MODELLING, DYNAMICS ANALYSIS AND CONTROL OF A MULTI-BODY SPACE PLATFORM

by

Bruno Marco Quadrelli

SUBMITTED TO THE DEPARTMENT OF
AERONAUTICS AND ASTRONAUTICS IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1992

Signature of Author____
Department of Aeronautics and Astronautics
March 12, 1992

Certified by____
Professor Andreas H. von Flotow
Thesis Supervisor

Accepted by____
Professor Harold Y. Wachman
Chairman, Department Graduate Committee

ABSTRACT

The equations of motion are obtained of a flexible multibody space platform in zero-g (in the Space Shuttle's Middeck) and in one-g (in the testing laboratory). These equations are linear in nodal displacements and rotations and non-linear in torque wheel rates and in articulated body relative angles and rates. Two analyses are carried out on the linearized zero-g model. One yields an empirical estimate of the precession frequency when flexibility is included in the model as a result of the spinning rotors. The other allows us to conclude that the effect of the base-body flexibility on the inertial payload pointing angle is not very significant, which justifies the decoupling of the rigid and the elastic equations. The one-g model also includes the internal dynamics of the suspension device used for ground testing. For the one-g model, a good agreement is found between the numerically obtained transfer functions and the experimentally obtained ones. This agreement validates the use of this low order model as evaluation model for control design. Finally, a time simulation is implemented in order to achieve trajectory tracking for one of the payloads undergoing a slew maneuver, when the bus is being stabilized in inertial space. A feedback /feedforward approach has been used to design the control laws and it is found that the magnitude of the Coriolis and centripetal terms is very small, and that trajectory tracking can be succesfully achieved. Finally, the balanced reduction algorithm has been applied in order to reduce the model to a lower order one, for purposes of non-linear multi-body simulation. A reduction of as much as 90% in computation time can be obtained for a given sensor/actuator distribution. The results obtained confirm the validity of the approach for use in the non-linear time simulation of multi-body systems. However, the results obtained also stress the necessity of correctly updating the mode shapes during the non-linear simulation.

ACKNOWLEDGEMENTS

I feel very much indebted to prof. S. Bergamaschi, of the Universita' degli Studi di Padova (Italy), because he tethered me to the science of dynamics and to the beauty of Space, to Dr. E.Lorenzini, of the Harvard-Smithsonian Center for Astrophysics, because I enjoyed working with him and because with him I developed my interest in research and in multibody dynamics, to my friends of Aeritalia, now Alenia (Torino- Italy), for their patience, to prof. A. von Flotow because he allowed me to come to MIT to learn a lot more of dynamics , and to prof. E. Crawley because of his generosity in allowing me to continue my studies toward my degree and to work for MACE. I am also very indebted to my family , who have always been supporting me morally from Italy, and to Caroline, who has done the same from within the United States.

TABLE OF CONTENTS

Page

Abstract....1
Ackowledgements....2
Table of Contents....3
List of Figures....5
List of Tables....7

Chapter 1:
1.1 Introduction....8
1.2 Thesis Overview....11

Chapter 2: Derivation of the Model in zero-g....13
Introduction....13
2.1 General description of the MACE....13
2.1.1 List of Assumptions for the zero-g model....15
2.2 Derivation of the equations of motion of the articulated part....19
2.2.1 System Kinematics....19
2.2.2 System Dynamics....23
2.3 Derivation of the equations of motion of the central node....27
2.3.1 System Kinematics....27
2.3.2 System Dynamics....29
2.4 Derivation of the equations of motion of the flexible part using the finite element method....33
2.5 .Analysis of the open-loop dynamics of the zero-g model....42
2.5.1 Undamped, unforced dynamics....42
2.5.2 Effects of the stored angular momentum on the flexible structure....46
2.5.2.1 Rough estimates of the precession frequency....46
2.5.3 Open-loop transfer functions....59
2.5.3.1 Approximate analysis of the influence of the base-body flexibility on the torque to payload pointing angle transfer function....60

Chapter 3: Derivation of the model in one-g....65
Introduction....65

3.1 List of assumptions used in deriving the one-g model......................... 65
3.2 Description of the suspension device and of the internal
dynamics of the suspension.. 68
3.3 Gravity effects on MACE...73
3.3.1 The modelling of the suspension cables....................................73
3.3.2 The derivation of the gravity geometric stiffness matrix of a
beam-column...74
3.3.3 The modelling of the gravity load on the flexible structure
and the articulated part...76
3.3.4 Linearized dynamics.
Derivation of the gravity stiffness matrix for the articulated bodies......78
3.4 The derivation of the equations of motion of the constrained system..........81
3.5 Dynamic analysis of the one-g model..85
3.5.1 Open-loop eigenfrequencies and mode shapes. Transfer functions......85

Chapter 4: Large angle maneuvers on MACE..96
Introduction...96
4.1 Numerical simulation...96
4.2 Trajectory tracking of the payload using a
feedback/feedforward approach...98
4.2.1 Equations of motion of MACE in zero-g.............. 98
Model simplification for feedforward design.............................99
4.2.2 Trajectory design..101
4.2.3 Derivation of the feedback/feedforward control law.....................102
4.3 Feedback linearizing attitude control in MACE.................................106
4.4 Model reduction for multi-body simulation using the
balanced modal reduction algorithm...111
4.5 Implementation of the feedback/feedforward control
laws in the linear and non-linear models. Results of the simulations........121

Chapter 5: Conclusions 132

References...134

Appendix..138
Appendix A1: Inertia matrices of the bodies of the articulated chain..................138
Appendix A2: Non-linear terms in the equations of motion of the
equations of motion of the articulated bodies.............................140
Appendix A3: Non-linear terms in the equations of motion of the gyrostat...........141
Appendix A4: Blow-up of Figure [2.9]..142
Appendix A5: MATLAB simulation code..143

LIST OF FIGURES

Page
1.1 The Middeck Active Control Experiment (MACE).......................................9
2.1 Coordinate frames in the articulated body...18
2.2 Schematic of Central Node...27
2.3 Finite elements degrees of freedom...33
2.4 to 2.9 Structural frequencies vs. angular momentum.................................52
2.10 to 2.11 Gimbal torque to inertial angle transfer functions..........................55
2.12 to 2.17 Transfer functions: all-in-plane vs. all-out-of-plane configurations.........56
2.18 Articulated body mounted on a flexible base..59
3.1 Schematic of the suspended test article...67
3.2 Schematic of suspension device...71
3.3 Suspension control loop...72
3.4 Suspension controller block diagram...72
3.5 Gravity-stiffened beam cable...76
3.6 to 3.11 Comparison of transfer function from experiment and
from analytical model...89
3.12 Transfer function from gimbal torque to inertial angle in one-g of model with
rigid suspension cable.. 95
4.1 Angular position, velocity and feedforward torque for slew maneuver..............102
4.2 Log. of Balanced singular values vs. mode number..................................114
4.3 to 4.4 Full order model and reduced order model transfer functions................115
4.5 a Animation of test article motion in zero-g during
slew maneuver without bus attitude controller...124
4.5b Animation of test article motion in zero-g during slew maneuver with bus

attitude controller..124
4.6 to 4.14 Linear time simulation results of slew maneuver with bus
attitude controller..125
4.15 Linear vs. non-linear time simulation: relative angle....................................130
4.16 Flow Chart of Program for Multi-Body Simulation....................................131
Appendix A4 Blow-up of figure [2.9]..142

LIST OF TABLES

Page
2.1 Structural bus parameters and gimbal mount and payload data (SI units).............15
2.2 Angular partial velocities for the articulated chain.......................................24
2.3 Angular linear velocities for the articulated chain...24
2.4 Partial velocities for the gyrostat...29
2.5 Structural frequencies [Hz] and mode shapes of zero-g model.
Hinged vs. Locked..44
2.6 Structural frequencies [Hz] and mode shapes of zero-g model.
Influence of Stored angular momentum (Hy = 1Nm)......................................45
3.1 Parameters of the block diagram of figure [3.3]...69
3.2 Structural frequencies [Hz] and mode shapes of the suspended test article............86
4.1 Structural frequencies: full order and reduced order model............................120

CHAPTER 1

1.1 INTRODUCTION

The focus of the present research is on developing a dynamic model for a small scale multi-body testbed representative of future generation multi-pointing platforms. The testbed under consideration is the Middeck Active Control Experiment (MACE), a flight experiment proposed by the Space Engineering Research Center at the Massachusetts Institute of Technology.

The objective of the flight experiment is to investigate and validate the modelling of the dynamics of a sufficiently complex, actively controlled multi-body spacecraft free floating in zero gravity. The essence of the MACE program is to identify and characterize the gravity influences both on orbit and during ground testing, and analytically predict the way these perturbations influence the closed-loop control problem. The motivations for conducting such experiments are twofold. First, analytical modelling is not sufficient to guarantee satisfactory performance of the real system. Second, to guarantee stability of the real system in the closed-loop design, the open-loop design is not sufficient in general. A solution is to perform closed-loop ground-based tests on reduced scaled models of the spacecraft.

The testbed under examination is therefore representative of the Class II-IV Control-Structure Interaction (CSI) Missions , Ref.[4], which includes many types of low-Earth orbit and geo-synchronous platforms, typical of an Earth Observation System. The system, shown in Figure [1.1], consists of a flexible space platform with a number of rigid or elastic appendages mounted on it. A typical platform, as it has been envisioned by the international space community, will have multiple articulated payloads carried on a flexible bus. Each payload may have separate pointing performance requirements on agility, accuracy and pointing stability. In particular, each payload may undergo independent maneuvers in a slew/pointing/track mode. In its general conception, this class of missions have the goal of achieving precision attitude control of the spacecraft, very fine pointing of each payload with respect to inertial space and vibration suppression.

The flight test article is stored in the mid-deck lockers of the Space Shuttle and can be assembled by the crew in a variety of configurations (straight line, L-shaped) to allow the implementation of different set of controllers from very simple to very complex ones. The laboratory test article is used for ground testing and to identify the effects of gravity upon the dynamics of the suspended platform. The structure on the ground is suspended by means of an electro-mechanically controlled, state-of-the-art suspension device. Effects present in the suspended test article are: added stiffness, mass and damping of the suspension system, modal coupling between the test article and the suspension system characteristics, different eigen-structure of the test article due to static pre-deformation, additional non-linear torquing on the articulated part.

The articulated part is composed of two gimballed payloads capable of independent operations. One axis of the two-axis gimballed payloads allows control in the vertical plane, defined by the bus centerline and the gravity vector, which is thought to be only slightly perturbed by gravity effects. The second axis controls the out-of-plane payload dynamics and will excite horizontal bending and torsional deformations, which are more susceptible to gravity influences. Large angle payload motions are required in the experiment because they would provide a source of non-linear dynamic behavior caused by the motion and the gravity coupling during the pointing, scanning or tracking operations. Here, *pointing* refers to the maintenance of some pre-specified orientation in an inertial frame, *tracking* refers to the following of some unknown (caused by exogenous sources to the controller) profile in an inertial frame and *scanning* refers to the following of a known profile or trajectory in an inertial frame. Clearly, any time a torque is applied to the payload an equal and opposite torque is applied to the rest of the spacecraft. Therefore the possibility exists of unwanted excitation of the natural rigid and flexible body modes of the spacecraft itself by internal degrees of freedom, and it is easy to understand the complexity of the control logic needed to stabilize the system in inertial space whenever the two payloads act independently. This topic is still area of active research in the control of multivariable systems. Furthermore, in the presence of a one-g field, the torque transmitted to the rest of the bus also includes components due to the gravity torquing itself. This is equivalent to saying that the equilibrium position of the suspended test article is further influenced by the motion of the payload.

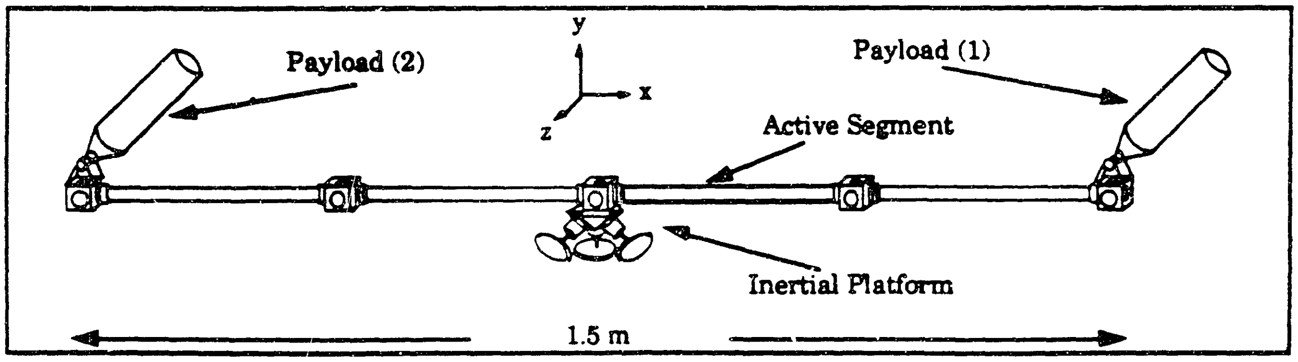


Figure [1]. The Middeck Active Control Experiment (MACE).

The geometry of this test article is sufficiently complex as to represent the different degrees of dynamic coupling. In particular, the bus is sufficiently flexible so that a number of modes lie within the control bandwidth and the damping level is typical of structures used for aerospace applications. The presence of the two gimballed payloads allows the implementation of multiple interacting control systems with independent objectives, and the installation of an articulating flexible appendage would enable the investigation of the dynamics of a telerobotic servicer. The control objectives are expressed in terms of pointing and scanning performance of the payload, and the performance metrics include the root-mean-square (rms) 2-axis angular position (stability) and angular rate (jitter) about a pointing line of sight or scanning profile, the time to complete the slew, the performance degradation of one payload when the other is acting.

The purpose of this thesis is to provide a three-dimensional dynamic model for both the on-orbit and the test article on the ground, rather than developing a general purpose numerical tool for handling multi-body flexible spacecraft. The equations of motion are derived using finite elements for the flexible bus and Kane's equations for the articulated part. The main reason for using Kane's method of generalized speeds, also known as the Lagrange's form of d'Alembert 's Principle, is that it combines the computational advantages of both Newton's laws and the Lagrangean formulation (in fact, tedious differentiations of scalar energy functions are avoided and the non-working constraint forces do not enter in the equations of motion) resulting in a set of equations of minimum dimension. Public domain general purpose software such as DISCOS uses this formulation and allows the user to simulate numerically the motion of dynamic systems of high complexity, also in closed-loop topological trees. Unfortunately, it does not allow the user to obtain any information about the analytical structure of the equations of motion, which could greatly help in understanding the dynamics of the system itself. This is the reason that motivated the procedure described in this thesis and which resulted in an ad hoc computer simulation program written in MATLAB. High fidelity structural modeling is in turn offered by the finite element method. Instead of using NASTRAN or ADINA to obtain the modal characteristics, a more direct approach has been adopted and an ad hoc finite element code has also been used. The advantages of using this procedure is that the resulting model is of lower order, which greatly simplifies the design of a suitable controller at a later stage of the design, and keeps the most important dynamic features of the structural test article (in particular the change in modal characteristics as the configuration of the articulated bodies changes in time) and that the computer code is simpler to interact with. A serious disadvantage is the fact that this code is not very flexible in accomodating dramatic changes in geometry, and that the architecture of the program is not optimized which in turn results in slow computational times. Nevertheless, computer simulation is often the only realistic means of investigating the stability and the performance of complex, multi-body dynamic systems which are expected to operate in space, but for which hardware ground testing presents difficult implementation issues. In addition, time-domain solutions to the control problem, especially when non-linearities are present, are a necessary (and by its nature, iterative) step in the design phase and are therefore computationally intensive. This means that the dynamic analyst must search for ways to simplify the equations and reduce the order of the model without altering the input/output properties of the system.

The procedure followed in this thesis yields a set of fully non-linear, coupled, and configuration dependent equations of motion. The model is then linearized about a certain geometric configuration and the effect of the modelled non-linearities is investigated, both in the zero-g and the one-g test article models. In particular, the validity of this reduced order model (with respect to an equivalent, but more sophisticated NASTRAN model) is examined. A feedback /feedforward decentralized approach for the control law is adopted to simulate the dynamic response of the structure to a reference trajectory for the slew of one of the articulated payloads. The response is then investigated and suggestions on a suitable feedback linearizing controller for the attitude of the spacecraft are made. Finally, the balanced reduction algorithm is adopted to bring the size of the model to a more manageable size, for purposes of non-linear time simulation.

1.2 THESIS OVERVIEW

Chapter 2 presents a description of the test article and of the suspension device. It also presents the derivation of the equations of motion of the three-dimensional structure in free space. The finite element method is used to model the flexible members and Kane's method is used to derive the dynamic equations of motion of the articulated payloads. An investigation on the influence of the bus flexibility on the pointing dynamics and on the influence of the stored angular momentum on the flexible dynamics is also carried out.

Chapter 3 presents the model of the structure on the suspension. The effect of gravity is included both in the flexible part and in the articulated part. Also, an analysis is made on how the eigenstructure changes due to gravity, and a comparison is carried out with experimentally obtained data on the real test article.

Chapter 4 shows the result of the implementation of a control law used to simulate the slew maneuver of one of the payloads and the comparison of the numerical results for the slew maneuver implemented in the zero-g model and in the one-g model. It also suggests feedback linearizing control laws for both the attitude control of the spacecraft and of the maneuvering of the articulated bodies with respect to the base body. Furthermore, and in order to reduce the time required by the non-linear time simulation, the balanced reduction algorithm has been applied to reduce the order of the model.

CHAPTER 2 : DERIVATION OF THE MODEL IN ZERO-G.

INTRODUCTION

The three-dimensional equations of motion of a spacecraft composed of flexible elements, articulated rigid payloads and torque-wheel attitude actuators are to be derived. After describing the system model and stating the assumptions made, the kinematics and the dynamics of each component are derived and the equations of motion are assembled.

This chapter is divided in 3 sections. Section 1 describes the features of the spacecraft to be modelled. Included in this Section are a description of the structural plant, consisting of line drawings and tables of relevant structural parameters and a description of the sensors and actuators and their placement on the structure.

Section 2 contains the derivation of the analytical model of the spacecraft in zero-g. It is divided into four subsections: derivation of the equations of motion of the articulated part (part 1) and of the central node (parts 2) using Kane's method, derivation of the finite element model for the flexible part (part 3) and assemblage of the equations independently derived with the two methods to obtain the equations of motion of the spacecraft fully non-linear in torque-wheel rates and in payload angle and rates.

In Section 3 approximate analyses of the open-loop dynamics of the zero-g model are carried out with the purpose of deriving the natural frequencies and the mode shapes. An analysis is made of the effect of the stored angular momentum on the flexible dynamics. Also an evaluation analysis of the effect of the bus flexibility into the pointing dynamics is developed.

2.1 General Description of the MACE

Figure 1 shows the MACE test article. This configuration is also the zero-g configuration. The 3-D model of the MACE considers the flexible bus supporting the payloads as four slender cylindrical Lexan beams connected at five rigid metallic nodes located at regular intervals on the beam. These nodes are actual elements which, in reality, serve the purpose to connect the different struts composing the bus and attach the sensors, actuators and payloads. They also allow one to change the geometry of the test article by adding additional elements. Each of the two end nodes supports a gimballed payload. A DC torque motor actuates each gimbal and the relative gimbal angle as well as the inertial angular rate of the payloads can be measured. On the central node, in the middle of the beam, an inertial rate gyro package and a set of reaction wheels are used to control the attitude of the spacecraft. The sensor package measures the inertial angular rate about the axes of a reference frame centered on the body.

The reference frame used to describe the geometry of the system is centered at the central node at a point which is located on the centerline of the structural bus. The X-axis is along the bus, pointing to the right, the Y-axis is vertical, pointing upward, and the Z-axis is perpendicular to the page facing outward (Figure 1.1). This reference is taken to be an inertial reference frame for the purposes of this work.

The test article actuators include two two-axis gimbals which drive the two payloads, three attitude control torque wheels, not aligned with the principal axes of the bus, and a two-axis piezoelectric bending member (active segment).

The sensors include two-axis rate gyros located on each payload, a three axis rate gyro at the attitude control location (inertial platform), two angle optical encoders on each gimbal axis, assorted strain gauges and a series of triaxial accelerometers distributed along the nodes of the structure.

Table 1 shows some geometric parameters of the structure. A more detailed description of the engineering specifications of the structural test article can be found in Ref.[2,3].
The Development Model (DM) of the test article, the one actually used in this thesis, is different from the real test article (in its flight configuration) in that only one articulated payload is present, the other being a non-articulated dummy gimbal of equivalent inertia properties. At the moment of completing this thesis,in fact, the second gimbal was still under construction. A detailed description of the DM model is given in Ref.[3].

The suspension system is described in detail in Ref.[4]. It is intended to minimize (theoretically to cancel) the effects involved in suspending the structure in one-g. To do so, it should provide at least a decade of separation between the suspension mode frequencies (or pendular frequencies) and the first flexible mode of the suspended test article. It consists of three steel cables attached to the structure at three equally spaced locations and driven by three equal independent actively controlled suspension devices. These,in turn, consist of a pneumatic control system to provide the reference position in inertial space, and an electro-mechanical control system to guarantee fine control of the position of the bus centerline about the reference.

Table 2.1. Structural bus parameters and gimbal mount and payload data (SI units)

Length of flex. beam	0.22886
Length of joint	0.15264
Outer/Inner diameter	0.0254/0.0190
Density	1189.77
Polar Moment of Inertia about X	2.8x10-8
Section Moment of Inertia about Y and Z	1.4x10-8
Mass per Unit Length	0.27
Young's Modulus [N/m2]	2.3x10+9
EI	32.2
Damping ratio	1.0% (uniform)
Mass of central node	8.78
Mass of 1st body in the chain	2.9918
Mass of 2nd body in the chain	2.9227
Mass of 3rd body in the chain	1.0616
Mass of payload (including sensor)	1.2979
Position of CM of payload from hinge	0.1745

2.2.1 LIST OF ASSUMPTIONS FOR THE ZERO-G MODEL.

The following is a list of the assumptions made in the derivation of the equations of motion of the various components of the spacecraft. An underlying assumption is that the mass/inertia properties of each component are assumed known. The bodies are assumed to have constant mass and inertia properties, and the moving parts possess only rotational freedom with respect to each other.

On the flexible part:

1) The flexible elements (Lexan beams) are modelled using nominally straight Bernoulli Euler finite elements of symmetric tubular cross section and uniform density.

2) Rotary inertia and shear effects are considered negligible, but torsion is taken into account.
3) Each flexible beam possesses 6 degrees of freedom at each finite element node.
4) Elastic deformations and deformation rates are assumed to be infinitesimal. This assumption allows one to neglect, under certain conditions, second order terms in the equations of motion. Furthermore, linear strain-displacement is assumed as constitutive relationship for each element.
5) One percent proportional damping is added to represent material damping.
6) The solid elements connecting together two flexible elements of the bus are modelled as rigid links. In, particular, they allow for relative rotations between the connecting points.

On the articulated part:

1)

The articulated part at nodes 1 and 5, represents the various stages of each gimbal-payload assembly and is modelled as a chain of 3 rigid bodies connected by frictionless revolute joints, each joint allowing one relative rotation of one body with respect to the previous one in the chain. This is motivated by the necessity of adapting the model to different stages in the design of the physical system. Body 1 is defined as the fixed part attached to the bus, consisting of the support of the gimbal mount. Body 2 is defined as the first stage of the motor, capable of rotation about a direction parallel to the spacecraft roll axis (X) and body 3 is defined as the assembly of the second stage of the motor plus the payload and rate gyro package attached to it. The latter is capable of rotation about a direction parallel to the pitch axis of the spacecraft (axis Z). Each separate body is allowed to have the most general inertia distribution with an offset of the location of the center of mass with respect to each pivot point. We will discuss later on how the behavior of the articulated body changes when it is CG mounted and when it is not.
The modelling of this separate entity is carried out considering the end node as the origin of the frame of the first body of the chain, thereby defining the first 6 degrees of freedom. The relative rotations of the next two bodies contribute two additional degrees of freedom. Kane's method is used to derive the equations of motion of this system of bodies. Next, the dynamics of the articulated payload are assembled to the dynamics of the rest of the spacecraft.

2)

The central node, node 3, where the inertial platform is located, is modelled as a rigid body on which three torque wheels, each capable of sustaining independent spin about its own axis, are mounted and oriented along a skewed (non-principal) reference frame. The dynamics of this separate body are described by 6 degrees of freedom associated with the finite element node fixed at node 3 (the three rotational displacements therefore describe the attitude of the spacecraft with respect to a reference frame aligned with the X-Y-Z frame) plus the three relative rates of each wheel. External perturbations are neglected, except for those acting at the actuator locations.

An important assumption concerns the definition of the reference frames associated with each rigid body of the system. The flexible bus has a coordinate system at each finite element node. For each finite element, a local reference frame can be embedded at each finite element node, describing the elastic displacements and rotations of each local element. Similarly, a global reference frame can be embedded at a particular reference point of the whole assembled structure. It would represent the frame to which we can refer the elastic deformations of each flexible member. Finally, one should also include an additional reference frame which characterizes the inertial or newtonian motion of the system in inertial space. The last would, for example, describe the attitude dynamics of the spacecraft considered as a rigid body during re-orientation maneuvers.
At the i-th node, denoted by F_i, the embedded local coordinate frame is identified by the unit vectors $(\mathbf{f}_1,\mathbf{f}_2,\mathbf{f}_3)_i$, describing a right handed triad. For convenience, we will refer to a system of three oriented unit vectors as a vectrix $\mathcal{F}_i$ (note that it is a column vector), in the manner described in Ref.[5].

The convention for the MACE studies considers as mentioned above a newtonian reference frame (N) with axis X along the undisplaced centerline of the bus, Y vertical upwards and Z which completes the right handed reference frame. The inertial coordinates of each node are defined with respect to this N frame $\mathcal{F}_N$, as shown in Figure [1.1]. Therefore, we assume that the transformation matrix relating $\mathcal{F}_i$ to $\mathcal{F}_N$ is an infinitesimal transformation, equal as shown below to $[\mathbf{I} + \theta^{\times}]$, where $\mathbf{I}$ is the identity matrix and $\theta^{\times}$ is the skew symmetric matrix of rotational deformational coordinates.

Consistent with the definitions used in the finite element methodology, we will call x, y, z (lower-case) the elastic linear displacements of the i-th node with respect to the $\mathcal{F}_N$ frame. In particular, x will describe the longitudinal displacement of the finite element at node i along the 1-direction (X), y the upward bending deflection along 2 (Y) and z the "out-of-plane" bending deflection along 3 (Z). These elastic displacement are nominally zero, i.e., in the undeformed configuration. Similarly, we will call θ_{1i}, θ_{2i}, θ_{3i} the elastic rotational displacements of the i-th node in the $\mathcal{F}_i$ frame with respect to the $\mathcal{F}_N$ frame . In particular, θ_{1i} will describe the torsion about the X axis, θ_{2i} the horizontal bending of the element about Y and θ_{3i} the vertical bending about Z. The sign convention for the rotations follows from the definition of the right-handed triad (positive counter-clockwise). These elastic rotations are also nominally zero.

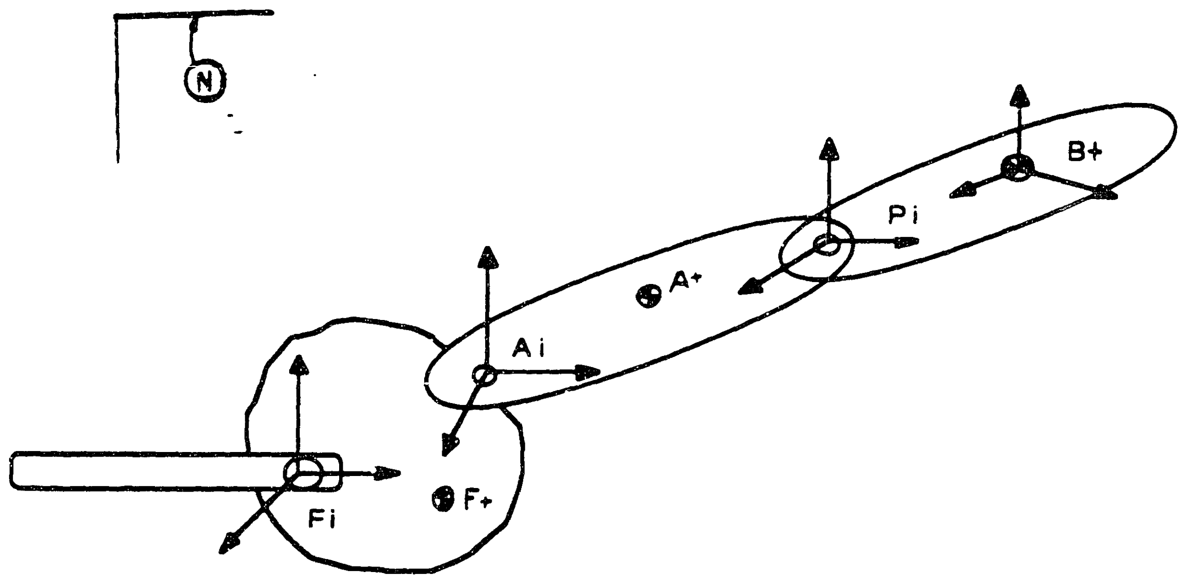


Figure (2.1). Coordinate frames in the articulated body.

2.2 DERIVATION OF THE EQUATIONS OF MOTION OF THE ARTICULATED PART.

2.2.1 System Kinematics

Following Kane (see Ref.[6]), with each rigid body is associated a reference frame. The articulated payload is described by a coordinate frame for each body in the chain, as shown in Figure (2.1). In particular, body 1 is described by the F_i frame of the end node, body 2 by the A_i frame centered at the first hinge and body three by the P_i frame centered at the second hinge. The orientation with respect to the F_i frame is defined by the payload pointing mechanism gimbal angles. For these angles, the desired orientation is user-specified. Also, frame B_i is centered at the payload's center of mass (CM). In conclusion, the inertial orientation of the payload may be determined by the orientation of the P_i frame with respect to the A_i frame, the orientation of the A_i frame with respect to the F_i frame, and the orientation of the F_i frame in inertial space. In order to completely define the kinematics of this 3-body assembly,we must evaluate a set of vector quantities, namely, the generalized speeds, which directly enter the equations of motion. These are defined, for a point P in N, by:

$$^{N}\mathbf{v}^{P} = \sum_{i=1}^{N} {}^{N}\mathbf{v}_r^{P}\, u_i + \mathbf{v}_t \tag{2.1}$$

where $^{N}\mathbf{v}^{P}$ is the velocity of P in N, $u_i = \dot{q}_i$ is the time derivative of the i-th generalized coordinate (generalized speed) and $^{N}\mathbf{v}_r^{P}$ is the r-th partial velocity of P in N.

Points F+, A+ and B+ denote the centers of mass of each body. Points F, A and P represent the origin of the reference frame embedded on each body. Note that none of these triads are centered on the centers of mass, and this is simply to accomodate geometric irregularities of the physical bodies. Therefore, frame A_i is, in general, related to frame F_i by the transformation Caf :

$$A_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\varphi & s\varphi \\ 0 & -s\varphi & c\varphi \end{bmatrix} F_i \tag{2.2}$$

where φ is the relative angle of body 2 with respect to body 1.
Similarly frame P_i is related to frame A_i by the transformation Cpa:

$$P_i = \begin{bmatrix} c\theta & s\theta & 0 \\ -s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} A_i \qquad (2.3)$$

where θ is the relative angle of body 3 with respect to body 2. Therefore, Cpf = Cpa Caf.
We use the convention that c=cos(.) and s=sin(.). Also, angular velocities are denoted by ω and linear velocities by v. For example, according to our notation, ${}^{N}\mathbf{v}^{P}$ stands for the velocity vector of point P in the reference frame N. Furthermore, let us establish the notation that $r_1 = r_x$, $r_2 = r_y$, $r_3 = r_z$, for the components of every vector r in a given frame F_i , which can be inferred from the text.

Kane's method requires the determination of the generalized speeds. These kinematical quantities uniquely define the motion of the rigid bodies. With reference to Figure [2.1], we have that:

$$\begin{aligned} {}^{N}\omega^{F} &= \begin{pmatrix} \dot\theta_1 & \dot\theta_2 & \dot\theta_3 \end{pmatrix} F_i \\ {}^{N}\omega^{A} &= \begin{pmatrix} \dot\theta_1+\dot\varphi & \dot\theta_2 & \dot\theta_3 \end{pmatrix} F_i \\ {}^{N}\omega^{P} &= \begin{pmatrix} \dot\theta_1+\dot\varphi & \dot\theta_2-\dot\theta s\varphi & \dot\theta_3+\dot\theta c\varphi \end{pmatrix} F_i \end{aligned} \qquad (2.4)$$

and, for the velocity of point F:

$$ {}^{N}v^{P} = \begin{pmatrix} \dot x & \dot y & \dot z \end{pmatrix} F_i \qquad (2.5)$$

Note, as introduced above, that the assumption of small elastic deflections allows one to approximate the transformation:

$$F_i = \left[\mathbf{I} - \theta^{\times}\right] N = \begin{bmatrix} 1 & \theta_3 & -\theta_2 \\ -\theta_3 & 1 & \theta_1 \\ \theta_2 & -\theta_1 & 1 \end{bmatrix} N \tag{2.6}$$

where $\theta = (\theta_1\ \theta_2\ \theta_3)^T$, and $(.)^x$ is the skew operator, by the identity matrix **I**.

The remaining velocities are given recursively by:

$$\begin{aligned} {}^{N}\mathbf{v}^{F+} &= {}^{N}\mathbf{v}^{F} + {}^{N}\boldsymbol{\omega}^{F} \times {}^{N}\mathbf{r}^{F+} \\ {}^{N}\mathbf{v}^{A+} &= {}^{N}\mathbf{v}^{F} + {}^{N}\boldsymbol{\omega}^{F} \times {}^{F}\mathbf{r}^{A} + {}^{N}\boldsymbol{\omega}^{A} \times {}^{A}\mathbf{r}^{A+} \\ {}^{N}\mathbf{v}^{B+} &= {}^{N}\mathbf{v}^{F} + {}^{N}\boldsymbol{\omega}^{F} \times {}^{F}\mathbf{r}^{A} + {}^{N}\boldsymbol{\omega}^{A} \times {}^{A}\mathbf{r}^{P} + {}^{N}\boldsymbol{\omega}^{P} \times {}^{P}\mathbf{r}^{B+} \end{aligned} \tag{2.7}$$

They become:

$$\begin{aligned} {}^{N}\mathbf{v}^{P} &= \left[\ \dot{x}+\dot{\theta}_2 r_3-\dot{\theta}_3 r_2 \quad \dot{y}+\dot{\theta}_3 r_1-\dot{\theta}_1 r_3 \quad \dot{z}+\dot{\theta}_1 r_2-\dot{\theta}_2 r_1\ \right] F_i \\ {}^{N}\dot{\mathbf{v}}^{A+} &= \left[\ \dot{x}+\dot{\theta}_2 A_2-\dot{\theta}_3 B_2 \quad \dot{y}+\dot{\theta}_3 C_2-\dot{\theta}_1 A_2-\dot{\varphi} C_1 \quad \dot{z}+\dot{\theta}_1 B_2-\dot{\theta}_2 C_2+\dot{\varphi} B_1\ \right] F_i \\ {}^{N}\mathbf{v}^{B+} &= \left[\ \dot{x}+\dot{\theta}_2 A_5-\dot{\theta}_3 B_5-\dot{\theta} C_5 \quad \dot{y}+\dot{\theta}_3 D_5-\dot{\theta}_1 A_5-\dot{\varphi} E_5+\dot{\theta} G_5 \quad \dot{z}+\dot{\theta}_1 B_5-\dot{\theta}_2 C_5+\dot{\varphi} F_5+\dot{\theta} H_5\ \right] F_i \end{aligned} \tag{2.8}$$

where $r_i = {}^{F}r_i^{F+}$, i=1,2,3 represent the components of the vector in the F_i frame. For convenience we have defined:

$$
\begin{gathered}
A_1 = {}^{A}r_1{}^{A+} \\
B_1 = {}^{A}r_2{}^{A+} c\varphi - {}^{A}r_3{}^{A+} s\varphi \\
C_1 = {}^{A}r_2{}^{A+} s\varphi + {}^{A}r_3{}^{A+} c\varphi \\[1em]
A_2 = C_1 + {}^{F}r_3{}^{A} \\
B_2 = B_1 + {}^{F}r_2{}^{A} \\
C_2 = A_1 + {}^{F}r_1{}^{A} \\[1em]
A_3 = {}^{A}r_1{}^{P} \\
B_3 = {}^{A}r_2{}^{P} c\varphi - {}^{A}r_3{}^{P} s\varphi \\
C_3 = {}^{A}r_2{}^{P} s\varphi + {}^{A}r_3{}^{P} c\varphi \\[1em]
A_4 = {}^{P}r_1{}^{B+} c\theta - {}^{P}r_2{}^{B+} s\theta \\
B_4 = {}^{P}r_1{}^{B+} s\theta c\varphi + {}^{P}r_2{}^{E+} c\theta c\varphi - {}^{P}r_3{}^{B+} s\varphi \\
C_4 = {}^{P}r_1{}^{B+} s\theta s\varphi + {}^{P}r_2{}^{B+} c\theta s\varphi + {}^{P}r_3{}^{B+} c\varphi \\[1em]
A_5 = C_3 + C_4 + {}^{F}r_3{}^{A} \\
B_5 = B_3 + B_4 + {}^{F}r_2{}^{A} \\
C_5 = C_4\, s\varphi + B_4\, c\varphi \\
D_5 = A_3 + A_4 + {}^{F}r_1{}^{A} \\
E_5 = C_3 + C_4 \\
F_5 = B_3 + B_4 \\
G_5 = A_4\, c\varphi \\
H_5 = A_4\, s\varphi \\
L_5 = C_4\, c\varphi - B_4\, s\varphi \\
M_5 = A_3 + A_4 \\
N_5 = C_3 + C_4 \\
O_5 = C_4\, s\varphi + 2B_4\, c\varphi \\
T_5 = B_3 + B_4 \\
W_5 = 2C_4\, s\varphi + B_4\, c\varphi
\end{gathered}
\tag{2.9}
$$

Note that so far the velocities and angular velocities are non-linear in hinge angles and rates.

2.2.2 System Dynamics

According to Kane's formalism, the dynamical equations of motion are given by:

$$F_r^* + F_r = 0 \tag{2.10}$$

where F_r^* is the generalized inertia force corresponding to the r-th generalized speed and F_r is the generalized active force. They are defined by the relations:

$$F_r^* = \sum_{i=0}^{Nbodies} \left[{}^N v_r^i \cdot F^{i*} + {}^N \omega_r^i \cdot T^{i*} \right]$$

$$F_r = \sum_{i=0}^{Nbodies} \left[{}^N v_r^i \cdot F^i + {}^N \omega_r^i \cdot T^i \right]$$

(2.11)

or the generalized inertia forces are given by the sum over all the bodies of the dot product of the linear partial velocities of each mass center with the generalized inertia force of each body plus the dot product of the partial angular velocity of each body with the generalized inertia torque. Similarly, the the generalized active forces are given by the sum over all the bodies of the dot product of the linear partial velocities of each point of application of force with the appropriate force plus the dot product of the partial angular velocity of each body with the active external torque acting on it. Here the partial velocities are defined to be the partial derivative with respect to the r-th quantity of the velocity itself. They are actually derived by inspection, and they are shown in the next tables:

Table 2.2. Angular Partial Velocities

r	${}^{N}\mathbf{w}_r^{F}$	${}^{N}\mathbf{w}_r^{A}$	${}^{N}\mathbf{w}_r^{P}$
1	0	0	0
2	0	0	0
3	0	0	0
4	$\mathbf{f}_1$	$\mathbf{f}_1$	$\mathbf{f}_1$
5	$\mathbf{f}_2$	$\mathbf{f}_2$	$\mathbf{f}_2$
6	$\mathbf{f}_3$	$\mathbf{f}_3$	$\mathbf{f}_3$
7	0	$\mathbf{f}_1$	$\mathbf{f}_1$
8	0	0	$-s\,\varphi\,\mathbf{f}_1 + c\,\varphi\,\mathbf{f}_3$

Table 2.3. Linear Partial Velocities

r	${}^{N}\mathbf{v}^{F+}$	${}^{N}\mathbf{v}^{A}$	${}^{N}\mathbf{v}^{B+}$
1	$\mathbf{f}_1$	$\mathbf{f}_1$	$\mathbf{f}_1$
2	$\mathbf{f}_2$	$\mathbf{f}_2$	$\mathbf{f}_2$
3	$\mathbf{f}_3$	$\mathbf{f}_3$	$\mathbf{f}_3$
4	$-{}^{F}r_3^{F+}\ \mathbf{f}_2 + {}^{F}r_2^{F+}\ \mathbf{f}_3$	$-A2\ \mathbf{f}_2 + B2\ \mathbf{f}_3$	$-A5\ \mathbf{f}_2 + B5\ \mathbf{f}_3$
5	${}^{F}r_3^{F+}\ \mathbf{f}_1 - {}^{F}r_1^{F+}\ \mathbf{f}_3$	$A2\ \mathbf{f}_1 - C2\ \mathbf{f}_3$	$A5\ \mathbf{f}_1 - D5\ \mathbf{f}_3$
6	$-{}^{F}r_2^{F+}\ \mathbf{f}_1 + {}^{F}r_1^{F+}\ \mathbf{f}_2$	$-B2\ \mathbf{f}_1 + C2\ \mathbf{f}_2$	$-B5\ \mathbf{f}_1 + D5\ \mathbf{f}_2$
7	0	$-C1\ \mathbf{f}_2 + B1\ \mathbf{f}_3$	$-E5\ \mathbf{f}_2 + F5\ \mathbf{f}_3$
8	0	0	$-C5\ \mathbf{f}_1 + G5\ \mathbf{f}_2 + H5\ \mathbf{f}_3$

Because, by definition $\mathbf{F}^* = m_k\,{}^{N}\mathbf{a}^{k}$ and $\mathbf{T}^* = \underline{\underline{J}}_k\ {}^{N}\alpha^{k} + {}^{N}\omega^{k} \times (\underline{\underline{J}}_k \cdot {}^{N}\omega^{k})$, where m_k is the mass of the k-th body and $\underline{\underline{J}}_k$ is its central inertia dyadic, it is possible to express the (i,j)-the entry of the inertia matrix as:

$$m(i,j) = \sum_{k=0}^{Nbodies} \left[m_k \left({}^{N}v_i^{k} \cdot {}^{N}v_j^{k} \right) + {}^{N}\omega_r^{k} \cdot \underline{\underline{J}}_k \cdot {}^{N}\omega_j^{k} \right] \tag{2.12}$$

therefore as the sum over the bodies of certain scalar quantities. The details of these passages are in Appendix A.

