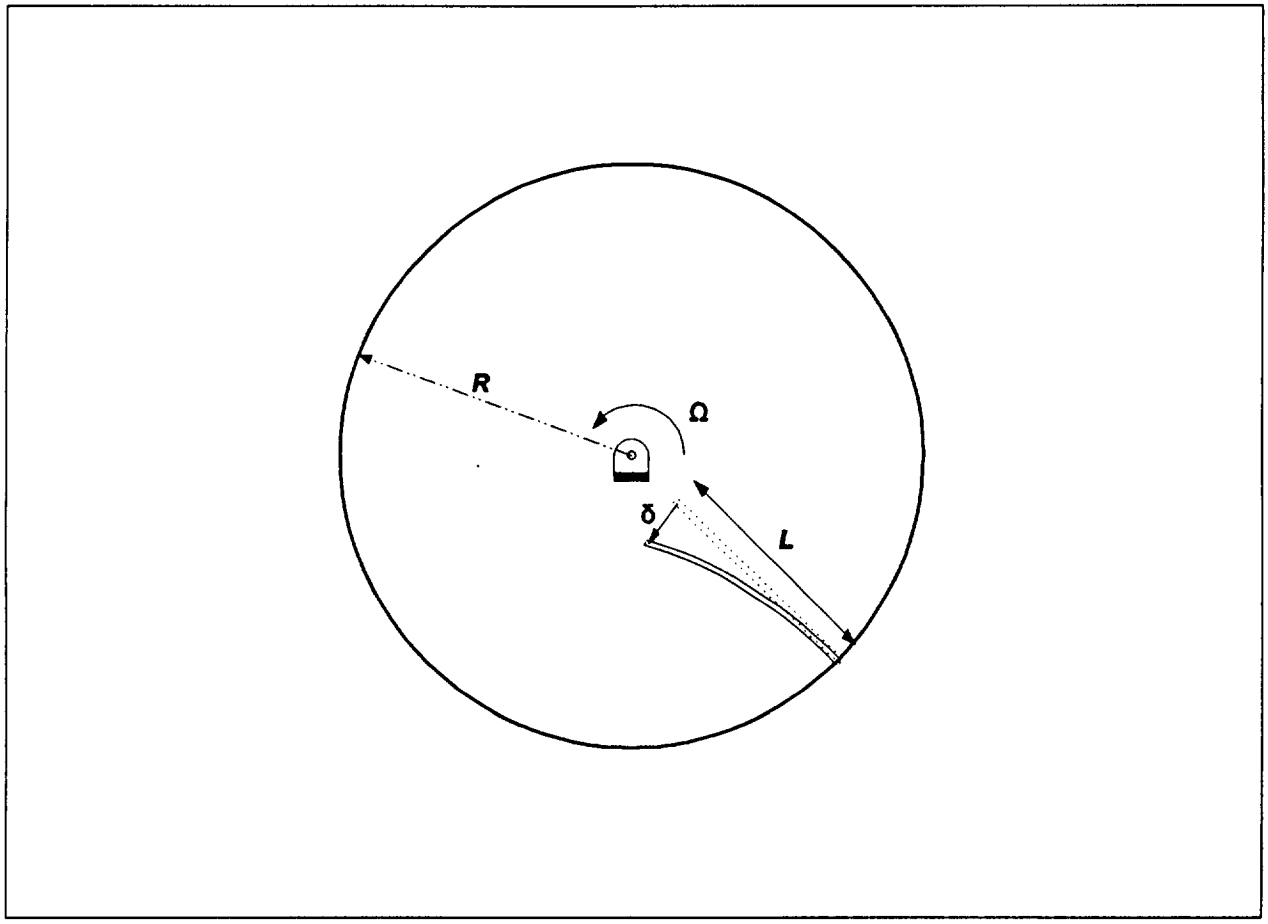


Figure 6. Beam Inside Rotating Hub.

Since this example is a two body problem, it is important to consider the inter-body forces as external loads on the beam for the purposes of computing the geometric stiffening terms. This is illustrated in the Figure 7, which is simply a diagram of the primary inter-body force, F_x.

For this example, two K_G matrices must therefore be computed. The first matrix must be computed to account for the rigid body spin rate of the hub (Ω), and the second is due to the F_x "contact" load between the two bodies.

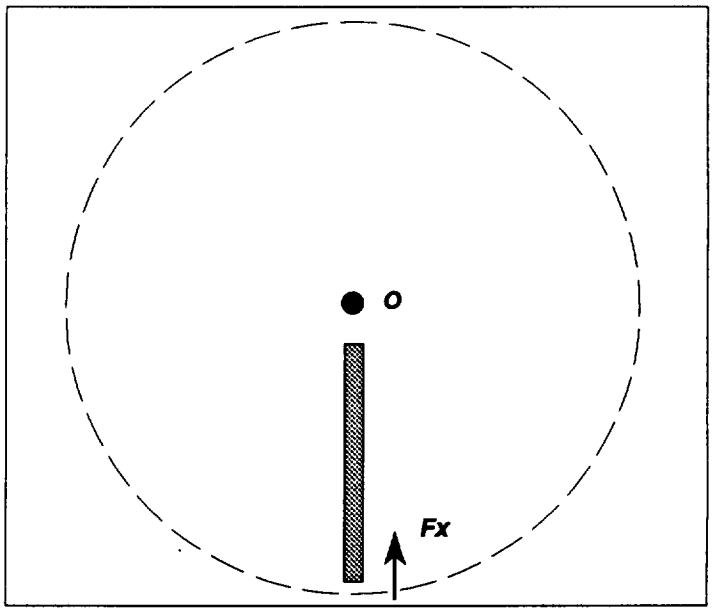


Figure 7. Primary Inter-Body Forces.

The user controller for this case must implement the following equation for the modal geometric stiffening forces -

$$\mathcal{F} = (\omega_z^2 \cdot K_{G1}^{\Omega} + F_x \cdot K_{G2}^{F_x}) \cdot \eta \qquad (20)$$

where the F_x term is contained in the FCONI array which is accessed in the user controller by adding the include file "..\inc\dbh.f" into the USCC.FOR file. It should also be noted that the contact forces in the FCONI array are expressed in the inertial coordinate frame and should therefore be converted to the Body 2 fixed frame.