The derivation of the equations of motion proceeds with the computation of the angular accelerations of each body (denoted by α) and the linear acceleration of each center of mass (denoted by a). These are compactly given by:

$${}^{N}\mathbf{a}^{F+} = {}^{N}\mathbf{a}^{F} + {}^{N}\omega^{F} \times ({}^{N}\omega^{F} \times {}^{F}\mathbf{r}^{F+}) + {}^{N}\alpha^{F} \times {}^{F}\mathbf{r}^{F+}$$

$${}^{N}\mathbf{a}^{A+} = {}^{N}\mathbf{a}^{F} + {}^{N}\omega^{F} \times ({}^{N}\omega^{F} \times {}^{F}\mathbf{r}^{A}) + {}^{N}\omega^{A} \times ({}^{N}\omega^{A} \times {}^{A}\mathbf{r}^{A+}) + {}^{N}\alpha^{F} \times {}^{F}\mathbf{r}^{A} + {}^{N}\alpha^{A} \times {}^{A}\mathbf{r}^{A+}$$

$${}^{N}\mathbf{a}^{B+} = {}^{N}\mathbf{a}^{F} + {}^{N}\omega^{P} \times ({}^{N}\omega^{P} \times {}^{P}\mathbf{r}^{B+}) + {}^{N}\omega^{A} \times ({}^{N}\omega^{A} \times {}^{A}\mathbf{r}^{P}) + {}^{N}\alpha^{P} \times {}^{P}\mathbf{r}^{B} + {}^{N}\alpha^{A} \times {}^{A}\mathbf{r}^{P} \qquad (2.13)$$

It is straight-forward at this stage to write the equations of motion of this composite body as $\mathbf{M}\,\ddot{\mathbf{u}} = \mathbf{F}$ where M is the inertia matrix, derived as explained above, and $\mathbf{F} = -F_{\text{non-linear}} + F_{\text{external}}$ is a (8x1) vector of Coriolis and centripetal forces and external forces and torques.

The vector of non-linear terms can be simply derived from the expression of the generalized inertia forces by grouping all the terms independent of the second-time derivatives of any generalized coordinate (because the terms containing the acceleration have already been taken into account in the inertia matrix) or by using linear and angular acceleration remainder terms, as explained in Ref.[7]. It is in the former way that they are assembled in the computer program. Notice that until this point we have made no assumptions concerning the magnitude of these non-linear terms, i.e. these are equations including non-linear payload relative angles and rates and Coriolis and centripetal terms in the deformation coordinates as well (even though given the smallness of the latter ones, they could simply be discarded without introducing a dramatic change in the equations). These terms are partially derived and included in Appendix A.

As for the external forces, they are given by

$$(\mathbf{F}_{\text{external}})_r = {}^{N}\omega_r^{F}.(-\tau_A) + {}^{N}\omega_r^{A}.(\tau_A) + {}^{N}\omega_r^{A}.(-\tau_P) + {}^{N}\omega_r^{P}.(\tau_P) + {}^{N}\omega_r^{F}.(T_1\,\mathbf{f}_1 + T_2\,\mathbf{f}_2 + T_3\,\mathbf{f}_3) + {}^{N}\mathbf{v}_r^{F}.(F_1\,\mathbf{f}_1 + F_2\,\mathbf{f}_2 + F_3\,\mathbf{f}_3) \qquad (2.14)$$

where τ_A is the actuator torque localized at point A and acting on body 2 (1st stage motor) and τ_P is the actuator torque localized at point P and acting on body 3 (2nd stage motor). Furthermore, $(T_1\,\mathbf{f}_1 + T_2\,\mathbf{f}_2 + T_3\,\mathbf{f}_3)$ and $(F_1\,\mathbf{f}_1 + F_2\,\mathbf{f}_2 + F_3\,\mathbf{f}_3)$ represent respectively the vector form of the resultant constraint torque and the resultant constraint force acting at the interface between body 1 and the flexible beam. It is easy to see that the contribution of the gimbal torques results in $(\mathbf{F}_{\text{external}})_r = 0$ for $r=1,2,3,4,5,6$ and $(\mathbf{F}_{\text{external}})_7 = \tau_A$, $(\mathbf{F}_{\text{external}})_8 = \tau_P$.

Denoting with the subscripts t, r and o respectively the terms related to the translational, rotational and internal relative rotation configuration degrees of freedom, and with the superscript x a skew symmetric quantity, these equations of motion can compactly be written as:

$$\begin{bmatrix} \text{diag(m)} & -\mathbf{C}^x & \mathbf{Jp} \\ \mathbf{C}^x & \mathbf{J} & \mathbf{Jr} \\ \mathbf{Jp} & \mathbf{Jr} & \mathbf{Jo} \end{bmatrix} \begin{pmatrix} \ddot{q}_T \\ \ddot{q}_R \\ \ddot{q}_o \end{pmatrix} + \begin{pmatrix} \mathbf{F}_{nlT} \\ \mathbf{F}_{nlR} \\ \mathbf{F}_{nlo} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{extT} \\ \mathbf{F}_{extR} \\ \mathbf{F}_{exto} \end{pmatrix} \tag{2.15}$$

where the boldface quantities in the inertia matrix are all functions of θ and φ, the articulation angles. Therefore, this inertia matrix is symmetric, positive definite, and configuration dependent. Denoting with $\mathbf{q}_a = (\mathbf{q}_t, \mathbf{q}_r)^T$ and $\mathbf{q}_o$ the terms related to the translational, rotational and internal rigid body configuration degrees of freedom, these equations are in the form that we will use to assemble them with the rest of the structure.

2.3 DERIVATION OF THE EQUATIONS OF MOTION OF THE CENTRAL NODE.

2.3.1 System Kinematics

With reference to Figure [2.2], we define an oriented triad *B* centered at a reference point of this composite body (to which we will refer to from now on as body B), and aligned with the (X,Y,Z) frame. This reference point will be chosen to be the F3 node, according to our previous convention.

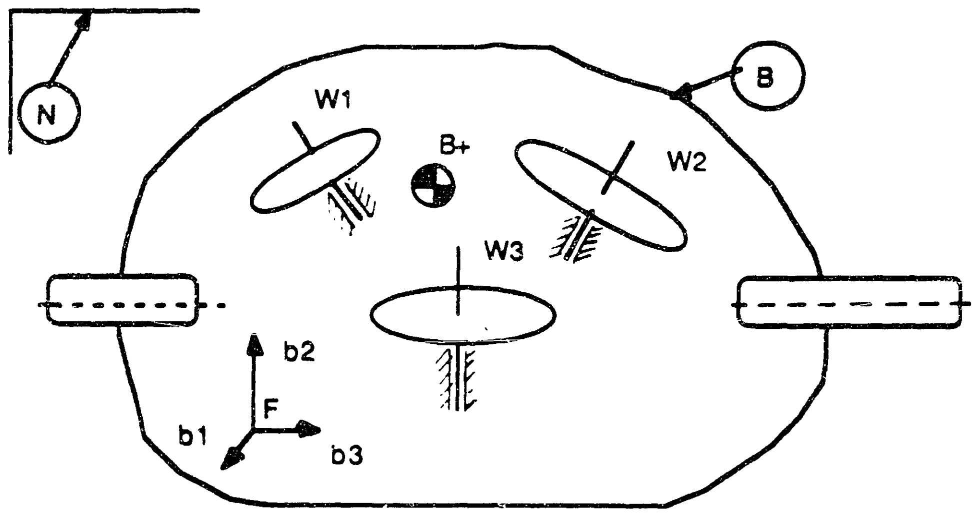


Figure (2.2). Schematic of Central Node

The procedure which we will follow to write the equations of motion of this system is similar to the procedure followed in Ref.[5] for a general n-rotor spacecraft. Indeed, the central node with the inertial platform is in all equivalent to a multispin vehicle in which the torque-wheel assembly is not aligned with the principal axes of the carrier. As in the general case,the wheels may be used for momentum transfer for attitude stabilization or attitude maneuvers of the spacecraft.

An important assumption here is that each wheel is symmetric about its spin axis, while the base-body can be of general shape. Each spin axis is in turn fixed in B, and each center of mass is arbitrarily located with respect to B+, the centroid of the whole system.

Therefore, for each wheel, say W_i, i=1,2,3, spinning with relative velocity ${}^{B}\omega^{W_i}$ about its axis, we define a reference frame W_i ($\mathbf{w}_{i1}$, $\mathbf{w}_{i2}$, $\mathbf{w}_{i3}$) in which axis $\mathbf{w}_{i1}$ is aligned with the wheel's own relative angular momentum. W_i is related to B by the transformation W_i = CwiB B , or:

$$W_i = \begin{bmatrix} p & q & r \\ l & m & n \\ f & g & h \end{bmatrix} B \tag{2.16}$$

where (p,q,r)i represent the constant direction cosines of the spin axis of the i-th wheel in B (and similarly, the other six quantities represent the direction cosines of the remaining two wheel's axes in B).

We can therefore introduce the velocity vector of B and B+ in N, coincident with the global reference frame of the bus, as:

$$\begin{gathered} {}^{N}\mathbf{v}^{B} = \left(\dot{x} \quad \dot{y} \quad \dot{z} \right) N = (u_1 \quad u_2 \quad u_3) N \\ {}^{N}\mathbf{v}^{B+} = {}^{N}\mathbf{v}^{B} + {}^{N}\omega^{B} \times {}^{F}\mathbf{r}^{B+} \end{gathered} \tag{2.17}$$

and the angular velocities of B in N and of each wheel in N as :

$$\begin{gathered} {}^{N}\omega^{B} = (u_4 \quad u_5 \quad u_6) N \\ {}^{N}\omega^{W_i} = u_{6+i}\, \mathbf{w}_{i1} + {}^{N}\omega^{B} = {}^{N}\omega^{B} + {}^{B}\omega^{W_i} \\ (i = 1,2,3) \end{gathered} \tag{2.18}$$

where u_{6+i} (i = 1,2,3) , represent the instantaneous angular velocity of each wheel about its axis relative to the body.

This enables us to write the table of partial velocities as follows:

Table 2.4. Partial Velocities

r	${}^{N}\mathbf{v}_r^{B+}$	${}^{N}\omega_r^{B}$	${}^{B}\omega_r^{W1}$	${}^{B}\omega_r^{W2}$	${}^{B}\omega_r^{W3}$
1	$\bar{\mathbf{n}}_1$	0	0	0	0
2	$\mathbf{n}_2$	0	0	0	0
3	$\mathbf{n}_3$	0	0	0	0
4	0	$\mathbf{b}_1$	0	0	0
5	0	$\mathbf{b}_2$	0	0	0
6	0	$\mathbf{b}_3$	0	0	0
7	0	0	$\mathbf{w}_{11}$	0	0
8	0	0	0	$\mathbf{w}_{21}$	0
9	0	0	0	0	$\mathbf{w}_{31}$

2.3.2 System Dynamics

The generalized inertia forces are given by equation (2.11), which for this case specialize into:

$$(F^*_r)_{\text{translational}} = m\,({}^{N}\mathbf{a}^{B+} \cdot {}^{N}\mathbf{v}_r^{B+})$$

$$(F^*_r)_{\text{rotational}} = \frac{{}^{N}d}{dt}(\underline{\underline{J}}^{B} \cdot {}^{N}\omega^{B}) \cdot {}^{N}\omega_r^{B} + \sum_{i=1}^{3} \frac{{}^{N}d}{dt}(\underline{\underline{J}}^{W_i} \cdot {}^{N}\mathbf{w}^{W_i}) \cdot {}^{N}\omega_r^{W_i}$$

$$F^*_r = (F^*_r)_{\text{translational}} + (F^*_r)_{\text{rotational}}$$
$$r = 1,\ldots,9 \tag{2.19}$$

Introducing the central inertia dyadic of B about B+ in B as $\underline{\underline{J}}^{B}$ and the central inertia dyadic of each wheel as $\underline{\underline{J}}^{W_i} = W_i^{T}\begin{bmatrix} J_{ai} & 0 & 0 \\ 0 & Jt_i & 0 \\ 0 & 0 & Jt_i \end{bmatrix} W_i$ where J_{ai} and Jt_i are the axial and transverse moment of inertia of the wheel about its spin axis, the inertia dyadic of the whole body, including the wheels is given by $\underline{\underline{J}}^{+} = \underline{\underline{J}}^{B} + \sum_{i=1}^{3} I^{BW_i}$, where the transformation

$W_i = C_{W_iB}\, B$ has been used.

Because $\frac{^N d}{dt}(\underline{\underline{J}}^{B} \cdot {}^N\omega^B) \cdot {}^N\omega_r^B = \underline{\underline{J}}^{B} \cdot {}^N\alpha^B + {}^N\omega^B \times (\underline{\underline{J}}^{B} \cdot {}^N\omega^B)$, and similarly for the other terms in equation (2.19), if we make the definitions:

$$\begin{aligned} A_i &= 2\, J_a\, (gl \; - mf)_i\, u_{6+i} \\ B_i &= 2\, J_a\, (hl \; - nf)_i\, u_{6+i} \\ C_i &= 2\, J_a (mh - ng)_i\, u_{6+i} \end{aligned} \tag{2.20}$$

we are now in a position to derive the equations of motion of this composite body. In particular, the rotational generalized inertia forces take the form:

$$\begin{aligned} (F^*_r)_{\text{rotational}} = \; & \{\underline{\underline{J}}^{+} \cdot {}^N\alpha^B + {}^N\omega^B \times (\underline{\underline{J}}^{+} \cdot {}^N\omega^B) + \\ & \sum_{i=1}^{3} [\; \underline{\underline{J}}^{i} \cdot {}^B\alpha^{W_i} + {}^N\omega^B \times (\underline{\underline{J}}^{i} \cdot {}^B\omega^{W_i})] \}\, {}^N\omega_r^B + \\ & \sum_{i=1}^{3} \{[\; \underline{\underline{J}}^{i} \cdot {}^N\alpha^B + {}^N\omega^B \times (\underline{\underline{J}}^{i} \cdot {}^N\omega^B) + \\ & \underline{\underline{J}}^{i} \cdot {}^B\alpha^{W_i} + {}^N\omega^B \times (\underline{\underline{J}}^{i} \cdot {}^B\omega^{W_i}))\;] \cdot {}^B\omega_r^{W_i} \} \end{aligned} \tag{2.21}$$

Introducing the generalized velocity vector $\mathbf{u} = (u_1, \ldots, u_9)^T$, the equations of motion take the form:

$$\mathbf{M}\,\dot{\mathbf{u}} + \mathbf{G}\,\mathbf{u} + \mathbf{F}_{\text{non-linear}} = \mathbf{F}_{\text{ext}} \tag{2.22}$$

where M and G are the inertia and gyroscopic matrix, given by:

$$M = \begin{bmatrix} \mathrm{diag}(m) & 0 & 0 \\ 0 & J^{+} & \begin{pmatrix} J_{a1}p_1 & J_{a2}p_2 & J_{a3}p_3 \\ J_{a1}q_1 & J_{a2}q_2 & J_{a3}q_3 \\ J_{a1}r_1 & J_{a2}r_2 & J_{a3}r_3 \end{pmatrix} \\ 0 & \begin{pmatrix} J_{a1}p_1 & J_{a2}p_2 & J_{a3}p_3 \\ J_{a1}q_1 & J_{a2}q_2 & J_{a3}q_3 \\ J_{a1}r_1 & J_{a2}r_2 & J_{a3}r_3 \end{pmatrix}^T & \begin{pmatrix} J_{a1} & 0 & 0 \\ 0 & J_{a2} & 0 \\ 0 & 0 & J_{a3} \end{pmatrix} \end{bmatrix} \tag{2.23}$$

$$G = -G^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \begin{matrix} 0 & \sum_1^3 A_i & \sum_1^3 B_i \\ -\sum_1^3 A_i & 0 & \sum_1^3 C_i \\ -\sum_1^3 B_i & -\sum_1^3 C_i & 0 \end{matrix} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The vector of non-linear terms is derived entirely in Appendix A. It involves Coriolis and centripetal terms in (u_4,u_5,u_6). As for the external forces, they are given by the contribution of the resultant constraint forces and torques at the interfaces between the rigid and the flexible parts and the contributions of the motor torques delivered to each wheel. Using our notation, therefore, we get:

$$(F_{\text{external}})_r = {}^{N}V_r^{B+} \bullet [f^{\text{body}}_{\text{external}} - f^{\text{body}}_{\text{constraint}}] +$$

$$ {}^{N}\mathbf{W}_r^{B} \bullet \left[\tau^{body}_{external} - \tau^{body}_{constraint} - \tau_7\, \mathbf{w}_{11} - \tau_8\, \mathbf{w}_{21} - \tau_9\, \mathbf{w}_{31} \right] + $$
$$ - {}^{B}\omega_r^{W1} \bullet \left[\tau_7 \;\; \mathbf{w}_{11} \right] + {}^{B}\omega_r^{W1} \bullet \left[\tau_8\, \mathbf{w}_{21} \right] + {}^{B}\omega_r^{W1} \bullet \left[\tau_9 \;\; \mathbf{w}_{31} \right] $$

(2.24)

which, considering that the constraint forces and torques will vanish after assembling with the rest of the structure, result in:

$$ (F_{external})_r = 0 \qquad (r=1,2,3) $$

$$ (F_{external})_4 = - \sum_{i=1}^{3} t_{6+i}\, p_i $$

$$ (F_{external})_5 = - \sum_{i=1}^{3} t_{6+i}\, q_i $$

(2.25)

$$ (F_{external})_6 = - \sum_{i=1}^{3} t_{6+i}\, r_i $$

$$ (F_{external})_r = \tau_i \qquad (r=7,8,9\,;\; i=1,2,3) $$

where τ_i is the i-th torque delivered by the i-th motor to the i-th wheel. Therefore, any torque which is delivered to each wheel by its motor, is also transmitted with the opposite sign to the base body and vice-versa, thereby providing a means for controlling the attitude of the spacecraft. This can be done actively, by feeding back base-body attitude angles and an application of this method will be given in chapter 4.

Denoting with $\mathbf{q}_s = (\mathbf{q}_t, \mathbf{q}_r)^T$ and w the terms related to the translational, rotational and wheel spin configuration degrees of freedom, these equations of motion can compactly be written in the form in which we will use them during assembling with the rest of the structure.
The inertia matrix here is not configuration dependent. However, the presence of the gyroscopic matrix G takes into account the capability for this body to store a finite amount of angular momentum. This peculiarity, as we shall see later on, changes the eigenstructure of the system in a particular way.

2.4 DERIVATION OF THE EQUATIONS OF MOTION OF THE FLEXIBLE PART USING THE FINITE ELEMENT METHOD.

A total of five finite element reference nodes are defined along the MACE test article. We call them F1 to F5. These points are evenly distributed along the bus. This layout is also aligned with the X axis of the (X,Y,Z) reference frame. The spacecraft bus is therefore divided into 4 equal flexible beams separated from each other by rigid elements. Each finite element node has 6 degrees of freedom/node before assembly, and for each element a Bernoulli-Euler schematization is chosen. If nele is the number of elements/beam, the number of nodes/beam is nnod=nele+1, and the number of degrees of freedom/beam is ndof=6 nnod. For all the flexible part, we have nflex= 6 numnp (where numnp=total number of nodal points). Our model has a total of 7 internal rigid body degrees of freedom (irbdof=7): 4 payload relative angles and 3 relative wheel speeds. Because a body in space has 6 rigid body degrees of freedom the total number of rigid body degrees of freedom is RB = irbdof + 6 , and the total number of degrees of freedom for the model nfree=irbdof+nflex. If nele=1, nfree=37 and RB=13.

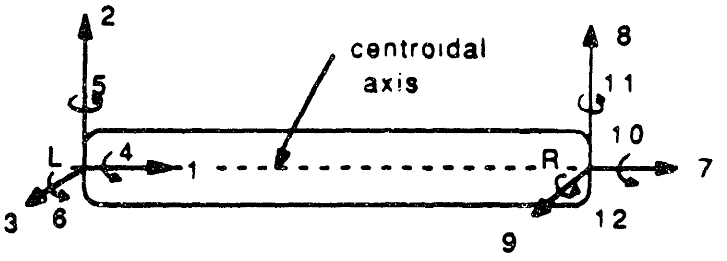


Figure (2.3). Finite element degrees of freedom

As from Figure (2.3), the degrees of freedom of each node are the (x,y,z) translations along the (X,Y,Z) axes and the infinitesimal rotations (θ_1, θ_2, θ_3)about the local axes at the node. All the nodes are numbered from left to right. This choice results in the inertia matrix Mele (12x12) and in the stiffness matrix Kele (12x12) of each single element to be given by:

$$M_{ele} = \frac{\rho AL}{420}\begin{bmatrix} 140 & 0 & 0 & 0 & 0 & 0 & 70 & 0 & 0 & 0 & 0 & 0 \\ 0 & 156 & 0 & 0 & 0 & 22L & 0 & 54 & 0 & 0 & 0 & -13L \\ 0 & 0 & 156 & 0 & -22L & 0 & 0 & 0 & 54 & 0 & 13L & 0 \\ 0 & 0 & 0 & \frac{140J_x}{A} & 0 & 0 & 0 & 0 & 0 & \frac{70J_x}{A} & 0 & 0 \\ 0 & 0 & -22L & 0 & 4L^2 & 0 & 0 & 0 & -13L & 0 & -3L^2 & 0 \\ 0 & 22L & 0 & 0 & 0 & 4L^2 & 0 & 13L & 0 & 0 & 0 & -3L^2 \\ 70 & 0 & 0 & 0 & 0 & 0 & 140 & 0 & 0 & 0 & 0 & 0 \\ 0 & 54 & 0 & 0 & 0 & 13L & 0 & 156 & 0 & 0 & 0 & -22L \\ 0 & 0 & 54 & 0 & -13L & 0 & 0 & 0 & 156 & 0 & 22L & 0 \\ 0 & 0 & 0 & \frac{70J_x}{A} & 0 & 0 & 0 & 0 & 0 & \frac{140J_x}{A} & 0 & 0 \\ 0 & 0 & 13L & 0 & -3L^2 & 0 & 0 & 0 & 22L & 0 & 4L^2 & 0 \\ 0 & -13L & 0 & 0 & 0 & -3L^2 & 0 & -22L & 0 & 0 & 0 & 4L^2 \end{bmatrix}$$

(2.26)

$$K_{ele} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EJ_z}{L^3} & 0 & 0 & 0 & \frac{6EJ_z}{L^2} & 0 & -\frac{12EJ_z}{L^3} & 0 & 0 & 0 & \frac{6EJ_z}{L^2} \\ 0 & 0 & \frac{12EJ_y}{L^3} & 0 & \frac{6EJ_y}{L^2} & 0 & 0 & 0 & \frac{12EJ_y}{L^3} & 0 & \frac{6EJ_y}{L^2} & 0 \\ 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 \\ 0 & 0 & \frac{6EJ_y}{L^2} & 0 & \frac{4EJ_y}{L} & 0 & 0 & 0 & \frac{6EJ_y}{L^2} & 0 & \frac{2EJ_y}{L} & 0 \\ 0 & \frac{6EJ_z}{L^2} & 0 & 0 & 0 & \frac{4EJ_z}{L} & 0 & \frac{6EJ_z}{L^2} & 0 & 0 & 0 & \frac{2EJ_z}{L} \\ -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{12EJ_z}{L^3} & 0 & 0 & 0 & \frac{6EJ_z}{L^2} & 0 & \frac{12EJ_z}{L^3} & 0 & 0 & 0 & \frac{6EJ_z}{L^2} \\ 0 & 0 & \frac{12EJ_y}{L^3} & 0 & \frac{6EJ_y}{L^2} & 0 & 0 & 0 & \frac{12EJ_y}{L^3} & 0 & \frac{6EJ_y}{L^2} & 0 \\ 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \\ 0 & 0 & \frac{6EJ_y}{L^2} & 0 & \frac{2EJ_y}{L} & 0 & 0 & 0 & \frac{6EJ_y}{L^2} & 0 & \frac{4EJ_y}{L} & 0 \\ 0 & \frac{6EJ_z}{L^2} & 0 & 0 & 0 & \frac{2EJ_z}{L} & 0 & -\frac{6EJ_z}{L^2} & 0 & 0 & 0 & \frac{4EJ_z}{L} \end{bmatrix}$$

where $L = L_{beam}$ /nele is the length of the element, ρ is the material density, E is the Young's modulus, G is the shear modulus, A is the cross sectional area, and J_x,J_y, and J_z are the section's area moments of inertia.

Given the distributed shape of the rigid bodies to be assembled with the elastic members, a master-slave relationship holds between the displacements of the reference points belonging to the flexible part and those of the point belonging to the rigid body. This is done to allow for relative displacements between one attachment point and another. Therefore, if **w** is the vector of translational displacements and θ is the vector of rotational displacements at any point Pi (coincident with the attachment point of a flexible beam to a rigid body, on which the reference node is at Qi), the following relationship holds:

$$\begin{pmatrix} \mathbf{w} \\ \theta \end{pmatrix}_{Pi} = {}^{Qi}T^{Pi} \begin{pmatrix} \mathbf{w} \\ \theta \end{pmatrix}_{Qi} = \begin{bmatrix} I & (-{}^{Qi}\mathbf{r}^{Pi})^{x} \\ 0 & I \end{bmatrix} \begin{pmatrix} \mathbf{w} \\ \theta \end{pmatrix}_{Qi} \tag{2.27}$$

where ${}^{Qi}\mathbf{r}^{Pi}$ is the radius vector from Pi to Qi.

The construction of the model proceeds as follows:

1) The matrices Mele and Kele are first built for each member, in the local reference frame, each being a 12x12 matrix;

2) the M_{loc} and K_{loc} matrices for each flexible beam are built in the local frame, each being of dimension 6 x (nele + 1);

3) the global mass and stiffness matrices are assembled into the global ones K_{flex} and M_{flex} by taking into account the presence of any rigid link, and using the transformation of equation (2.27). In other words, introducing for each element the matrices (L=left node of element, R=right node) $T_{(i,j)} = \begin{bmatrix} T_i^L & 0 \\ 0 & T_i^R \end{bmatrix}$ for (i,j) = 1,...,[6 x (nele+1)] we have that $K_{flex}(i,j) = T^T_{(i,j)} K_{loc}(i,j) T_{(i,j)}$ and $M_{flex}(i,j) = T^T_{(i,j)} M_{loc}(i,j) T_{(i,j)}$.

4) The mass and inertia of every rigid body, i.e., the intermediate rigid elements and the articulated part, is then lumped into the global inertia matrix. This is done by writing $M_{global}(i,j) = M_{flex}(i,j) + M_k$ for (i,j) = 1,...,(6 for rigid body, 8 for articulated body) and for k = 1,..., (Number of rigid and articulated bodies).

5) The order of the state vector is re-arranged as in:

$$\mathbf{q} = [(x,y,z,\theta_1, \theta_2, \theta_3)_{F1}, \ldots,(x,y,z,\theta_1, \theta_2, \theta_3)_{Fn}, (\varphi, \theta)_1, (\varphi, \theta)_2, (w_1, w_2, w_3)]^T = [\mathbf{q}_e\ \mathbf{q}_o\ \mathbf{q}_w]^T$$

where the subscripts e,o and w stand for "elastic","internal rigid body", " wheel rates" degrees of freedom. In this way, the equations of motion of the spacecraft, with flexibility, torque-wheel attitude controllers and independent rigid payloads take the more general form:

$$\begin{bmatrix} M_{ee} & M_{eo}(q_o) & M_{ew} \\ M_{oe}(q_o) & M_{oo}(q_o) & M_{ow} \\ M_{we} & M_{wo} & M_{ww} \end{bmatrix} \begin{pmatrix} \ddot{q}_e \\ \ddot{q}_o \\ \dot{q}_w \end{pmatrix} + \begin{pmatrix} [G_{ee}(\dot{q}_w) + D_{ee}]\,\dot{q}_e + K_{ee}\,q_e \\ 0 \\ 0 \end{pmatrix} + $$
$$+ \begin{pmatrix} \mathbf{F}_e(q_o, q_e, \dot{q}_o, \dot{q}_e) \\ \mathbf{F}_o(q_o, q_e, \dot{q}_o, \dot{q}_e) \\ 0 \end{pmatrix}_{N\,L.} = \begin{pmatrix} \mathbf{F}_e \\ \mathbf{F}_o \\ \mathbf{F}_w \end{pmatrix}_{\text{Control}}$$

(2.28)

where the subscripts "N.L." and "control" stand for the non-linear and the actuation forces respectively.

Clearly, the inertia matrix is symmetric, positive definite, and configuration dependent. Conversely, the stiffness matrix is symmetric and positive semi-definite. The rigid body modes, in fact, span the null space of the stiffness matrix. These are the translations and rotations of the spacecraft along the (X,Y,Z) axes and the motions of the articulated and rotating parts.

The modal analysis on the previous equation considering only the homogeneous, undamped and non-gyroscopic part gives a matrix of mode-shapes Φ of dimension (nfree x nfree) and a matrix of eigenfrequencies $\Omega = \mathrm{diag}(0,\ldots,0,w_1,w_2,\ldots,w_{nflex})$, with $w_i > 0$, which satisfy:

$$(K - M\Omega^2)\,\Phi = 0 \qquad (2.29)$$

The columns of Φ contain both the linear and the angular displacements of the finite element nodes in modal form and the rigid body modes. Whenever one of the payloads is locked to the bus, the equivalent internal rigid body modes are zero, which is the same as deleting the equivalent rows and columns from the M and K matrices in global coordinates. The state vector can be ordered so that the corresponding rigid body mode shapes are given by:

$$\Phi_x = [\; I_{(ndofxndof)} \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \;]^T$$
$$\Phi_y = [\; 0 \mid I_{(ndofxndof)} \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \;]^T$$

$$\Phi_z = [\, 0 \mid 0 \mid I_{(ndof \times ndof)} \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \,]^T$$

$$\begin{aligned}
\Phi_{\theta 1} &= [\, 0 \mid -r_{yi} \mid r_{zi} \mid I_{(ndof \times ndof)} \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \,]^T \\
\Phi_{\theta 2} &= [\, -r_{xi} \mid 0 \mid -r_{zi} \mid 0 \mid I_{(ndof \times ndof)} \mid 0 \mid 0 \mid 0 \mid 0 \,]^T \\
\Phi_{\theta 3} &= [\, -r_{xi} \mid r_{yi} \mid 0 \mid 0 \mid 0 \mid I_{(ndof \times ndof)} \mid 0 \mid 0 \mid 0 \,]^T
\end{aligned}$$

$$\begin{aligned}
\Phi_{\theta\varphi 1} &= [\, 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid I_{(2 \times 2)} \mid 0 \mid 0 \,]^T \\
\Phi_{\theta\varphi 2} &= [\, 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid I_{(2 \times 2)} \mid 0 \,]^T \\
\Phi_{w} &= [\, 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid I_{(3 \times 3)} \,]^T
\end{aligned} \tag{2.30}$$

where $\mathbf{r}_i = (r_x, r_y, r_z)_i^T$ is the matrix of 3 column vectors and nnod columns representing the components of the vector from the origin of the (X,Y,Z) frame to the i-th finite element node located on the structure. Note that these displacement functions are not orthogonal, however they are linearly independent and can be normalized and used as a basis. It has been proved in Ref.[8,9,10], with reference to gyroelastic vehicles, that the inclusion of the gyroscopic matrix in the eigenproblem yields a new set of rigid body modes. For a gyroelastic vehicle there exist, in fact, modes which correspond to zero frequencies but do not have zero deformational energy. They are also called scleromorphic modes and describe uniform rotations of the vehicle with the elastic parts in a deformed state. With the mode shapes of equation (2.30), it is clear that because there are only torque actuators available, the translational degrees of freedom of the base body are in fact uncontrollable from the inputs. They are also undisturbable. In reality, when articulated payloads are also present, the translational dynamics of the vehicle's center of mass are coupled to the rotational dynamics of the payloads. This is equivalent to say that, for a non-CG mounted payload, its inertial rotation is coupled to the translation of the bus, in the zero-g model. This means that the non-observable/non-controllable dynamics are linear combinations of the rigid body modes enumerated above. Furthermore, one should in reality also take into account the presence of instrumentation cables, which further couple the translational and rotational degrees of freedom of the vehicle in zero-g and suppress six rigid-body modes.

The damping matrix is a symmetric, positive, semi-definite damping matrix introduced to represent material damping in the structure. As specified in the engineering requirements data, 1% of uniform damping is desired in all modes of interest. This can generally be accomplished by introducing (see Ref.[11]) a damping matrix of the form (Raleigh damping) $D = \alpha M + \beta K$ where M and K represent the assembled mass and stiffness matrix of the flexible part before lumping the mass of the rigid bodies. Normally, the constants α and β would have to be chosen to produce specified modal damping factors for two given modes, i.e. computed by solving the system of equations: $\alpha + \beta\, \omega_i^2 = 2\, \omega_i\, \zeta_n$ for given damping ratio ζi . The α M term gives a contribution in the damping factor of the i-th mode which is inversely proportional to the i-th natural frequency, while the β K term gives a contribution in the damping factor of the i-th mode which increases linearly with the i-th natural frequency. Therefore, α and β are coefficients which "shape" the location of the eigenfrequencies in the left-hand complex plane.. Unfortunately, this method gives unrealistic damping levels for the higher modes, i.e. far away from the specified value (for example, some modes can be critically damped). An alternative is to chose the damping matrix on the basis of the mode shapes, as follows. Substitute $q = \Phi\, \eta$,where η is the time dependent vector of modal coordinates, in the equations of motion. The choice $D= 2\, \zeta\, (\Phi^{-T}\, \Omega\, \Phi^{-1}$) will guarantee uniform damping equal to z in all modes.

The equations of motion therefore in modal form become:

$$\ddot{\eta} + 2\, \zeta\, \dot{\eta} + \omega_i^2\, \eta = \Phi^T\, B_c\, \mathbf{F}_{ext} \tag{2.31}$$

Note that the rigid body dynamics can be written as:

$$\ddot{\eta}_r = B_r\, \mathbf{F}_{ext} \tag{2.32}$$

where r=1,...,13 and B_r represents a forcing distribution matrix of dimension (13x7), because there are 7 inputs. Since only torque actuators are available, it has a left nullspace of dimension 3 (the three vehicle translations). Following the steps of Ref [12], carrying out a singular value decomposition (SVD) of B_r yields $B_r = U\, \Sigma\, V^T$ with U(13x13) and V(7x7) unitary and S(13x7) is of the form:

$$\Sigma = \begin{bmatrix} \mathrm{diag}(\sigma_i) & 0 \\ 0 & 0 \end{bmatrix} \tag{2.33}$$

with i=1,...,10, and $\sigma_i > 0$. If we redefine the rigid body modal coordinates as $\eta_{r\,new} = U^{-1}\, \eta_r$ and $B_{rnew} = U^{-1}\, B_r$ it is clear that the new control distribution matrix has zero entries correspondingly to the uncontrollable rigid body modes. This procedure can be used to discard any undisturbable coordinates that do not participate in the control performance.

A few comments are now necessary before proceeding with the analysis.

First, note that we have treated the dynamics as linear-time invariant and non-gyroscopic. One could also say that configuration dependent terms in the inertia matrix and gyroscopic and non-linear terms in the equations of motion are first order perturbational quantities and, on the basis of this assumption, they may be neglected from the solution of the eigenvalue problem. Note that in general the presence of devices capable of storing internal angular momentum influences the q_e degrees of freedom only.through the skew-symmetric matrix Gee.and the mode-shapes are obviously changed. An exhaustive analysis of the effects of distribution of internal angular momentum on a flexible structure is given by Hughes et al.in Ref [8,9,10].
Note, however, that the equations of motion can be re-written as:

$$M_{ee}\,\ddot{q}_e + M_{er}\,\ddot{q}_r + (G_{ee} + D_{ee})\,\dot{q}_e + K_{ee}\,q_e = F_{NL}e + F_{ext}e$$

$$M_{re}\,\ddot{q}_e + M_{rr}\,\ddot{q}_r = F_{ext\,r} + F_{cr} \tag{2.34}$$

and, using the second equation gives:

$$(M_{ee} - M_{er}\,M_{rr}^{-1}\,M_{re})\,\ddot{q}_e + (G_{ee} + D_{ee})\,\dot{q}_e + K_{ee}\,q_e = F_{N.L}e + F_{ext}e - M_{er}\,M_{rr}^{-1}\,(F_{ext\,r} + F_{cr}) \tag{2.35}$$

in the flexible dynamics. The second term in the inertia matrix is clearly configuration dependent. However when the changes of configuration of the spacecraft do not alter dramatically the eigenstructure of the system, this term can be considered as a first order perturbation in the time series expansion of the M_{ee} matrix. This dynamics is linear time invariant but not asymptotically stable in general (because of the presence of rigid body modes).

Second, M_{ee}, M_{er}, M_{re} and M_{rr} are dependent on the configuration of the pointing payloads. Therefore, the eigenproblem itself is configuration dependent and the model is linear time-varying only if the non-linear terms are discarded.

Third, because of the particular structure of the spacecraft, the sub-partitions M_{ow} and M_{wo} are matrices with zero entries. They would be non-zero only if the inertial platform was mounted on an articulated part of the structure, or if instead of torque wheels we had Control Moment Gyros. Therefore, the equations of motion in the wheels degrees of freedom are in a sense uncoupled from the rest of the structure. This facilitates considerations on the cancellation of the gyroscopic effects by means of active control, better explained in chapter 4.

Fourth, the presence of a slewing flexible appendage, instead of a rigid payload, would introduce more complexity. In particular, additional entries in $\mathbf{M}_{oe}$ would be dependent on the configuration of the appendage, and additional entries in Kee would give rise to motion stiffening terms.

Fifth, components of the vector of Coriolis and centripetal terms arising from payload motion also excite the elastic coordinates $\mathbf{q}_e$. Consistent with the "ruthless" linearization approach, presented in Ref.[13,14], the equations of motion of the payloads can be simplified to:

$$\mathbf{M}_{oo}\,\ddot{\mathbf{q}}_o + \mathbf{F}_{n.l.o} = \mathbf{F}_{external} \tag{2.36}$$

i.e., the effect of the flexible dynamics is neglected when the payload moves below a certain empirical rotational speed limit. This is specified as follows: the magnitude of the spinning rate has to be one order of magnitude less than the fundamental bending frequency of the supporting body. Further considerations on the rigid-flexible coupling between the base body and the articulated payload are made in section 2.5.

Sixth, one must note that the presence of instrumentation cables, always present in the testing of structures, alters somehow the dynamics in the sense that the stiffness and damping characteristics of the structured are increased at low frequency. The analysis of such effect will not be considered here. However, Ref. [15] presents an experimentally derived curve which plots the applied force vs.the strain measured on a long thin copper cable. At negative strain ($\varepsilon = -10^{-3}$) the applied force, of the order of 1 N, is still positive because of the presence of wrinkles and stranding in the cable. This suggests that for very slow, precise maneuvers of the free-floating test article and for cables of a certain size, one should consider a more careful modelling.

The model of the test article derived until now will be from now on termed the zero-g model. As we have seen, it is a model which is linear time-invariant in the structural dynamics, but non-linear in the articulated dynamics. In particular, it is non-linear in the torque wheel rates and in the payload(s) angle and rates.

At this stage, the fully non-linear model can be linearized about a given payload configuration, and the wheel relative angular velocities can be treated as constants with the additions of small perturbations. This results in the LTI (Linear-Time-Invariant) model, which can therefore be cast in the familiar state-space form:

$$\dot{x} = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}(G+D) \end{bmatrix} x + \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix} u$$

(2.37)

$$y = \begin{pmatrix} C_q & C_{\dot{q}} \end{pmatrix} x + D\,u$$

where $x = (q\ ,\ \dot{q})^T$ is the state vector.

In this description there are 7 inputs, namely:
- 4 payload gimbal torques (one in-plane, the other out-of-plane, for each payload);
- 3 torque-wheels electric motor torques;
- (additionally, when the active piezoelectric member is operational, there is also a differential torque acting between the nodes at which the member is attached).

The following measurements are used:
- payload relative angle;
- payload inertial rate, given by the sum of the angle at the extreme node of the bus plus the relative angular rate at the gimbal torque location;
- attitude angular rates of the spacecraft, given by the three nodal angular displacements at the central node;
- three-components of accelerations at each node;
- (additionally, when the active piezoelectric member is operational, there is also an angular gradient between the nodes at which the member is attached simulating the output of strain gages).

This LTI model is used in the derivation of the various inputs to outputs transfer functions, task which is carried out in the following section.

To conclude this section, the equations of motion of the zero-g model (and later of the one-g model) were programmed in Matlab programming language and implemented in a Sun station. The choice of using Matlab was made because it has been proven to be a very efficient matrix algebra computation code, ideal therefore for dynamic analysis of complex dynamic systems. The code developed for this purpose allows one to enter interactively data concerning the geometry of the system and initial conditions such as nominal wheel speeds and payload gimbal angles relative to the base bus. The data were taken from the Development Model used for the MACE studies at the Space Engineering Research Center at MIT.

2.5 ANALYSIS OF THE OPEN-LOOP DYNAMICS OF THE ZERO-G MODEL

2.5.1 Undamped, unforced dynamics.

In this section, we will analize the open-loop dynamics of the zero-g model. To obtain the exact system frequencies and mode shapes, a configuration dependent eigenvalue problem has to be solved. The eigenanalysis is performed for both cases in which the payloads are hinged and locked to the base structure.

To do so, first we solve the eigenproblem, and this implies that we must linearize the dynamics, by considering the state vector as the sum of a nominal value plus a perturbation quantity, as in $y = y_o + \delta y$, where δ denotes a perturbation from the equilibrium. The nominal attitude of the spacecraft in inertial space is the zero vector, the nominal elastic displacements $\mathbf{q_o}$ are also zero, the nominal torque wheel rates are also zero (or equal to a constant value for purposes of studying the behavior of the system during precessional motion) and finally the nominal payload gimbal angles are user-defined, corresponding to a given pointing configuration. Afterwards, we compute the structural frequencies and the mode shapes for the system in the form of equation (2.28), but considering the undamped, unforced system and neglecting non-linear and gyroscopic terms. Therefore it is in the form:

$$(K - M\,\Omega^2)\,\Phi = 0 \qquad (2.38)$$

Table [2.5] shows a comparison of the undamped structural frequencies of the zero-g model in 2 cases: when payload #1 and payload #2 are hinged, and when only payload #1 is locked, for two extreme payload configurations: one is a "all in-plane" configuration, with payloads pointing vertically down at right angles with respect to the X axis (90 degrees in-plane and 0 degrees out-of-plane) , the other is a "all out-of-plane" configuration, in which the radius vector of the payload's CG has a component along the Z axis as well (45 degrees in-plane and 45 degrees out-of-plane) Note how the frequencies (and, of course, the mode shapes) change when the payloads are locked instead of hinged. For instance, when the payloads are locked to the bus, the structural frequencies decrease in magnitude with respect to the case when the payloads are hinged, because the equivalent inertia seen by the flexible part has actually increased. This change is more evident in the lowest frequencies, and it also varies with the configuration about which we linearize the model. Note also the way the frequencies are clustered in packets of closely spaced frequencies. This is typical of most space structures. Because of the configuration of the spacecraft, with two massive non CG-mounted hinged/locked rigid bodies articulated to the flexible base, there is a significant bending/torsion coupling in the horizontal plane. Also, asymmetric modes predominate, since the mass/inertia distribution is not symmetrically located about the origin of the X-Y-Z frame. However, one should always be cautious, and warned that, as a rule, in finite element analysis the upper half of the numerically obtained frequency spectrum is not correct. This error is due only to discretization. Errors due to modelling or parameter values may make even the lowest eigenfrequency and its eigenvector incorrect.

The eigenstructure also changes when a finite amount of angular momentum is stored in the wheels. But this analysis is carried out in the following section.

Table 2.5. Structural frequencies in Hz of zero-g model. Hinged vs. Locked.

mode number	all-in-plane p.#1 hinged	all-out-of-plane p.#1 hinged	mode shape *	mode number	all-in-plane p.#1 locked	all-out-of-plane p.#1 locked
14	1.5137	1.5177	1st v.b.	12	1.6328	1.6338
15	1.8233	1.8199	1st h b/t	13	1.7278	1.7178
16	2.5714	2.5644	2nd h b/t	14	2.7169	2.7118
17	5.3712	5.2251	2nd v b	15	5.4907	5.3994
18	7.6958	7.8297	1st t	16	6.4753	6.4923
19	8.1434	7.8800	2nd t	17	7.9686	8.1411
20	9.0227	9.1459	3rd v b	18	9.6580	9.4801
21	10.8332	10.9813	3rd h b/t	19	11.7373	12.2286
22	11.8136	11.9110	4th v b	20	12.6402	12.6224
23	11.8810	12.3277	4th h b/t	21	13.9466	13.9566
24	37.8817	37.8921	3rd t	22	37.7402	37.7403
25	39.1623	39.2047	5th h b/t	23	39.1570	39.2006
26	39.3982	39.3936	5th v b	24	40.6079	40.5296
27	41.6141	41.5159	6th v b	25	41.8738	41.8653
28	44.1339	44.2227	6th h b/t	26	44.0670	44.0760
29	47.0640	47.7613	7th h b/t	27	47.0621	47.7536
30	92.9527	88.8969	7th v b	28	72.6618	71.9388
31	116.4011	115.9661	8th v b	29	115.1362	113.7073
32	121.0165	120.2351	9th v b	30	117.6711	117.6707
33	129.1448	126.2835	10th v b	31	124.6395	123.7937
34	193.4474	193.4719		32	193.1601	193.1658
35	193.9108	194.0919		33	193.8929	194.0817
36	320.1595	317.9980		34	310.6938	310.5861
37	331.9648	330.6217		35	331.1780	329.9926

* v = vertical; h = horizontal; b = bending; t = torsion.

Table 2.6. Structural frequencies in Hz of zero-g model. Influence of stored angular momentum (H_v =1 Nm).

mode number	all-in-plane p.#1 hinged	all-in-plane p.#1 hinged H_v =1 Nm	mode shape *	mode number	all-in-plane p.#1 locked	all-in-plane p.#1 locked H_v =1 Nm
13	0	0.0839	precession frequency	11	0	0.0937
14	1.5137	1.5138	1st v.b.	12	1.6328	1.6328
15	1.8233	1.8216	1st h b/t	13	1.7278	1.7276
16	2.5714	2.5714	2nd h b/t	14	2.7169	2.7167
17	5.3712	5.3898	2nd v b	15	5.4907	5.5014
18	7.6958	7.6966	1st t	16	6.4753	6.4798
19	8.1434	8.1434	2nd t	17	7.9686	7.9701
20	9.0227	9.0224	3rd v b	18	9.6580	9.6650
21	10.8332	10.833	3rd h b/t	19	11.7373	11.7370
22	11.8136	11.8077	4th v b	20	12.6402	12.6523
23	11.8810	11.9050	4th h b/t	21	13.9466	13.9466
24	37.8817	37.8818	3rd t	22	37.7402	37.7403
25	39.1623	39.1623	5th h b/t	23	39.1570	39.1571
26	39.3982	39.3983	5th v b	24	40.6079	40.6081
27	41.6141	41.6150	6th v b	25	41.8738	41.8745
28	44.1339	44.1339	6th h b/t	26	44.0670	44.0670
29	47.0640	47.0640	7th h b/t	27	47.0621	47.0621
30	92.9527	92.9527	7th v b	28	72.6618	72.6618
31	116.4011	116.4012		29	115.1362	115.1362
32	121.0165	121.0166		30	117.6711	117.6712
33	129.1448	129.1449		31	124.6395	124.6395
34	193.4474	193.4474		32	193.1601	193.1602
35	193.9108	194.9109		33	193.8929	193.8930
36	320.1595	320.1596		34	310.6938	310.6939
37	331.9648	331.9648		35	331.1780	331.1780

* v = vertical; h = horizontal; b = bending; t = torsion.

2.5.2 Effects of the Stored Angular Momentum on the Flexible Structure.

In order to see how the dynamics change when the wheels spin at a non-zero rate, we will solve the eigenproblem considering the LTI model in state space form. Even if the equations of motion for a gyroscopic system resemble those of a damped system, in the first case the matrix pre-multiplying the vector of generalized velocities is skew symmetric, while in the second case, when the damping is viscous the matrix is symmetric. Additionally, while in certain cases it is possible using a similarity transformation to diagonalize a viscously damped system (case of proportional damping), it is not possible to do the same for a gyroscopic system. The situation is more complicated when the system is damped and gyroscopic. One could use a perturbation analysis to quantify the deviation of frequencies and mode shapes of the damped-gyroscopic system from those of the system in equation (2.38). Reference [17] analyses exactly this case. Instead, we abandon any possibility of performing any classical modal analysis. Rather, as shown in Ref.[18,19], the system can indeed be diagonalized by transforming it into a convenient first order matrix form, and we consider the model in the state space form. The eigenvalue problem is therefore defined by:

$$(s\,I - A)\,\Psi = 0 \tag{2.39}$$

where the effect of damping (1% uniform) and the gyroscopic effects are present in the A matrix. The result of this analysis shows not only that the structural frequencies are aligned along the line of 1% uniform damping in the complex plane, as expected, but that one of the rigid body frequencies has been substituted by a non-zero one: the precessional frequency. This frequency represents the precession frequency of the vehicle, including the flexibility of the bus, when the torque wheels spin at a given rate. In general, the resultant angular momentum of this "gyric" distribution is arbitrarily oriented in inertial space.

2.5.2.1 Rough Estimates of the Precession Frequency.

For a gyrostat mounted on a rigid, non-spinning carrier of central principal inertia dyadic $\underline{\underline{J}} = \mathrm{diag}(J_i)$, i=1,2,3, if we denote with h the magnitude of the relative angular momentum vector of direction $\mathbf{a}$ in inertial space, we have that the equations of motion for the attitude perturbations θ are:

$$\underline{\underline{J}}\,\ddot{\theta} + \dot{\theta} \times (h\,\mathbf{a}) = \mathbf{0} \qquad (2.40)$$

The eigenvalue problem becomes :

$$\det\begin{bmatrix} s^2 J_1 & sh\,\mathbf{a}_z & -sh\,\mathbf{a}_y \\ -sh\,\mathbf{a}_z & s^2 J_2 & sh\,\mathbf{a}_x \\ sh\,\mathbf{a}_y & -sh\,\mathbf{a}_x & s^2 J_3 \end{bmatrix} = 0 \qquad (2.41)$$

By expanding the determinant in scalar form and removing the four zero frequencies, corresponding to an arbitrary reference attitude of the vehicle, a double non-zero root is left which can be written as:

$$\Omega_{precession} = h\sqrt{\frac{\mathbf{a}^T \underline{\underline{J}}\, \mathbf{a}}{\det \underline{\underline{J}}}} \qquad (2.42)$$

Note that this frequency has been derived under the assumption of rigid spacecraft and undamped motion.and that it lies on the jω axis of the complex plane. However, as the wheel's spin increases, and therefore the resultant stored angular momentum increases, the rigid model becomes invalid For a rigid spacecraft, if Jt is the transverse moment of inertia and $h = J_w\,\omega$ (one wheel only), one gets $\Omega_{precession} = \frac{h}{Jt}$. Taking Jt/J_w to be about 2, and measuring ω in rpm, the frequency in Hz is ν = rpm/30. Therefore, during spin-up, there will be resonance with most of the low-frequency modes of a conventional space structure, specially the anti-symmetric modes. In the MACE structure, the low frequency bending/torsion mode in the horizontal plane is a potential candidate for resonance when h increases.

When the presence of structural flexibility is taken into account, as when N_{app} flexible appendages are present, the equations of motion change. In the case of MACE, the appendages would be the flexible beams at the sides of the central node. In particular, following the steps of Ref.[9,10], assuming that the body reference frame is centered at the center of mass of the whole vehicle, and that vehicle translations are uncoupled from vehicle rotations (even if we know that this is not precisely the case for MACE, when the hinges are free), the unforced rotational equations of motion can be written as :

$$\underline{\underline{J}}\ \ddot{\theta} + \dot{\theta} \times (h\, a) + \sum_{i=1}^{N_{app}} H_{jn}\, \ddot{\eta}_{jn} + \dot{h} = 0$$

$$\ddot{\eta}_{jn} + (2\, \zeta_{jn}\, \omega_{jn})\, \dot{\eta}_{jn} + \omega^2_{jn}\, \eta_{jn} + \sum_{i=1}^{N_{app}} H^T_{jn}\, \ddot{\theta} = 0 \qquad (2.43)$$

$$(j = 1, \ldots, N_{retained\ modes} \quad ; \quad n = 1, \ldots, N_{appendages})$$

where the matrix $H_{jn} = \sum_{i=1}^{Nretained} \left[\int_{Appendages} r^x\, \Phi_{jn}(r)\, dm \right]$ is known as the modal angular momentum coefficient and represents the contribution of the angular momentum of the appendage to the rotational equation, and $\Phi_{jn}(r)$ is the mode shape of the j-th mode of the n-th appendage at the location r from the vehicle's center of mass. The assumption of uncoupled translations and rotations is a consequence of certain symmetry properties of the structure which make the modal linear momentum coefficient

$P_{jn} = \sum_{i=1}^{Nretained} \left[\int_{Appendages} \Phi_{jn}(r)\, dm \right]$ equal to zero. The matrix $\underline{\underline{J}}$ represents the inertia of the whole vehicle in the undeformed configuration and we assume a constant distribution of angular momentum, i.e., $\dot{h}$. Taking the Laplace transform of the last equation, assuming zero initial conditions, and zero external forces, and combining the resulting expressions, one obtains the characteristic equation:

$$\det \left[s^2 \underline{\underline{J}} - s\, h^x - s^2 \sum_{i=1}^{N_{retained}} \frac{s^2 H_{jn}\, H^T_{jn}}{(s^2 + 2\, \zeta_{jn}\, \omega_{jn} s + \omega_{jn}^2)} \right] = 0$$

(2.44)

Therefore the effects of flexibility enter exclusively through the last term, where ω_{jn} represent the natural frequencies of the constrained modes, i.e. the frequencies of the appendages with the base fixed in inertial space. Setting $s = i\,\omega$ ($i = \sqrt{-1}$), and expanding the determinant, one obtains the expression:

$$\Omega_{\text{precession with flexibility}} =$$

$$\sqrt{\frac{a^T \underline{\underline{J}}\, a\, h^2 + a^T \omega^2 \displaystyle\sum_{i=1}^{N_{\text{retained}}} \frac{H_{jn} H^T_{jn}}{(\omega_{jn}^2 - \omega^2 + i\, 2\, \zeta_{jn}\, \omega_{jn}\omega\,)}\; a\, h^2}{\det\left[\underline{\underline{J}} + \omega^2 \displaystyle\sum_{i=1}^{N_{\text{retained}}} \frac{H_{jn} H^T_{jn}}{(\,\omega_{jn}^2 - \omega^2 + i\, 2\, \zeta_{jn}\, \omega_{jn}\omega\,)}\right]}} \tag{2.45}$$

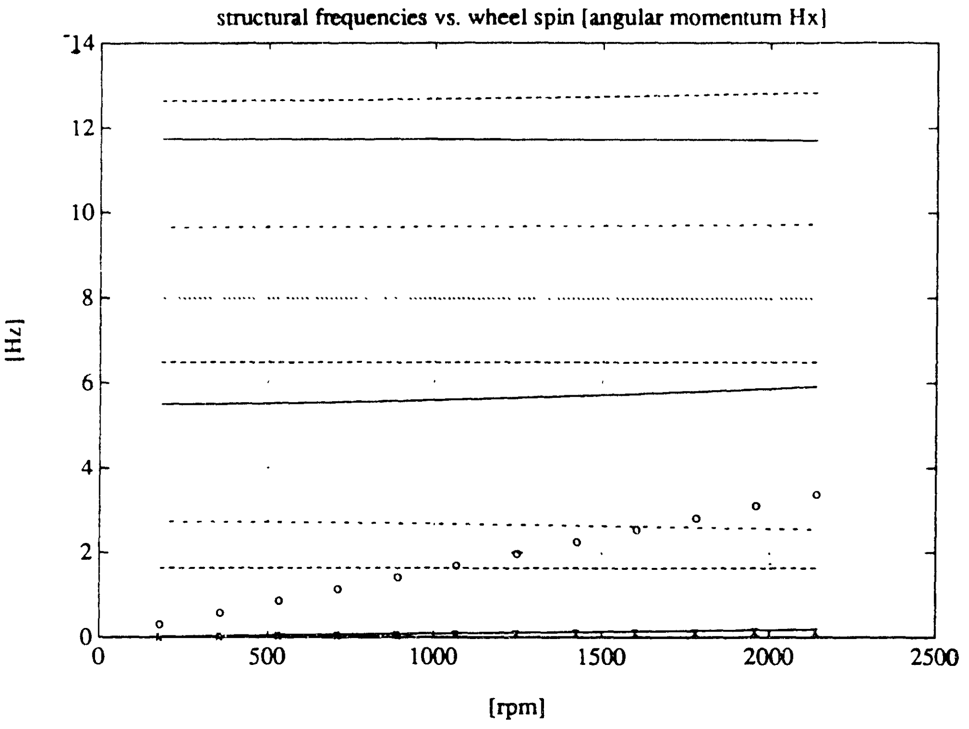
Note that the first term under the square root sign represents the precession frequency of the spacecraft considered as a rigid body while the second term represents the contribution due to the flexibility. As before, when the appendages are sufficiently light, this expression recovers the precession frequency of the rigid bus without flexible appendages. Similarly, when the spacecraft is rigid, the sums are zero and the inertia is the inertia of the spacecraft in the undeformed configuration. In terms of precession frequency, the flexible appendages therefore constrain the rigid body dynamics and this frequency is increased. We know also from Ref.[8,9] that the constrained frequencies are lower than the unconstrained ones. The precession frequency would be however lower of the first unconstrained frequency for small angular momentum. For example, when the resultant angular momentum is directed perpendicularly to the bus centerline (along the Y axis), the summation under the square root is inversely proportional to the bending stiffness of the appendage. Conversely, when the resultant angular momentum is directed along the bus centerline (along the X axis), it is inversely proportional to the torsional stiffness of the appendage. Table [2.6], shows the damped structural frequencies of the test article in two situations: no stored angular momentum, and for the angular momentum directed along the Y axis and with magnitude 1 Nm. This value of angular momentum was chosen so as to make the wheels spin at about 103 rpm, which is within the wheel's motor capabilities. Note the alteration in almost the whole eigenstructure, specially in the lower frequencies,

and the value of the first frequency is the value of the precession frequency obtained as predicted by the considerations of the last paragraph. In particular, for the case in which the payload #1 is hinged, the "rigid" precession frequency would be 0.0674 Hz , while the influence of flexibility increases the value to 0.0839 Hz. This is almost a rigid body mode. Therefore, as a rule of thumb, when the precession frequency is well below the first structural resonance, then the entire spacecraft behaves as rigid and one can compute the precession frequency using equation (2.42). Conversely, if the precession frequency lies within the first structural resonances of the spacecraft, then, being the structure flexible, the precession frequency must be computed using equation (2.45), where the inertia is the one of the rigid central body only and the appendages add stiffness through the second term in the square root. Increasing the wheels speed also increases h and therefore the precession frequency is increased. In the limit, the frequencies of the unconstrained modes approach those of the constrained appendages. However, the change in the structural frequencies is inappreciable when the wheels speed is limited to the actual maximum available speed delivered by the motors, i.e. a few hundred r.p.m. Loci of the structural frequencies when the angular momentum (in its 3 components) is made to increase from zero to a very high (however unrealistic) value, i.e. equivalent to thousands of r.p.m. for the wheel speed, nevertheless show that all the frequencies are slightly altered by this effect , some increased and others decreased.
Figures [2.4] to [2.9] show precisely these loci, where the frequencies are measured in Hz and the wheel's angular speed in rpm. In these graphs, the line denoted by crosses (+) represents the locus of the precession frequency evaluated according to equation (2.42) while the line denoted by circles (o) represents the same quantity but evaluated according to expression (2.45). Some observations ca be inferred from these loci. First, the slope of the curves is always positive, or zero, denoting stiffening of the relative modes, which tend to approach the constrained modes of the appendages. An exception to this behavior occurs when the resultant angular momentum is along the X axis, for which the mode at 2.7 Hz (2nd horizontal bending/torsion) is de-stiffened. Second, as the parameter on the abscissa of the diagrams is varied, some modes interchange their order, which means that some modes are more sensitive to others to the variation, and this probably depends on the symmetry characteristics of the vehicle. From a close-up examination of these parametric graphs, especially when the horizontal axis covers at least three orders of magnitude, one can see that some of the curves actually do touch, others do not. It has been found in the literature, Ref. [11], that when two approaching curves represent a symmetric and a skew-symmetric mode they touch at the crossing point, but they do not touch if both curves represent either symmetric or skew-symmetric modes. In particular, for low speeds, the

sum under the root sign in equation (2.45) is approximately inversely proportional to the bending/torsional stiffness, but for high speeds it tends to be directly proportional. This makes sense, since for high speeds the natural frequencies of the system approach those of the constrained modes. At the crossing points, the modal shape is actually transferred from one mode to another, going from a symmetric mode to a skew-symmetric one, as the precession frequency increases. Also, when this happens, the precession goes from prograde to retrograde (or viceversa). In particular, note that for MACE this crossing happens always starting from below from an un-symmetric mode. The curve which emanates from the origin is the precession mode locus, and it is well matched by equation (2.42), and this fact is well shown in the figures [2.4] to [2.9]. An extended analysis of the gyroscopic effects of a generic distribution of stored angular momentum on a flexible structure has already been made; see, for example, Ref. [8,9,10,11].

From this point on, and for reference, the value of the resultant stored angular momentum will nominally be fixed to zero.

Figure 2.4 and 2.5



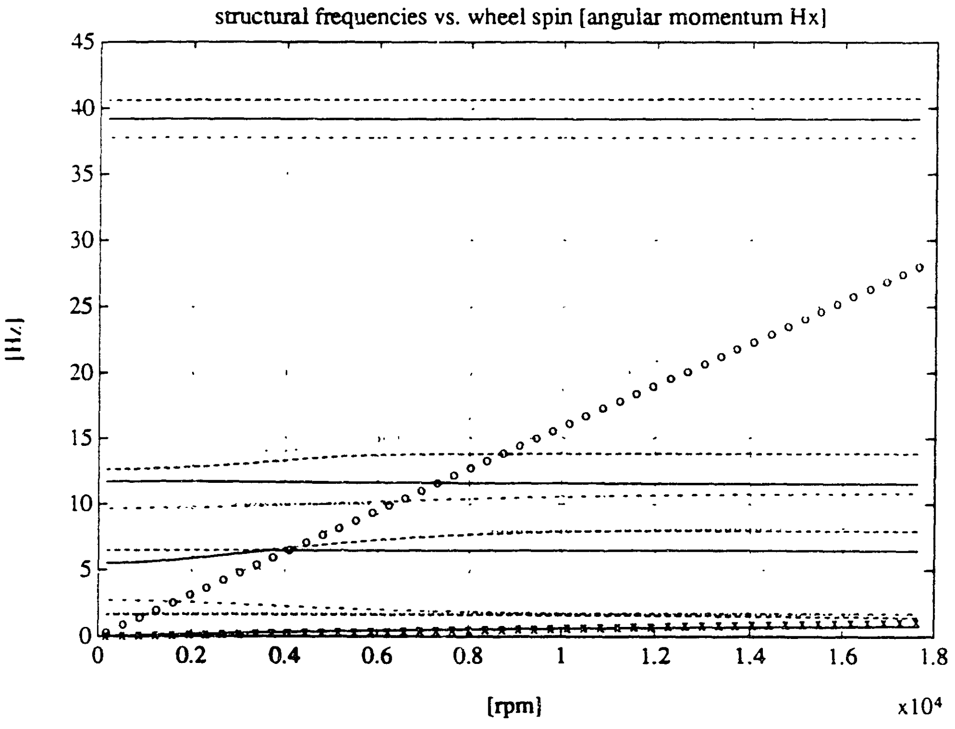
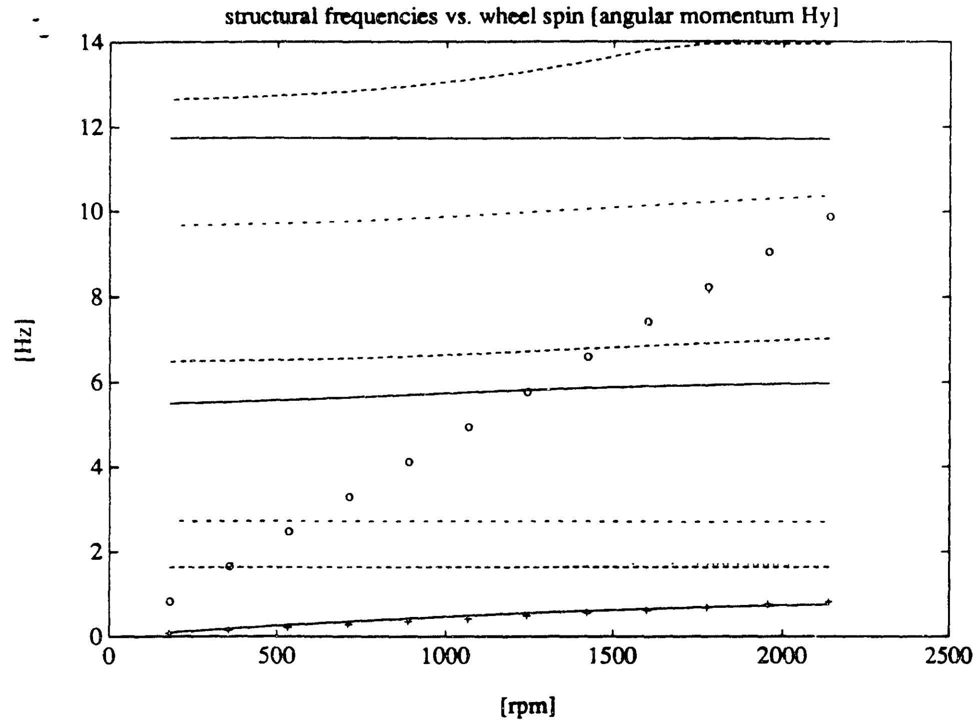


Figure 2.6 and 2.7



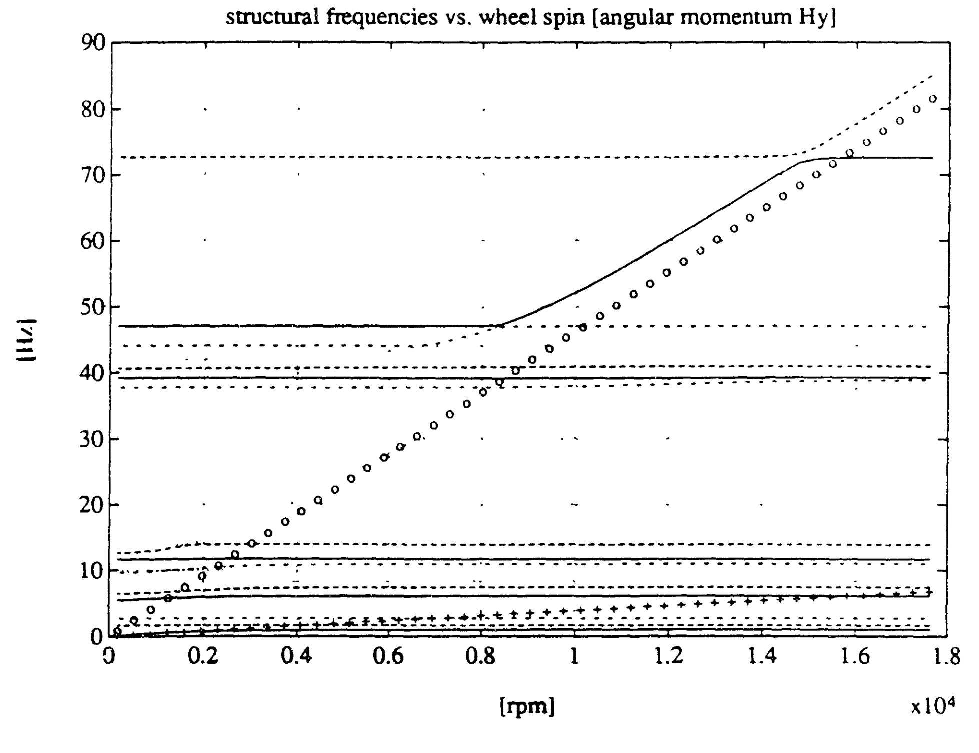
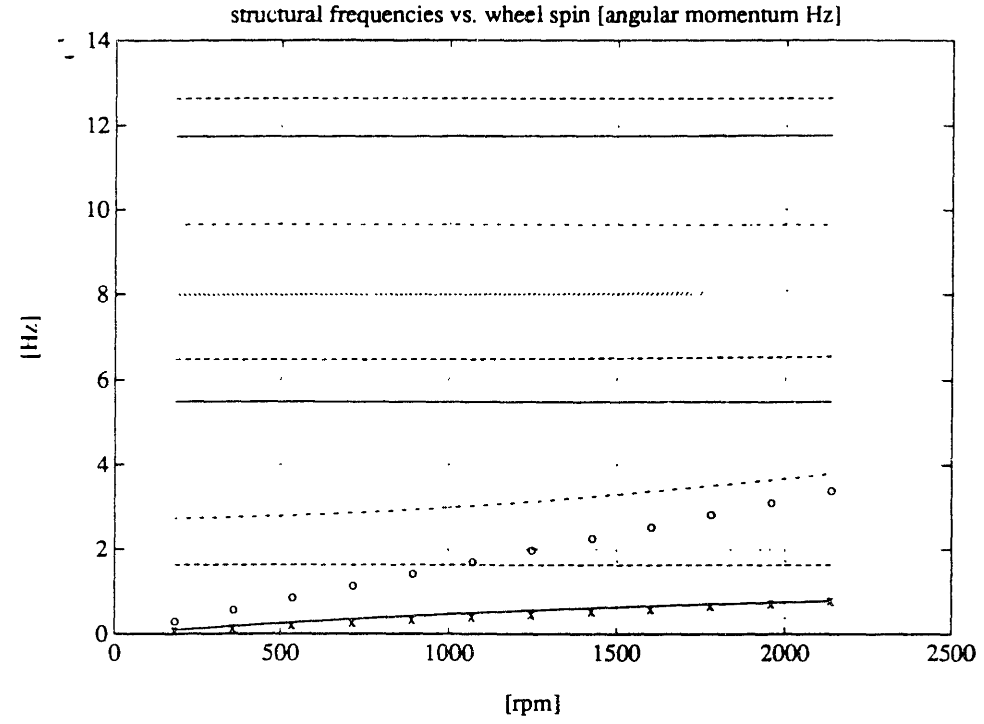


Figure 2.8 and 2.9



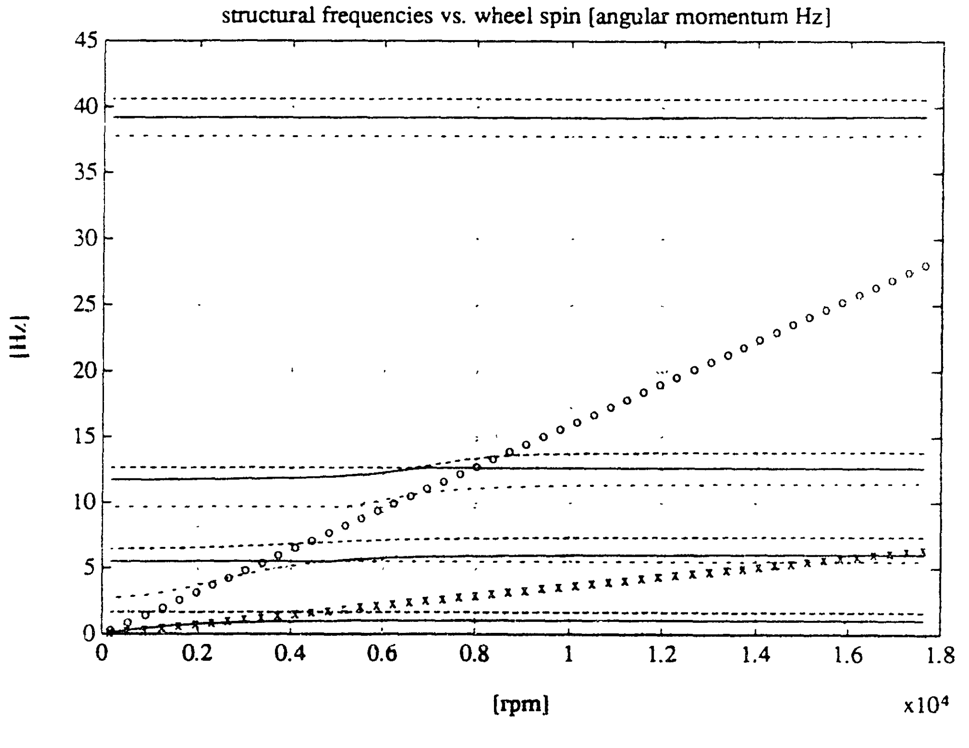
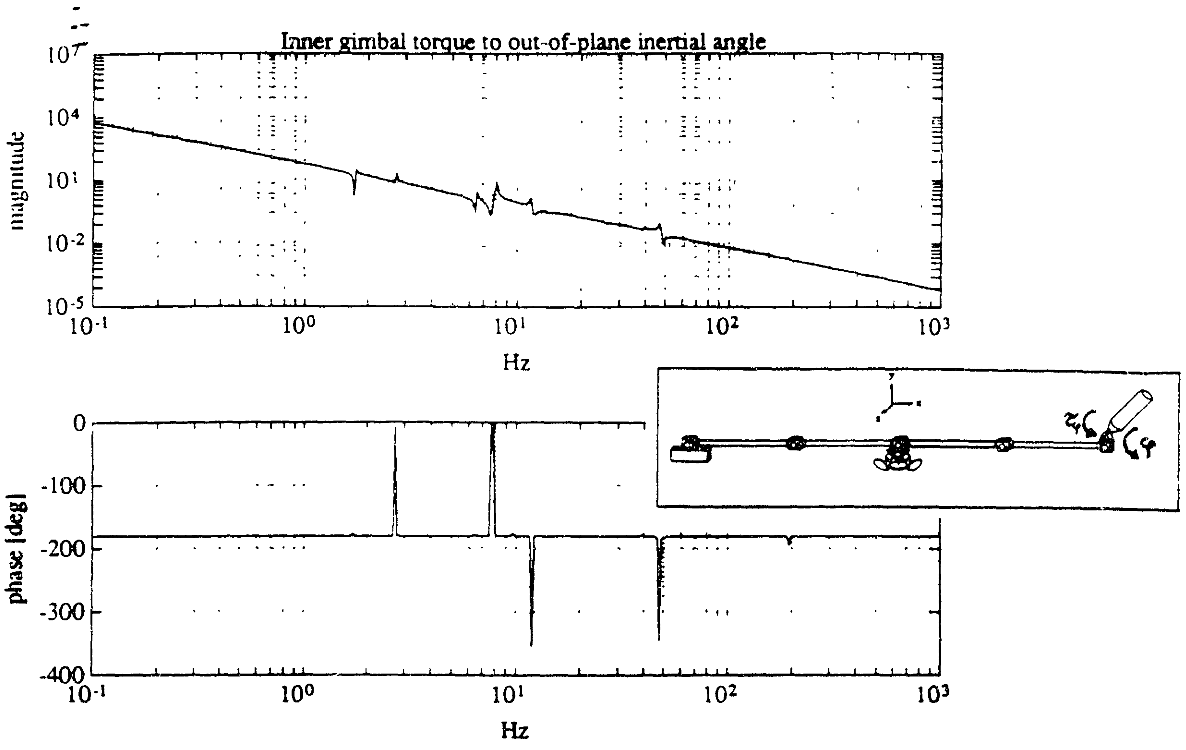


Figure 2.10 and 2.11



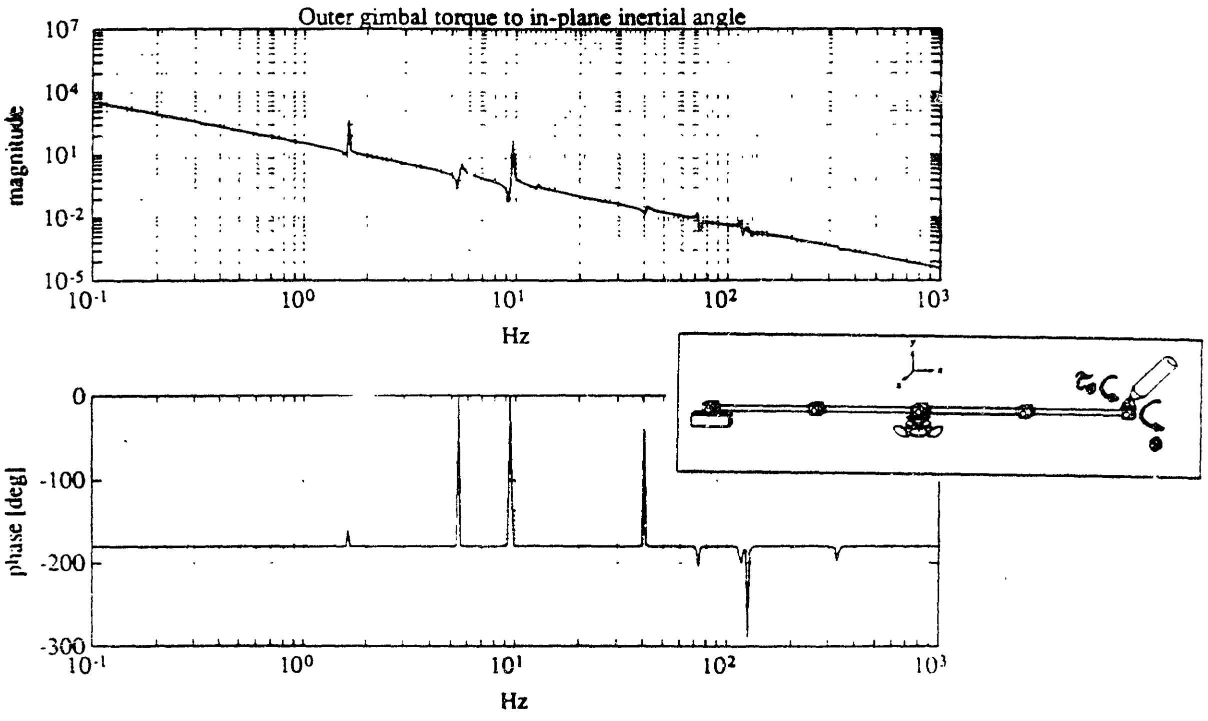
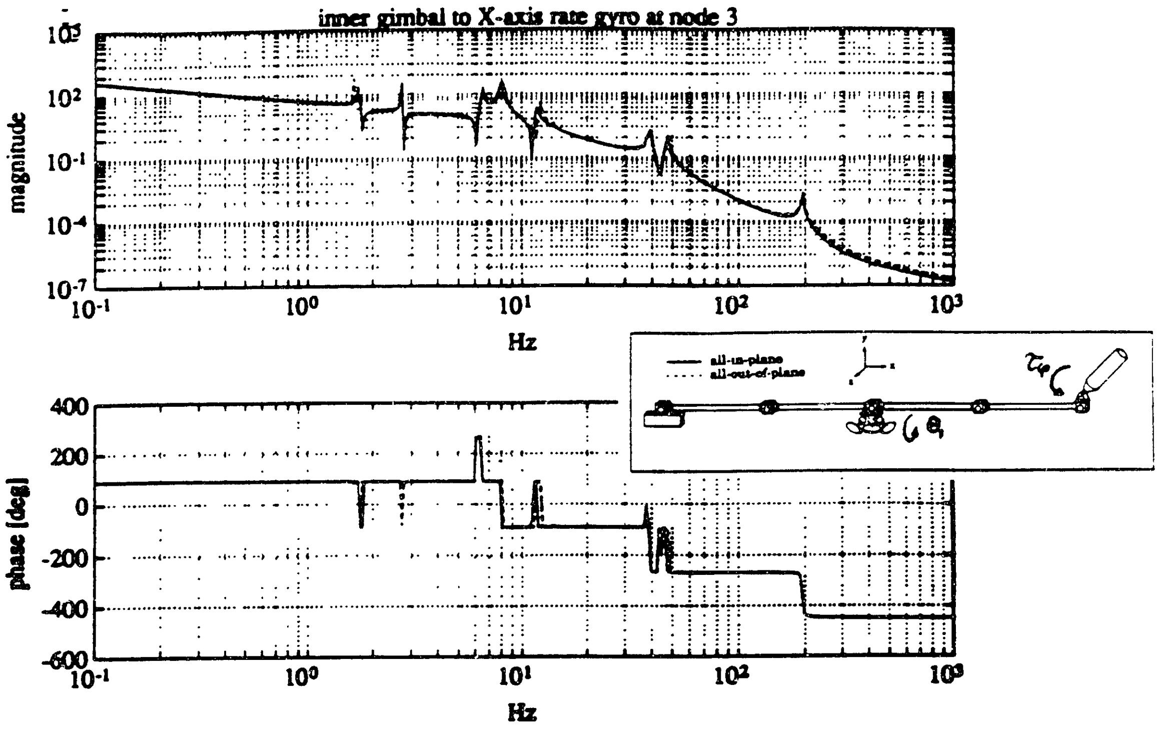


Figure 2.12 and 2.13



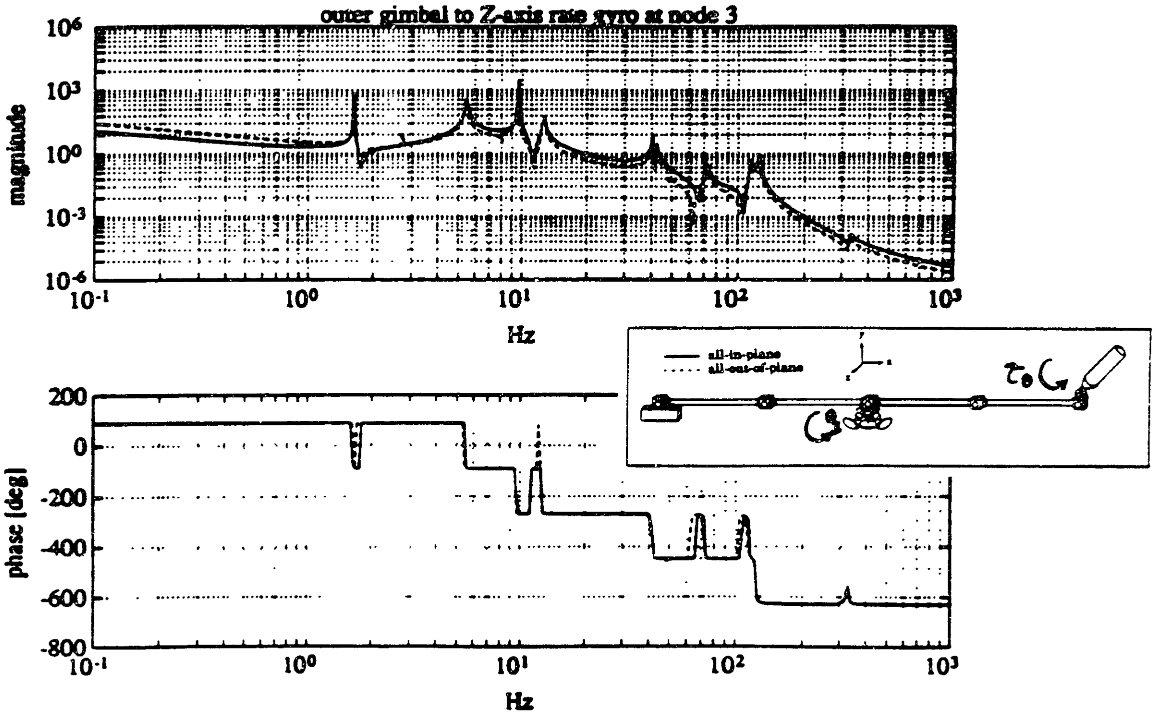
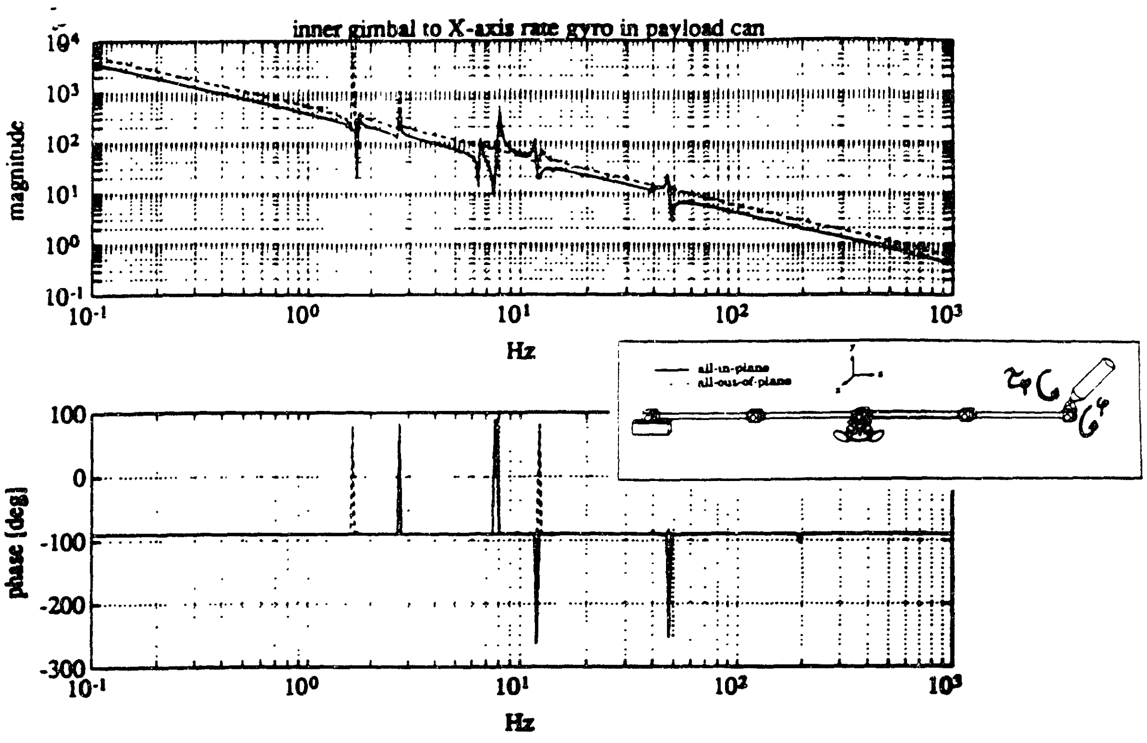