Figure 8 is a plot of the tip response of the beam for two different steady-state Ω's. The geometric stiffening terms are included in both cases. As can been seen, the beam has an unbounded response at Ω = 2.19 rad/sec, however, it is stable at Ω = 2.18 rad/sec. This is in agreement with the theoretical result of Ω_{cr} = 2.186 rad/sec.

An interesting point to note is that there is no elastic response of the beam until t = 2 sec. This is due to the fact that a small "tap" is given to the beam at t = 2 sec. Without this "tap", no response occurs because there is no initial imperfection (or disturbance) to "push" the beam into the solution bifurcation. This behavior is typical when dynamic solution techniques are applied to buckling problems.

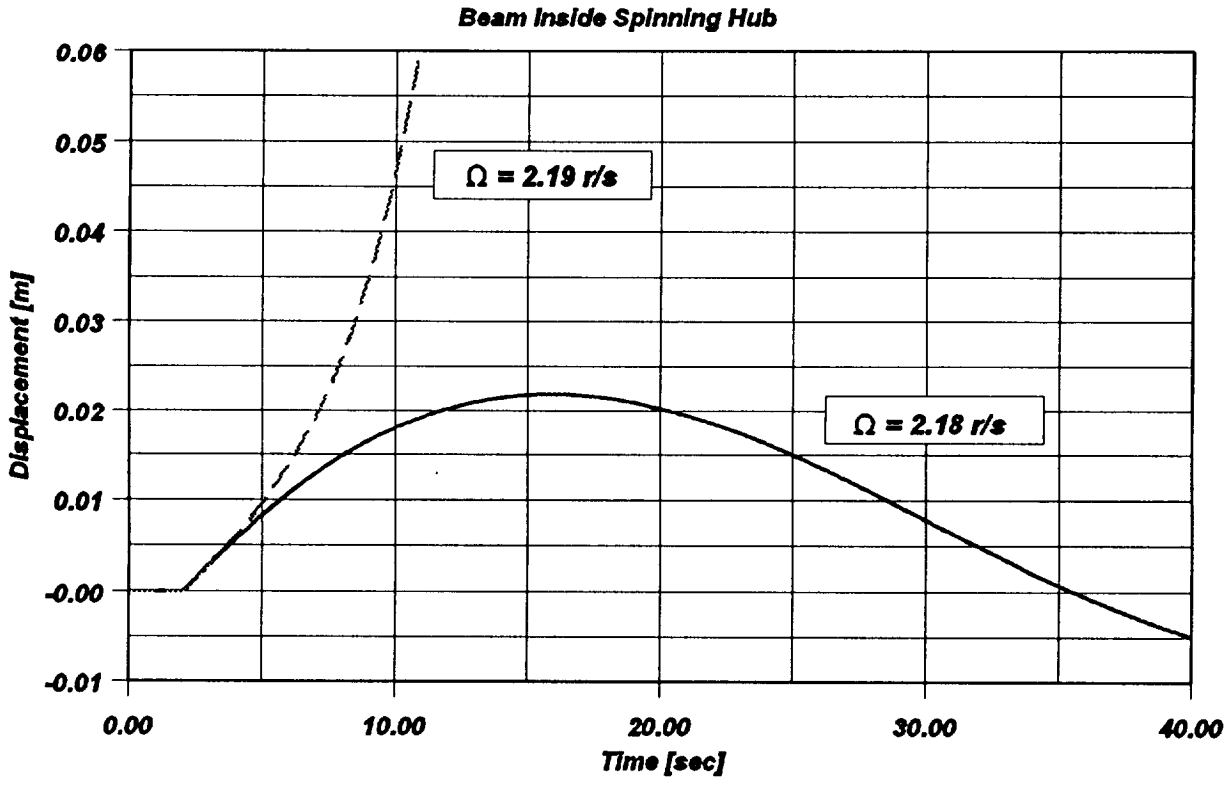


Figure 8. Tip Response of Beam.

4.1.4.3 Plate Buckling.

In this final example, the critical buckling load for a simply supported plate will be determined. The inertial loads do not play a role in the geometric stiffening terms; however, the running load across the top and bottom of the plate is obviously the prime driver for the calculation of the geometric stiffening terms.

Figure 9 illustrates the buckling problem to be examined in TREETOPS. The flexibility of the plate is modeled with 10 modes; therefore, 11 special nodes are required. These nodes are collocated at the center of the plate, and have the designations of SN20 through SN30. Node 32 is located slightly above the center of the plate, and is used for monitoring the plate deflection.