Figure 2.14 and 2.15



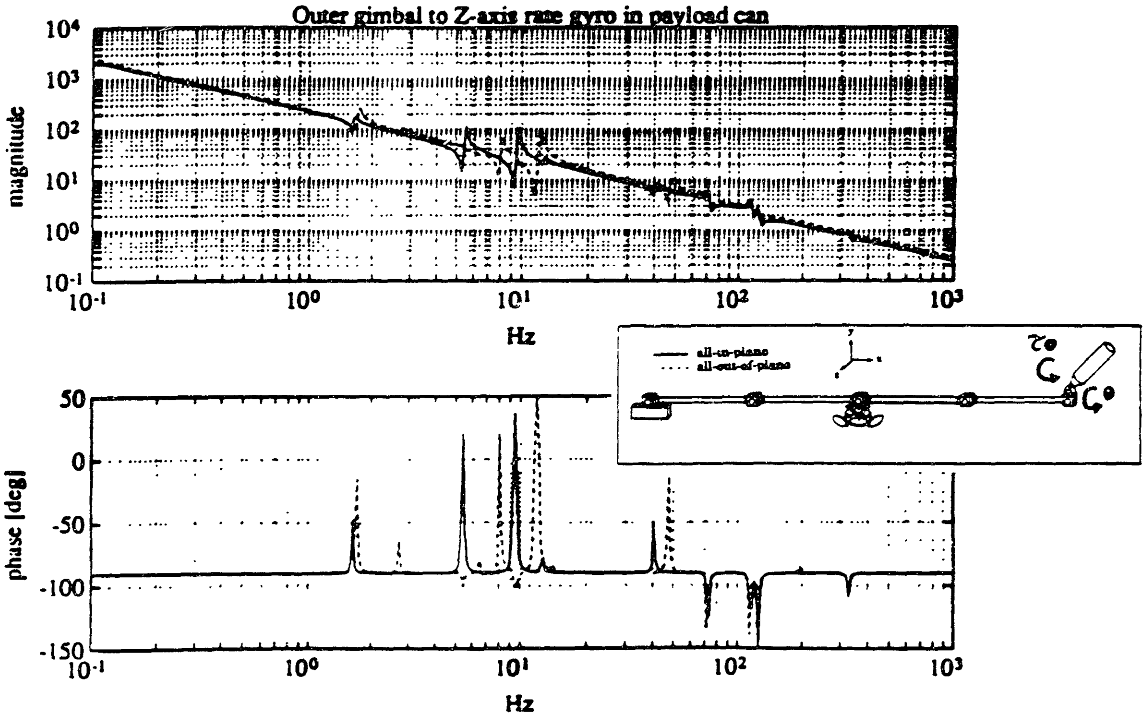
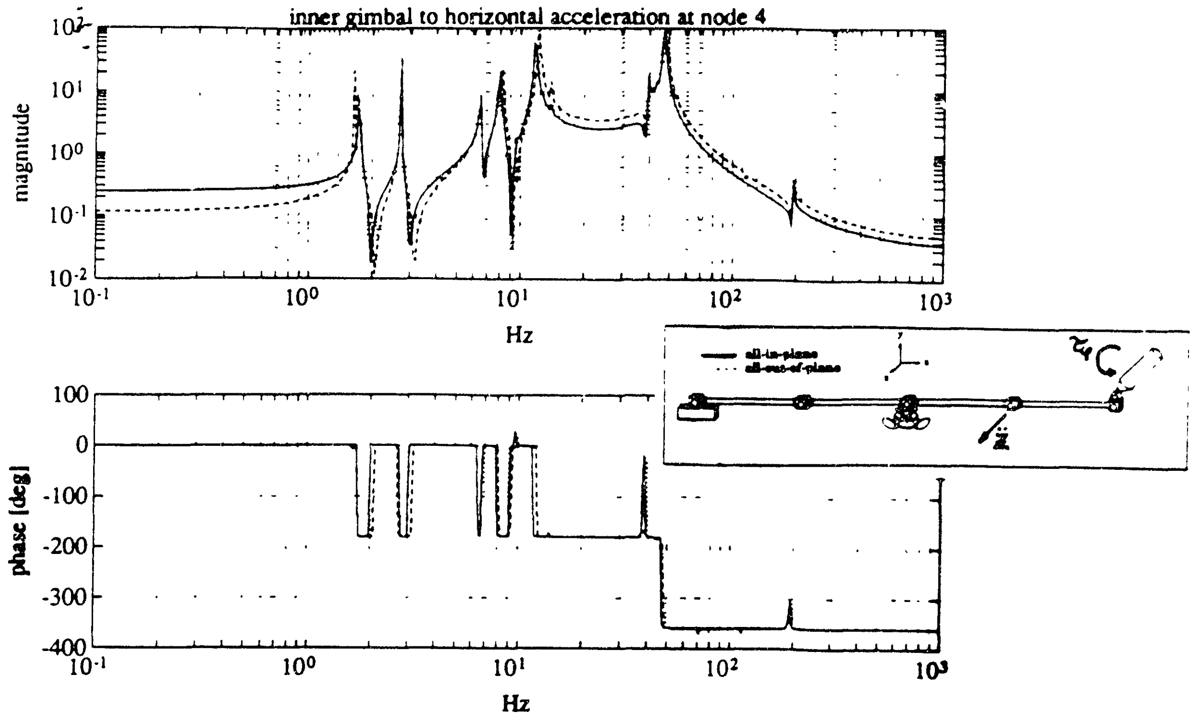
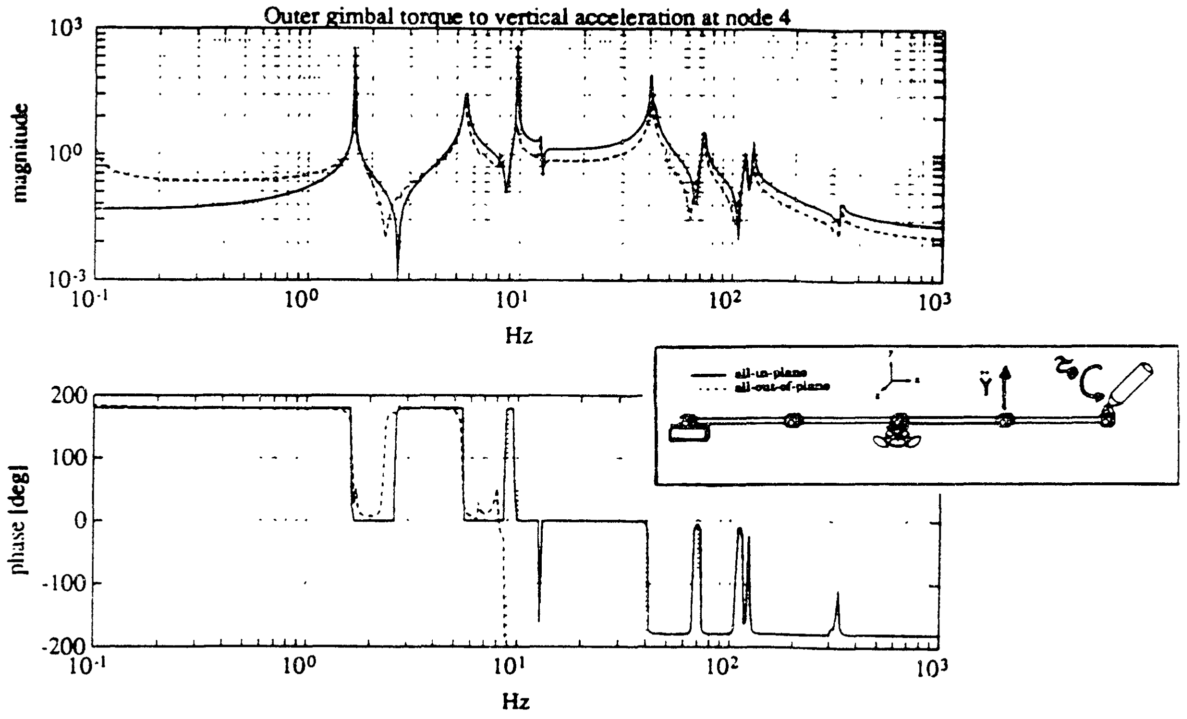


Figure 2.16 and 2.17



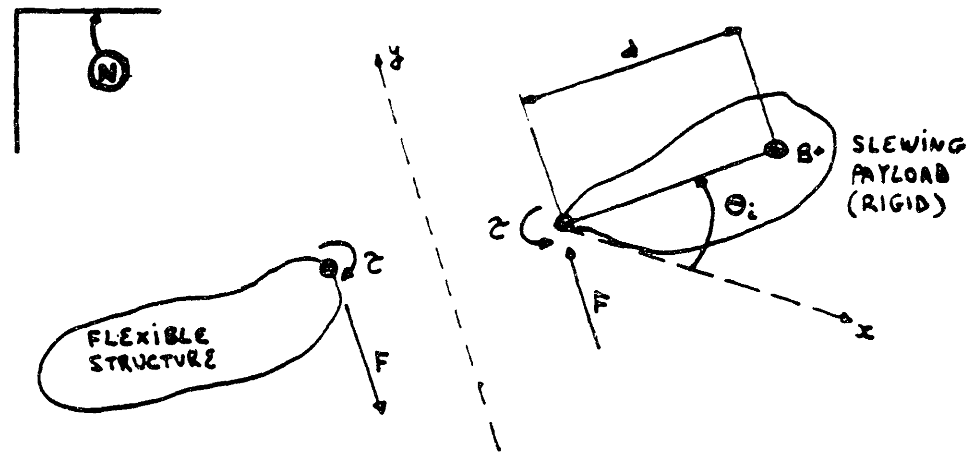


2.5.3. Open-loop Transfer Functions.

From the model in state space form it is possible to obtain the matrix of transfer functions from all of the inputs to all of the outputs. It is given by $G(s) = C\,(s\,I - A)^{-1}\,B + D$, where the feed-through D term is present because we are also measuring accelerations at certain structure locations. Note that in this form we can compute, for a given configuration of the articulated part, the open-loop transfer functions considering the effect of damping and gyroscopic terms. Therefore, superimposing different plots of the same transfer function for different dynamic conditions,i.e. wheel rates and gimbal angles, would ideally show how the perturbed model differs from the nominal one. We take the reference model as the one in which payload number 1 (at the right) is substituted by a dummy gimbal, while payload number 2, located at the left, is hinged and pointing towards the - Y direction at right angles with respect to the structure (i.e., $\varphi = 0$ and $\theta = -90$ degrees). Note also that we neglect sensor and actuator dynamics. The one-g model is treated differently, because of the significant coupling at low frequencies between the suspension dynamics and the test article dynamics.

Figures [2.10] and [2.11] show the transfer function in magnitude and phase from gimbal torque to inertial pointing angle for a configuration in the vertical plane, the reference configuration. Note also that the payload is not CG-mounted and that the effect of the base flexibility is weak, giving rise to a situation of nearly pole-zero cancellation. Figure [2.12] to Figure [2.17] show overlays of the same and other transfer functions for the "all-in-plane" and the "all-out-of-plane" configurations, which shows the change in modal content.

Figure 2.18



To see how the base flexibility influences in general the pointing dynamics of an articulated body mounted on a flexible base, consider the generic system depicted in Figure [2.18]. In general, all flexible structures with dual (i.e., of the same type) collocated sensor/actuator pairs exhibit the useful property that the poles and zeros of the plant transfer function (or the driving point compliance) alternate along the imaginary axis of the complex plane, keeping the phase bounded within zero and 180 degrees. This is a strong property for lightly damped structures, and occasionally also holds when the duality requirement is violated but collocation is not. For a non-collocated sensor/actuator pair, there is intervening flexibility. The initial order depends on the plant, but it always begin with a pole in the sequence for an *in the loop minimum phase mode* (Ref. [20]), typical of a series of an inertia-spring-inertia system in which the actuator is on the first mass, but the sensor measures a linear combination of the motions of inertia 1 and 2 The sequence begins with a zero in the case of an *appendage mode* , in which both sensor and actuator are collocated on inertia 1. This non-collocation can arise stability issues in some cases.

2.5.3.1. Approximate Analysis of the Influence of the Base Flexibility on the Torque to Payload Pointing Angle Transfer Function.

For the generic structure of Figure [2.18], the equilibrium of torques about the payload CG gives:

$$\tau - F\,d = J\,\ddot{\theta} \tag{2.46}$$

where J is the moment of inertia of the payload about its CG, θ is the inertial angle with respect to the direction X fixed in space and d is the offset from the pivot. The equilibrium of forces is given by:

$$F = m\,(\ddot{y} + d\,\ddot{\theta}\,) \tag{2.47}$$

where m is the payload's mass and y the displacement of the pivot and the input/output relationship for the structure yields:

$$\ddot{y} = H_{yF}\, F + H_{y\tau}\, \tau \tag{2.48}$$

Combining the last three equations one gets the following expression for the transfer function form torque τ to inertial angle θ :

$$\frac{\theta(s)}{\tau(s)} = \left[\frac{1 - m\, H_{yF} - md\, H_{yt}}{J\,(1 - m\, H_{yf}\,) + md^2} \right] \frac{1}{s^2} \tag{2.49}$$

This expression shows how the base flexibility appears and also tells us that its effect is negligible when m is very small (very light payload) or when d = 0 (CG-mounted payload). Basically, the difference between the flexible and rigid body transfer functions depends on the ratio of the payload inertia (or of the flexible appendage inertia, should it be present) to base-body inertia characteristics.

An extended form of this relation can be obtained as follows. Consider the structure only. The transfer functions H_{yF} and H_{yf} , in the absence of damping, take the form:

$$H_{yF} = \sum_{i=1}^{\infty} \frac{\Phi_i\, \Phi_i}{m_i\,(s^2 + \omega_i^2)}$$

$$H_{yf} = \sum_{i=1}^{\infty} \frac{\Phi_i\, \Phi'_i}{m_i\,(s^2 + \omega_i^2)} \tag{2.50}$$

where Φ_i and Φ'_i are the modal deflection and slope at the sensor/actuator location (which is collocated), and m_i is the modal mass for the i-th mode of the flexible structure, equal to $\Phi^T_i M \Phi_i$. Introduce now the quantities $\bar{m}_i = \frac{m_i}{\Phi_i\, \Phi_i}$ and $\bar{c}_i = \frac{m_i}{\Phi_i\, \Phi'_i}$. They represent the inverse of the modal residues of the two transfer functions, the former with the dimensions of [Kg] or mass and always positive for a dual and collocated transfer function and the latter with the dimensions of [Kgm] or first moment of inertia, and not always necessarily positive because the modal slope can be negative. Therefore, in the vicinity of the j-th eigenfrequency, we have that (J is the complex unit)

$$H_{yF} = \frac{1}{\bar{m}_j\,(s^2 + \omega_j^2)} + \sum_{i=j+1}^{\infty} \frac{1}{\bar{m}_i\,\omega_i^2} + \sum_{i=1}^{j-1} \frac{1}{\bar{m}_i\,(J\omega_i)^2} = \frac{1}{\bar{m}_i\,(s^2 + \omega_i^2)} + p(\text{small}) \tag{2.51}$$

where the second and third summations represent the smaller terms pertaining to the static contribution of, respectively, the lower modes (stiffness dominated) and the higher modes (inertia dominated). Similarly, we have that:

$$H_{yf} = \frac{1}{\bar{c}_j\,(s^2 + \omega_j^2)} + q\,(\text{small}). \tag{2.52}$$

If now we make $A = \frac{m}{\bar{m}_j}$ and $B = \frac{md}{\bar{c}_j}$, we obtain that:

$$\frac{\theta(s)}{\tau(s)} = \left\{ \frac{(s^2 + \omega_j^2)(1 - m\,p - md\,q) - A - B}{(s^2 + \omega_j^2)(J + md^2 - mJ) - AJ} \right\} \frac{1}{s^2} \tag{2.53}$$

From this transfer function one can evaluate the pole-zero spacing in the vicinity of the j-th eigenfrequency. After some algebra, it results in:

$$(p - z)_j = \omega_j \left\{ \sqrt{\left[1 - \frac{A\,J}{\omega_j^2(J + md^2\,)}\right]} - \sqrt{[1 - A - B]} \right\} \tag{2.54}$$

which tends to zero when the payload is very light, when it is CG mounted and when the modal displacement and slopes at the sensor/actuator location approach zero.

These considerations are useful when dealing with vibration isolation of precision pointing structures from a flexible base, in which the mount (gimbal) represents a critical point in the disturbance path and the modal spectrum of the flexible base is generally poorly known. It is obviously better to mount a pointing instrument at a point where the modal displacement is small, theoretically zero (node).

Since near pole-zero cancellation bounds the phase excursion of the plant transfer function, broadband pointing should be possible in the presence of uncertain flexibility. One, in fact, wishes to design controllers that, in the limit, ignore the flexibility of the base. To achieve this, one should start from design considerations by using a payload mass to modal mass ratio at a given frequency which is small compared to the damping ratio (i.e., $D = \frac{m/m_i}{2\,\zeta}$ small), thereby imposing a bound equal to $\phi = -2 \tan^{-1}(\frac{D}{2})$ on the phase and equal to $M = \sqrt{\{1 + D^2\}}$ on the magnitude excursions at that frequency when the control loop is closed (see Ref. [21]). Therefore, the effect of high damping is similar to that of a light payload if one wants to reduce the impact of the structural flexibility on the control of the pointing of the payload.

For a CG-mounted payload, however, one has:

$$\tau = J\,\ddot{\theta}_{\text{inertial}}$$

$$M\,\ddot{q}_e + (G + D)\,\dot{q}_e + K\,q_e = -B\,\tau \tag{2.55}$$

where we have also taken into account damping and gyroscopic forces. Therefore, one can say that the pole-zero cancellation is perfect when the inertial angle comes into play, as there is no effect of flexibility. Using the fact that $\theta_{\text{inertial}} = \theta_{(\text{relative to base body})} + C\,q_e$, taking Laplace transforms of all three equations and combining, one gets that:

$$\frac{\theta_{(\text{relative to base body})}(s)}{\tau(s)} = \frac{1}{J\,s^2} + C\,[M\,s^2 + (G+D)\,s + K\,]^{-1}\,B \tag{2.56}$$

Generalizing to a non CG-mounted payload, one can see that the gyroscopic and damping distribution on the base-body also affects in some measure the gain/phase excursion of the torque to relative angle transfer function at a given frequency.

CHAPTER 3: DERIVATION OF THE MODEL IN ONE-G

Introduction

In this chapter we will derive a model of the multi-body test article in one-g, e.g., including the dynamics of the actively controlled suspension device and the effect of gravity on the suspended test structure. The model at this stage is fully non-linear in payload angles and rates, wheel angular rates and, for the articulated part, including non-linear terms proportional to the gravitational constant g.

This chapter is divided into 3 sections. In section 3.1, a description is made of the assumptions used in deriving the one-g model. In section 3.2, a description of the suspension device is presented and a model of its internal dynamics is provided. In section 3.3 an analysis of the gravity effects on MACE is presented. First, we take into account the gravity stiffening on the suspension cables. Next, we introduce the gravitational load on the rest of the structure and on the articulated part. To obtain eigenfrequencies and mode shapes of the new dynamic system, the model must be linearized about a certain payload configuration. In the linearized model, the gravity effect on the multi-body part appears as a gravity stiffening/destiffening matrix which has to be calculated for every configuration of the articulated part. In section 3.4., the eigenproblem is solved with reference to a given suspended test article configuration (in the reference MACE configuration) and a reference set of open loop transfer functions of the model are compared to the equivalent set produced by a more refined NASTRAN model and to the same set of transfer functions obtained experimentally from the real suspended test article. In this way, the goodness of the model existing until now is verified, and justified on the basis of the fact that both the zero-g and the one-g models are lower order models which capture entirely the most significant aspects of the multi-body dynamics of the test article.

3.1 LIST OF ASSUMPTIONS USED IN DERIVING THE ONE-G MODEL.

For the model of the flexible spacecraft and of the articulated part, the same assumptions presented in section 2.2 still hold. The model of the whole structure is nevertheless different in the sense that, as shown in Figure [3.1], an active suspension system is included, and the whole system is in turn subject to the gravity loading.

1)

Each steel suspension cable contributes to the system dynamics with its own flexibility (violin modes). Since it is a long thin element of circular cross section, the choice was made to model it as if it were a beam-cable element, or a straight Bernoulli-Euler beam stiffened by an axial loading caused by gravity. The justification is presented later on, but it relies on the fact that if $\sqrt{(EI/T)}$, where E is the Young modulus, I the cross sectional inertia and T the cable tension equal to $\frac{Mg}{3}$ (M=mass of the suspended test article), is much lower than than the cable length then the bending stiffness should not be important, and practically the cable behaves as an idealized string.

2)

As from the engineering specifications, Ref.[4], each suspension device has its own internal dynamics, consisting of the pneumatic and electro-mechanic control loops. The pneumatic control loop basically acts as a static spring stiffness and gives a reference position. The electro-mechanical control loop represents the mass-cancelling control loop, and feeds back the vertical acceleration of a point of the suspension carriage to keep the test article centered about the reference position. This internal dynamics represents the only actuator dynamics that was taken into account in assembling the suspension dynamics to the test article dynamics.

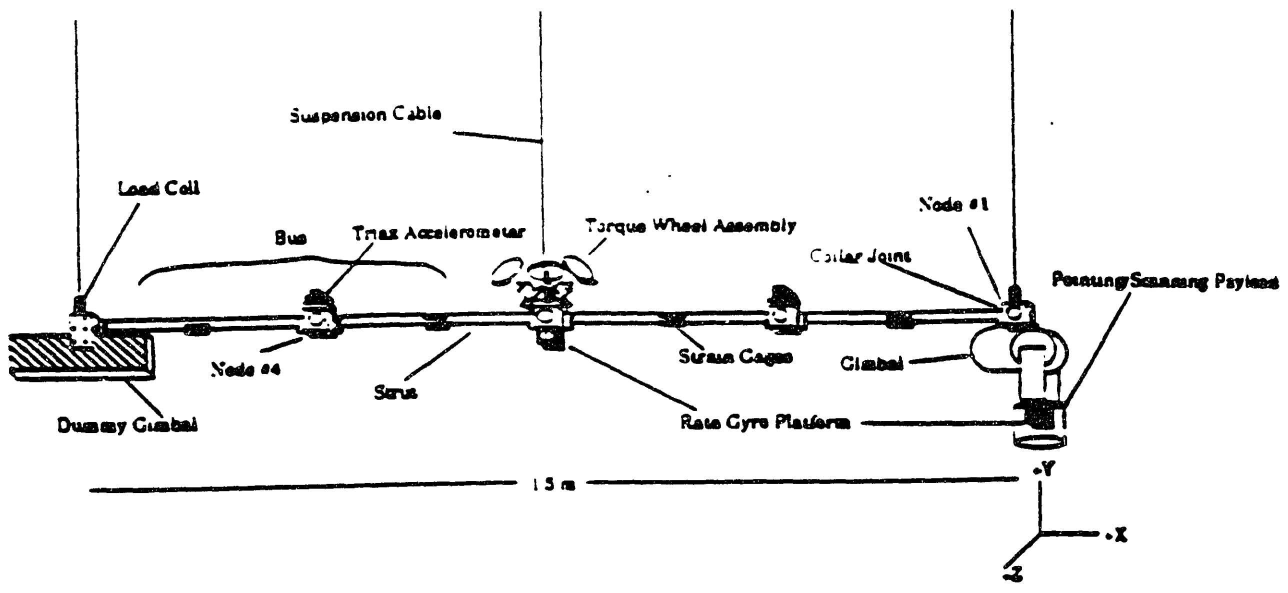
3)

The presence of an offset between the cable attachment point and the structural nodes of the bus was also taken into account. Each suspension cable is hinged above the bus centerline at a point which is capable to rotate relative to the nearest finite element node. This hinge allows three relative rotations, but torsion about the cable centerline was neglected because the cable itself is free to rotate about the hinged points at the top (suspension device) and at the bottom (bus attachment).

4)

The distributed gravity loading on the flexible bus was approximated by consistent nodal loads acting on the vertical plane (X-Y plane) only. On the articulated payload, the gravity loading causes an equivalent gravity stiffening/destiffening effect, depending on its orientation and proportional to the vertical offset between the center of mass of each body and the hinge immediately before in the topological chain. Clearly, this effect appears only when the articulated dynamics is linearized about a certain configuration.

Figure 3.1



3.2 Description of the suspension device and of the internal dynamics of the suspension.

In this section, a brief description of the suspension device is given and the state space equations of its internal dynamics are derived.

As shown schematically in Figure [3.2], the suspension device coupled to each suspension cable, consists of a combination of a passive pneumatic part and an active electro-mechanical part. The pneumatic part contains a 1.48 in^2 air piston fed from an external tank (30 gal) through a precision valve. It forms the part which supports the whole weight and it is equivalent to an air spring with no static stiffness, therefore it can be in equilibrium in any vertical position of the piston. The suspension carriage is the only moving part and can only translate vertically on non-contact bearings with no friction (this is achieved by a controlled leak rate which forms an air film around the piston). This zero-friction capability avoids non-linear contamination of the dynamics of the suspended test article. The test-article hangs from the lower of the two cross members of the carriage, while the piston directly moves the upper member. On the carriage, there are mounted position (LVDT), velocity and acceleration sensors as well as a 2.43 inches stroke electromagnetic linear actuator. This vertical stroke is basically the maximum vertical displacement that the suspended test article can undergo. For stability of the mass cancelling acceleration loop, passive damping of the cable stretch mode is required. This is achieved by incorporating a number of layers of viscoelastic material in the lower cross member of the carriage.

The electro-mechanical part, instead, does not carry any weight, but it provides a small static stiffness to keep the test article centered about the reference. This subsystem can be viewed as a second equivalent spring in parallel with the series spring-dashpot combination of the air spring. Figure [3.3] shows a block diagram of the active part. A low-pass elliptic filter is used in the displacement loop to enhance the stiffness below 0.7 Hz. Because it introduces a phase lag, a velocity feedback loop is also added to the displacement loop. Finally, to provide the cancelling effect of the carriage mass, acceleration is fed back with positive sign. This mass cancellation loop ensures that the mass added by the moving part of the suspension device is only bounded to be about 0.5 - 3% of the mass of the payload.
Table [7] gives the values of the parameters of the block diagram.

Table 3.1 Parameters of the block diagram of Figure [3.3].

gain of linear actuator	1.4 [lbf/Amp]
gain of power amplifier	0.62 [Amps/Volt]
gain of LVDT	1 - 11 [Volts/in]
gain of velocity feedback loop	4.17 [Volts/in/sec]
gain of accelerometer	0.7 [Volts/g]
external excitation signal	0.868 [lbf/Volt]
gain of low pass filter	0.49 [Volts/Volt]
low pass filter specs	see Ref [ ]

With these data, it is straightforward to derive the state-space representation of the dynamic suspension controller. In particular, because the current development model of the test article in one-g includes feeding back acceleration+displacement, one can compute the poles of a single suspension controller as :

$$-40.42535,\ -20.21267,\ 0,\ 0\ \text{[rad/sec]}$$

Two of them are zero because the input is the vertical acceleration of the suspension carriage, which has to be integrated twice to get the displacement. It was in fact chosen, for the sake of analytical simplicity (only one sensor), not to measure directly the displacement. The other two poles are in the LHP and they are critically damped poles ($\zeta = 1$).

The internal dynamics of the suspension can be written in state space form as:

$$\begin{aligned} \dot{z} &= A_c\, z + B_c\, u_c \\ y_c &= C_c\, z + D_c\, u_c \end{aligned} \tag{3.1}$$

where z represents the internal dynamics state vector (4x1) and the subscript c refers to the suspension "controller". In addition, if u denotes the input from the suspension controller and u_o all the other inputs to the system, the test article dynamics can be written as:

$$\dot{x} = A\,x + B\,u + B_o\,u_o$$

$$y = C\,x + D\,u + D_o\,u_o \tag{3.2}$$

This construction is shown in Figure [3.4].

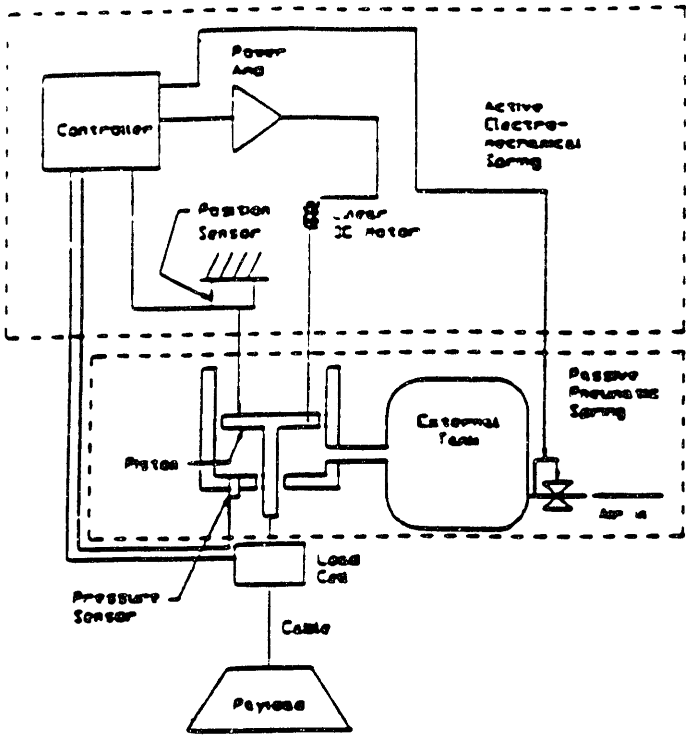
It is clear that $u_c = P\,y$, where P is an output distribution matrix which selects the acceleration of the carriages among the other outputs of the control system, and that $u = -y_c$ Manipulating these equations, and introducing the expression $Q = \frac{1}{1 + D_c P D}$, one obtains the integrated (suspension controller + suspended test article) equations of motion in state space form as:

$$\begin{pmatrix} \dot{x} \\ \dot{z} \end{pmatrix} = \begin{bmatrix} A\text{-}BQD_cPC & \text{-}BQC_c \\ B_cPC\text{-}B_cPDQD_cPC & A - B_cPDQC_c \end{bmatrix} \begin{pmatrix} x \\ z \end{pmatrix} + \begin{bmatrix} B_o - BQD_cPD_o \\ B_cPD_o - B_cPDQD_cPD_o \end{bmatrix} u$$

$$Y = \begin{bmatrix} C - DQD_cPC & \text{-}DQC_c \end{bmatrix} \begin{pmatrix} x \\ z \end{pmatrix} + [D_o - DQD_cPD_o]\,u_o \tag{3.3}$$

Note that the new dynamics has been appended to the original dynamics, thereby introducing 12 new states, 4 for each one of the three suspension devices.
From these one can analize the linearized dynamics of the test-article suspended in a one-g field. To do so, however, one must first completely derive the equations of motion of the test article under the stiffening/destiffening effect of gravity and this is the purpose of the next sections.

Figure 3.2 a and b



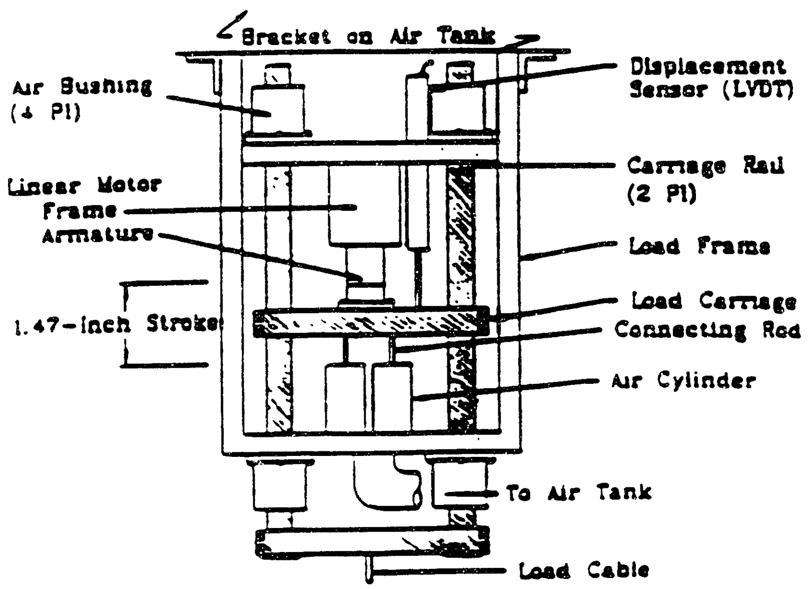
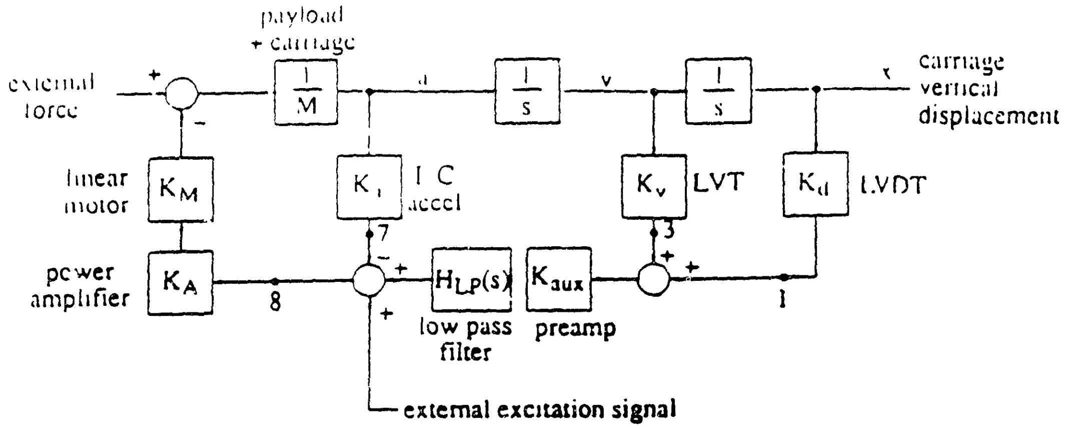
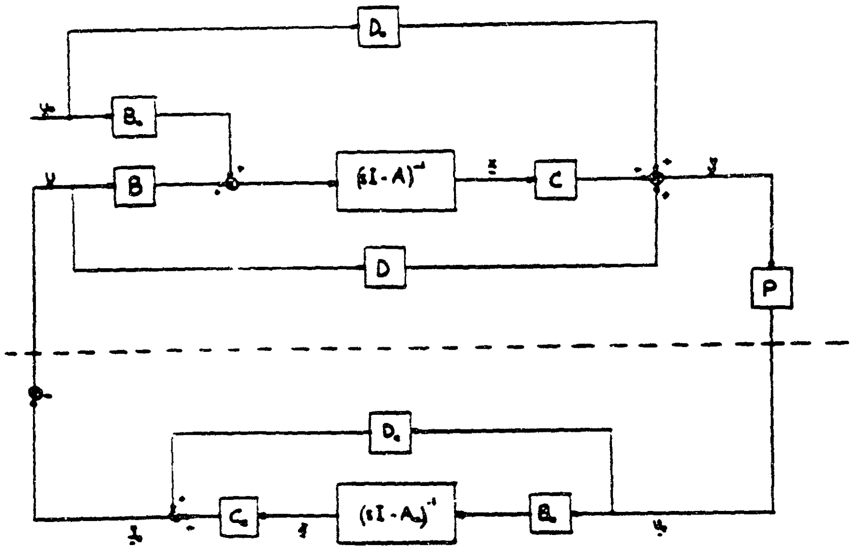


Figure 3.3 and 3.4





3.3 Gravity Effects on MACE

As a rule, the suspension device has to support the test article in the laboratory environment and, at the same time, approximate the free-free boundary conditions of the in-orbit operations. However, the presence of the gravity field acting on a flexible structure can cause different perturbations: stiffening or destiffening along the gravitational vector (which can be modelled as a perturbation in the stiffness matrix), transverse dynamic buckling and modal coupling induced by the static sag on a horizontal flexible member (static droop), which requires a redefinition of the reference structure because it involves finite displacements. In addition, the presence of a suspension device introduces, in general, additional effects, namely: added stiffness, inertia and damping of the suspension, more complex suspension dynamics such as cable modes, modal coupling with the suspension dynamics (equivalent to the introduction of additional stiffnesses at the attachment points) and dynamic torques resulting form the offset of the center of mass of the supported bodies about the plane of the suspension, because of the general geometric shape of the suspended article. As a rule of thumb, it is desirable to have an order of one decade separation between the suspension pendular frequencies and the test article fundamental For the MACE test article, which has a fundamental frequency in zero-g at 1.638 Hz approximately, this means that the pendular modes must be located at most at 0.2 Hz. For the given mass properties, this corresponds to a cable length of about 4.6 meters.

3.3.1 The modelling of the suspension cables.

As mentioned in section 3.2, the bending length $\sqrt{\frac{EI}{\text{Tension}}}$ of a long thin steel cable axially loaded by a weight is much lower than its length and than the wavelength of the equivalent string wave. This reasoning allows us to say that the suspension cables behave mostly like idealized strings loaded in tension, rather than elements in bending only. The elastic dynamics is therefore dominated by the axial tension. However, the gravity stiffening matrix has to be derived considering the rotational degrees of freedom involved in the bending of the equivalent beam. This is done in the next section.

3.3.2.. The derivation of the gravity geometric stiffness matrix of a beam-column.

The total stiffness matrix for a finite element under axial load is the sum of the Bernoulli-Euler stiffness matrix plus the geometric stiffening matrix proportional to the axial load. For a two-dimensional member, of length h and subject to an axial load of intensity P(x), the potential energy is given by (Ref. [12]):

$$V = \frac{1}{2}\int_0^h EJ\,(y\ )^2\,dx \ + \frac{1}{2}\int_0^h P(x)\,(y\ )^2 dx$$

(3.4)

Introducing the lateral deflection $y(x) = L_1\, w_1 + L_2\, h\, \theta_1 + L_3\, w_2 + L_4\, h\, \theta_2 = L^T q$ and the nodal lateral (w_1 and w_2) and rotational (θ_1 and θ_2) displacements, the geometric stiffening matrix is given by :

$$K_g = \int_0^h P(x)\ L'\ L'^T\ dx \tag{3.5}$$

where L' is the derivative taken with respect to the variable x of the vector of interpolation polynomials. Using Hermite cubics as interpolation polynomials, of the form:

$$\begin{aligned}
L_1 &= 1 - 3(\tfrac{x}{h})^2 + 2(\tfrac{x}{h})^3\\
L_2 &= \tfrac{x}{h} - 2(\tfrac{x}{h})^2 + (\tfrac{x}{h})^3\\
L_3 &= 3(\tfrac{x}{h})^2 - 2(\tfrac{x}{h})^3\\
L_4 &= -(\tfrac{x}{h})^2 + (\tfrac{x}{h})^3
\end{aligned}$$

(3.6)

and a linear distribution of the axial load as $P(x) = W + \mu\, g\, x$ (μ = linear density, W=suspended weight) neglecting axial displacements one obtains ($p_o = \mu g$):

$$K_g = \frac{1}{h}\begin{bmatrix} \frac{6}{5}W + \frac{6}{10}P_0 & h(\frac{1}{10}W + \frac{1}{10}P_0) & -(\frac{6}{5}W + \frac{6}{10}P_0) & h\frac{1}{10}W \\ h(\frac{1}{10}W + \frac{1}{10}P_0) & h^2(\frac{2}{5}W + \frac{1}{30}P_0) & -h(\frac{1}{10}W + \frac{1}{10}P_0) & -h^2(\frac{1}{30}W + \frac{1}{60}P_0) \\ -(\frac{6}{5}W + \frac{6}{10}P_0) & -h(\frac{1}{10}W + \frac{1}{10}P_0) & \frac{6}{5}W + \frac{6}{10}P_0 & -h\frac{1}{10}W \\ h\frac{1}{10}W & -h^2(\frac{1}{30}W + \frac{1}{60}P_0) & -h\frac{1}{10}W & h^2(\frac{2}{15}W + \frac{1}{10}P_0) \end{bmatrix} \tag{3.7}$$

The extension to a three-dimensional element is straightforward and involves the remaining degrees of freedom of the element, as shown in Figure [3.5].

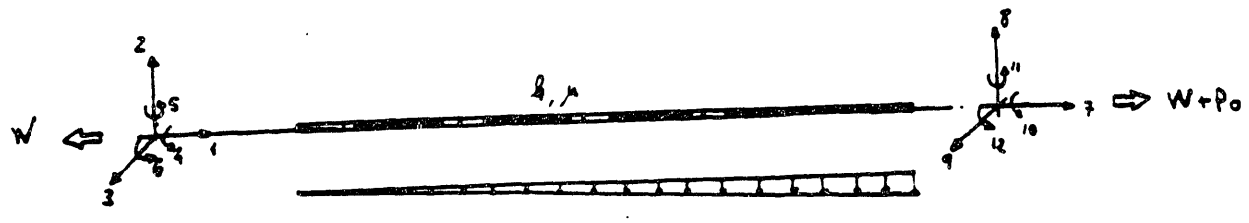
The 12x12 inertia and stiffness matrices of the beam-cable element are added to the stiffness matrices of the Bernoulli-Euler element and then the boundary conditions are imposed. In this case, the cable is hinged at both extremities, with 3 degrees of freedom of rotational motion, and furthermore, displacements in the X and Z directions are precluded in the hinge located at the top, while the displacement in the Y direction is allowed by the suspension carriage. This re-defines the eigenproblem for the suspension cable as:

$$(-\omega^2 M + K_{de} + K_g)\, q = 0 \tag{3.8}$$

Note that in this case the load P(x) is tensile, hence all the frequencies of the suspension cable are increased. Note also that if the load were compressive instead of tensile, one could recover the Euler buckling load as a result of the rotational freedoms permitted. Clearly, the exact value of the buckling load for those boundary conditions would be approached from above as more finite elements are included in the analysis. Reference [22] analyzes a two-dimensional example of this instance.

It remains now to assemble the equations of motion of the cables to those of the suspended test article.

Figure 3.5



3.3.3 The modelling of the gravity load on the flexible structure and the articulated part.

On each flexible member, of length h, the consistent nodal loads caused on the vertical plane by gravity can be derived as follows. First, we assume that the suspension control system has been tuned so that the centroidal line of the spacecraft bus is aligned with a reference horizontal line (the zero line). This implies that each flexible element is defined about an undeformed static reference line, or that there is no static pre-deformation. This is one of the perturbations induced by the one-g field and it basically induces a static sag to the horizontal member. Then we derive the consistent nodal loads as:

$$F_{nodal} = \int_0^1 L^T \begin{pmatrix} 0 \\ P_Y \\ 0 \end{pmatrix} h\, d\xi = -P_Y h \begin{pmatrix} 0 \\ \frac{1}{2} \\ 0 \\ 0 \\ 0 \\ \frac{h}{12} \\ 0 \\ \frac{1}{2} \\ 0 \\ 0 \\ 0 \\ -\frac{h}{12} \end{pmatrix} \tag{3.9}$$

where $\xi = \frac{x}{h}$, L^T is a (12x3) matrix of shape functions (Hermite cubics) and P_Y is the gravity load per unit length directed along -Y.

Using Kane's approach, we can derive the expression of the gravity load acting on the articulated part. For a chain of rigid bodies, the generalized active force due to gravity is given by:

$$(\mathbf{F}_{gravity})_r = \sum_{k=1}^{Nbodies} {}^N\mathbf{v}_r^k . \left(- m_k\, g\, \mathbf{n}_2 \right) \tag{3.10}$$

where $\mathbf{n}_2$ is the unit vector along the positive Y axis. Recalling the structure of the state vector, and the fact that for the choice of the global reference frame $\mathbf{n}_2 = \mathbf{f}_2$, one obtains:

$$F_{1g} = 0$$
$$F_{2g} = -g\,(m1 + m2 + m3)$$

$$F_{3g} = 0$$
$$F_{4g} = -g\,[\, m1\, (\, -\, {}^{F}r_3{}^{F+}) + m2\, (-A_2) + m3\, (\, -A_5)\,]$$
$$F_{5g} = 0$$
$$F_{6g} = -g\,[\, m1\, (\, {}^{F}r_1{}^{F+}) + m2\, (C_2) + m3\, (\, D_5)\,]$$
$$F_{7g} = -g\,[\, m2\, (-C_1) + m3\, (\, -E_5)\,]$$
$$F_{8g} = -g\, m3\, G_5$$
(3.11)

Similarly, if the radius vector from node 3 to the center body is r , for the center node we have:

$$F_{1g} = 0$$
$$F_{2g} = -m\, g$$
$$F_{3g} = 0$$
$$F_{4g} = m\, g\, r_3$$
$$F_{5g} = 0$$
$$F_{6g} = -\, m\, g\, r_1$$
(3.12)

These entries form the vectors of non-linear terms due to gravity and it is used in the non-linear time simulation.

However, for purposes of analyzing the linearized dynamics of the system under gravity, one needs to linearize the dynamics about a given configuration of the articulated bodies. The effect of gravity is not any more represented by these non-linear terms derived above. Instead, one must consider the gravity stiffening/destiffening terms arising when the payload is in a different equilibrium configuration. This is the purpose of the next section.

3.3.4 Linearized Dynamics. Derivation of the gravity stiffness matrix for the articulated bodies.

To form the linearized version of non-linear equations of motion, one must expand all the functions of the perturbations involved in the linearization into power series in these perturbations, and drop higher order terms. In particular, we are interested in obtaining the linearized effect of gravity on the articulated part. The linearization of the configuration dependent terms of the inertia matrix results simply in a constant term and a time dependent term for each entry. However, linearizing the vector of gravity forces entails expanding sines and cosines in series. If one simply does so, the series expansion produces a constant vector and non-symmetric matrix proportional to g, which obviously destroys the symmetry of the equations. The symmetry will be preserved if the second order terms appearing in the energy expressions will appear in the dynamical equations to first order. Recall that the gravitational field is conservative, and the system is natural, according to the definition presented in Ref. [12]. Therefore, the kinetic and potential energy are symmetric quadratic forms in the generalized coordinates. To have the correct terms to first order in the generalized coordinates, one must keep terms until second order in the expressions of the energy. Therefore, for the resulting linearized dynamics to be symmetric, part of the constant vector and of the non-symmetric stiffness matrix must equilibrate the reaction forces at equilibrium. Note also that we are linearizing the articulated part about a non-equilibrium configuration in a gravity field. This means that the gimbal actuators are applying equivalent reaction torques at the pivot to mantain the payload pointed. If the payload points upward, the net effect is a destiffening and the situation is similar to that of an inverted pendulum.

The conventional procedure adopted in the Kane's method consists of developing fully non-linear expressions of all the angular velocities and the linear velocities of the bodies and points of interest. The partial linear and angular velocities must be derived from these non-linear expressions. Then one proceeds to linearize all the angular and linear velocities, by expanding each generalized coordinate as $q = q_{ref} + \varepsilon$ where ε is a perturbation, and one can then form linearized accelerations. The partial velocities are then linearized, and the linearized versions of the generalized active and inertia forces can be derived.

First, we must project the gravitational force along the axes of body 1, as done for the other forces in chapter 2. Therefore we must introduce a transformation between the F_i and the N frame. Because we have assumed small deflections, we can write that:

$$F_i = \begin{bmatrix} 1 & -\theta_3 & \theta_2 \\ \theta_3 & 1 & -\theta_1 \\ -\theta_2 & \theta_1 & 1 \end{bmatrix} N \tag{3.13}$$

and, therefore, $\mathbf{n}_2 = \theta_3\mathbf{f}_1 + \mathbf{f}_2 - \theta_1\mathbf{f}_3$. Similarly, the kinematical relation between the angular velocities and the nodal rotations is the identity matrix, to first order. Substituting in the expression of the gravity forces, the constraint forces and torques at the interface between the rigid/flexible part, and the constraint reaction forces caused by the presence of the suspension cable (attached at a vertical distance η_2 from the node), one obtains the linearized expressions of the generalized active forces. However, one must also impose equilibrium with the rest of the structure. This means that the constraint forces and torques at the interface between the rigid/flexible part vanish with the same forces but with opposite direction at the interface, and that at equilibrium, when $\theta_{1ref} = \theta_{2ref} = \theta_{3ref} = 0$, the cable supports the weight of the underlying part. It also supports the components of the weight force in the other two directions (but these are infinitesimal quantities). Imposing these force equalities in the expansion of the generalized active forces results in vector of loads which, in matrix form, yields the symmetric geometric stiffening terms caused by gravity. For this case, this matrix assumes the form:

$$K_{g\ \text{multi-body}} = g \begin{bmatrix} k_{11} & 0 & 0 & k_{14} & k_{15} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & k_{33} & 0 & k_{35} \\ k_{14} & 0 & 0 & k_{44} & k_{45} \\ k_{15} & 0 & k_{35} & k_{45} & k_{55} \end{bmatrix} \tag{3.14}$$

where the columns involve only the rotational degrees of freedom, namely $\theta_1,\theta_2,\theta_3,\varphi,\theta$ and where, denoting with the subscript o the linearized expressions of the quantities defined in section 2.2 and evaluated at equilibrium:

$$k_{11} = m1\ {}^{F}r_2^{\ F+} + m2\ ({}^{F}r_2^{\ A} + B_{10}) + m3\ ({}^{F}r_2^{\ A} + B_{40}) - (m1+m2+m3)\ \eta_2$$
$$k_{14} = m2\ (B_{10} + B_{40})$$

$$
\begin{aligned}
k_{15} &= m3\ C_{40} \\
k_{33} &= k_{11} \\
k_{35} &= m3\ A_{40} \\
k_{44} &= k_{14} \\
k_{45} &= k_{15} \\
k_{55} &= m3\ A_{40}\ c\varphi_0
\end{aligned}
\tag{3.15}
$$

Similarly, for the central node, the very same procedure outlined above would yield the linearized stiffness matrix as:

$$
K_{g3} = g \begin{bmatrix} k & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & k \end{bmatrix}
\tag{3.16}
$$

where $k = m\,({}^{B}r_2^{B+} - \eta_2)$. Note that when the torque wheel assembly is mounted upwards, the center of mass of the center node lies above the bus centerline, and the gravity effect is to destiffen the structure.

These gravity stiffening terms are added to the finite element stiffness matrix in global coordinates, once the global stiffness matrix for the whole structure is obtain from the formulation. This is done in the next section.

3.4. The derivation of the equations of motion of the constrained system.

Note, at this point, that we are deriving the dynamic model of a gravitationally coupled structure which is forming a closed topological tree. To derive these dynamical equations, we will follow a procedure commonly used in multi-body dynamics. In particular, to assemble the equations of motion of the whole structure, one has to obtain the equations of motion, in global coordinates, of each component. The equations of motion of the zero-g model have been derived in chapter 2, and they are already defined in global coordinates.

Schematically, they are given by:

$$M_s \ddot{q} + (G_s+D_s)\,\dot{q} + K_s\, q = F_s\, u \tag{3.17}$$

where q is the state vector described at the end of chapter 2. Incorporating the geometric stiffening effect on each cable, and transforming to global coordinates, one obtains the equations of motion for the suspension cables in the same set of coordinates as the rest of the structure. They have the same form of equation (3.17), but with $G = 0$. It is important to note here that the fact that the structure is suspended in a one-g field allows one to consider the transformation matrix from the local reference frame at each cable end node to the global coordinates of the structure as if it were an infinitesimal transformation, and therefore it is basically given by a permutation matrix. This is equivalent to assuming infinitesimal rigid body pendular motions induced by the suspension.

To assemble to the rest of the structure (i.e., the zero-g model of the structural test article), we must take into account the kinematical constraints imposed by the hinged connection at each attachment point. These constraints impose a relative rotation between the point where the cable is hinged and the nearest base-body finite element node, the coordinates of which are actually used in the state vector. This construction was followed because it was chosen from the beginning that the nodes of the base body lie on the rigid elements along the bus centerline. One can always use a classical approach and describe the constraint equations as $A\, q_{tot} = 0$ where now q_{tot} contains the global states of each the components to be assembled. Here, the non-square matrix A denotes the Jacobian of the holonomic constraint equation. Next, we follow the procedure outlined in Ref.[23]. Using a vector Λ of Lagrange multipliers, one can write the constrained component equations of motion as:

$$M_s \ddot{q} + (G_s+D_s)\,\dot{q} + K_s\, q = F_s\, u + A^T \Lambda$$

(3.18)

$$A\, \dot{q}_{total} = 0$$

If we now introduce a non-square transformation matrix W mapping q_{total} (whose entries are independent generalized coordinates) into the state vector of the constrained system, i.e. $q_{total} = W\, q_c$, we can show that $AW = 0$. Therefore, the constraint equation is satisfied automatically if the choice of W satisfies $AW = W^T A^T = 0$. In our case, W is the product of a permutation matrix with the transformation from local coordinates of the cable to global coordinates of the structure. Because of the assumption of small pendular motion, this choice results in a full rank matrix W which is, basically, another permutation matrix. Therefore, substituting into the component equations of motion and premultiplying by W^T we obtain the parameters of the constrained system as:

$$
\begin{aligned}
M_{system} &= W^T M_{comp} W \\
K_{system} &= W^T K_{comp} W \\
G_{system} &= W^T G_{comp} W \\
F_{system} &= W^T F_{comp}
\end{aligned}
\tag{3.19}
$$

where the subscript comp denotes the component parameters before imposing constraints.

At this point, a first solution of the eigenproblem, provides a matrix of mode shapes and a diagonal matrix of eigenfrequencies which is used to build the damping matrix, in the manner described in chapter 2. In this case, one percent material damping is also added to the suspension cables.

The equations of motion of the constrained system before incorporating the internal dynamics of the suspension (which introduces additional 12 states) can be written as follows. Denoting with the subscripts "e", "a", "o" and "w" respectively the global coordinates of the nodes, the rotational freedoms at the hinges, the relative gimbal angles and the wheel spin, and with the subscripts "N.L.", "c" and "g" the non-linear, control and gravity terms, we obtain:

$$
\begin{bmatrix} M_{ee} & M_{ea} & M_{eo} & M_{ew} \\ M_{ae} & M_{aa} & M_{ao} & M_{aw} \\ M_{oe} & M_{oa} & M_{oo} & M_{ow} \\ M_{we} & M_{wa} & M_{wo} & M_{ww} \end{bmatrix}
\begin{pmatrix} \ddot{q}_e \\ \ddot{q}_a \\ \ddot{q}_o \\ \ddot{q}_w \end{pmatrix}
+ \left(\begin{bmatrix} G_{ee} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
+ \begin{bmatrix} D_{ee} & 0 & 0 & 0 \\ 0 & D_{aa} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right)
\begin{pmatrix} \dot{q}_e \\ \dot{q}_a \\ \dot{q}_o \\ \dot{q}_w \end{pmatrix} +
$$
$$
\begin{bmatrix} K_{ee} & K_{ea} & 0 & 0 \\ K_{ae} & K_{aa} & 0 & 0 \\ 0 & 0 & K_{oo} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\begin{pmatrix} q_e \\ q_a \\ q_o \\ q_w \end{pmatrix}
+ \begin{pmatrix} F_e \\ 0 \\ F_o \\ 0 \end{pmatrix}_{N.L.}
= \begin{pmatrix} F_e \\ 0 \\ F_o \\ F_w \end{pmatrix}_{C}
+ \begin{pmatrix} F_e \\ 0 \\ F_o \\ 0 \end{pmatrix}_{g}
\tag{3.20}
$$

Note that $M_{ao} = M_{oa}^T = M_{aw} = M_{wa}^T = M_{ow} = M_{wo}^T = 0$. Also note the presence of the gravity stiffening terms K_{oo} on the articulated part, which make the equations of motion gravitationally coupled, but these terms appear only when the dynamics is linearized about a payload configuration, as explained in the previous section. Aerodynamic drag and gimbal bearing friction during payload slewings, although not difficult to model, have not been included in the generalized active forces for the sake of simplicity. Experience has shown that the modelling phase is prone to continuous revisions and improvements, but one should not enter into much more detail once the most fundamental aspects of the dynamics have been captured.

3.5. Dynamic analysis of the one-g model.

3.5.1. Open-loop eigenfrequencies and mode shapes. Transfer functions.

Table [3.2] shows the frequencies and mode shapes of the test-article suspended in a one-g field. Note the appearance of the low frequency suspension pendular modes. Because the rigid body modes have been stiffened by gravity, there are no more zero frequencies except those of the free-spinning wheels. Further, the precession frequency is masked by the suspension pendular modes. Towards higher frequencies, the behavior of the zero-g model is recovered. Also, there is only a slight difference between the test article eigenstructure in one-g and the one in zero-g, as one could see by comparing the numbers from table[2.5] in chapter 2 and table [3.2]. The reason is that this eigenproblem has been solved linearizing the dynamics about the equilibrium state. This means that there are distributed loads along the flexible beams and end constraint torques due to the offset of the CG of the articulated part, which also manifest a geometric stiffening effect.
Figures [3.6a] to [3.11a] show two of the open-loop transfer functions represented in continuous line, in a comparison with the equivalent ones of the zero-g model, represented in dotted line The gravity field effect on the linear model introduces an equivalent DC gain, and it is clear that this must be so because the suspension must not interact with the suspended structure, except at low frequency. In fact, the device tries to mimic the zero-g behavior.

Figures [3.6b] to [3.11b] show a comparison between the experimentally obtained transfer functions corresponding to the figures above and those derived from the one-g model. In particular, in these plots the continuous line represents the behavior of the equivalent high fidelity NASTRAN model and the dotted line the experimental data. The matching is rather good, especially at low-middle frequencies and the suspension modes as well as the pole-zero structure until 100 Hz approximately are well reproduced captured. However, some mismatch starts to appear in the range 10 Hz and above, where instead of capturing a pole-zero pair in the region around 9 Hz, a zero-pole sequence is represented instead. This may be due to th, more detailed modelling permitted by the use of NASTRAN, in particular in the assembling of the composite rigid bodies and therefore in the computation of the actual centers of mass. However, for the purposes of deriving a lower order model of reasonable fidelity, the results just presented are satisfying.

Table [3.2] Structural Frequencies and Mode Shapes of the Suspended Test Article.

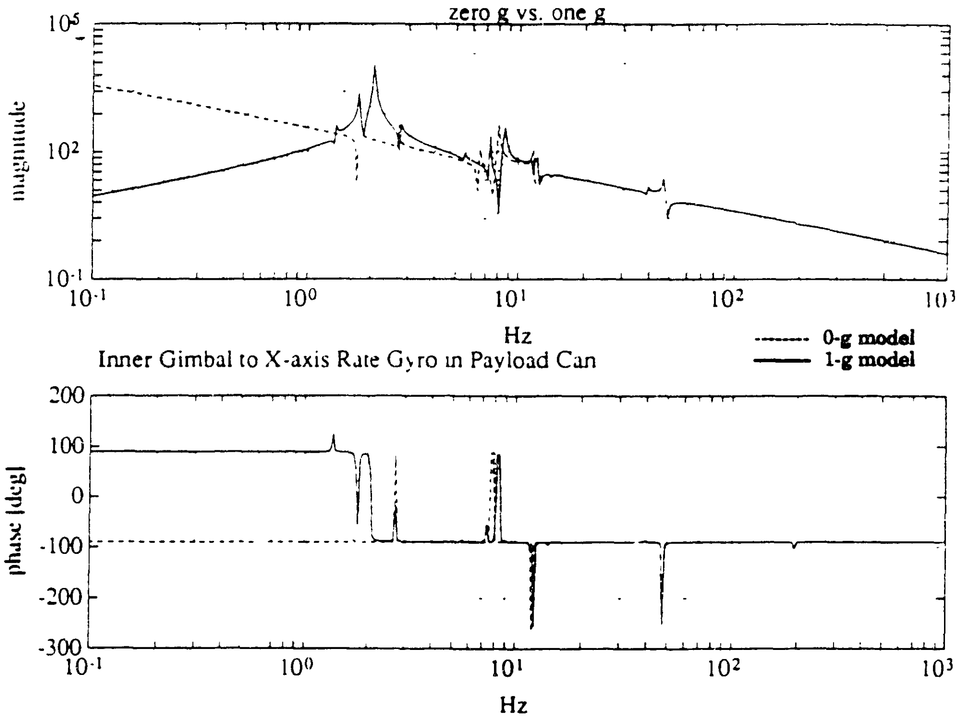
mode number	frequency [Hz]	mode shape
1	0	rigid body (wheel)
2	0	rigid body(wheel)
3	0	rigid body(wheel)
4	0.2550	pendular
5	0.2736	pendular
6	0.2779	pendular
7	0.5663	bounce mode
8	0.6221	pendular
9	1.3544	pendular
10	1.5494	pendular
11	1.7517	pendular
12	1.7616	1st v b of bus
13	2.0676	1st h b/t
14	2.7370	2nd h b/t
15	5.5216	1st b of susp. cable
16	5.5228	1st b of susp. cable
17	5.5345	1st b of susp. cable
18	5.5346	1st b of susp. cable
19	5.5447	1st b of susp. cable
20	5.5489	1st b of susp. cable
21	5.6264	2nd v b
22	7.3116	1st torsion
23	8.5234	2nd torsion
24	9.6334	3rd v b
25	11.5418	3rd h b/t
26	11.6951	2nd b of susp. cable
27	11.6953	2nd b of susp. cable
28	11.6996	2nd b of susp. cable

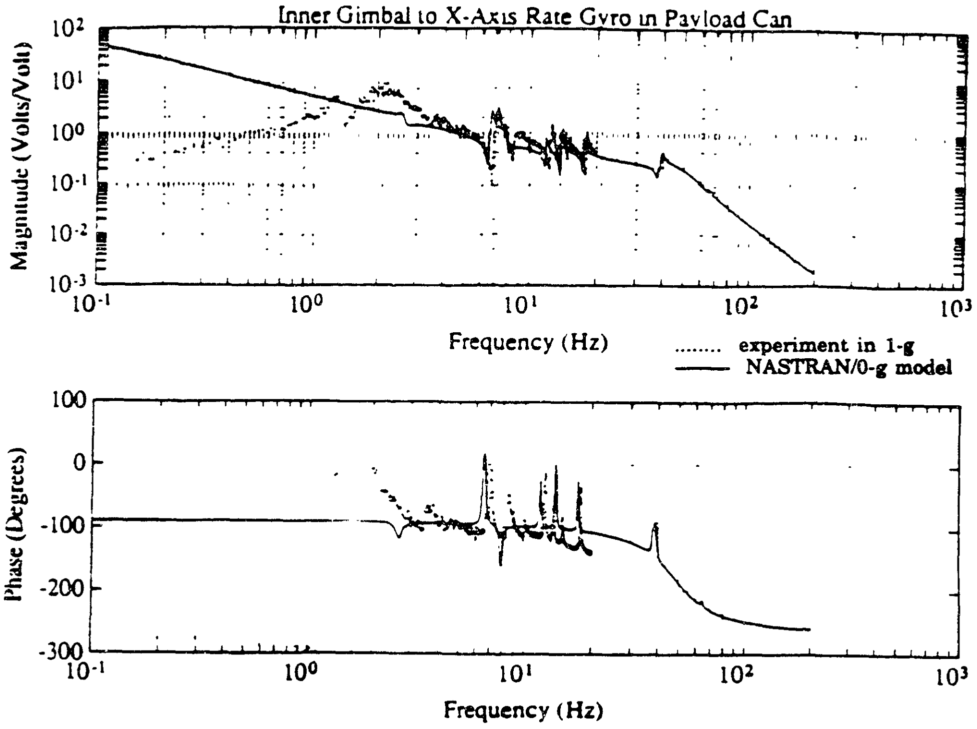
29	11.7008	2nd b of susp. cable
30	11.7018	2nd b of susp. cable
31	12.0991	2nd b of susp. cable
32	12.7399	4th v b
33	13.9495	4th h b/t
34	37.7449	3rd torsion
35	39.1587	5th h b/t
36	40.2281	5th v b
37	41.6882	6th v b
38	44.0652	6th h b/t
39	47.0336	7th h b/t
40	71.7161	7th v/b
41	76.1501	
42	77.1458	
43	77.3637	
44	115.1039	
45	117.7176	
46	124.7451	
47	193.1589	
48	193.8744	
49	310.6729	
50	331.0982	

* Test article in the all-in-plane configuration with payload #1 locked and no internal distribution of angular momentum.(mode shapes are difficult to visualize, also because they are coupled to the displacements of the payload and of the suspension cables, therefore only the frequencies are reported here).

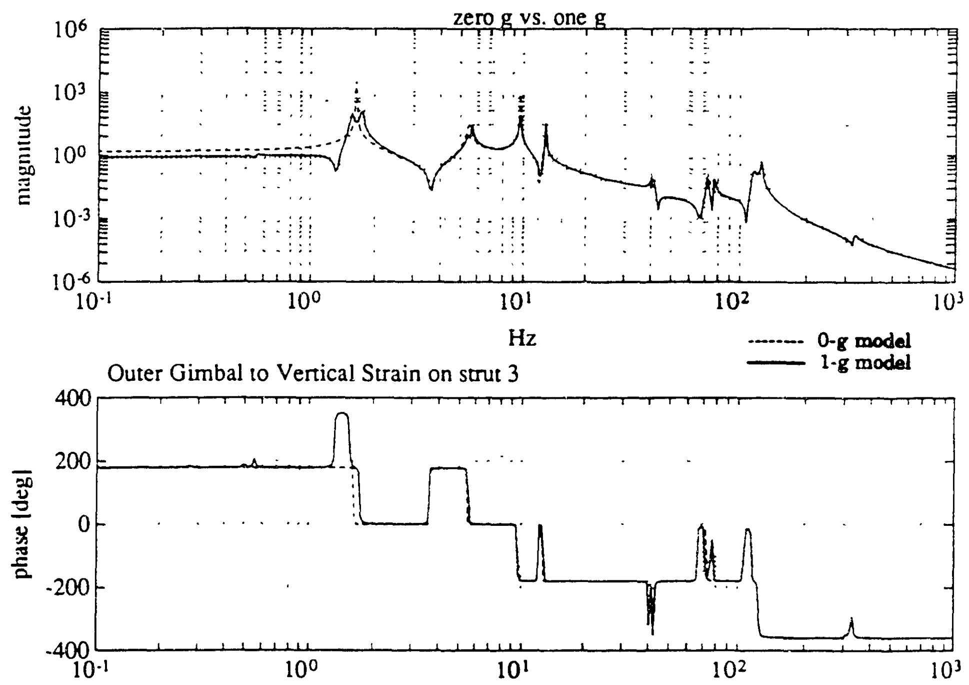
Finally, figure [3.12a] shows the same transfer function of figure [2.10]. Again, the difference is only at low frequencies. Figure [3.12b], instead, shows the transfer function from gimbal torque to inertial pointing angle (configuration vertical down) for a simplified, two-dimensional model of the suspended test article, with geometric parameters in all equal to those of the real test article. The system with a rigid suspension cable would be therefore a rigid triple pendulum stiffened by gravity. In particular, the model would somehow be representative of the out-of-plane pointing dynamics of an articulated body mounted on a ground testing suspension. The dotted line considers the suspension cable rigid, while the continuous line represents the behavior when the cable is flexible, and represented with only one finite element derived as discussed in section 3.3.2. The ripples at 0.2 Hz represent the suspension modes and those at 11 Hz the first violin modes of the suspension cable. The poles at about 1 and 6 Hz are the resonant frequencies of the compound rigid pendulums formed by the end body and the intermediate node, respectively. Because the overall dynamic behavior is almost unchanged,i.e. the magnitude excursion of these suspension effects is rather insignificant , one can deduce that modelling the suspension cables as rigid rods is, indeed, a rather good approximation.

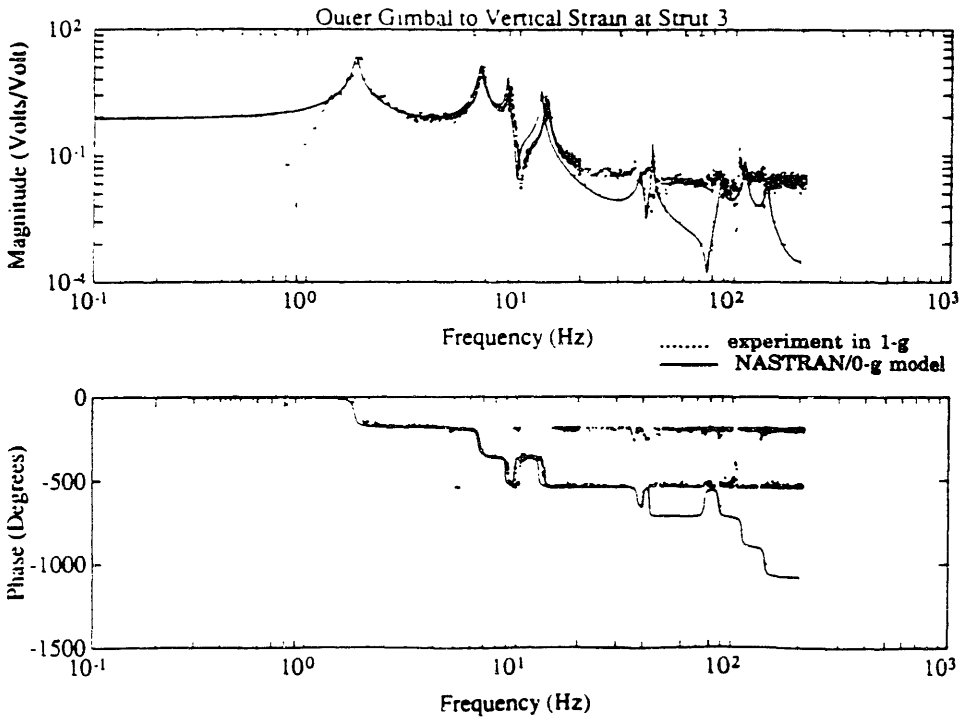
Figures 3.6 a and b



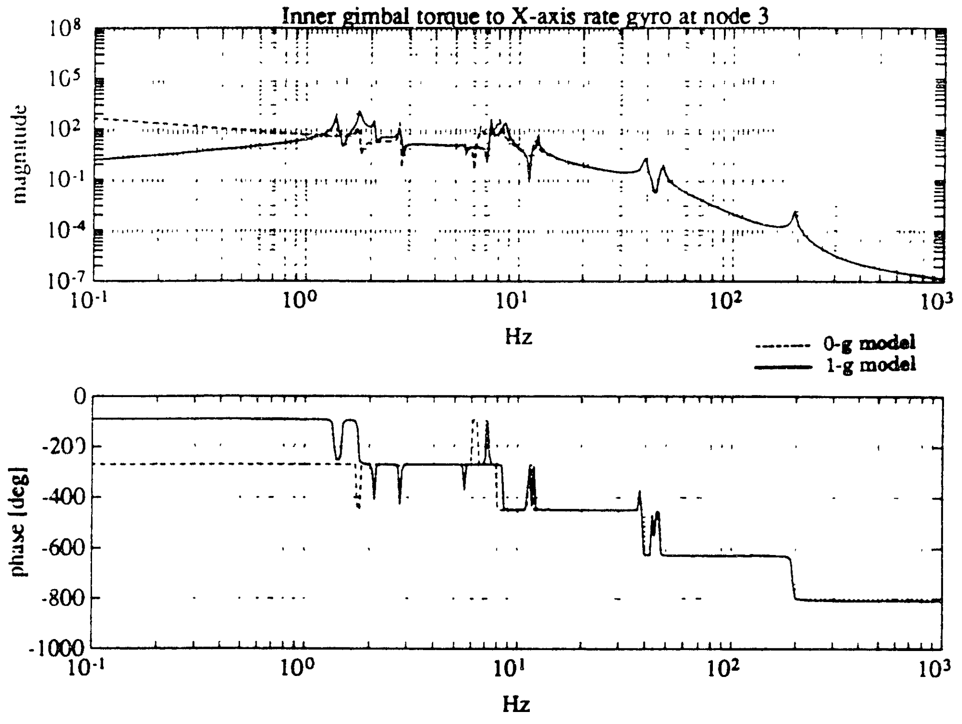


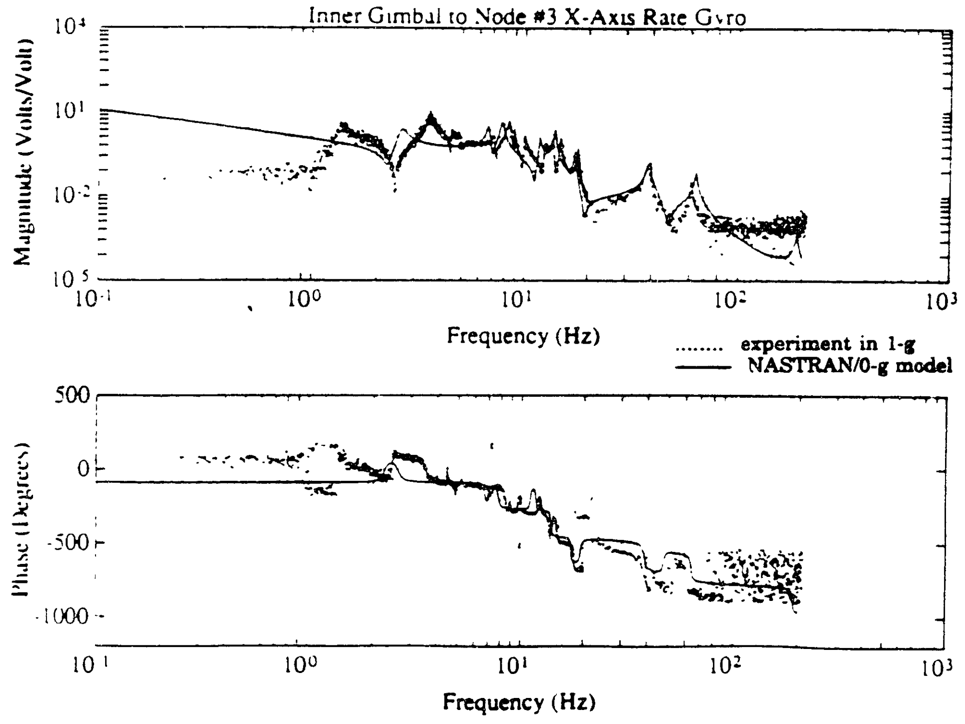
Figures 3.7 a and b



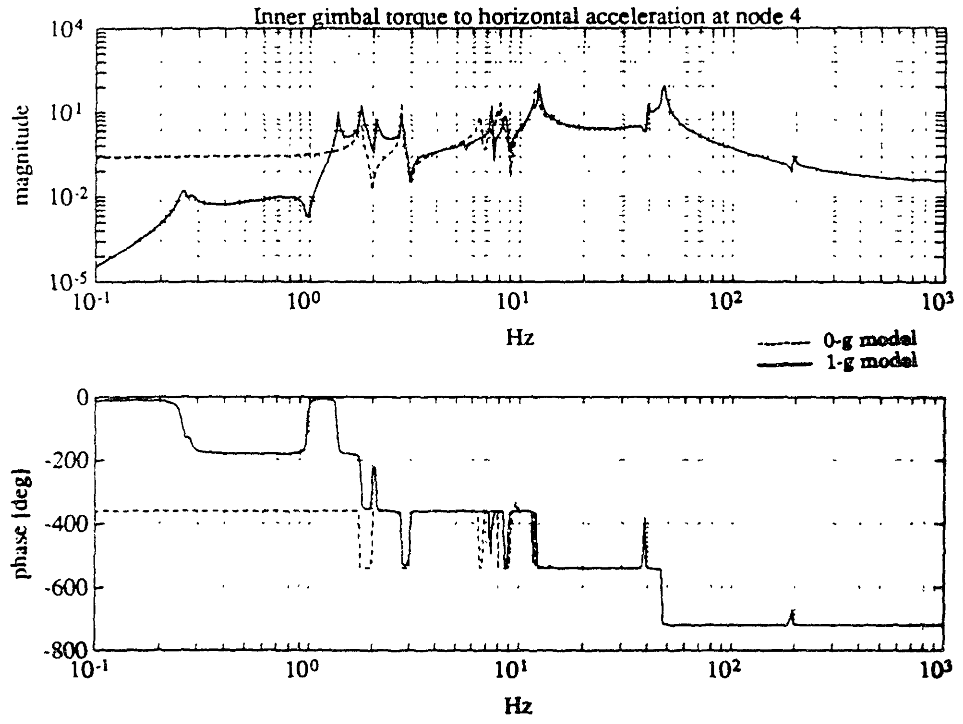


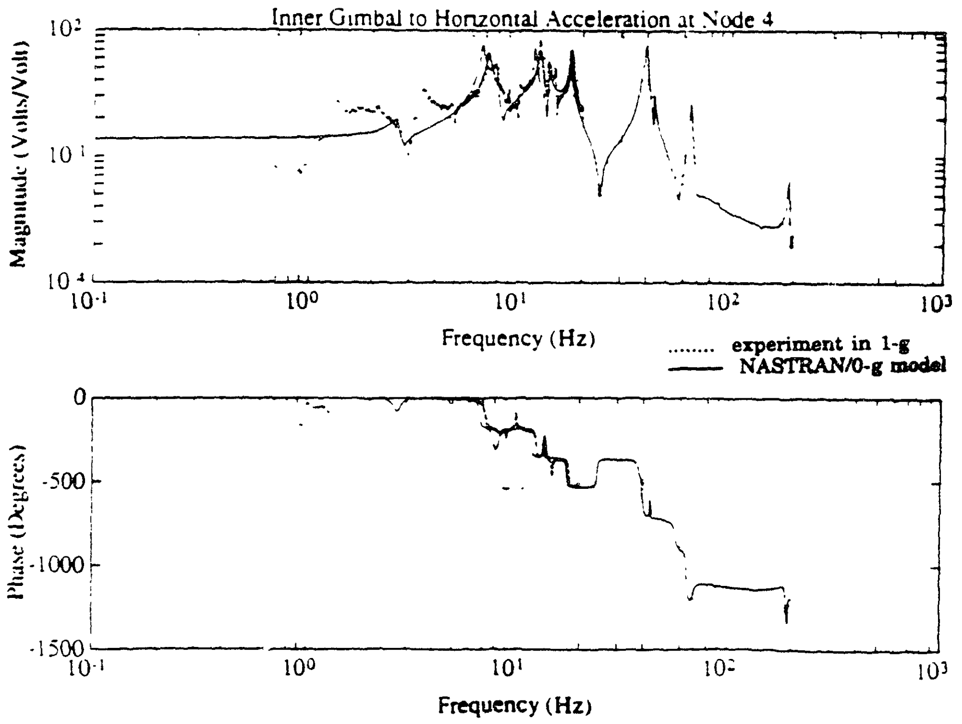
Figures 3.8 a and b



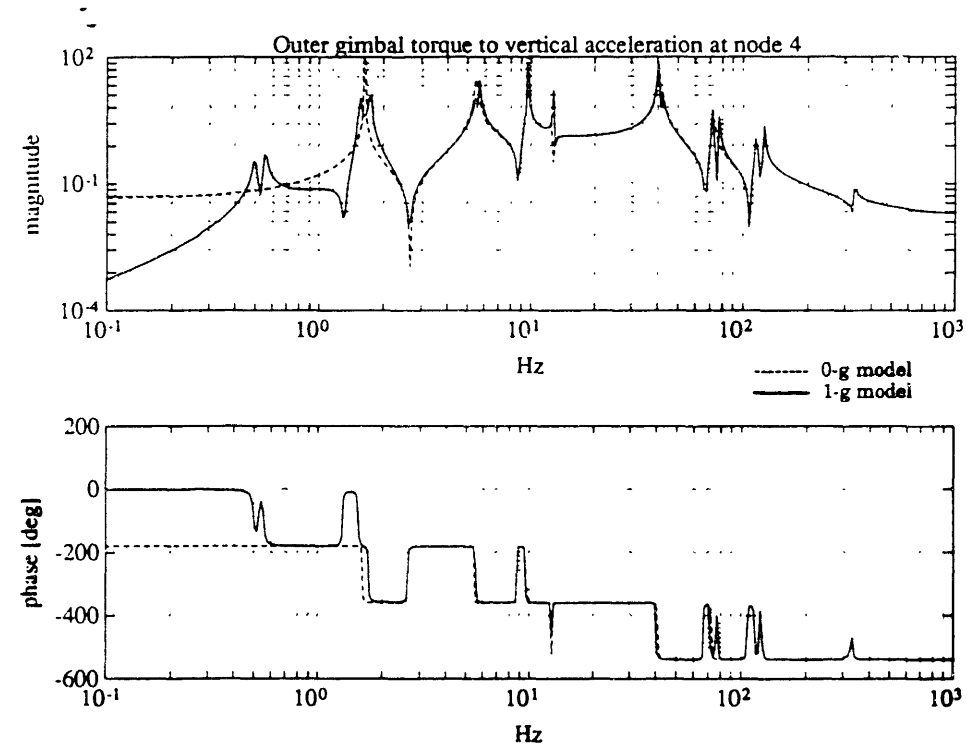


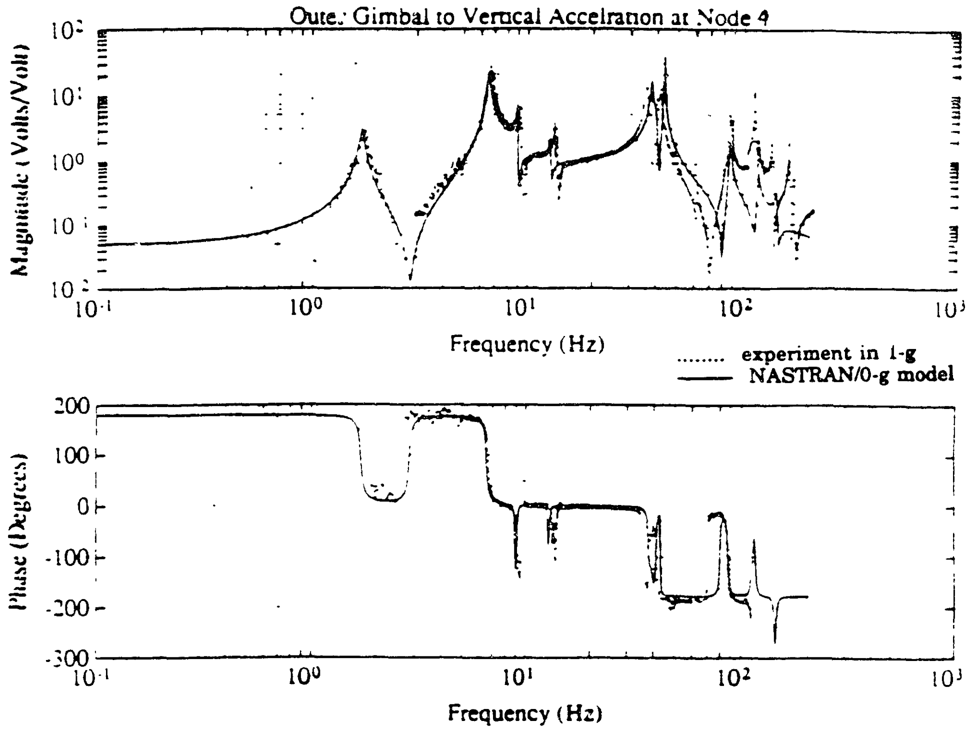
Figures 3.9 a and b



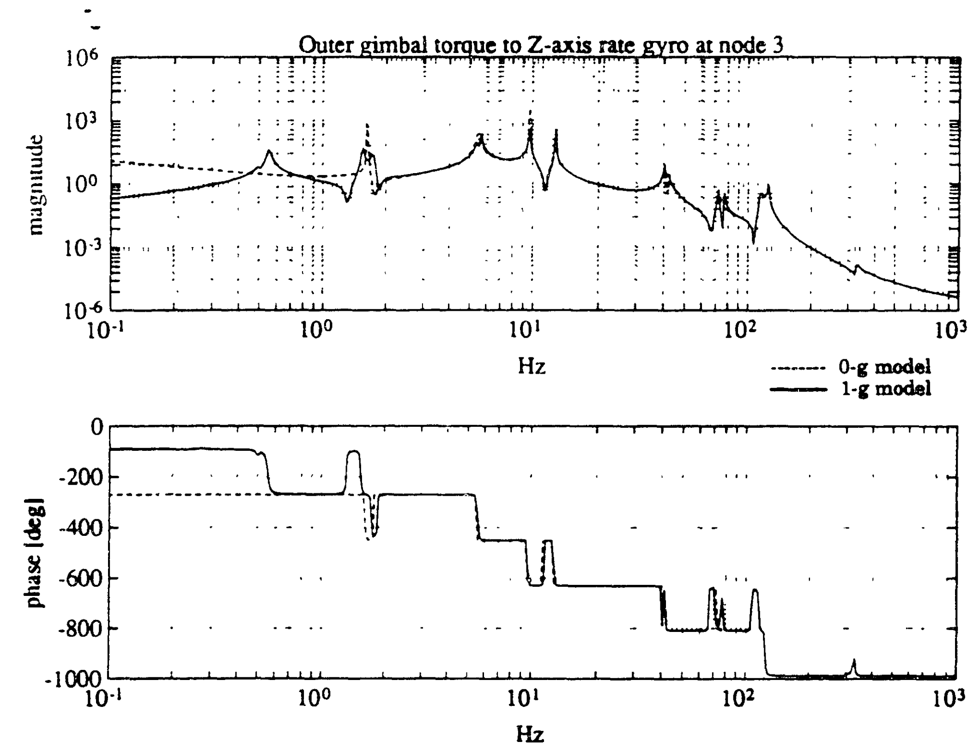


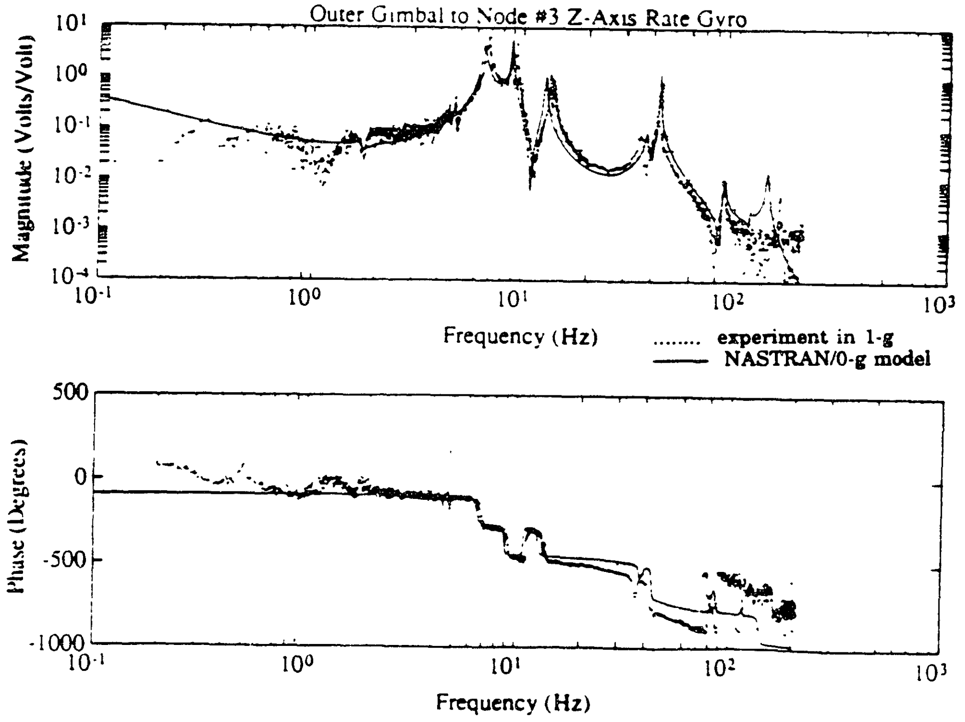
Figures 3.10 a and b



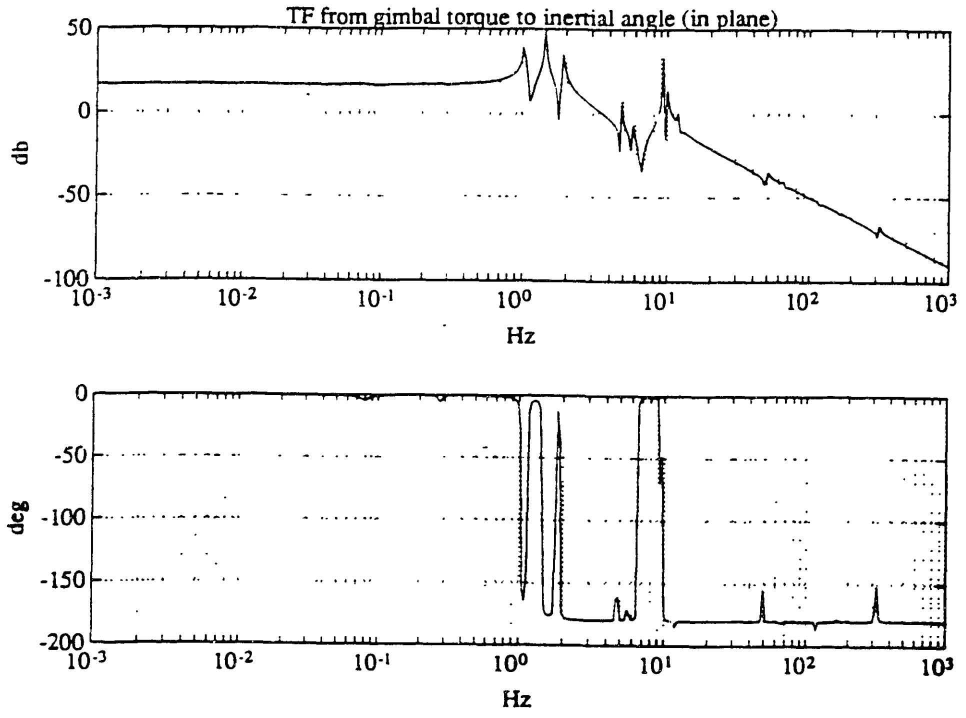


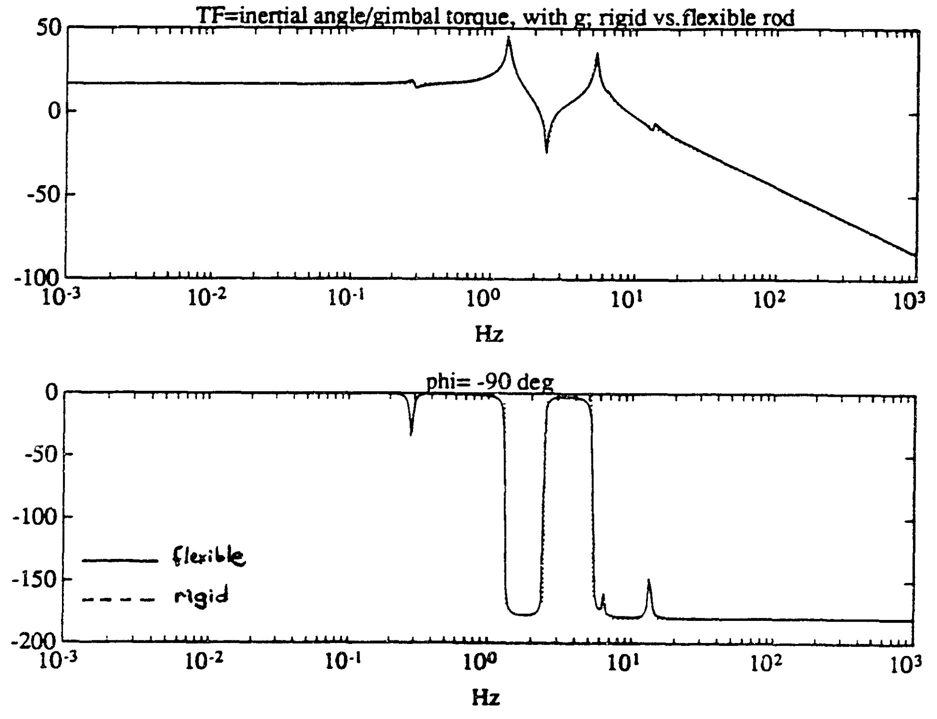
Figures 3.11 a and b





Figures 3.12 a and b





CHAPTER 4: LARGE ANGLE MANEUVERS ON MACE

Introduction.