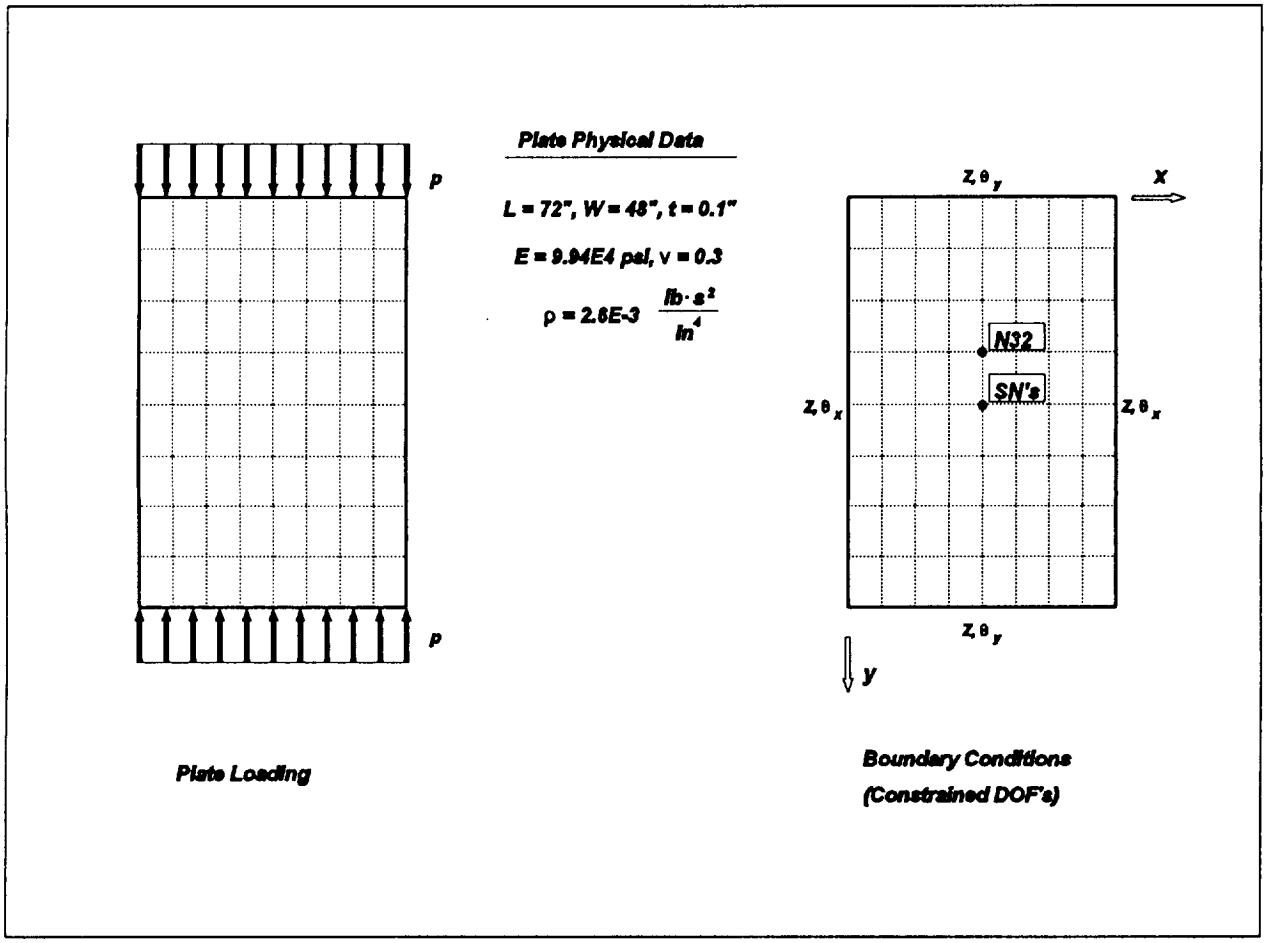
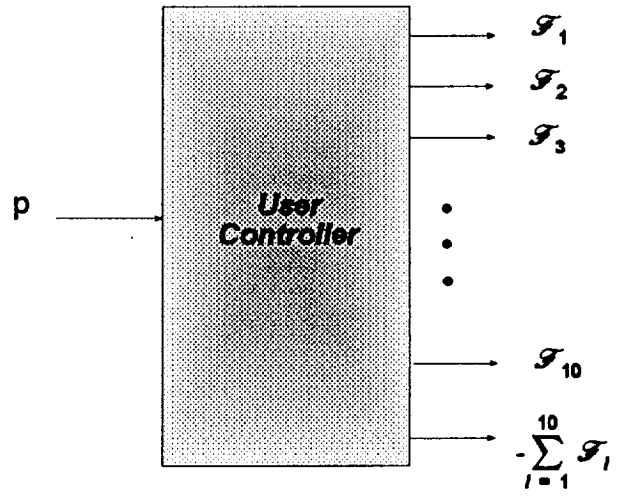


Figure 9. Plate Buckling Problem.

The table shown below contains the modal gains for the special nodes. The gains associated with all other directions are zero. As can be seen, SN20 is the "rigid body" node since it has no gain in any flexible mode. SN21 through SN30 are designed to effect modes 1 through 10, respectively.

Translational Modal Gains for Special Nodes										
Node No.	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7	Mode 8	Mode 9	Mode 10
	δx	δx	δx	δx	δx	δx	δx	δx	δx	δx
SN20	0	0	0	0	0	0	0	0	0	0
SN21	1	0	0	0	0	0	0	0	0	0
SN22	0	1	0	0	0	0	0	0	0	0
SN23	0	0	1	0	0	0	0	0	0	0
SN24	0	0	0	1	0	0	0	0	0	0
SN25	0	0	0	0	1	0	0	0	0	0
SN26	0	0	0	0	0	1	0	0	0	0
SN27	0	0	0	0	0	0	1	0	0	0
SN28	0	0	0	0	0	0	0	1	0	0
SN29	0	0	0	0	0	0	0	0	1	0
SN30	0	0	0	0	0	0	0	0	0	1

The user controller for this problem is straightforward. As indicated in the figure below, the only input to the controller is the running load, p. This load is generated by a Function Generator, and has a magnitude which is set in the <...>.int file. (The <...>.int is on the accompanying disk.) As is the usual case, the modal forces associated with the geometric stiffening term are the outputs of the user controller.



The user controller is responsible for computing the modal geometric stiffness terms according to the equation given below.

$$\mathcal{F} = (p \cdot K_{G2}^{p}) \cdot \eta \qquad (21)$$

Note that in this buckling problem, only a single geometric stiffness matrix is required—namely, the K_{G2} associated with a unit running load value (i.e., p = 1).

Figure 10 is a plot of the transverse deflection of N32 for two different values of p. The disturbance in both cases is a simple pulse (or "tap") delivered to N32 at t = 1 sec. As can be seen in the plot, the p = 0.170 lb/in case exhibits an unbounded response, whereas the p = 0.165 lb/in case is bounded. In other words, this TREETOPS study indicates that the critical buckling load is somewhere between p = 0.165 and 0.170 lb/in. The NASTRAN buckling solution produces a critical value of p = 0.167 lb/in for the plate model—which is supported by the TREETOPS solution.