Chapter 2 has presented the derivation of the equations of motion for the non-linear model of the test article in zero-g, including fully non-linear terms in payload angles and rates, and wheel speeds.

Chapter 3 has presented the derivation of the non-linear model of the test article in one-g, by augmenting the model derived in chapter 2 with the additional internal dynamics of the suspension device and the presence of the suspension cables.

In this chapter, both a linear and a non-linear time simulation of the dynamics of the system in zero-g and one-g are carried out, with the main objective of testing a feedback linearizing control law which ensures the stabilization of the attitude of the spacecraft and a globally asymptotically stable tracking error dynamics for the articulated body. The purpose of this study is to simulate the slew maneuver of one of the payloads with the presence of the bus attitude controller, and quantify the degradation of performance due to gravity and suspension, in the sense of deviation of the inertial pointing angle from the designed value at the end of the maneuver. The residual vibration is not intended to be minimal in the sense of minimizing a given performance index, but of very small magnitude. Numerical results show the comparison between these two analyses. In addition, the balanced reduction algorithm has been used to reduce the order of the model to a more manageable size for purposes of numerical simulation.

4.1. Numerical Simulation.

The equations of motion for the zero-g model and the one-g model were derived in chapter 2 and 3, respectively. They were programmed in Matlab programming language and they were implemented in a Sun workstation. The codes are enclosed in Appendix A4, together with the most important sub-programs and user-defined functions required to run both the linear and the non-linear time simulation. The integration subroutine is "ode23.m", which is one of the functions of the Matlab software package, and is based on a Runge-Kutta scheme with stepsize control. The driving programs to run the simulation accept , interactively, data concerning the initial configuration of the payloads and the position of the suspended test article in the N frame. They also allow the user to select the kind of controller bandwidth (high or low) and the type of trajectory that the slewed payload will follow, as described in the next section. In the non-linear simulation, and to avoid the solution of an eigenproblem at each step, a precomputed matrix of uniform viscous damping equal to 1% on each mode (and computed as described in chapter 2) is loaded at the time the initial conditions are read. Therefore, the damping matrix is considered to be independent of the actual mode shape during the whole simulation. In the linear simulation however, the model is linearized about a pre-specified payload configuration and use is made of the normalized modal coordinates. The file "animation.m" allows to plot three views (front, top and side) of the test article, both in zero-g and in one-g, undergoing the motion resulting from the payload slew.

4.2. TRAJECTORY TRACKING OF THE PAYLOAD USING A FEEDBACK/FEEDFORWARD APPROACH.

In this section we will derive a set of feedback linearizing control laws to achieve trajectory tracking of one of the articulated payloads performing a slewing maneuver according to a given time profile, and a similar control law for the spacecraft attitude such that the dynamics are stabilized around the zero state. The control strategy is, therefore, to stabilize the bus in inertial space and to reorient the payload with respect to the bus in such a fashion that the resulting effect of the bus vibration on the payload pointing is small. However, no optimization is carried out such as a linear quadratic regulator design (LQR) of the control law so that the resulting vibration is actually minimal. For further study, see Ref.[24,26], and Ref.[25]. The latter gives a thorough bibliographical description of the state of the art in single-body and multi-body maneuvers performed in space structures.

We first recall the equations of motion derived in chapters 2 and 3. For simplicity, we consider only the equations of motion in zero-g, and the control laws will be derived with reference to this model. They will then be implemented unchanged on the one-g model (with the exception of adding a non-linear feedback term which is expected to cancel the gravity torques). A comparison follows between the numerical results obtained with the linear LTI model and those obtained with the non-linear time simulation. This comparison is intended to show any discrepancies caused by the non-linear effects. The last section of this chapter presents the balanced reduction algorithm as a means to reduce the order of the model, and a way is introduced to incorporate configuration-dependent reduced-order modal data into the non-linear equations of motion for purposes of performance verification using numerical integration forward in time.

4.2.1 Equations of motion of MACE in zero-g

The equations of motion of the spacecraft, with flexibility, torque-wheel attitude controllers and independent articulated rigid payloads are given by:

$$\begin{bmatrix} M_{ee} & M_{eo}(q_o) & M_{ew} \\ M_{oe}(q_o) & M_{oo}(q_o) & M_{ow} \\ M_{we} & M_{wo} & M_{ww} \end{bmatrix} \begin{pmatrix} \ddot{q}_e \\ \ddot{q}_o \\ \dot{q}_w \end{pmatrix} + \begin{pmatrix} [G_{ee}(\dot{q}_w) + D_{ee}]\, \dot{q}_e + K_{ee}\, q_e \\ 0 \\ 0 \end{pmatrix} + \\ + \begin{pmatrix} F_e(q_o, q_e, q_o, \dot{q}_e) \\ F_o(q_o, q_e, \dot{q}_o, \dot{q}_e) \\ 0 \end{pmatrix}_{N.L.} = \begin{pmatrix} F_e \\ F_o \\ F_w \end{pmatrix}_{Control}$$

(4.1)

where the subscripts "N.L." and "control", stand for the non-linear and the actuation forces respectively. The state vector has been ordered so that:

$$q = [(x,y,z,\theta_1, \theta_2, \theta_3)_{F1}, \ldots,(x,y,z,\theta_1, \theta_2, \theta_3)_{Fn}, (\varphi, \theta)_1, (\varphi, \theta)_2, (w_1, w_2, w_3)\,]^T = [q_e\ q_o\ q_w]^T$$

(4.2)

where the subscripts e,o and w stand for "elastic","internal rigid body", " wheel rates" degrees of freedom. It is very important to note the functional dependencies of the different terms of the equations of motion, because the knowledge of these terms enables one to simplify the derivation of the control law. Note the dependency of the inertia matrix on the configuration of the articulated bodies. Also, the gyroscopic matrix is dependent on the torque wheel speeds. In addition, the non-linear terms which are made up of Coriolis and centripetal terms, are functions of the payload angles and rates, and of the node rotational displacements as well. These latter terms are typically Coriolis and centripetal terms in the node rotational velocities. However, retaining these terms non only complicates the derivation of the equations of motion, but also represents a complicating factor during the simulation of the slew maneuver. Also, for slow maneuvers (with respect to the fundamental frequency of the base-body) it has been proven that these terms are not necessary in the equations. These facts substantiate the considerations of the next paragraphs.

Model Simplification for Feedforward Design.

The equations of motion for the articulated body are therefore:

$$M_{oe}\, \ddot{q}_e + M_{oo}\, \ddot{q}_o + F_{NL}\,(\dot{q}_e\,, \dot{q}_o\,, q_e\,, q_o) \;=\tau \tag{4.3}$$

where q_e = base-body deformational coordinates (finite element generalized displacements) and q_o = in-plane and out-of-plane relative gimbal angles (measured by the encoders).

However, the previous analysis (section 2.5) conducted on the collocated transfer function from gimbal torque to inertial gimbal angle shows that:

$$\frac{q_o\,(s)}{\tau(s)} \simeq \frac{1}{J\,s^2} \tag{4.4}$$

where J is the inertia of the payload about its center of mass, or the effect of the base flexibility on the pointing dynamics in zero-g is small. This is because, as shown previously, there occurs an instance of nearly pole/zero cancellation.

This allows one to consider the inertial contribution of q_e in the equation as a small quantity and therefore to drop the contribution of $M_{oe}\, \ddot{q}_e$. Therefore, we can consider the pointing payload as an independent rigid body.

Also, for slow payload maneuvers (i.e., when $\dot{q}_{omax} < \frac{1}{10}\, \omega_{fundamental}$ of the flexible base structure consider at rest in an inertial reference frame) we can, in principle, also neglect the contribution of the base-body elastic dynamics in F_{NL}., since these terms will be at least one order of magnitude smaller than the non-linear terms dependent on the articulation rate.

This is equivalent to the process of applying the further step of "ruthless linearization", namely, "brutal linearization" to the equations of motion (see Ref.[14,15]), in which all non-linear terms involving the elastic deflections and speeds are ignored. Furthermore, this is equivalent to making a spectral separation argument as the contribution of flexibility in the rigid body equations is neglected (because it constitutes fast dynamics), but the contribution of the rigid body generalized coordinates (slow dynamics) in the flexible equations is retained (because the system is configuration dependent and the eigenstructure actually depends on the internal rigid body dynamics). Other arguments, however, which have implications in the design methodology, support the simplification mentioned above. In particular, it is generally true that the modes above the control bandwidth tend not to be important for controller stability considerations, even though they can alter the performance requirements since they cause the phenomenon of spillover. Fortunately, for simple constant gain feedback , the spillover phenomenon is not an issue, and we need not be concerned with the influence of high frequency modes. It is also generally true that sensor and actuator non-linearities, and other kinds of un-modelled dynamics, have a larger influence in the response than the elastic non-linear terms.

In conclusion, to design a reference slew maneuver for the articulated body, it is sufficient to consider the independent, simplified dynamics:

$$M_{OO}\, \ddot{q}_O + F_{NL}\, (\dot{q}_O\, , q_O) = \tau \qquad (4.5)$$

4.2.2 TRAJECTORY DESIGN

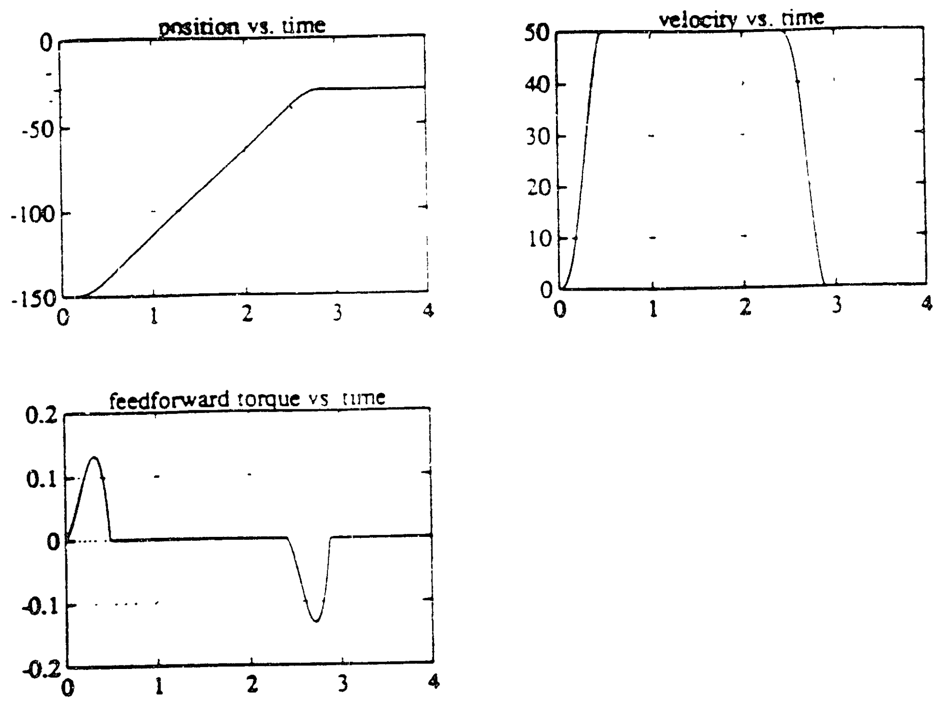
Consider a "reference" slew maneuver, to investigate the effects of the non linear terms during the slew. From the Engineering Model Hardware Requirements Document, Ref.[2], the limit to the slew range is fixed to 120 degrees, both in-plane and out-of-plane. Also, the maximum slew speed is fixed to 50 degrees/sec, and the minimum acceleration to 50 degrees.sec to be achieved in 10 degrees from rest. Therefore, a profile was chosen as shown in Figure [4.1], with the following features:

- velocity ramp from 0 degrees/sec to 50 degrees/sec within the first 10 degrees (and down);

- in-plane maneuver from -150 degrees to -30 degrees (or a full 120 degrees slew).

Because the trajectory is symmetric with respect to the halfway point in time, in which the payload is pointing along the vertical down, we use quintic polynomials in time (Ref.[27]) for the lift-off and set-down ramps and a linear segment for the intermediate path. The maximum torque during the slew is 0.12 Nm. The choice of the quintic polynomials allows one to impose six end boundary conditions in terms of angle, angular velocity and acceleration so that the resulting trajectory is a smooth function of time. The angular position in degrees, the angular velocity in degrees/second and the feedforward torque in Nm are shown in Figure [4.1].

Figure 4.1



4.2.3.DERIVATION OF THE FEEDBACK/FEEDFORWARD CONTROL LAW

As shown above, the dynamics is configuration dependent. It is also strongly non-linear in centripetal and Coriolis terms during the maneuver.

If possible, one should try to make the dynamics exponentially asymptotically stable. One way to do this is through a feedback/feedforward linearizing control law (see Ref. [28,29]).

The feedforward part cancels the non-linearities which are possible to cancel and makes the system follow the desired profile. This approach is known as computed torques in robotics. The feedback part, essentially a proportional/derivative control law, stabilizes the tracking error dynamics during the maneuver by overcoming any kind of structured uncertainty (parameter errors on the model) but it is limited by requirement that the frequency content of the control signal does not spill over into the region of the unmodeled (or unstructured) dynamics, which for this 3D model of MACE begins about 100 Hz (i.e., the point where the discrepancy between the experimental and the analytical data starts to grow dramatically). This limit in bandwidth is lowered also by the presence of neglected time delays in sensors and actuators, which were not included in the derivation of the model. For example, the rate gyros have a nominal bandwidth of about 50 Hz, the accelerometers above 300 Hz. Therefore, this model is assumed to be accurate until a frequency of 100 Hz, after which actuator and sensor dynamics and unstructured uncertainty in the flexible modes cause magnitude and phase errors.

The justification to using this approach can be given as follows. For a second order system described by:

$$M(q)\,\ddot{q} + C(q, \dot{q})\dot{q} + F_{gravity}(q) = \tau \tag{4.6}$$

choosing the control torque as in:

$$\tau = M(q)\,v + C(q, \dot{q})\dot{q} + F_{gravity}(q) \tag{4.7}$$

will transform the original equation into the new system:

$$\ddot{q} = v \tag{4.8}$$

If one now introduces the tracking error $\tilde{q} = q - q_{desired}$ and one makes:

$$\tau = M\,\ddot{q}_{desired} - K_P\,\tilde{q} - K_D\,\dot{\tilde{q}} + C(q,\dot{q})\,\dot{q} - F_{gravity}(q) \tag{4.9}$$

one also obtains

$$M\ddot{\tilde{q}} + K_D\,\dot{\tilde{q}} + K_P\,\tilde{q} = 0 \tag{4.10}$$

or the tracking error dynamics are linearly, asymptotically, exponentially stable. This very simple and practical approach has the inconvenience of depending on the knowledge of an exact model of the plant. In fact, we can only cancel exactly those terms of the equations which we know very well. It also implies that the gains of the PD part will be pushed up to compensate for this lack of information and the problem of actuator saturation will eventually pose a limit to the bandwidth of this controller. More elegant approaches would be to design a robust controller based on the theory of sliding mode control, whose main objective is to counteract both parameter and unmodeled uncertainty, but in reality this would not be really justified since the effect of the uncertainty (i.e., case of a robotic arm picking up a load instead of a pointing payload) in this case is related only to the poor knowledge of the base-body flexible dynamics. In practice, the dynamics of equation (4.10) is never perfectly realized, since the non-linear Coriolis and centripetal forces are not fully cancelled (they depend also on $\dot{q}_e$ and q_e) and we also have the inertia terms due to flexibility, which we assumed negligible. This is enough for us to expect some tracking error at steady state. As we will see in the next sections, the control law of equation (4.9) is applied to a system in which the mode shapes are not updated during the simulation. The performance will therefore be not the expected one since the model of the system is not the correct one.

Here, K_P and K_D are symmetric and positive definite matrices, equal to a constant multiplied by the identity matrix. As it is commonly done in trajectory control, K_D is chosen to get good performance based on a critically damped response, i.e. $\zeta = 0.707$, or to avoid overshoot when the end point has been reached. K_P is chosen based on the fundamental flexible frequency of the base-body. The suggestion was made in a previous non-linear study of the 2D MACE dynamics using DISCOS (see Ref.[30]) that high bandwidth (with respect to the 1st bending frequency, equal in the 3D Matlab model to 1.638 Hz) collocated control gives more pointing performance than the low bandwidth one, at the expense of exciting higher modes in the flexible bus. In this study, different cases were analyzed for different control topologies. In particular, a centralized sensing scheme, in which the inertial angle of the payload was inferred from the attitude of the spacecraft (assuming the base-body to be rigid), was compared to a localized sensing scheme, in which the inertial angle of the payload was measured by an inertial platform located at the payload's center of mass. A low bandwidth PD loop was also closed at the bus attitude control location. In any case, i.e. centralized or localized sensing scheme, the dynamics during position control of one of the end payloads when the other acts as a disturbance, was shown to be globally asymptotically stable. For this reason, a high bandwidth of 16.3 Hz , i.e. ten times the fundamental of the bus, was chosen and the gains for implementing the PD loop in the in-plane maneuver are:

$$\begin{aligned} K_P &= 452.57 \ \text{kg m}^2/\text{sec}^2 \\ K_D &= 6.230 \ \text{kg m}^2/\text{sec} \end{aligned} \qquad (4.11)$$

Equation (4.9) represents the actuator torque provided by the gimbal motor to execute the maneuver. This control law can be applied to the zero-g model after cancelling the gravity term from the expression.

Deleting the gravity term in equation (4.9) and applying the control law to the one-g model, i.e. without feeding forward gravity torques, allows one to evaluate the degradation in performance (pointing accuracy) caused by the gravity field during the slew maneuver of the payload when mounted in the suspended testbed. However, when the one-g model is considered, one must also take into account the coupling with the suspension dynamics. It turns out that the mass cancellation loop in the suspension controller adds to unstable poles to the rest of the system, therefore when the suspension is activated the gain of the mass cancelling loop has to be tuned (decreased), depending on the application, so as to guarantee a stable dynamics. Since this operation is done manually by the operator of the suspension devices, the choice has been made for the simulations of the behavior of the test article to exclude the electro-mechanical part of the control, and adjust the air spring of the pneumatic pistons so that the bus centerline is horizontal at rest. Since the mass distribution in the testbed is not uniform, for example, the accelerometer packages on different nodes of the structure do not weigh the same, the equivalent stiffnesses of these springs are different. To guarantee the horizontal alignment between the nodes of the structure, at least within a static displacement of 1 mm from the rightmost to the leftmost node, the spring stiffnesses are (from right to left):

$$\begin{aligned} K_1 &= 120.0458 \text{ N/m} \\ K_2 &= 138.2746 \text{ N/m} \\ K_3 &= 119.7072 \text{ N/m} \end{aligned} \tag{4.12}$$

Note also that in this situation, we cannot expect the payload to perfectly follow a given profile defined in inertial space. In order to do so, the suspension device has to reproduce the zero-g behavior. In addition, a feedforward/feedback control law such as (4.9) would make the dynamics asymptotically stable assuming that the gain matrices are positive definite. In other words, in the case of the one-g model, the inertial angle of the payload is given by the sum of the suspension angle, the hinge angle, the nodal rotation at the attachment with the bus and the relative angle. Of all these quantities, only the relative angle is measured, and therefore, we cannot fully cancel the effect of the suspension with a feedforward term if this is not done by the suspension itself, i.e., by the mass cancellation loop.

4.3 FEEDBACK LINEARIZING ATTITUDE CONTROL IN MACE.

In this section we derive an attitude control law for the flexible spacecraft in zero-g. To do this, we consider, for simplicity, the equations of motion of the spacecraft with payloads locked, so that we assume that they represent the coupling between the bus and the torque-wheel attitude controller for an infinitely slow maneuver of the payload. These equations are given by:

$$M_{ee}\,\ddot{q}_e + M_{ew}\,\dot{q}_w + (G_{ee} + D_{ee})\,\dot{q}_e + K_{ee}\,q_e + F_{NL} = F_C \qquad (4.13)$$

$$M_{we}\,\ddot{q}_e + M_{ww}\,\dot{q}_w = \tau$$

where:

$q_e = (\ldots, \Omega, \ldots)^T$ are the base-body generalized finite element displacements and, treated as infinitesimal quantities, $\Omega = (\theta_1, \theta_2, \theta_3)^T$ are the attitude angles;

$q_w = (\omega_1, \omega_2, \omega_3)^T$ is the vector (3x1) of the wheel's relative spin;

τ = wheel's motor torques vector (3x1)

F_C = the components in base-body axes of the wheel's torques (with minus sign) = $(\ldots, -R\,\tau, \ldots)^T$;

$R = \begin{bmatrix} p_1 & p_2 & p_3 \\ q_1 & q_2 & q_3 \\ r_1 & r_2 & r_3 \end{bmatrix}$ distribution matrix of the wheel torques in body axes (constant and invertible);

$G_{ee}(q_w)\dot{q}_e$ = matrix form of vector cross product $\Omega \times H$ (H is the total angular momentum vector);

$M_{ew} = M_{we}{}^T \neq 0$ because the wheels are not aligned with the principal axes of the bus;

$F_{NL}(\theta_1, \theta_2, \theta_3)^T$ = small non-linear terms in attitude variables.

From equation (4.13b) one gets $\dot{q}_w = M_{ww}{}^{-1}\,(\tau - M_{we}\,\ddot{q}_e)$, and substituting into equation (4.13a) one obtains:

$$(M_{ee} - M_{ew}\,M_{ww}{}^{-1}\,M_{we})\,\ddot{q}_e +$$

$$(G_{ee} + D_{ee})\,\dot{q}_e + K_{ee}\, q_e + F_{NL} = [\, (\ldots,- R \,,\ldots)^T \; - \; M_{ew} \; M_{ww}^{-1} \,]\, \tau = T\, \tau \tag{4.14}$$

which shows the effect of the control wheels torques upon the bus dynamics.
The new inertia matrix $M^* = (M_{ee} - M_{ew}\; M_{ww}^{-1}\, M_{we})$ is symmetric, positive definite non-singular and constant and T is a rectangular control distribution matrix, which is also equal (by construction) to -2 R. Note that we can "extract" the attitude variables by making:

$$\Omega = \Lambda\, q_e \tag{4.15}$$

where Λ is a (3xnflex) rectangular matrix. Therefore, using the pseudo-inverse, one obtains $q_e = \Lambda^+\, \Omega$.

Following similar steps as in section 4.2, we can choose a feedforward/feedback control law also for the attitude control. In particular, we also want to cancel the effect of the gyroscopic coupling. Pre-multiplying equation (4.14) by Λ and re-arranging, yields:

$$\begin{aligned}(\Lambda\, M^*\Lambda^+)\, \ddot{\Omega} + (\Lambda\, D_{ee}\, \Lambda^+)\, \dot{\Omega} + \; (\Lambda\, K_{ee}\, \Lambda^+)\; \Omega &= \\ &= \Lambda\, T\, \tau \; - (\Lambda\, G_{ee}\, \Lambda^+)\, \dot{\Omega} = v\end{aligned} \tag{4.16}$$

and if one makes

$$v = \Lambda\, T\, \tau \; - (\Lambda\, G_{ee}\, \Lambda^+)\, \dot{\Omega} = - K_P\; \Omega \; - \; K_D\, \dot{\Omega} \tag{4.17}$$

the following feedback-linearizing control law results:

$$\tau \; = (\Lambda\, T\,)^{-1}\, [\, - K_P\; \Omega \; - \; K_D\, \dot{\Omega} + (\Lambda\, G_{ee}\, \Lambda^+)\, \dot{\Omega}\,] \tag{4.18}$$

which, if substituted back into the original equation will guarantee that, under the assumption that the model is perfectly known and with positive gains, the attitude angles Ω and the elastic coordinates will go to zero as the time goes to infinity.

The nominal attitude configuration is the zero vector. This is also the reference state for attitude stabilization.

A more direct approach can also be followed which results in an equivalently useful control law. Choosing

$$\tau = - R^{-1} K_P \Omega - R^{-1} K_D \dot{\Omega} + R^{-1} Gee^{*} \dot{\Omega} \tag{4.19}$$

where K_P = (3 x nflex), K_D = (3 x nflex) are gain matrices (all zeros except at the three actuator locations, where the sub-partition of the gain matrices are positive definite), and where Gee^{*} represents the sub-partition of Gee corresponding to the attitude angles, one achieves feedback stabilization, gyroscopic coupling cancellation and additional cancellation of non-linear terms depending on attitude rates. In this way the attitude dynamics is asymptotically exponentially stable. In fact, combining equations (4.15) and (4.14) one obtains an expression very similar to equation (4.10).

The difference is in the presence of additional inertia terms, which however do not affect stability. One must note that the term $R^{-1} Gee^{*} \dot{\Omega}$, in the simplified case in which the stored angular momentum is constant, represents a linear feedback term. However, since the eigenvalues of the gyroscopic matrix are purely imaginary, the net effect is to move the closed-loop poles back to the origin of the complex plane but along the $j\omega$ axis. The problem therefore with this gyroscopic cancellation scheme is that if the cancellation is not done perfectly, the closed loop poles can migrate into the right half of the complex plane, resulting in unstable behavior. In other words, this scheme has no phase margin. However, for small stored angular momentum, the precession pole is very close to the origin and the last term in equation (4.15) is dominated by the rate feedback term, which suggests that the errors during the cancellation should not represent a real problem. Incidentally, one should also note that errors during this gyroscopic cancellation scheme can arise also from an incorrect measurement of the attitude rates (in this example they are approximately given by the central node angular velocities).

The determination of the gain matrices in this case is more subtle. An example of the gain determination but for an eigenaxis rotational maneuver of a rigid spacecraft using the quaternion instead of the more familiar Euler angles is given in Ref.[31]. Following the assumptions made in this paper, attitude regulation can be achieved using linear feedback of the attitude angles and rates, provided the gain matrices are a function (usually a linear function) of the principal inertia matrix of the whole spacecraft and are diagonal matrices.

They must also be positive definite for stability. The constants of proportionality are chosen based on the desired second order dynamics of the Euler angle, which represents the shortest angular path between two different orientations. Approximately, its linearized second order dynamics is given by (Ref.[31])

$$\ddot{\theta} + d\,\dot{\theta} + k\,\theta/2 = 0 \tag{4.20}$$

where $d = 2\,\zeta\,\omega_o$ and $k/2 = \omega_o^2$ where $\zeta = 0.707$ for critically damped response and ω_o is the bandwidth. Choosing a settling time of 20 seconds gives a bandwidth of 0.09 Hz, which is well below the first flexible frequency and even below the suspension pendular modes of the one-g model. The stiffness and damping coefficients of the Euler angle second order dynamics turn out to be equal to $k = 2.1064$ and $d = 1.4511$, and these constants enter in the proportional and the derivative gains respectively. The gain matrices are then obtained from $K_P = k\,J^{-1}$ and $K_D = d\,J^{-1}$ as:

$$K_P = k\begin{bmatrix} 0.8568 & 0 & 0 \\ 0 & 31.6076 & 0 \\ 0 & 0 & 28.9433 \end{bmatrix}$$
$$K_D = d\begin{bmatrix} 0.5903 & 0 & 0 \\ 0 & 21.7747 & 0 \\ 0 & 0 & 19.9393 \end{bmatrix} \tag{4.21}$$

Since the effect of the control torques is distributed onto the body axes through the R matrix, we premultiply the gains obtained above by R^{-1} so that a positive definite matrix is formed. We also premultiply the Gee $\dot{q}_e$ term by R^{-1}.

Note that all the variables can be measured and used in the control law. In particular, tachometers on the wheels measure q_w, hence G_{ew}, and a rate-gyro package measures Ω, which enters into q_e and is the vector quantity directly involved in this control law. For simplicity, only the gyroscopic coupling terms are cancelled in equation (4.15), also because the remaining non-linear terms are very small when only regulation about the origin of the state space is involved. Note, finally, that in using this approach, we are not able to cancel the inertia terms caused by $M_{ew}\,\ddot{q}_w$, since the angular acceleration of the wheels is not available from the measurements.

Equation (4.19) is the control law actually used for attitude stabilization, although equation (4.18) also achieves attitude stabilization.

4.4 MODEL REDUCTION FOR MULTI-BODY SIMULATION USING THE BALANCED REALIZATION ALGORITHM.

The computer implementation of the control laws derived in the previous section have shown that the typical duration of a non-linear simulation of a slew maneuver (4 seconds) requires approximately 40 hours of computing time, while it takes only a few minutes for the linear case. This is due to two reasons: first, the programming environment in which the non-linear simulations were run is not particularly well suited for this task; second, all the states of the model were kept in the simulation (70 states for the zero-g model, 100 states for the one-g model). This suggests the fact that if higher fidelity modelling is required, which implies including more states, the task of carrying out a simulation of the multi-body dynamics becomes almost impossible.

To alleviate this problem, a model reduction approach is needed. However, the model reduction procedures currently available deal primarily with linear systems which one can express in the state space form. A multi-body system, such as MACE, is instead an example of a geometrically non-linear problem, since large angle articulations are present between the base-body and the articulating payloads. The process of model reduction of non-linear systems is still an open area of research, even though useful indications on how to treat such systems have already been announced in the literature. The conclusion is that it is always necessary to determine the modes to be retained for each component by using the information on the modes of interest at the system level. The projection and assembly method for multi-body component reduction is an example which uses this practice. Using this method, one first chooses the modes of interest at the system level, projects these modes on each component, reduces the order of the component model (at the component level) and then reassembles the reduced component models into a new reduced system model. When a component body is articulated with respect to the rest of the structure, the method is not valid any more. In Ref. [34] , the authors take the Galileo spacecraft in examination. The model reduction technique of projecting and assembling the system modes can however still be applied if different system modal data are available for a certain number of configurations of the articulated bodies, usually the most significant ones. Therefore, for each configuration, one can apply the procedure outlined above, and thus obtain input data for the non-linear simulation code in terms of these reduced order configurations.

Another approach, commonly employed in the model reduction of linear systems, is the balanced model reduction algorithm (see Ref.[32]). This different method is used to reduce the order of the model with the objective of, ultimately, design a lower order controller for the plant. Using this approach, a new set of modes, the second order or Hankel singular values, are derived for the model. These are obtained through a similarity transformation which brings the system into a form (balanced form) in which the controllability and observability grammians of the linear-time invariant system are equal and diagonal, i.e. balanced. These grammians are the solutions to the matrix Lyapunov equations:

$$AW_c + W_cA^T + BB^T = 0 \quad \text{and} \quad W_oA + A^TW_o + C^TC = 0 \tag{4.18}$$

The balanced modes which correspond to small elements in the diagonal of these grammians can be truncated since they correspond to the least controllable and the least observable modes of the system. To support this methodology is the fact that for lightly damped structures (such as MACE and other space structures), the modal representation becomes asymptotically balanced when damping approaches zero and when there are no closely spaced frequencies. This is also the case when small gyroscopic or circulatory forces are present, as discussed in Ref. [33], since their effect is generally consider as a first order perturbation to the natural frequencies and mode shapes of the original system. However, the resulting reduced order model has been proven not to be optimal in any sense. Unfortunately, the close-spaceness criterion is in general violated for MACE, which exhibits clustered frequencies.

According to these indications, we can carry out this procedure on the LTI zero-g model of MACE reduced to modal form. Therefore, from equations (2.37) and (2.38), we can solve the eigenproblem for an articulated configuration of interest during the motion of the system and find the diagonal matrix Ω^2 of square natural frequencies (including the zero frequencies of the rigid body modes) and the matrix of mode shapes Φ, normalized to satisfy the mass and stiffness orthonormality conditions $\Phi^T M \Phi = I$ and $\Phi^T K \Phi = \Omega^2$. Neglecting the gyroscopic effect and assuming uniform damping, the equations of motion in modal state space form are:

$$\dot{\eta} = \begin{bmatrix} 0 & I \\ -\Omega^2 & -2\zeta\Omega \end{bmatrix} \eta + \begin{bmatrix} 0 \\ \Phi^T B \end{bmatrix} u$$

$$y = \begin{pmatrix} C_q\Phi & C_{\dot{q}}\Phi \end{pmatrix} \eta + D\,u \tag{4.18}$$

The next step is to partition the A, B C and D matrices into the "rigid" part, which contains the zero frequencies, and the "flexible" part. The balanced reduction algorithm operates on the flexible part only, since the balanced singular values are inversely proportional to the modal damping, and therefore would be infinite for a rigid mode. After the algorithm has been applied, the rigid body part can be appended back to the truncated flexible model.

The algorithm applied for MACE, in two different configurations, namely, the all-in-plane and the all-out-of-plane configurations, produces the balanced singular values shown in Figure [4.2]. Because of the five orders of magnitude of difference between the largest and the smallest singular values, a logarithmic scale has been used. This plot shows that, presumably, the flexible modes until the 15th state contribute significantly in the actuators to sensors transfer functions, and that this contribution, for two different payload configurations, is similar in the two cases. A significant difference occurs in the 11th and 12th states, which differ strongly because in the all-out-of-plane configuration the horizontal-bending/torsion coupling is more significant. This means discarding the last 9 flexible modes from the model. In doing this, some typical open-loop all-in-plane and all-out-of-plane transfer functions evaluated before and after the reduction process are depicted in Figures [4.3] to Figure [4.4], which shows that the last 9 frequencies of this model do not contribute significantly to these non-collocated transfer functions. The collocated transfer functions from gimbal torque to gimbal angle do not show any appreciable difference, and this is due again to the nearly pole/zero cancellation. Table [4.1] shows the structural frequencies of the all-in-plane and of the all-out-of-plane configurations before and after modal truncation. Note how some modes have been ranked differently by the algorithm and this is due to the change of configuration. However, these plots and the table were obtained by considering the presence of the strain actuator attached to one of the central beams of the bus. Since this actuator applies two localized moments at two adjacent nodes, the high frequency vertical bending mode at 194 Hz is activated. Therefore, it is an important mode which has to be retained for the control design. Retaining this frequency would not make much sense if the purpose is to reduce the simulation time when only the gimbal torques are active, therefore modified B and D matrices were considered which do not include this type of actuation. A new analysis of the system using balanced truncation shows that the first 6 modes (i.e., the frequencies until 12 Hz) capture the most significant vertical bending and horizontal bending/torsion modes. This, in turn, means that the smallest integration step has now been reduced to 10^{-2} seconds, approximately.