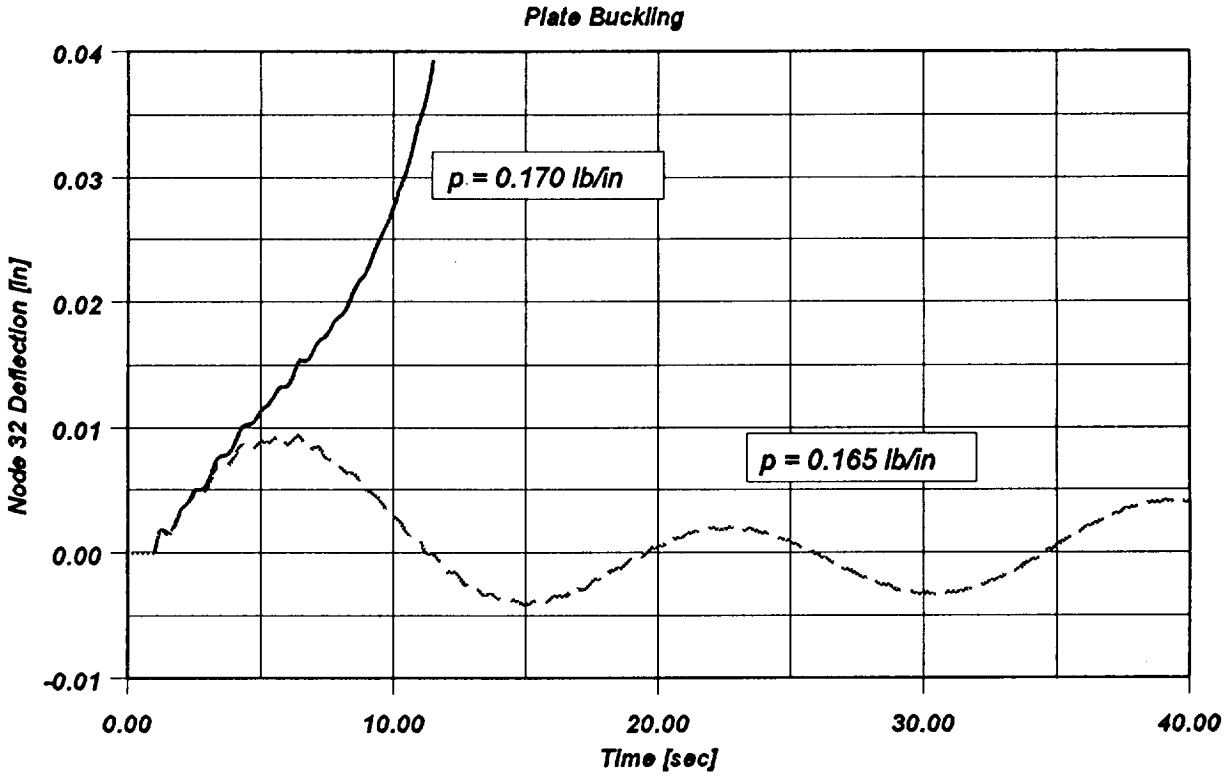


Figure 10. Response of Simply Supported Plate to Running Loads.

4.1.5 Summary.

This document has demonstrated the successful integration of geometric stiffening terms into the TREETOPS multibody dynamics package. The procedure has been developed in such a manner as to accommodate not only motion stiffening (due to inertial loads), but also geometric stiffening due to external loads acting on the bodies. This allows the procedure to be applied to multibody systems in which the inter-body forces are significant.

A set of tools have been developed (fimake5.m and modal1.m) which ease the computational burden placed upon the user when calculating the inertial load vectors and the <...>.flx files. However, obtaining the individual geometric stiffness matrices remains the user's responsibility. Fortunately, many of the popular FEM packages (e.g.,

NASTRAN, ANSYS, etc.) are capable of producing the required geometric stiffness matrices.

The current methodology for computing the modal geometric stiffening terms is carried out inside the "user controller" within TREETOPS. The inputs to the controller are the various rigid body accelerations and angular rate values, as well as the magnitudes of the external loads. The outputs of the controller are the modal forces associated with the geometric stiffness terms. The modal forces are applied through a series of "jet" actuators to a set of special nodes which possess modal gains "designed" in such a way as to make it possible to selectively drive the individual modes of the body in question.

4.2 Mass Integrals for Multibody Simulations

4.2.1 Introduction

This report will present a theoretical discussion on the calculation of mass or modal integrals common to the equations of motion of multibody systems. An algorithm and accompanying FORTRAN program will also be described to compute these terms from lumped or consistent mass matrices.

4.2.2 Derivation of Mass Integrals

Shown in Figure 11 is an undeformed generic flexible body. Let the continuous body be discretized into a system of p nodal bodies, with mass m_I and inertia I_I, and n total degrees of freedom; $\boldsymbol{R}$ is the position vector from the inertial frame O to the body fixed frame B; and ρ_I is the body fixed vector locating the undeformed position of the nodal body I relative to B. Body deformation is defined through degrees of freedom at the nodes. Each node may have up to six degrees of freedom. The nodal degrees of freedom are placed in a state vector $\boldsymbol{X}$.

$$X = \begin{bmatrix} DOF\ 1 \\ DOF\ 2 \\ DOF\ 3 \\ \vdots \\ DOF\ n \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ \theta_{x1} \\ \theta_{y1} \\ \theta_{z1} \\ \vdots \\ \theta_{zp} \end{bmatrix}$$

where x_1 is the x translational degree of freedom for node 1 and θ_{zp} is the z rotational degree of freedom for node p. A degree of freedom map, available from most finite element programs, will be used to characterize the model. This map denotes for each degree of freedom a direction label, one through seven, and an associated node number. The degree of freedom type or direction is as follows :

$$x = 1 \quad y = 2 \quad z = 3$$
$$\theta_x = 4 \quad \theta_y = 5 \quad \theta_z = 6$$

A value of 7 is used for generalized degrees of freedom in reduced or synthesized models.