Figure 4.2

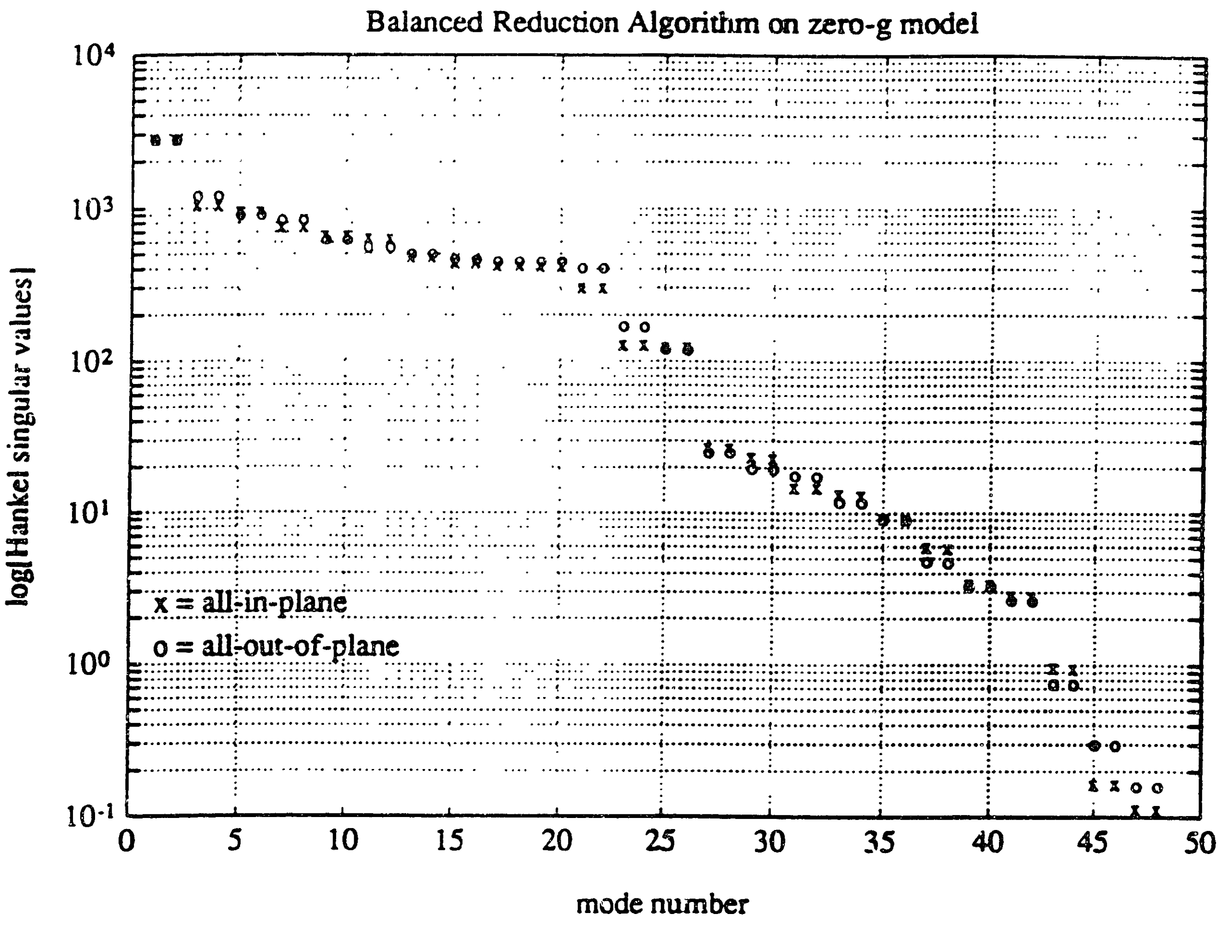
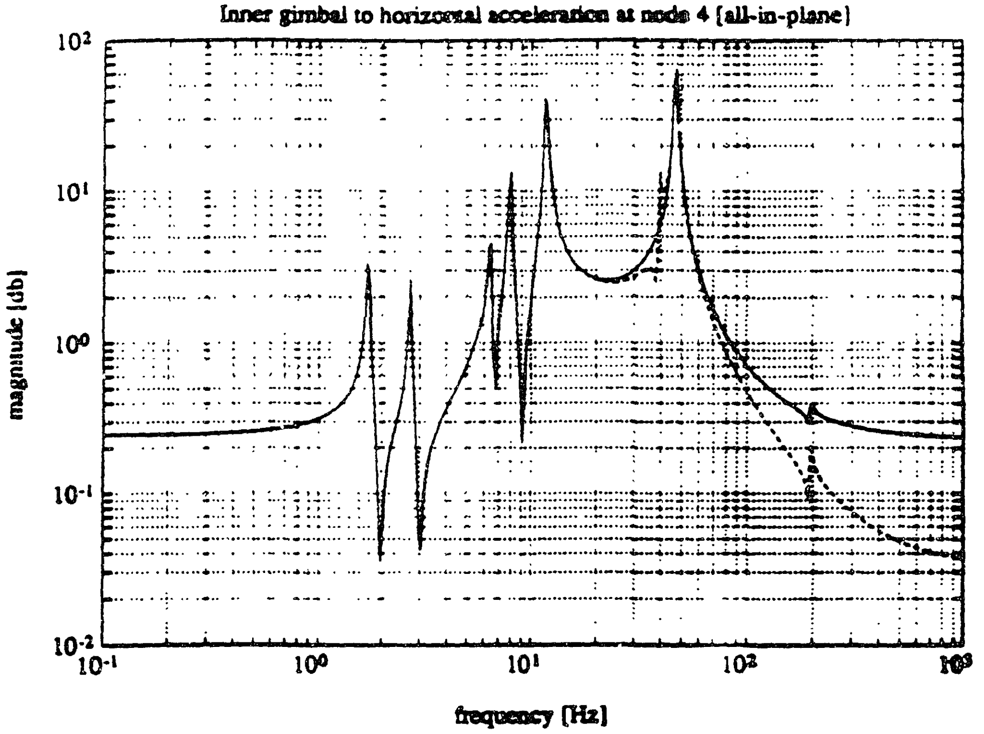

Balanced Reduction Algorithm on zero-g model
log[Hankel singular values]
mode number
x = all-in-plane
o = all-out-of-plane
10^4
10^3
10^2
10^1
10^0
10^-1
0
5
10
15
20
25
30
35
40
45
50

Figure 4.3a and 4.3b

------- full order model
——— reduced order model



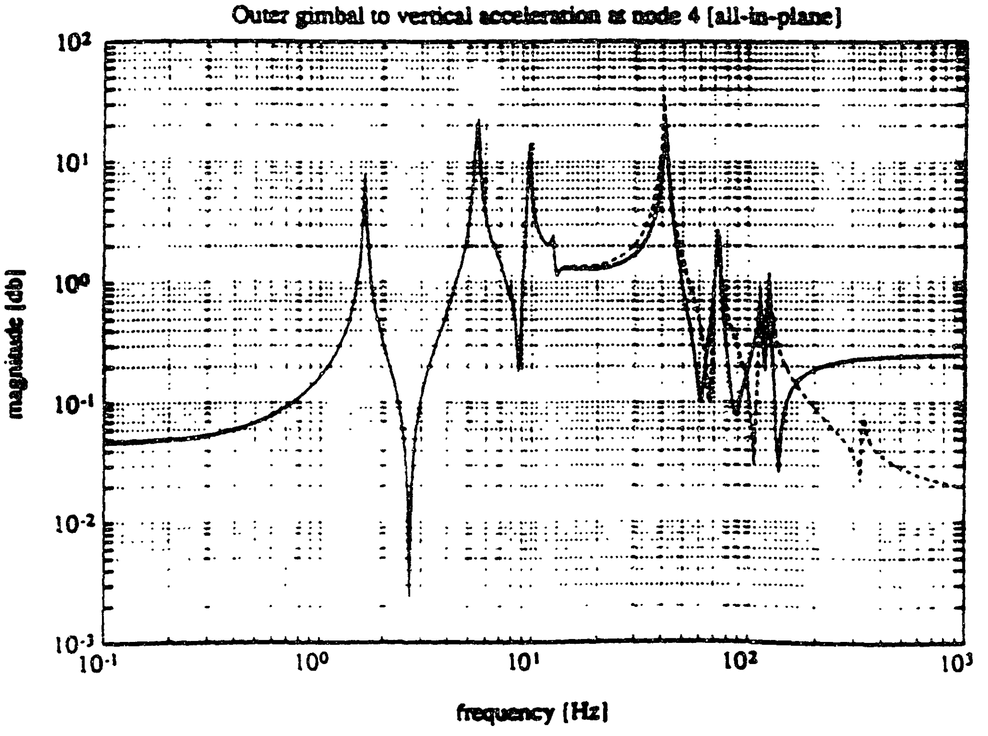
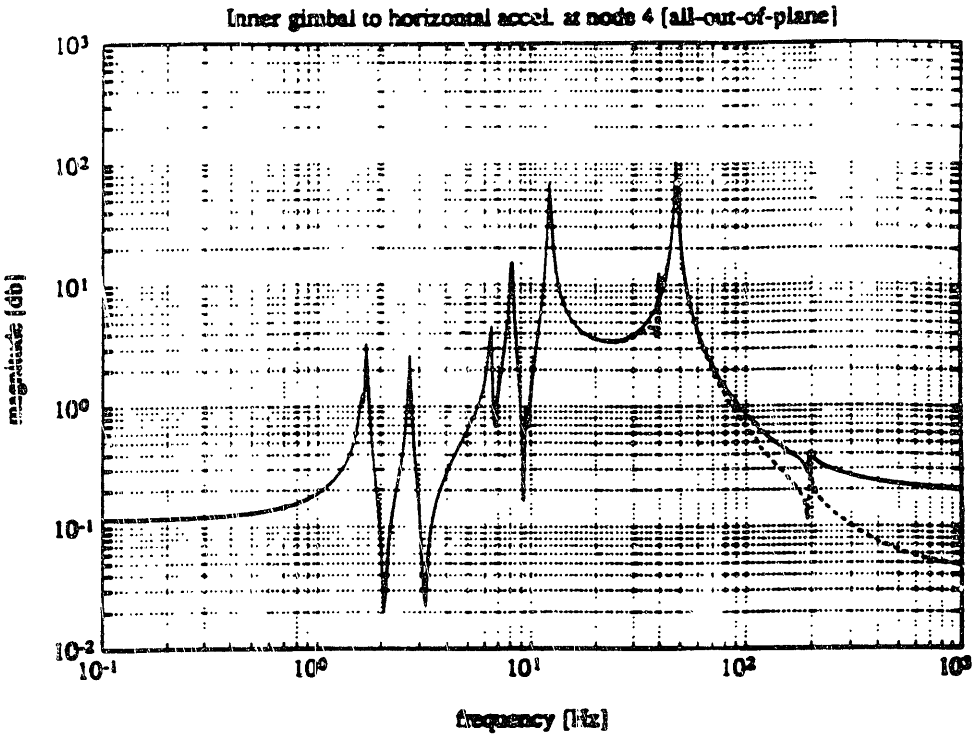
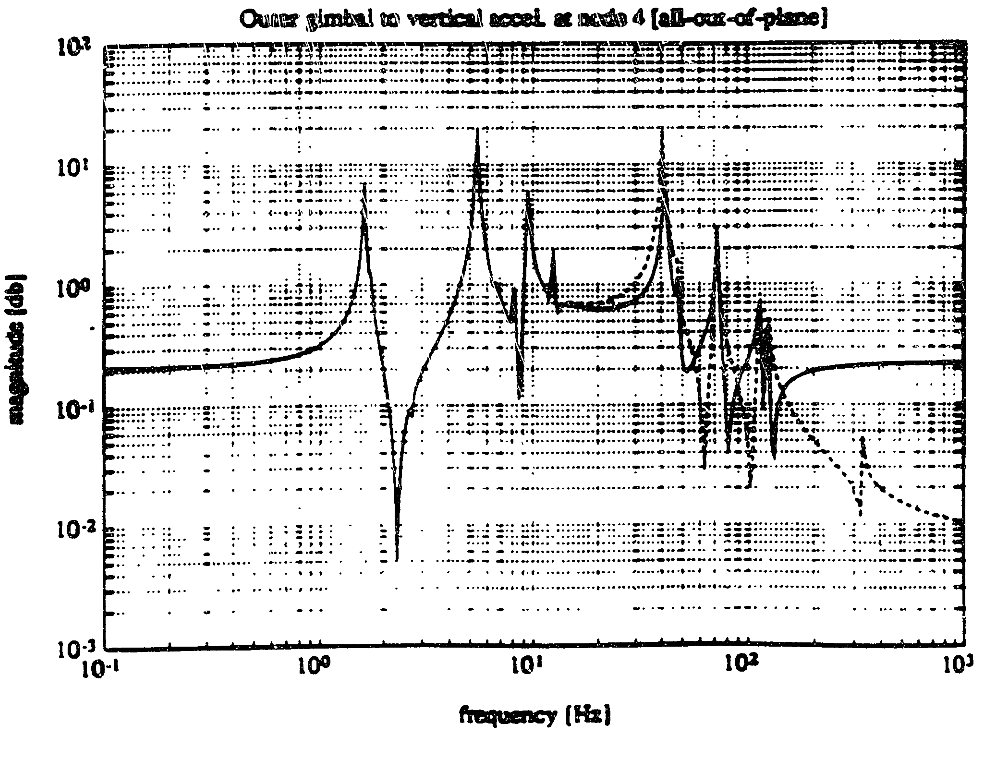


Figure 4.4a and 4.4b

-------- full order model
——— reduced order model





In conclusion, we have shown that, for purposes of linear control design, a reduced order model with 15 states can be used if the strain actuator is included, and only 6 modes can be retained if the purpose is to conduct slew maneuvers only. Alternatively, the same reduced order model could be used for non-linear control, as in the case of non-linear time simulation of large angle motions of the articulated bodies and for the purpose of reducing the simulation time, because as we have seen the effect of the flexibility of the base on the articulated motion is not very large. This is precisely the model reduction scheme implemented in the program " MACE0g.m ", enclosed in Appendix A4, and which we describe in the following paragraphs. For this purpose, and adopting the terminology commonly used in the literature, we identify the flexible beam of MACE as the "component" flexible body and the "system" modes are the modes of the whole structure, with the hinged payload in a specified configuration. This also means that the motions of the articulated part cause changes in the frequencies and mode shapes of the flexible bus, and for purpose of computer simulation, the modes must be updated. However, updating the mode shapes during the simulation is not as easy as it may seem. It involves the correct matching between the component mode sets corresponding to the different configurations, and it must be done "in-the-loop". Reference [35] presents a method to accomplish this for systems of articulated flexible bodies by assuming that the component mode position and velocities do not change during the update, in order to preserve the continuity of the sensor measurements and of the response of the system to the actuator inputs. The authors in Ref.[35], however, base their updating scheme on neglecting certain rigid/flexible coupling terms in the equations of motion (equivalent to neglecting effects such as, among others, the gimbal axis reorientation due to flexible motion) and this approximation holds only for slow appendage motion (compared to their fundamental frequency). The difficulty stems from the fact that we do not know which new initial conditions to assume when the mode shapes have been updated. Clearly, keeping the system modal set unaltered during the simulation and equal to the set corresponding to the premaneuver state makes is a valid approximation only when the change of frequencies and modal shapes with the payload configuration is small. This will not be true when the effect of the changes of configuration is significant. Therefore, one can argue that if the up-dating is done at each step, then the correct eigenstructure will be available during the whole slew of the payload and this problem would be somehow alleviated. In this case, however, we always have to begin the next step with the correct modal structure, and numerically this cannot be done exactly, thereby introducing an error. The implication is that the control design derived for the linear model does not guarantee the specified performance in the non-linear model. This issue is revisited at the end of this section.

First, the mode shape matrix obtained by solving the undamped, non-gyroscopic eigenvalue problem for the hinge-free case is reordered and partitioned into its "flexible" and "rigid" parts. In this step one must include the six rigid body modes of the system (the three translations and the three rotations) into the flexible part. This is because the only flexible component of this system is the flexible bus, which contains also the system modes that we want to reduce. The remaining rigid body modes (articulated modes and wheel rotations) are included into the rigid partition. The flexible partition, which is a rectangular matrix of dimensions (number of rigid and flexible modes) x (6 + number of flexible modes), has its columns truncated to the number of modes retained after the balanced reduction procedure has been applied, plus the original six rigid body modes, which do not change. Reappending back to the partition including the remaining rigid body modes, one obtains the new (reduced) mode shape matrix, which is now a non-square matrix. The singular value decomposition is the used to obtain its pseudo-inverse, which is needed in the multibody code to transform the full order state vector into the reduced order state vector. This inversion is applied at the beginning and at the end of the integration, since the integrator works only with the reduced order model. After this truncation, the full order mass, stiffness and damping (and gyroscopic) matrices are "projected" into the reduced order component model, by pre and post - multiplying by the reduced mode shape matrix. Similarly, the vector of generalized external forces is pre-multiplied by the transpose of the reduced order mode shape matrix. If the slew maneuver of the articulated body is done only in the vertical plane (X-Y plane), only one set of modes corresponding to the "all-in-plane" configuration can be used with reasonable approximation, since the change in the dynamics characteristics has been shown to be small (see chapter 2). Alternatively, if one desires to slew the payload also in the direction of the Z axis, one should include at least another set of modes corresponding to the "out-of-plane" configuration.

Therefore, during one half of the maneuver one can use the first set of modes and the second one in the next part. In the program "MACE0g.m" the reduced order state vector is reorganized into the full order state vector during the assembling of the equations of motion. This is done to deal more easily with the configuration-dependent inertia matrices and with the vector of non-linear terms . After the equations of motion have been assembled, the time derivative of the state vector is sent to the integrator routine in the reduced order form. After one integration loop, the reduced order modal state vector is transformed in the full order state vector after premultiplication by the mode shape matrix of the current configuration, which for small amplitude slew maneuvers can be the mode shape matrix of the pre-maneuver state, but for large angle slews needs to be re-scheduled for a certain number of payload configurations. How to smoothly incorporate these reduced order modal updates in a fully non-linear model is still an open issue and it is proposed as further research.

The same simulations described in section 4.5 have been implemented using the reduction procedure described above. It has been found that, for the variables of interest during the slew maneuver, namely, the inertial gimbal angle and the attitude angles, the difference with respect to the results obtained using the full order model is almost negligible. This is, again, due to the small influence of the base flexibility on the motion of the articulated body. More important, the computation time has been reduced of 90% approximately. For these reasons, one should always attempt to use model reduction procedures whenever faced with computationally intensive simulations of complex, multibody systems.

Table [4.1] Structural frequencies: full order and reduced order models. All-in-plane and All-out-of-plane configurations.

All-out-of-p. configuration		All-in-plane configuration	
full order	retained mode	full order	retained mode
1.6338	yes	1.6328	yes
1.7178	yes	1.7278	yes
2.7118	yes	2.7169	yes
5.3994	yes	5.4907	yes
6.4923	yes	6.4753	yes
8.1411	yes	7.9686	yes
9.4801	yes	9.6580	yes
12.2286	yes	11.7373	yes
12.6224	yes	12.6402	yes
13.9566	yes	13.9466	no
37.7403	no	37.7402	no
39.2006	no	39.1570	no
40.5296	no	40.6079	no
41.8653	no	41.8738	yes
44.0760	no	44.0670	no
47.7536	yes	47.0621	yes
71.9388	yes	72.6618	yes
113.7073	yes	115.1362	yes
117.6707	no	117.6711	no
123.7937	yes	124.6395	yes
193.1658	no	193.1601	no
194.0817	yes	193.8929	yes
310.5861	no	310.6938	no

4.5 IMPLEMENTATION OF THE FEEDBACK/FEEDFORWARD CONTROL LAWS IN THE LINEAR AND IN THE FULLY NON-LINEAR MODEL. RESULTS OF THE TIME SIMULATIONS.

The following plots show the result of non-linear time simulations of the slew both on the zero-g model and in the one-g model of MACE using the Matlab software. For comparison, also the linear time invariant model was also used. To explicitly see the influence of the non-linear terms, some simulations were run assuming no feedforward of the non-linear terms and of the gravity terms. Also, and for purposes of comparison, some simulations were run with and others without the bus attitude controller.
For example, figure [4.5a] and [4.5b] show a series of snapshots of the vertical motion of the bus and the articulated payload during the slew in the zero-g model. This slew maneuver was made to last 2.88 seconds, and the total simulation time is 4 seconds. The simulation on the LTI model takes only a few minutes, whereas the non-linear time simulation reaches 35 hours of computation time. This is because the full model was used in the simulation (70 states in the zero-g model, 100 states in the one-g model), and also because of the high frequency modes which caused the integration stepsize to be of the order of 10^{-3} seconds and smaller.

In figure [4.5a], the attitude control of the bus is not operational, and represents the results of the non-linear simulation (practically coincident with the same result obtained from the linear case). The attitude of the bus is operational in figure [4.5b], which is a result obtained from the LTI model. As one can see from figure [4.5a], the deviation of about 4 degrees at the end of the maneuver is caused by the rotation of the bus as a result of the oppposite torque applied on the bus by the gimbal motor. Note also from figure [4.5a] that there is a slight X-translation of the bus, which shows the coupling between the translational and rotational rigid body modes. This coupling has been discussed in chapter 2.

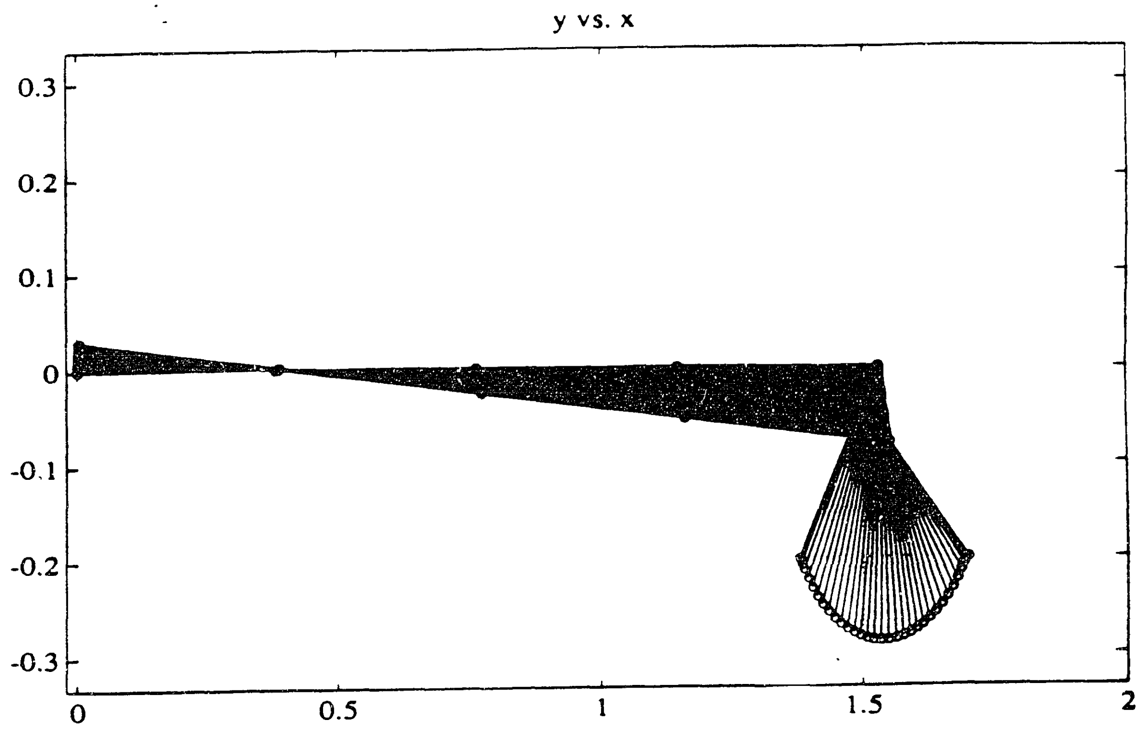
Figures [4.6] to [4.14] show the LTI response of the zero-g model and the one-g model when both the slew maneuver and the attitude regulation control are operational. To investigate the near term behavior, 10 seconds of simulation were deemed to be sufficient. For simplicity, a settling time of 4 seconds was assumed in the determination of the attitude control gains, corresponding to a bandwidth of 0.45 Hz for the attitude control loop. This still results in a stable closed-loop LTI system, although a full stability analysis using root-locus tecniques was not done. However, the non-linear simulation using this bandwidth and the truncated mode shape matrix became unstable after only 1 second. This behavior is explained below.

Figure [4.6a] and [4.6b] depict the inertial pointing angle of the payload at the end of the maneuver in the zero g case for two different situations: a CG-mounted payload and a non-CG mounted payload. In the second case, the response is considerably more oscillatory, however the exact end value of -30 degrees is not completely achieved within 10 seconds since there is a residual bus vibration with the period of the first bending mode. Note that in this amount of time, both the attitude control and the gimbal torquer are acting simultaneously, hence the combined effect of the two is visible in the plots.
The attitude angles and the nodal vertical displacements are depicted in figures [4.7] to [4.10]. In the one-g case, one should note the low frequency modulation of the vibration resulting from the pendular motion of the suspension system, with a frequency of 0.57 Hz. Also, the motion is remarkably more damped than in the zero-g case. Since the mass distribution along the bus is not uniform, at steady-state there is a residual displacement and tilting of the bus with respect to the horizontal axis. In the zero-g case, with attitude control, the payload follows the tracking profile very closely, while in the one-g case there is some deviation, a maximum of about 5 degrees, which is presumably caused by the fact that we are not cancelling gravity torques during the slew (since the control feeds back only linear terms). Also the response in the one-g case is influenced by the pendular oscillation of the suspension. These differences are shown in figure [4.11]. Figures [4.12] and [4.13] show the effect of the application of the balanced reduction algorithm to the LTI system. Practically, for an in-plane maneuver, the effect of retaining the first 6 modes does not cause a significant difference in the response, and this conclusion substantiates the use of this tecnique also for non-linear time simulation.

Figure [4.14] depicts, for the zero-g model with bus attitude control with a 0.09 Hz bandwidth (settling time equal to 20 seconds), the inertial angle of the payload obtained with the simulation using the LTI model (continuous line) and the same angle obtained from the non-linear time simulation using the truncated set of modes corresponding to the all-in-plane configuration. In addition, there is no up-dating of the mode shapes during the simulation. A flow chart showing the steps of the non-linear simulation code is presented in figure [4.16]. The result of figure [4.14] shows the different performance in the non-linear case and this is due to the fact that we are impinging the same control logic on virtually two different systems. This result, therefore, confirms that a means for updating the truncated system mode set must be included during the simulation. This task, however, will not be pursued here.

Figure [4.15] represents the relative angle obtained from the linear and the non-linear model when modal truncation was not applied and non-linear terms are not cancelled using feedback linearization. Therefore, the non-linear model is now the correct model with the correct control logic applied to it. Besides requiring almost 35 hours of computing time (for a maneuver only 4 seconds long!), this plot shows that the response is damped as in the linear case and that the non-linear terms in the equations of motion (equation (4.5)) do not represent a dramatic effect in the response of the system, since they cause a deviation in the relative angle of approximately 1 degree at most. We remark that these Coriolis and centripetal terms were obtained including also the effect of the nodal velocities. However, we should note that for accurate pointing of the payload, the requirements on the pointing performance are expressed as the root mean square (rms) value of the pointing angle and this is a value commonly specified to be less than, for example, 0.02 degrees. For accurate pointing therefore, we need to cancel also the non-linear terms and therefore use the control law proposed in equation (4.9).

Figure 4.5a and 4.5b



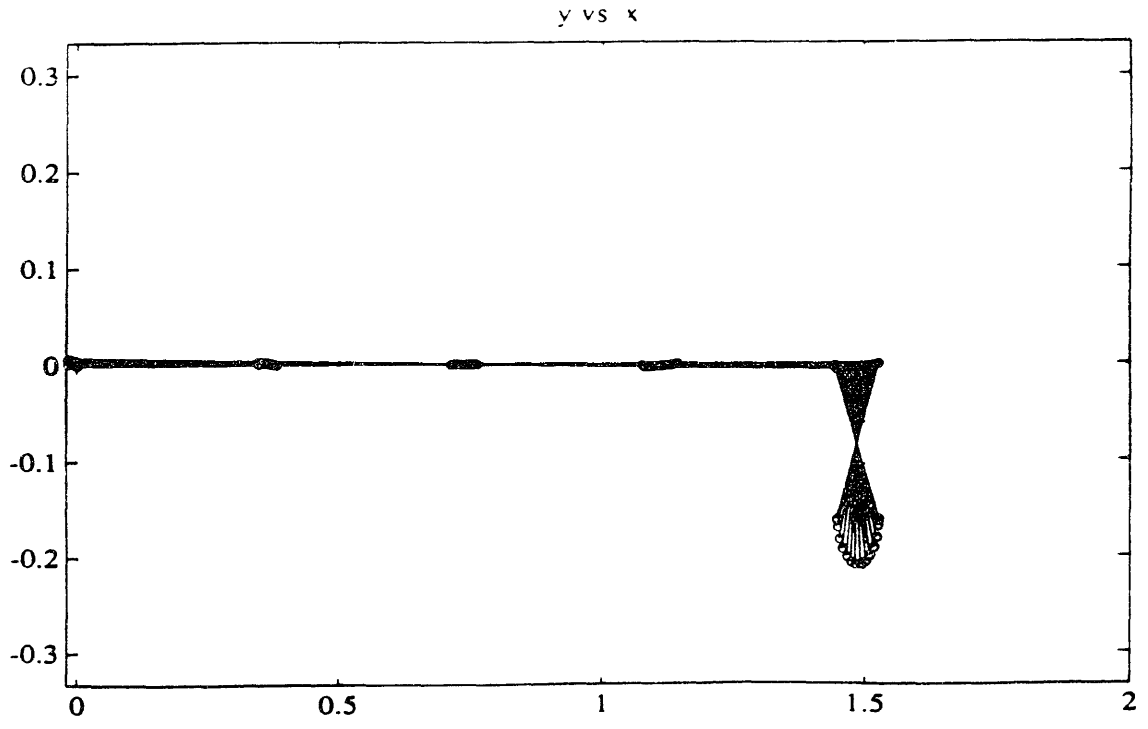
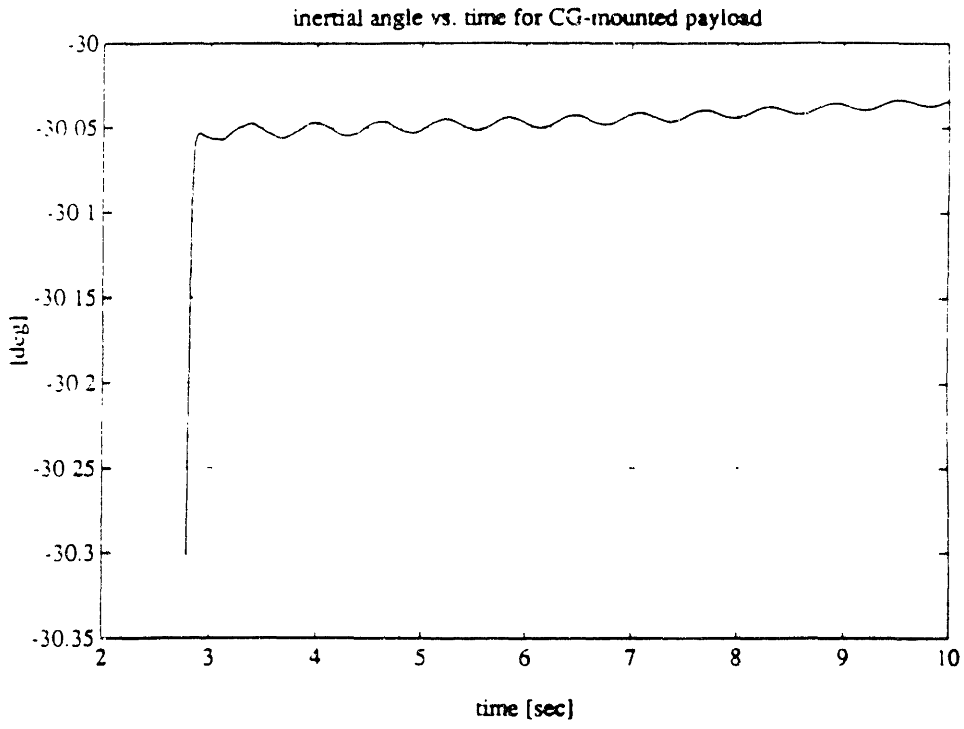


Figure 4.6a and 4.6b



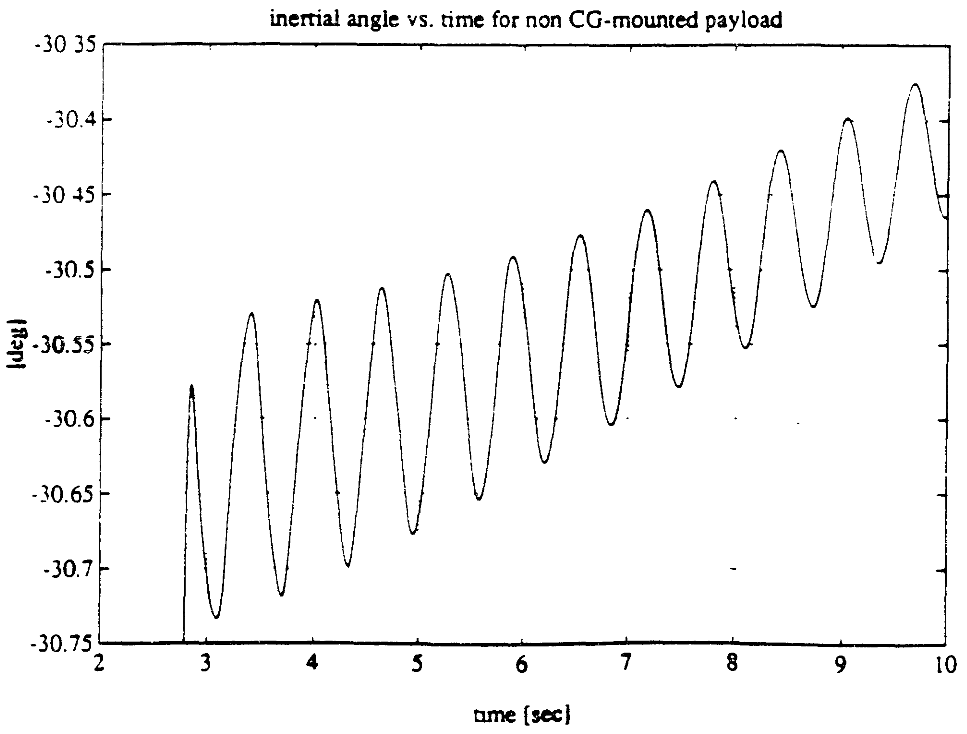
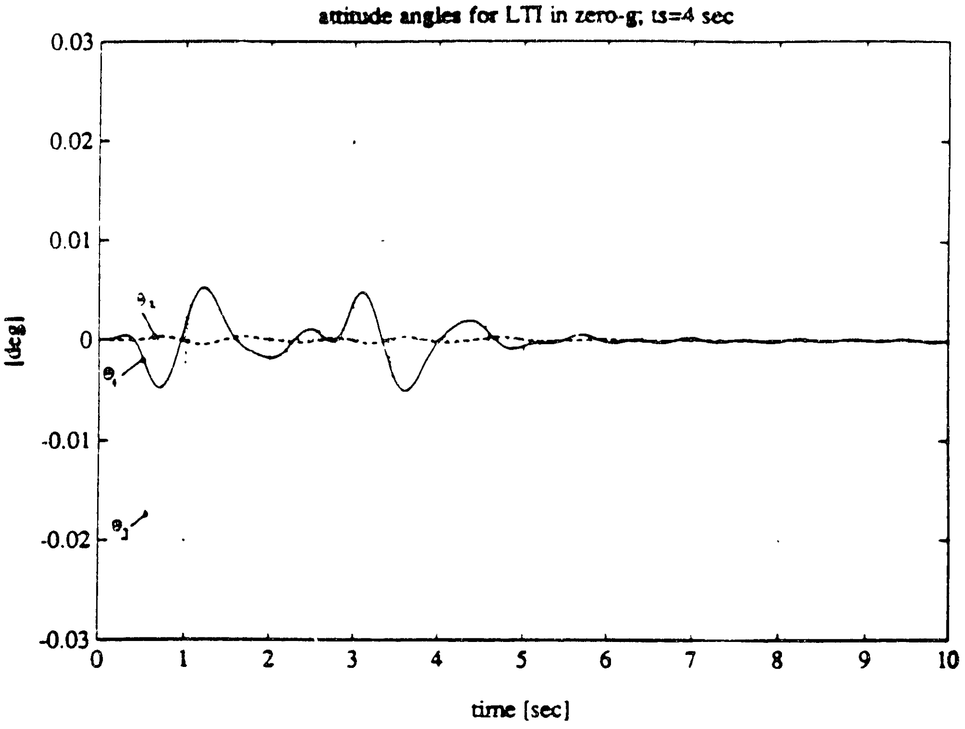


Figure 4.7 and 4.8



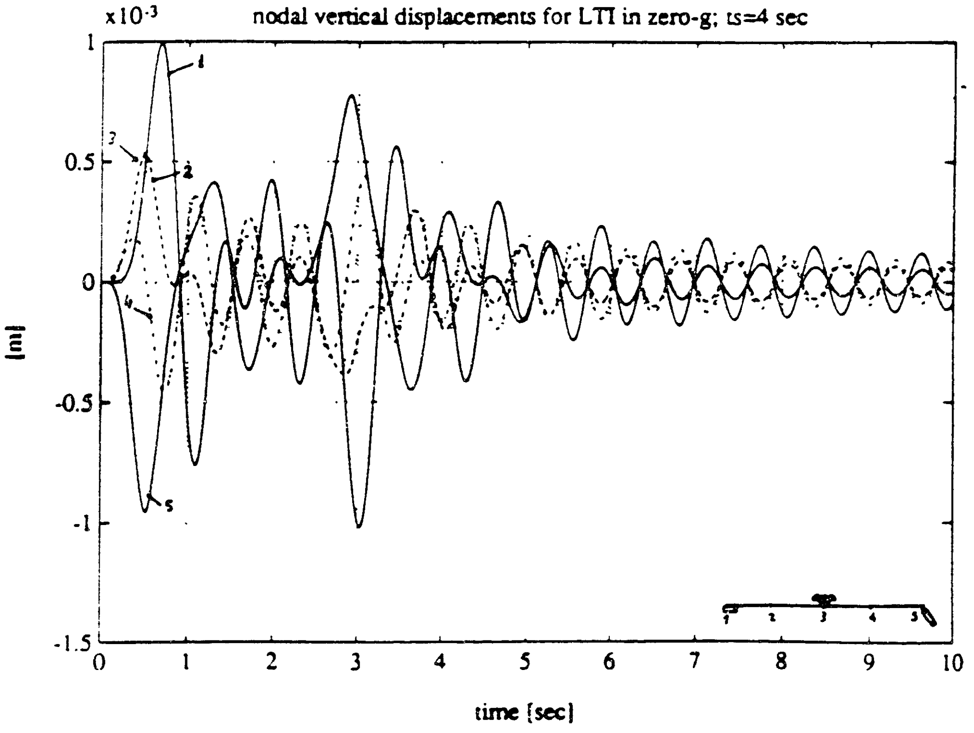
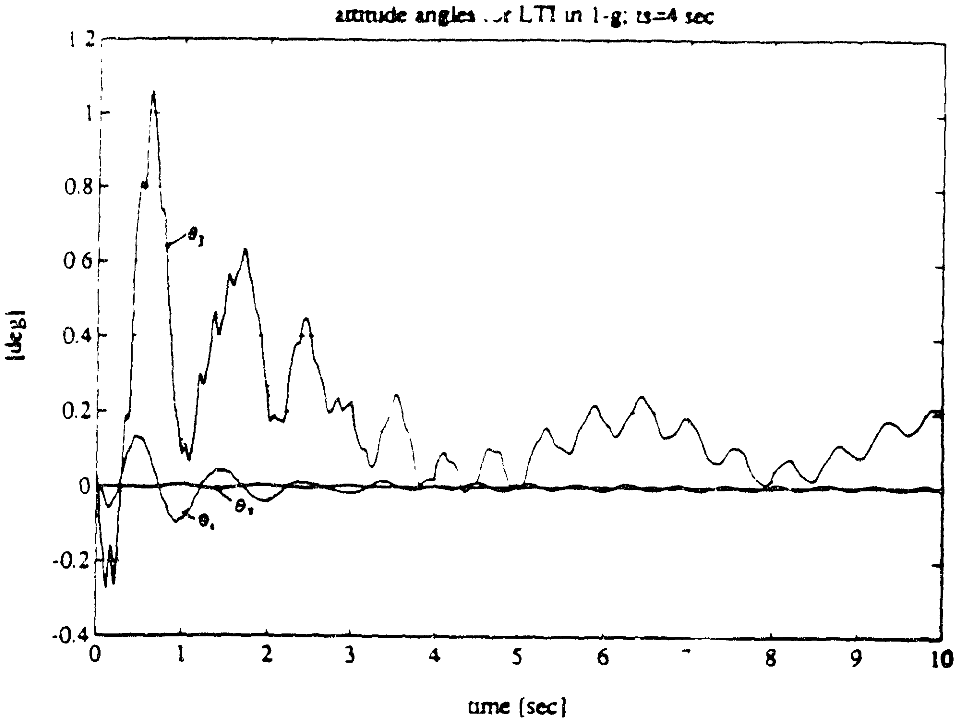


Figure 4.9 and 4.10



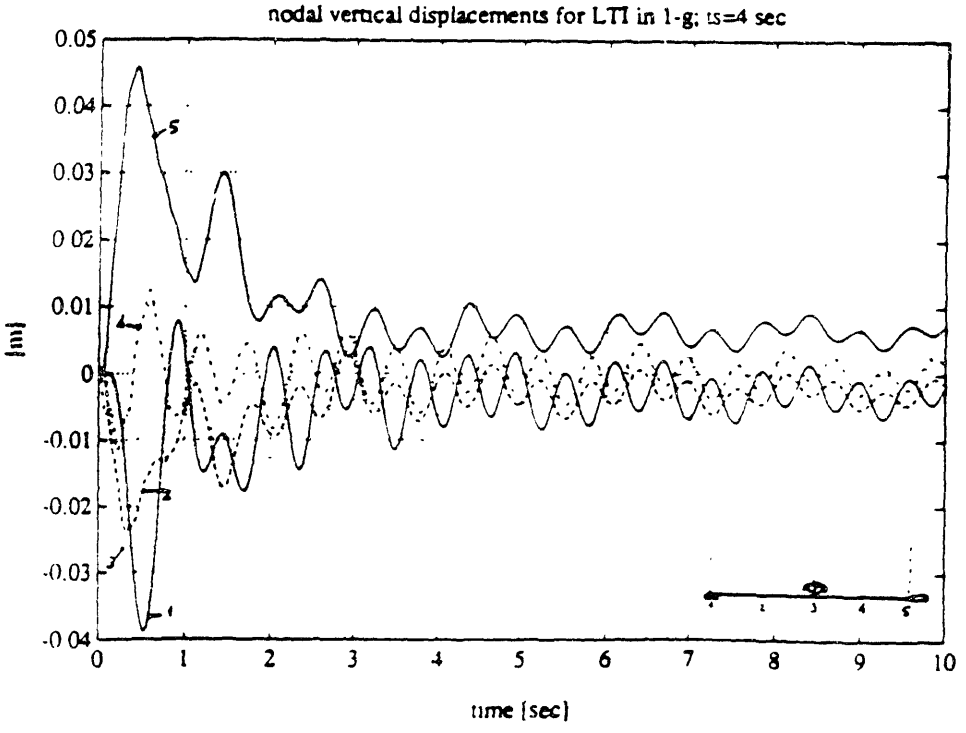


Figure 4.11

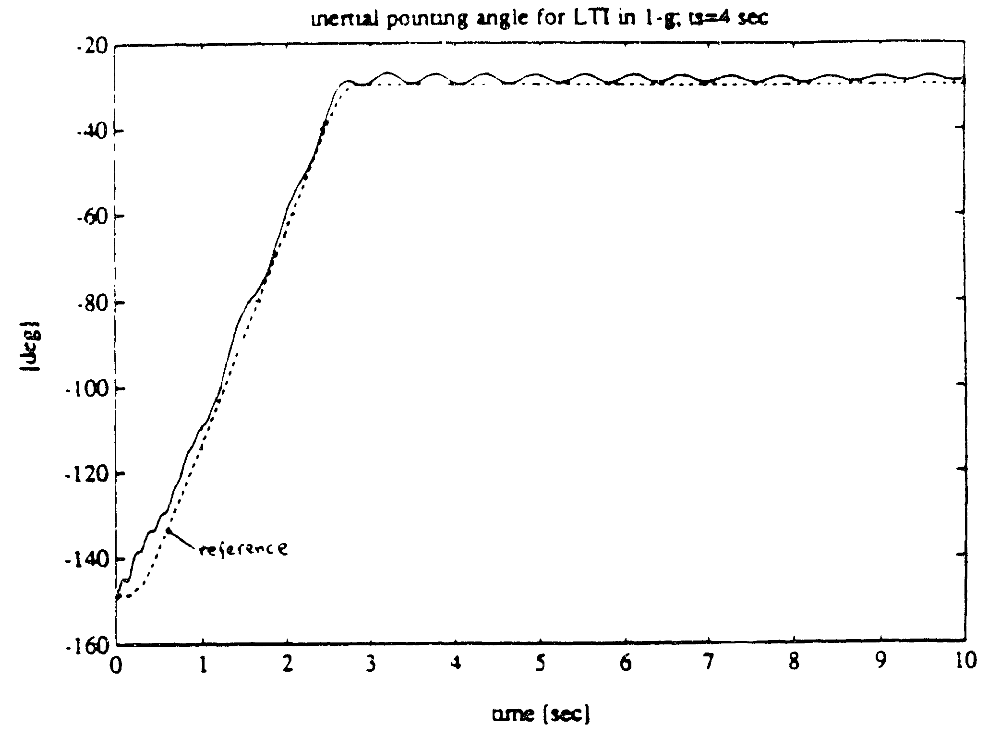
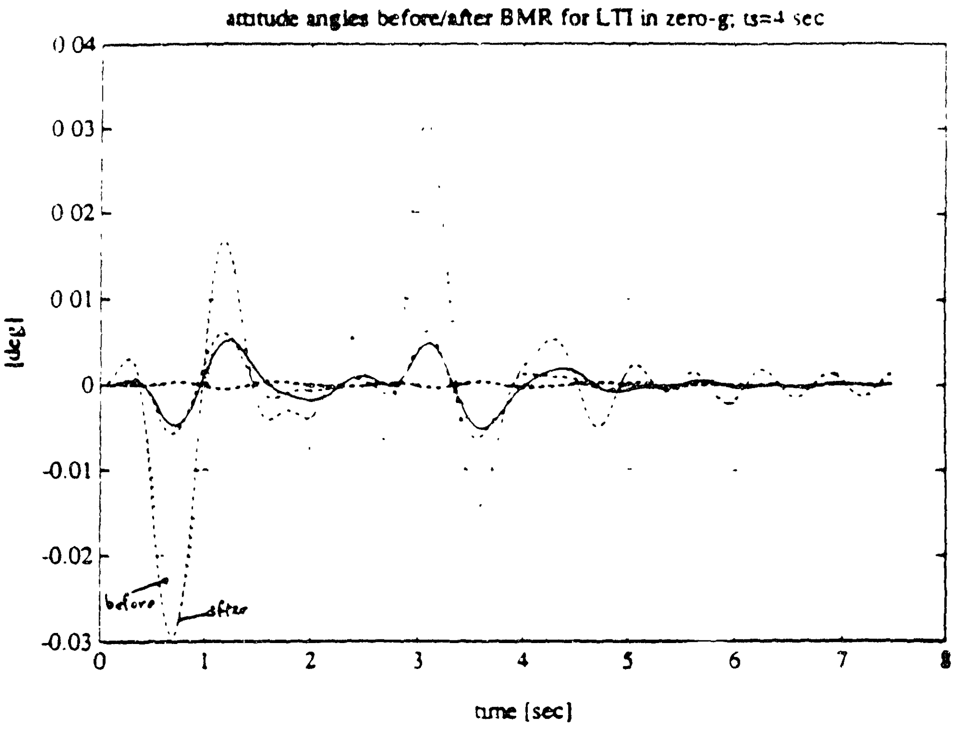

inertial pointing angle for LTI in 1-g; ts=4 sec
-20
-40
-60
-80
-100
-120
-140
-160
(deg)
0
1
2
3
4
5
6
7
8
9
10
time (sec)
reference