Figure 12 is the deformed flexible body. The translational and rotational deformation vectors of nodal body I are δ_I and θ_I, respectively. These deformations are assumed to be small and compatible with linear strain-displacement relations.

The deformed nodal body position relative to the inertial frame O is

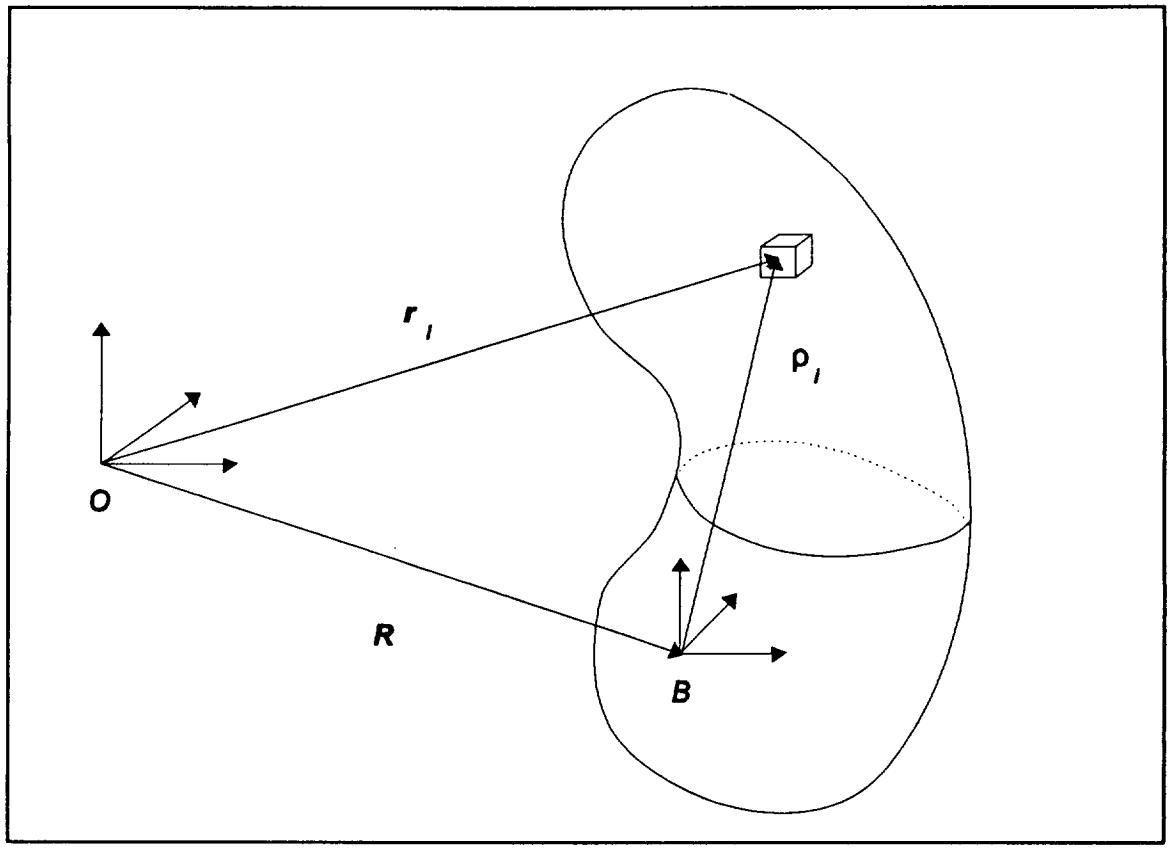


Figure 11: Undeformed Flexible Body

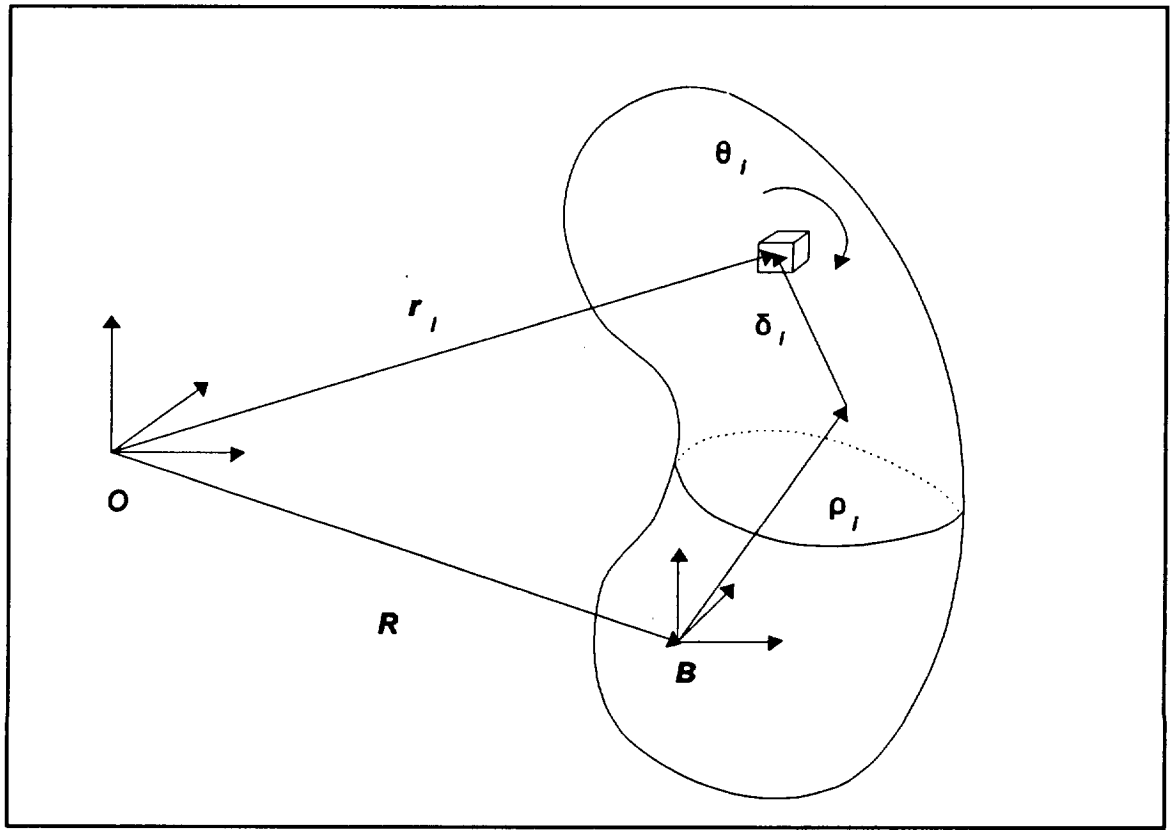


Figure 12: Deformed Flexible Body

$$r_I = R + \rho_I + \delta_I$$

The inertial translational velocity of nodal body I is

$$\dot{r}_I = \dot{R} + \omega \times (\rho_I + \delta_I) + \dot{\delta}_I$$

where ω is the angular velocity of the body fixed frame B with respect to O and $\dot{\delta}_I$ is the time rate of change of the deformation relative to B.

The kinetic energy of the system can be written as

$$\hat{T} = \sum_{I=1}^{p\ nodes} \frac{1}{2}\left[m_I \dot{r}_I \cdot \dot{r}_I + (\omega + \dot{\theta}_I)^T I_I (\omega + \dot{\theta}_I) \right]$$

The elastic deformations will now be described using the assumed modes technique. In this technique, the elastic deformations of the flexible body are approximated through a linear combination of the products of spatial shape functions and generalized time coordinates. The shape functions are typically a subset of normal modes derived from a discretized or finite element model of the component.

For this derivation, there are at most *n* shape functions for the *n* degree of freedom discretized structure. The translational and rotational deformations for nodal body I are written as

$$\delta_I = \sum_{J=1}^{n\ modes} \phi_{IJ} \eta_J$$

$$\theta_I = \sum_{J=1}^{n\ modes} \psi_{IJ} \eta_J$$

where ϕ_{Ij} and ψ_{Ij} are the translational and rotational 3x1 vectors for the j-th shape function evaluated at the location of nodal body I, and η_j is the associated generalized time coordinate. This is essentially a coordinate transformation from the discretized displacements to a set of modal coordinates. The model is reduced in size by truncating the number of modes or shape functions used in the coordinate transformation.

Let Φ be the matrix of eigenvectors of mode shapes. Each column of Φ corresponds to a mode and is associated with an eigenvalue. Each row of Φ corresponds to a model degree of freedom.

$$\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1n} \\ \psi_{11} & \psi_{12} & \cdots & \psi_{1n} \\ \vdots & \vdots & \cdots & \vdots \\ \phi_{p1} & \phi_{p2} & \cdots & \phi_{pn} \\ \psi_{p1} & \psi_{p2} & \cdots & \psi_{pn} \end{bmatrix}$$

The following mass integral terms are defined as

$$\Gamma_{0j} = \sum_{I=1}^{p\ nodes} m_I \phi_{Ij}$$

$$\Gamma_{1j} = \sum_{I=1}^{p\ nodes} \left(m_I \rho_I x \phi_{Ij} + I_I \psi_{Ij} \right)$$

$$I_{2jk} = \sum_{I=1}^{p\ nodes} m_I \left(\phi_{Ij}^T \phi_{Ik} 1 - \phi_{Ij} \phi_{Ik}^T \right)$$

$$I_{1j} = \sum_{i=1}^{p\ nodes} m_i \left(\rho_i^T \phi_{ij} 1 - \phi_{ij} \rho_i^T \right)$$

$$\Gamma_{2jk} = \sum_{i=1}^{p\ nodes} m_i \phi_{ij} x \phi_{ik}$$

where 1 is a 3x3 identity matrix.

The computation of these terms is straight forward given the modes, DOF map, nodal geometry, and mass / inertia distribution at each node. The mass / inertia distribution is readily available for lumped mass matrices. However, most finite element programs employ consistent mass matrices. Consistent mass matrices are not diagonal as are lumped mass matrices. The mass is usually distributed using the same shape functions used in the finite element stiffness matrix. Therefore, an algorithm is desired to compute the mass integral terms for a consistent mass matrix. This is best arrived at through momentum arguments.

The system linear and angular momentum vector can be written as

$$\boldsymbol{P} = \boldsymbol{M}\dot{\boldsymbol{X}}$$

where $\boldsymbol{M}$ is the mass matrix from the finite element model and $\dot{\boldsymbol{X}}$ is the time derivative of the nodal degrees of freedom or nodal velocities. From the mass integral definitions, the total linear momentum vector for the system is

$$P_T = \sum_{j=1}^{n\ modes} \Gamma_{0j} \eta_j$$

The total angular momentum vector for the system is

Now, the system momentum vector will be expressed in terms of modal coordinates.

$$P = M\dot{X} = M\Phi\eta$$

$$P_A = \sum_{I=1}^{p\ nodes} P_{AI} + \rho_I \times P_{TI}$$

The system momentum vector for the j'th mode is simply

$$P_J = M\Phi_J\eta_J = \begin{bmatrix} \vdots \\ P_{TIJ} \\ P_{AIJ} \\ \vdots \end{bmatrix}$$

where P_{TIJ} and P_{AIJ} are the system linear and angular momentum vectors for node I and mode j. The system momentum vector can be written as

$$P = \sum_{J=1}^{n\ modes} P_J$$

The total system linear momentum vector is therefore

$$P_T = \sum_{J=1}^{n\ modes} \sum_{I=1}^{p\ nodes} P_{TIJ} = \sum_{J=1}^{n\ modes} \Gamma_{0J}\eta_J$$