Figure 4.12 and 4.13



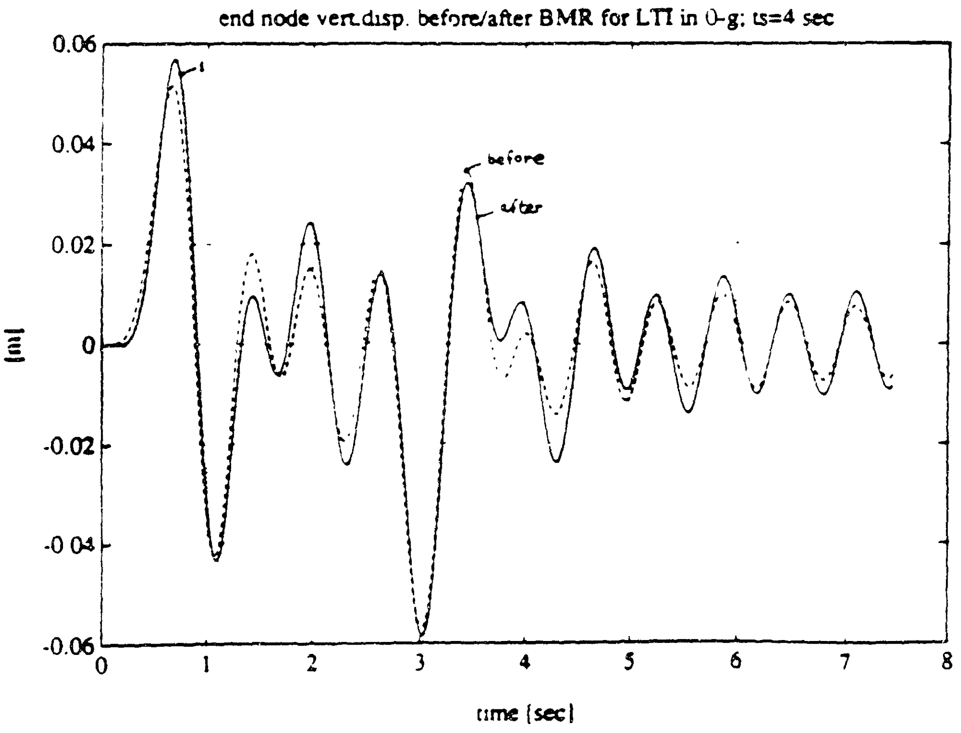
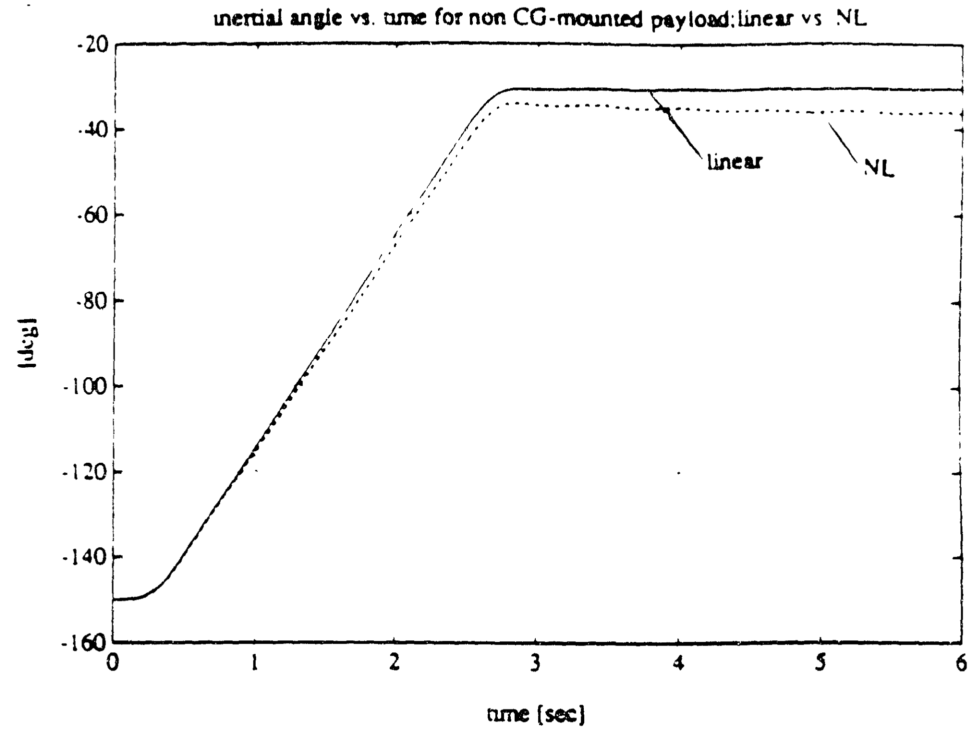


Figure 4.14 and 4.15



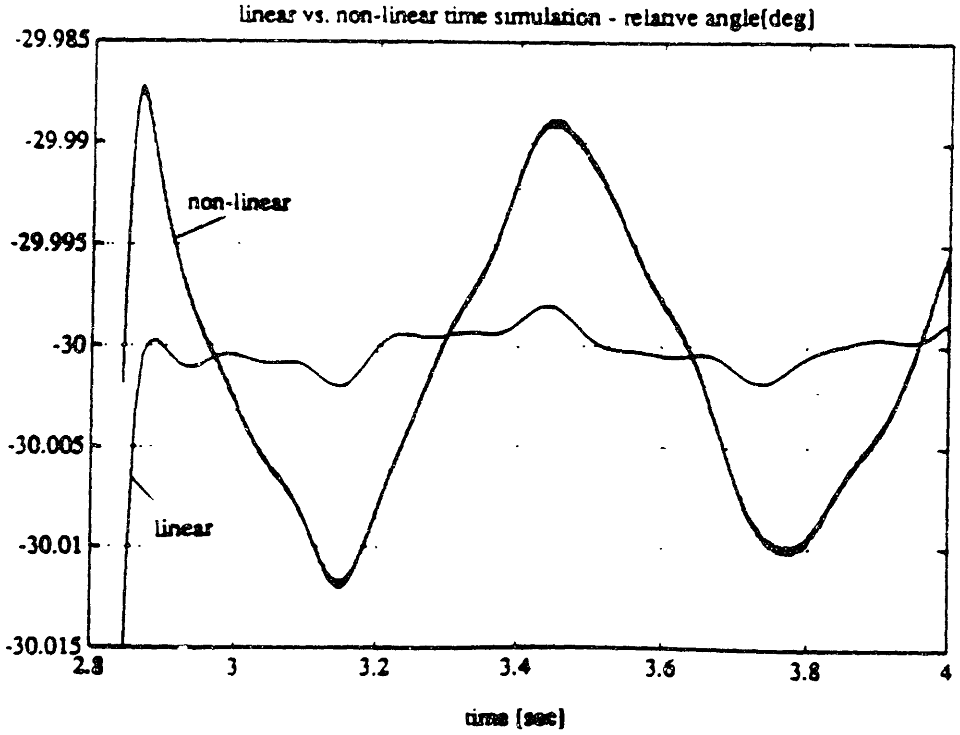


Figure 4.16

<u>FLOW CHART OF PROGRAM FOR MULTI-BODY SIMULATION</u>

0) compute the new full state vector at step k as:

$$x_k = \Phi_o \eta_{k-1}$$

where Φ_o is the truncated set of pre-maneuver modal shapes;

1) find q_{des}, $\dot{q}_{des}$ and $\ddot{q}_{des}$ for payload trajectory;

2) compute inertia terms dependent on q_{ok};

3) assemble inertia matrix at step k M_k ;

4) assemble finite elements at step k ;

5) compute gyroscopic matrix G_k at step k ;

6) load damping matrix D_o of the pre-maneuver state;

7) compute vector of non-linear terms F_{NLk} at step k ;

8) compute control vector at step k as:

$$F_{Ck} = f\,(q_{ok}, \; \dot{q}_{ok}, \; \Omega_k, \; \dot{\Omega}_k);$$

9)write equations of motion in modal form at step k , using the truncated set of pre-maneuver system modes Φ_o :

$$\eta_k = \Phi_o^{-1} x_k \qquad \dot{\eta}_k = \Phi_o^{-1} \dot{x}_k$$

$$M_{red} = \Phi_o^T M_k \Phi_o \qquad K_{red} = \Phi_o^T K_k \Phi_o \qquad GD_{red} = \Phi_o^T (G_k + D_o) \Phi_o$$

$$F_{red} = \Phi_o^T [\, b_c \; F_{Ck} - F_{NLk} \,]$$

$$\ddot{\eta}_k = M_{red}^{-1} [\, - K_{red} \, \eta_k - GD_{red} \, \dot{\eta}_k + F_{red} \,]$$

10) send to the integrator $\dot{\eta}_k$ and $\ddot{\eta}_k$;

CHAPTER 5: CONCLUSIONS

Two different analytic models of a multi-body spacecraft testbed were obtained. One model is the zero-g model, which is representative of the test article free-floating in the Middeck of the Space Shuttle, except that the umbilicals are not modelled. This model is linear in nodal displacements and rotations but non-linear in toque wheel angular speeds and articulated bodies relative gimbal angles and rates. The finite element method was used to assemble the equations of motion of the flexible bus, and Kane's method of generalized speeds was used to assemble the equations of motion of the torque-wheel assembly and of the articulated bodies. For this zero-g model, two different analyses were conducted on the model linearized about two reference payload configurations, the "all-in-plane" and the "all-out-of-plane" configurations with hinges free and locked. The first analysis has allowed us to determine an empirical estimate of the precession frequency of the spacecraft by taking into account the effect of the flexibility in the precessing behavior of the test article. The second, has allowed us to quantify the effect of the base-body flexibility upon the pointing dynamics of the articulated payloads. Even if the payloads are rather massive and non-CG mounted, the residual effect of the flexibility is so small that, for all practical purposes, the transfer function from gimbal torque to inertial pointing angle, both in the in-plane and the out-of-plane directions, is very close to a double integrator.

The other model is the one-g model, which is instead representative of the dynamics of the test article when suspended in the laboratory from a state-of-the-art suspension device. The suspension own internal dynamics were included into the non-linear model, thereby aumenting the states of the original model. It consists of a displacement and mass cancellation loop which attempts to mimick the zero-g behavior of the whole suspended system. The stiffening effect of the one-g gravity field on the suspension cables and on the flexible multibody test article was also included in the model. Finally, a comparison was made of the transfer functions obtained with this method on the one-g model with the same ones obtained experimentally on the laboratory and the agreement was found to be rather good. Therefore, a low order model carefully obtained by a combination of Kane's method and of the finite element method can be used with profit as an evaluation model for control design.

The linear time invariant model and the fully non-linear models derived here were implemented on a Sun workstation using the modularity offered by the Matlab software. A feedback/feedforward approach was used in designing a control law for the gimbal motor torque which enables the payload to undergo a full 120 degrees slew maneuver in the vertical plane. The technique of "ruthless linearization" was followed in simplified the dynamics of the articulated bodies for purposes of analyzing the tracking error dynamics. The method of feedback linearization was proposed in order to cancel non-linear Coriolis, centripetal and gravity terms arising during the motion so that trajectory tracking is achieved. A similar approach was followed for attitude regulation, in which we want to stabilize the attitude of the spacecraft in inertial space. In this case, the idea was proposed of introducing an additional term in the torque wheel motor control law which attempts to cancel the gyroscopic coupling terms from the equations of motion. The results from the linear simulation show that bus attitude stabilization and trajectory tracking are successfully achieved. Furthermore, a comparison of the linear and the non-linear time simulations shows that the magnitude of the Coriolis and centripetal terms are unimportant for the maneuver under investigation, both in zero-g and in one-g, but that in the one-g case, it is required to cancel out also the gravity terms since the payload is required not to deviate from the end position. However, since the order of the model was not reduced, the non-linear simulation time turned out to be prohibitively large, and this makes the analysis and design of the payload trajectories a cumbersome task. For this reason, the balanced modal reduction algorithm was applied to the zero-g model with the conclusion that a model of the flexibility of much lower order can be used without altering substantially the input/output properties of the system. This reduced order model has been incorporated in the non-linear multibody code with the result that almost a 90% savings in computation time has been obtained, thereby allowing the analysis in the time domain of the non-linear behavior of other types of slew maneuvers (for example, combined pointing/tracking of two independent payloads). However, the results of the non-linear simulation show that the behavior of the reduced order non-linear system is slightly different than the expected one because the system mode shapes are not being updated during the simulation. This suggests that further research is required to incorporate the updated reduced order modal characteristics of the system during the non-linear simulation.

REFERENCES

[1] Graves, P.C., Joshi, S.M.:"Modelling and Control of Flexible Space Platforms with Articulated Payloads", Proceedings of the 3rd Annual NASA/DoD CSI Conference, San Diego, Ca., 1989, pp. 181-210.

[2] Experiment Requirements Document, MACE-1-101, March 10, 1991.

[3] Miller,D.W.,de Luis,J.,Crawley,E.:"Dynamics and Control of Multipayload Platforms: the Middeck Active Control Experiment (MACE)", paper no.IAF-90-292, 41st Congress of the International Astronautical Federation, October 6-12, 1990, Dresden,GDR.

[4] Saarmaa,Erik:"MACE DM Dynamic Testing & Modelling", version 3.1, October 17,1991, MIT SERC.

[5] Hughes,P.C.:"*Spacecraft Attitude Dynamics*", Wiley, 1986.

[6] Kane, T.R, Levinson,D.A.: *"Dynamics: Theory and Applications"*, McGraw-Hill Book Company,1983.

[7] von Flotow,A.,Lecture Notes, Dept. of Aeronautics and Astronautics, MIT.

[8] Hughes,P.C.,Sharpe,H.N.:"Influence of Stored Angular Momentum on the Modal Characteristics of Spacecraft with Flexible Appendages", *Journal of Applied Mechanics*, Dec. 1975, pp.785-788.

[9] Hughes,P.C.:"Dynamics of Flexible Space Vehicles with Active Attitude Control", *Celestial Mechanics*, vol.9, 1974, pp.21-39.

[10] Hughes,P.C.:" Modal Identities for Elastic Bodies, with Application to Vehicle Dynamics and Control", *Journal of Applied Mechanics*, March 1980, vol.47, pp.177-184.

[11] D'Eleuterio,G.M.T., Hughes,P.C.,:"Dynamics of Gyroelastic Continua", ASME *Journal of Applied Mechanics*, vol. 51, June.1984,pp.415-422.

[12] Meirovitch,L..:"*Analytical Methods in Vibrations*",MacMillan Publishing Co., 1967.

[13] Mercadal,M.:"Analysis of the MACE First Sample Problem", Report 7-91-R, June 1991, MIT SERC.

[14] Padilla,C.E.,von Flotow,A.H.:"Further Approximations in Flexible Multibody Dynamics", Proceedings of the AIAA/ASME/ASCE/AHS/ASC 32nd Structures, Structural Dynamics and Materials Conference, April 1991.

[15] Padilla,C.E.:"Non-linear Strain-Displacement Relations in the Dynamics of a Two-link Flexible Manipulator", M.S.Thesis, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, May 1989.

[16] von Flotow,A.:"Some Approximations for the Dynamics of Spacecraft Tethers",reprinted form the *Journal of Guidance, Control and Dynamics*, vol.11,no.4, July-Aug.1988,pp.357-364.

[17] Hagedorn, P.:"The Eigenvalue Problem for a Certain Class of Discrete Linear Systems: A Perturbation Approach", Proceedings of the 4th VPI&SU/AIAA Symposium, Blacksburg, VA, June 6-8, 1983.

[18] Skelton,R.E., Hughes, P.C.:"Modal Cost Analysis for Linear Matrix-Second-Order Systems", *Journal od Dynamic Systems, Measurement, and Control*, Sept. 1980, vol.102,,pp.151-158.

[19] Hughes, P.C., Skelton, R.E.:"Controllability and Observability of Linear Matrix-Second-Order Syetems", *Journal of Applied Mechanics*, June 1980, vol.47, pp. 415-420.

[20] Spanos, J.T.,:"Control-Structure Interaction in Precision Pointing Servo Loops", *Journal of Guidance, Control and Dynamics*, vol.12, no.2, March-April 1989,pp.256-263.

[21] Garcia, J.G., Sievers, L.A., von Flotow, A.:"Broadband Positioning Control of Small Payloads Mounted on a Flexible Structure", to be published.

[22] Przemieniecki,J.S.:"*Theory of Matrix Structural Analysis*",Dover Publications,New York,1968.

[23] Bernard, D.E. :"Projection and Assembly Method for Multibody Component Model Reduction", *Journal of Guidance, Control and Dynamics*, vol.13, no.5, Sept-Oct. 1990, pp.905-912.

[24] Junkins,J.L.,Rahman,Z.H.,Bang,H.:"Near-Minimum-Time Control of Distributed Parameter Systems: Analytical and Experimental Results", *Journal of Guidance, Control and Dynamics*, vol.14, no.2, March-April 1991, pp.406-415.

[25] Meirovitch, L., Kwak,M.K.:"On the Maneuvering and Control of Space Structures", Invited Paper, *Dynamics of Flexible Structures in Space*, Edited by C.L.Kirk and J.L.Junkins, Computational Mechanics Publications & Springer Verlag, 1990.

[26] Junkins,J.L.,Turner,J.T.:"*Optimal Spacecraft Rotational Maneuvers*", Elsevier,Amsterdam, The Netherlands, 1986.

[27] Craig,J.J.,"*Introduction to Robotics: Mechanics and Control*", Addison-Wesley Publishing Co.1986.

[28] Slotine,J.J.,Li,W.:"*Applied Nonlinear Control*",Prentice Hall, 1991.

[29] Li,D.:"Non-Linear Slew-Maneuvering Control of Multi-Flexible-Body Systems", PhD Thesis, Dept. of Mechanical Engineering, Massachusetts Institute of Technology, June,1991.

[30] Padilla,C.E.:"Non-linear Modelling, Simulation and Preliminary Control of the Baseline MACE Test Article", Final Report, MIT Space Engineering Research Center, August 28,1990.

[31] Wie,B.,Weiss,H., Arapostathis,A.:"Eigenaxis Rotational Maneuver via Quaternion Feedback", to be published.

[32] Spanos,T.J., Tsuha, W.S.:"Selection of Component Modes for the Simulation of Flexible Multibody Spacecraft", paper AAS 89-438, presented at the AAS/AIAA Astrodynamics Specialist Conference, Stowe, VT, August 7-10, 1989.

[33] Blelloch,P.A.,Mingori,D.L.,Wei,J.D.:"Perturbation Analysis of Internal Balancing for Lightly Damped Mechanical Systems with Gyroscopic and Circulatory Forces", *Journal of Guidance, Control and Dynamics*, vol.10, no.4, July-August 1987, pp.406-410.

[34] Eke,O.F., Man,G.K:"Model Reduction in the Simulation of Interconnected Flexible Bodies", paper AAS 87-455.presented at the AAS/AIAA Astrodynamics Specialist Conference, Kalispell, MN, Aug.10-13, 1987.

[35] Jones,R.E.:"Multi-Flex Body Dynamics for Control Design", Proceedings of the Workshop on Multibody Simulation, G.Man, R.T.Laskin, editors, NASA/JPL D-5190, vol.1, pp.354-382,April 15, 1988.

APPENDIX

A1. INERTIA MATRICES OF THE BODIES OF THE ARTICULATED CHAIN.

The inertia matrix is given by:

$$m(i,j) = \sum_{k=0}^{Nbodies} \left[m_k \left({}^N v_i^k \cdot {}^N v_j^k \right) + {}^N \omega_r^k \cdot \underline{\underline{J}}_k \cdot {}^N \omega_j^k \right] =$$

$$\sum_{k=0}^{Nbodies} [M_{trans} + M_{rot\cdot}]_k$$

(A1.1)

<u>Body 1</u>:

Denoting with $\mathbf{r} = (r1 \;\; r2 \;\; r3)^T = {}^F r^{F*}$ and with $\underline{\underline{J}}$ its inertia dyadic, we have:

$$M_{trans} = M1 \begin{bmatrix} 1 & 0 & 0 & 0 & r3 & r2 \\ 0 & 1 & 0 & -r3 & 0 & r1 \\ 0 & 0 & 1 & r2 & -r1 & 0 \\ 0 & -r3 & r2 & r2^2 + r3^2 & -r1r2 & -r1r3 \\ r3 & 0 & -r1 & -r1r2 & r1^2 + r3^2 & -r2r3 \\ -r2 & r1 & 0 & -r1r3 & -r2r3 & r1^2 + r2^2 \end{bmatrix}$$

$$M_{rot\cdot} = \begin{bmatrix} 0 & 0 \\ 0 & \underline{\underline{J}} \end{bmatrix}$$

(A1.2)

Body 2:
Using the symbols of chapter 2 and denoting with $\underline{\underline{J}}$ its inertia dyadic, we have:

$$Mtrans = M2 \begin{bmatrix} 1 & 0 & 0 & 0 & A2 & -B2 & 0 \\ 0 & 1 & 0 & -A2 & 0 & C2 & -C1 \\ 0 & 0 & 1 & B2 & -C2 & 0 & B1 \\ 0 & -A2 & B2 & A2^2+B2^2 & -B2C2 & -A2C2 & A2C1+B1B2 \\ A2 & 0 & -C2 & -B2C2 & A2^2+C2^2 & -A2B2 & -B1C2 \\ -B2 & C2 & 0 & -A2C2 & -A2B2 & C2^2+B2^2 & -C1C2 \\ 0 & -C1 & B1 & A2C1+B1B2 & -B1C2 & -C1C2 & C1^2+B1^2 \end{bmatrix}$$

(A1.3)

$$Mrot = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & J11 & J12 & J13 & J11 \\ 0 & 0 & 0 & J21 & J22 & J23 & J21 \\ 0 & 0 & 0 & J31 & J32 & J33 & J31 \\ 0 & 0 & 0 & J11 & J21 & J31 & J11 \end{bmatrix}$$

Body 3:
Using the symbols of chapter 2 and denoting with $\underline{\underline{J}}$ its inertia dyadic, we have:

$$Mtrans = M3 \begin{bmatrix} 1 & 0 & 0 & 0 & A5 & -B5 & 0 & -C5 \\ 0 & 1 & 0 & -A5 & 0 & D5 & -E5 & G5 \\ 0 & 0 & 1 & B5 & -D5 & 0 & F5 & H5 \\ 0 & -A5 & B5 & A5^2+B5^2 & -B5D5 & -A5D5 & A5E5+B5F5 & -G5A5+H5B5 \\ A5 & 0 & -D5 & -B5D5 & A5^2+D5^2 & -A5B5 & -D5F5 & -A5D5-D5H5 \\ -B5 & D5 & 0 & -A5D5 & -A5B5 & D5^2+B5^2 & -D5E5 & B5C5+D5G5 \\ 0 & -E5 & F5 & A5E5+B5F5 & -D5F5 & -D5E5 & E5^2+F5^2 & -E5G5+F5H5 \\ -C5 & G5 & H5 & -G5A5+H5B5 & -A5C5+D5H5 & B5C5+D5G5 & -E5C5+F5H5 & C5^2+G5^2+H5^2 \end{bmatrix}$$

(A1.4)

$$
M_{rot} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & J11 & J12 & J13 & J11 & J13c\theta+J12s\theta \\
0 & 0 & 0 & J21 & J22 & J23 & J21 & J23c\theta+J22s\theta \\
0 & 0 & 0 & J31 & J32 & J33 & J31 & J33c\theta+J32s\theta \\
0 & 0 & 0 & J11 & J21 & J31 & J11 & J13c\theta+J12s\theta \\
0 & 0 & 0 & J13c\theta+J12s\theta & J23c\theta+J22s\theta & J33c\theta+J32s\theta & J13c\theta+J12s\theta & [illegible]
\end{bmatrix}
$$

A2. NON-LINEAR TERMS IN THE EQUATIONS OF MOTION OF THE ARTICULATED BODIES.

The non-linear terms in the equations of motion of the articulated part can be thought of as being composed of translational and rotational terms. In particular, using the notation of chapter 2, one obtains:

$$
(F_r)_{trans} = M1 \left[{}^{N}v_r^{F*} \left({}^{N}\omega^{F} \times \left({}^{N}\omega^{F} \times {}^{F}r^{F*} \right) \right) \right] +
$$

$$
M2\, {}^{N}v_r^{A*} \left[\left({}^{N}\omega^{F} \times \left({}^{N}\omega^{F} \times {}^{F}r^{A} \right) \right) + \left({}^{N}\omega^{A} \times \left({}^{N}\omega^{A} \times {}^{A}r^{A*} \right) \right) \right] +
$$

$$
M3\, {}^{N}v_r^{B*} \left[\left({}^{N}\omega^{F} \times \left({}^{N}\omega^{F} \times {}^{F}r^{A} \right) \right) + \left({}^{N}\omega^{A} \times \left({}^{N}\omega^{A} \times {}^{A}r^{P} \right) \right) + \left({}^{N}\omega^{P} \times \left({}^{N}\omega^{P} \times {}^{P}r^{B*} \right) \right) \right]
$$

(A1.5)

$$
(F_r)_{rot} = {}^{N}\omega_r^{F} \left[{}^{N}\omega^{F} \times \left(J1\, {}^{N}\omega^{F} \right) \right] +
$$

$$
{}^{N}\omega_r^{A} \left[{}^{N}\omega^{A} \times \left(J2\, {}^{N}\omega^{A} \right) \right] +
$$

$$ {}^{N}\omega_{r}^{P} \left[{}^{N}\omega^{P} \times \left(J3\ {}^{N}\omega^{P} \right) \right] $$

Note that they include not only non-linear terms in angles and rates of the articulation, but also the non-linear terms in the finite element elastic displacements and rotations. These terms are, therefore, fully non-linear in all the generalized coordinates. The extended expression of these terms will not reported here for the sake of brevity, as their transcription is very cumbersome. They, however, can be found in the listing of the files Fnonlin.m and fz.m. The first file computes these vector of non-linear terms by building up each part, as it is usually done with the Kane's method. The second, contains all the terms.

A3. NON-LINEAR TERMS IN THE EQUATIONS OF MOTION OF THE GYROSTAT.

The vector (9x1) of non-linear terms in equation (2.22), of chapter 2, is is made of zero entries except at the degrees of freedom corresponding to the rotational coordinates of the fixed body. The quantity $Q_i = [(J_a - J_t)\ (u_4\, p + u_5\, q + u_6\, r)]_i$ and, denoting with J the components of the inertia matrix of the whole body (including the wheels), one obtains:

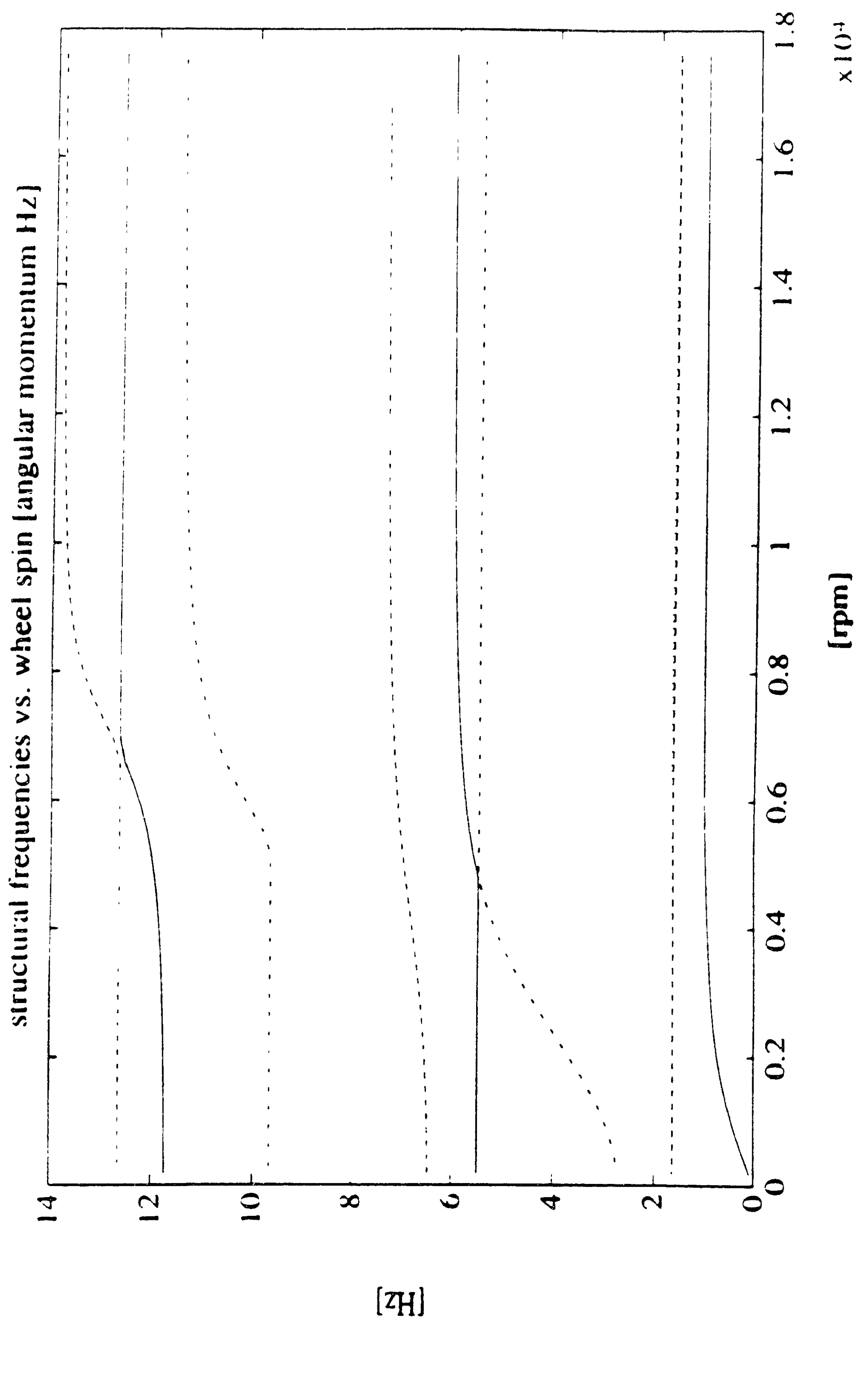
$$F_4 = J_{13}\, u_4 u_5 - J_{12}\, u_4 u_6 + J_{23}(u_5^2 - u_6^2) + u_5 u_6 (J_{33} - J_{22}) + \sum_{i=0}^{N} Q_i\, [\, u_5(gl - mf) + u_6(hl - nf)]_i$$

$$F_5 = J_{12}\, u_5 u_6 - J_{23}\, u_4 u_5 + J_{13}(u_6^2 - u_4^2) + u_4 u_6 (J_{11} - J_{33}) + \sum_{i=0}^{N} Q_i\, [\, u_4(fm - lg) + u_6(mh - ng)]_i$$

$$F_6 = J_{23}\, u_4 u_6 - J_{13}\, u_5 u_6 + J_{12}(u_4^2 - u_5^2) + u_5 u_4 (J_{22} - J_{11}) + \sum_{i=0}^{N} Q_i\, [\, u_5(gn - mh) + u_4(fn - lh)]_i$$

(A1.6)

APPENDIX A.4 : BLOW-UP OF FIGURE [2.9]



APPENDIX A.5 : MATLAB SIMULATION CODES

```
%-----------------------------------------------------------------
%                MACE 3-D model in zero-g
%
% Author        : Marco B. Quadrelli
% Date          : October 17, 1991
% Last revision: December 12, 1991
%-----------------------------------------------------------------
% Notes : This version considers the
%         (support + inner stage + outer stage + payload)
%         as a chain of three bodies connected by two 1dof
%         revolute joints.
%         The three bodies are:
%         - joint + support
%         - gimbal 1
%         - gimbal 2 + payload.
%         The program works only when
%         nelew=1 and nelef=-1(no flexible appendage yet), but
%         nele can be chosen to be any number.
%         The central node is allowed to store a finite amount of
%         internal angular momentum.
%         All units in S.I.
%-----------------------------------------------------------------
%
%          torque  -> /   \                  0  -> hinged payload
%         wheels     /  \ / \               /
%      1=======2=======3=======4=======5/
%     /                               \
%   /                                 \--> spacecraft bus
%  0
%=================================================================
% the state vector is:
%__________________________________________________________________
%                description                +    d.o.f.
%___________________________________________+______________________
%  X = [  (x,y,z,theta1,theta2,theta3)_1    |          1:ndof-6
%         (x,y,z,theta1,theta2,theta3)_2    |     ndof-5:ndof
%         (x,y,z,theta1,theta2,theta3)_3    |     ndof+1:2*ndof-6
%         (x,y,z,theta1,theta2,theta3)_4    |   2*ndof-5:3*ndof-12
%         (x,y,z,theta1,theta2,theta3)_5    |  3*ndof-11:ntotflex
%         (phi,theta)_#1                    |    ntot-3:ntot-2
%         (phi,theta)_#2                    |    ntot-1:ntot
%         (omega1,omega2,omega3)_wheels     |    ntot+1:ntot+3
%         ----------------------------------|-----------------------
%         and all the first derivatives     |    ntot+19:2*(ntot+18)]
%         ----------------------------------|-----------------------
%=================================================================

%              ENTER INPUT DATA

%-----------------------------------------------------------------
% Define parameters and size of arrays
%-----------------------------------------------------------------
tupi=2*pi;rad=pi/180;invrad=1/rad;invrpm=(2*pi)/60;rpmm=1/invrpm;
grav=9.8065;wbig=(logspace(-1,3,400));
rdum = [0;0;0];
%=================================================================
%nele = input(' number of elements per flexible Lexan beam [1 or 2]: ');
nele=1;
%nelef = input(' number of elements for flexible appendage [1]: ');
nelef=-1;

%pl_locked=1;
pl_locked=input('payload #1 free [0],locked (no moving mass) [1]:   ');


npay1=2;          % # of rigid body dof from payload 1
```

```
npay2=2;          % # of rigid body dof from payload 2
nwheels=3;        % # of rigid body dof from torque wheels

nnod= nele+1;     % # of nodes per Lexan beam
ndof= 6*nnod;     % # of dof allowed per Lexan beam
ntot= 24*nele+10;% # of dof allowed on bus + 2x2 payloads rotations
ntotflex= ntot-4;% # of flexible dof allowed on bus only
totaldof= ntot+3;% total # of on bus including 3 dof of torque wheels
nodi=ntotflex/6; % total # of nodes in spacecraft bus

nelef=-1;
nnodf= nelef+1;     % # of nodes per flexible appendage beam
ndoff= 6*nnodf;     % # of dof allowed per flexible appendage beam
nodif=ndoff/6;      % total # of nodes in flexible appendage beam
numnp=nodi+nodif;   % # of nodal points of all structure
nflex=numnp*6;

% rigidbody = total # of rigid body dof allowed by the structure:
%        bus + 2*payloads + 3*wheels          in free space
% nfree = total # of dof allowed by the structure:
%        bus + 2*payloads + 3*wheels
% in free space (before imposing constraints)
   nfree=nflex + npay1 + npay2 + nwheels;
   rigidbody=6+npay1+npay2+nwheels;

%===============================================================
% 1st rotation at node 1: angle phi is Cw about a1 -- out-of-plane XY
% starts from the Z axis of the bus
% 2nd rotation at node 1: angle theta is Cw about p3-- in-plane XY
% starts from the X axis of the bus
% 1st rotation at node 5: angle phi is CCw about a1 -- out-of-plane XY
% starts from the Z axis of the bus
% 2nd rotation at node 5: angle theta is CCw about p3-- in-plane XY
% starts from the X axis of the bus
%===============================================================
  phi1  =...
input('out-of-plane angle phi1[from XY plane towards +Z] (deg): ')*rad;
  theta1=...
input('in-plane angle theta1[from XZ plane towards +Y] (deg): ')*rad;
  phi5  =..
input('out-of-plane angle phi5[from XY plane towards +Z] (deg): ')*rad;
  theta5=...
input('in-plane angle theta5[from XZ plane towards +Y] (deg): ')*rad;
f1=phi1;t1=theta1;fi5=phi5;t5=theta5;
%===============================================================
%solveig=input('solve eigenproblem ?, yes [1], no [0]:    ');
solveig=1;

%choice2=...
%menu('enter',...
%'components of angular momentum in MACE body axes','wheels speeds');
choice2=1;
if choice2==1,
  Hcomp=[0;1;0];
  %Hcomp=...
  %input(...
  %'enter components of angular momentum in body axes [N.m.sec]:   ');
elseif choice2==2,
  spin1= input(' steady angular velocity of wheel 1 [rpm]:  ');
  spin2= input(' steady angular velocity of wheel 2 [rpm]:  ');
  spin3= input(' steady angular velocity of wheel 3 [rpm]:  ');
  %in rad/s
  spin1=invrpm*spin1;spin2=invrpm*spin2;spin3=invrpm*spin3;
end
spinMACE1=0;spinMACE2=0;spinMACE3=0;
spinMACE1=invrpm*spinMACE1;spinMACE2=invrpm*spinMACE2;
```

```
spinMACE3=invrpm*spinMACE3;
%precfre=input('need to estimate precession freq.?: yes[1] no[0]:   ');
precfre=1;

smorza=2;
damping=1;
%damping=menu('choose type of damping','modal','prop to K',...
%              'prop to M','old type','decoupling');
%CG=menu('is each payload','CG mounted ?','non CG mounted ?');
CG=2;
%gimbal1=menu('new coordinates of reference point of gimbal 1?',...
%              'yes,change','no,use default');
gimbal1=2;
%center=menu('gimbals rotation axes are centered',...
%             'no','yes');
center=2;
%pollo=menu('pivots lie on bus centerline','yes','no');
pollo=2;
%=================================================================
statespace=1;
%statespace=input('state space model, yes [enter 1], no [enter 0]:   ');
if statespace==1,
 bodeplots=menu('do you want Bode plots?','open-loop','closed-loop(N.A.)',...
               'no Bode plots');
 if bodeplots==1,
  bodeplotsOL=1;bodeplotsCL1=0;
 elseif bodeplots==2,
  bodeplotsOL=0;bodeplotsCL1=1;
 elseif bodeplots==3,
  bodeplotsOL=0;bodeplotsCL1=0;
 end
% compzero=input('compute zeros, yes [1], no [0]:   ');
% compsigma=input('compute singular values, yes [1], no [0]:   ');
% salva=menu('save results in zero_g45.mat ?','yes','no');
% salva1=menu('save results in zero_g1.mat ?','yes','no');
 compzero=0;compsigma=0;
 salva=2;salva1=2;
end

timsim=input('do you want linear time simulation, yes[1], no[0]:   ');
if timsim==1,trajectory=7;end;
%if timsim==1,
%trajectory=...
%menu('choose trajectory',...
%'constant step (sudden step to end)',...
%'cosine',...
%'sine',...
%'exponential (shaped step to end)',...
%'quintic polynomial for rest-to-rest',...
%'shaped (smooth) step in specified time',...
%'ref. MACE large angle slew (requires min.3 sec.of simulation)',...
%'Lissajous Figure',...
%'ref. smoothed bang-bang slew');
%end
modal_redux=input('do modal reduction?, yes[1], no[0]:   ');
%modal_redux=0;
%=================================================================
%  these are the 2 angles made by each wheel rotation axis
%  with the X axis of the bus
%           alfa_i - in plane ;  beta_i out-of-plane
%  which is fixed in the L frame -X Y Z- (in radians)
%=================================================================
%ruote=menu('wheels box symm. axis pointing towards:','-Y','+Y');
ruote=2;
if ruote==1,
alfa1=90*rad;beta1=-35.3*rad;
```

```
alfa2=-30*rad;beta2=-35.3*rad;
alfa3=-150*rad;beta3=-35.3*rad;
elseif ruote==2,
alfa1=90*rad;beta1=35.3*rad;
alfa2=-30*rad;beta2=35.3*rad;
alfa3=-150*rad;beta3=35.3*rad;
end
%==============================================================
%  transformation matrix between Fwheel-i and FB
[RwL11,RwL21,RwL31,RwLL1]=rotation(beta1,0,-alfa1);
[RwL12,RwL22,RwL32,RwLL2]=rotation(beta2,0,-alfa2);
[RwL13,RwL23,RwL33,RwLL3]=rotation(beta3,0,-alfa3);
%==============================================================
[pw1,qw1,rw1,lw1,mw1,nw1,fw1,gw1,hw1,Aw1,Bw1,Cw1,RwL1]=...
       WitoBody(alfa1,beta1);
[pw2,qw2,rw2,lw2,mw2,nw2,fw2,gw2,hw2,Aw2,Bw2,Cw2,RwL2]=...
       WitoBody(alfa2,beta2);
[pw3,qw3,rw3,lw3,mw3,nw3,fw3,gw3,hw3,Aw3,Bw3,Cw3,RwL3]=...
       WitoBody(alfa3,beta3);
Tratt=[pw1 pw2 pw3;qw1 qw2 qw3;rw1 rw2 rw3];
invTratt=inv(Tratt);
%==============================================================
%==============================================================
%    End of interactive input data
%==============================================================
%==============================================================

LEGGIGEO    % enter vectors and geometric information

%==============================================================
%==============================================================
if precfre==1,
  RUOTE       % compute precession frequency
elseif precfre==0,
  spin1=0;spin2=0;spin3=0;
end
%==============================================================
%==============================================================
% Transformation matrices relating inertial velocities at nodes
% placed a given distance apart (R=right,L=left of node)
%==============================================================
[TR1]=transform([0.07632;0;0]);[TL5]=transform([-0.07632;0;0]);
[TL2]=transform(rL2);[TR2]=transform(rR2);TL4 = TL2;TR4 = TR2;
%==============================================================
% if central node F.E. node is at the cg
%[TL3]=transform(-rcm3);[TR3]=transform(rRL3-rcm3);
%==============================================================
% if central node F.E. node is on bus centerline
[TL3]=transform(rL3);[TR3]=transform(rR3);
%==============================================================
%==============================================================
% assemble mass matrices
%==============================================================

%disp