Now define the product of the mass matrix and the mode shape vector as

$$M\Phi_J = A_J$$

The system momentum vector from the j'th mode can then be expressed as

$$P_J = A_J\eta_J$$

where

$$A_j = \begin{bmatrix} A_{1j} \\ A_{2j} \\ \vdots \\ A_{pj} \end{bmatrix}$$

and

$$A_{ij} = \begin{bmatrix} A_{Tij} \\ A_{Aij} \end{bmatrix}$$

$A_{ij}\eta_j$ is the momentum vector at the i'th node from the j'th mode. The total linear momentum vector for the system is

$$P_T = \sum_{j=1}^{n\,modes} \sum_{i=1}^{p\,nodes} A_{Tij}\dot{\eta}_j$$

The first mass integral for the j-th mode is therefore seen to be

$$\Gamma_{0j} = \sum_{i=1}^{p\,nodes} A_{Tij}$$

From similar reasoning, the total angular momentum vector of the system is written as

$$P_A = \sum_{j=1}^{n\,modes} \sum_{i=1}^{p\,nodes} \left(P_{Aij} + \rho_i \times P_{Tij}\right)$$

$$P_A = \sum_{j=1}^{n\,modes} \sum_{i=1}^{p\,nodes} \left(A_{Aij} + \rho_i \times A_{Tij}\right)\dot{\eta}_j$$

The total angular momentum may also be written in terms of the second mass integral Γ_{1j} in modal coordinates as

$$P_A = \sum_{j=1}^{n\,modes} \Gamma_{1j}\,\eta_j$$

The second mass integral is therefore seen to be

$$\Gamma_{1j} = \sum_{i=1}^{p\,nodes} \left(A_{Aij} + \rho_i \times A_{Tij}\right)$$

The remaining mass integral can be expressed in terms of A_{Tij} as

$$\Gamma_{2_{jk}} = \sum_{i=1}^{p\,nodes} A_{Tij} \times \phi_{ik}$$

$$I_{1j} = \sum_{i=1}^{p\,nodes} m_i\left(\rho_i^T A_{Tij} 1 - A_{Tij}\,\rho_i^T\right)$$

$$I_{2_{jk}} = \sum_{i=1}^{p\,nodes} m_i\left(A_{Tij}^T \phi_{ik} 1 - A_{Tij}\,\phi_{ik}^T\right)$$

4.2.3 Software User's Guide

A FORTRAN program based on the equations presented in Section 2.0 has been written to compute the mass integrals for discretized flexible bodies and output the TREETOPS .FLX file. This program is composed of the following files : `massint8.for`, `input.for`, `flx_out.for`, `lib_flx.for`, `info.inc`, `inpt.inc`, `outpt.inc`. The main program is in the file `massint8.for`. It first calls the subroutine `input` to read the data, computes the generalized or modal matrices and mass integrals, and then call the subroutine `flx_out` to write the .FLX file. The subroutine `input` is in the file `input.for` and `flx_out` is in `flx_out.for`. `lib_flx.for` contains a library of matrix / vector manipulation routines. `info.inc`, `inpt.inc`, and `outpt.inc` are include files used to pass the necessary data between the various routines. `info.inc` contains the parameter statements for the maximum number of modes, nodes, and degrees of freedom.

Upon running the program (type `massint` depending on the compiler used), the user will be queried for the names of three files. The first file name is an ascii file containing information which describes the model size, degree of freedom map, discretized model node numbers, number of modes, and assorted output information. The contents of this file will be described in more detail in the next paragraph. The second file is also an ascii file which houses the discretized or finite element model mass, stiffness, and damping matrices. The matrices are read in `input.for` as follows :

```
      open(unit=1, file = file1, status = 'old')

cc  Read mass, stiffness, and damping matrices in physical
coordinates
      do i=1,ndof
        read(1,*) (mass_mat(i,j),j=1,ndof)
      enddo

      do i=1,ndof
        read(1,*) (stiff_mat(i,j),j=1,ndof)
      enddo

      if(idamp.eq.1) then
        do i=1,ndof
          read(1,*) (damp_mat(i,j),j=1,ndof)
        enddo
      endif

      do i = 1,ndof
        read(1,*) (phi(i,j),j=1,nmode_sys)
      enddo

      close(1)
```

The third file name is that of the output .FLX file. The name should be the root of the .INT file and include the .FLX suffix.

The model information file input data format is as follows :

```
cc  Read model size and FEM DOF map
      read(2,*) nnode,ndof,nmode_sys
      do i=1,ndof
         read(2,*) inod(i), idof_type(i)
      enddo

cc  Read nodal geometry matrix
      read(2,*) (r_offset(i),i=1,3)
      do i=1,nnode
        read(2,*) (node_mat(i,j),j=1,3)
      enddo
```

```
cc  Body id number
      read(2,*) bid

cc  Modal integral option number (0,1,2)
      read(2,*) modopt(1,bid)

cc  Body mass
      read(2,*) mass

cc  Damping matrix option
cc  1 = read damping matrix in physical coordinates
cc  2 = create damping matrix from diagonals of generalized
cc      mass and stiffness matrices
      read(2,*) idamp
      if(idamp .eq. 2) read(2,*) zeta

cc  Number of modes to output
      read(2,*) nmode_out

cc  Modal selection vector flag
cc  1 = output modes in sequnetial order
cc  2 = use selection array'
      read(2,*) nselect

        if(nselect.eq.1) then
         do i = 1,nmode_out
          nmdind(i) = i
         enddo
        else
         if(nmode_out.gt.nmode_sys)then
           write(6,*)'trying to write out more modes than you
                     read in!'
           write(6,*)'review input deck - this programs is
                     stopping'
           stop
         endif

cc  Read nmdout mode indices to be output
cc  These indices must be in ascending order
         do i = 1,nmode_out
           read(2,*) nmdind(i)
         enddo
        endif

cc   Read in number of nodes to be output from discretized
```

```
model
      read(2,*) node_out

cc  Read in desired node numbers for modal output - phi and
psi
      do i = 1,node_out
         read(2,*) inod_out(i)
      enddo

cc  Read in special node option for geometric stiffening
cc  1 = no special nodes for geometric stiffening
cc  2 = output gains for special nodes
      read(2,*) spec_node
```

The first line in this file reads in the number of nodes in the discretized model, `nnode`; the number of degrees of freedom in the discretized model, `ndof`; and the number of modes in the file computed from the discretized model, `nmode_sys`. The second set of data is the degree of freedom map. For each degree of freedom, the software reads in an associated node number, `inod(i)`, and a degree of freedom type, `idof_type(i)`. The next line of data is the offset vector, `r_offset(i)`. This vector allows the user to compute the mass integrals about a point different from the origin used to compute the coordinates for the discretized nodes. The offset vector is labeled ***R*** in Figure 11 and translates the origin of the nodal geometry from ***O*** to ***B***. Note that TREETOPS assumes that the mass integrals are computed about the origin of the body fixed frame used to describe the nodal geometry in the .INT file. The next set of data are the coordinates for the nodes of the discretized model, `node_mat(i,j)`. Each row corresponds to the x, y, z components of the position for node i. The body identification number, `bid`, is the next line of data. After this line follows the mode

option number (0,1,2), `modopt(1,bid)`. A value of 0 will direct the software to compute all mass integrals; a value of 1 will compute only the zero and first order terms; and a value of 2 will output only the zero order modal integrals. The body mass, `mass`, is input next to scale the alpha modal integrals. Following the mass is the damping matrix option flag `idamp`. A value of 1 directs the code to read in a physical damping matrix after the stiffness matrix. A value of 2 will enable the code to compute a diagonal generalized damping matrix from the product of twice the input value of zeta and the square root of the ratio of the generalized stiffness matrix diagonal and the generalized mass matrix diagonal. The next line of data is the value of zeta, `zeta`, if the value of `idamp` is 2. Otherwise, the next line is the number of modes to be output in the .FLX file, `nmode_out`. The user may select which modes to output through the modal selection vector. A value of 1 for `nselect` will direct the code to output the first `nmode_out` modes in the data file. A value of 2 will direct the code to read in the selection array `nmdind(i)` as an array of *ascending* output mode numbers, one per line. The next line of data is the number of discretized model nodes to be output, `node_out`. After this is the array of desired output node numbers, `inod_out`, in *ascending* order and one per line. Finally, the last line of data in the special nodes flag, `spec_node`, for use in incorporating geometric stiffness matrices into the model. A value of 1 for this flag will cause the code to output no special nodes. A value of 2 will direct the software to write out *nmode + 1* special nodes. The first special node is the rigid body node and will have zero modal gains. The remaining special nodes are given a modal gain of 1 for x translation for one mode each. The special nodes are appended to

the end of the *normal* nodes in the .INT file.

The user of this code may readily modify `input.for` for specific data bases. A set of example data files is supplied with this software to test it upon installation to a particular computer system. The model information file name is `test.inp`. The discretized data file name is `test.mat`. The output .FLX file is `test.flx` and matches the data computed using `modal1.m` in MATLAB.

5.0 Conclusions

This report has described the enhancements and additions to the TREETOPS suite of dynamic analysis tools by Oakwood College. The Software Design Document, inclusion of geometric stiffness into TREETOPS, as well as the development of user-friendly modal integral tools have been presented.

The Software Design Document provides a description of the argument lists and internal variables used in each subroutine of the TREETOPS suite. Geometrically nonlinear response problems can be addressed by incorporating a specially designed "User Controller" into the suite. Both Fortran and MATLAB forms of user-friendly modal integral tools were delivered under the provisions of contract NAS8-40194.

Additionally, a new set of User and Theory Manuals were prepared under this contract. These items, along with the others listed, should provide the inexperienced user more confidence in the use of the TREETOPS package. The Software Design Document is intended to allow the advanced user a more rapid and complete immersion into the code. This will no doubt ease the burden of future analysts who attempt to provide further enhancements to the TREETOPS suite of codes.

References

[1] Honeywell, "Users Manual for CONTOPS VAX Application," Clearwater, Florida, January, 1987

[2] Honeywell, "Software Design Document," Clearwater, Florida, 1985

[3] Dynacs Engineering Co., "TREETOPS Software Verification Manual," Florida, 1988

[4] Honeywell, "Users Manual for TREETOPS," Clearwater, Florida, June, 1984

[5] Dynacs Engineering Co., "TREETOPS Theory Manual," Florida, November 29, 1990

[6] Dynacs Engineering Co., "Geometric Nonlinearity in TREETOPS," NAS8-38092, Report Number : r 30-30, December 16, 1991

[7] Howsman, T. G., "Dynamic Analysis of a Class of Geometrically Nonlinear Multibody Systems," Ph.D. Dissertation, University of Alabama in Huntsville, 1993

List of Acronyms

NASTRAN	National Aeronautics and Space Administration Structural Analysis
DOS	Disk Operating System
MSC	Macneal-Schwendler Corporation
COTR	Contracting Officer's Technical Representative
FEM	Finite Element Model
DMAP	Direct Matrix Alter Program

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	23 Aug 96	Final Report, 28 Jul 94-31 Jan 96

4. TITLE AND SUBTITLE
Research of "TREETOPS Structural Dynamics Controls Simulation System Upgrade"

5. FUNDING NUMBERS
Contract #NAS8-40194

6. AUTHOR(S)
Dr. Rose M. Yates
Director of Grants and Contracts

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Oakwood College
Huntsville, AL 35896

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
NASA/MSFC
Marshall Space Flight Center, AL 35812

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Under the provisions of contract number NAS8-40194, which was entitled "TREETOPS Structural Dynamics and Controls Simulation System Upgrade", Oakwood College contracted to produce an upgrade to the existing TREETOPS suite of analysis tools. This suite includes the main simulation program, TREETOPS, two interactive preprocessors, TREESET and TREEFLX, an interactive post processor, TREEPLOT, and an adjunct program, TREESEL.

A "Software Design Document", which provides descriptions of the argument lists and internal variables for each subroutine in the TREETOPS suite, was established. Additionally, installation guides for both DOS and UNIX platforms were developed. Finally, updated User's Manuals, as well as a Theory Manual, were generated.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	58
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR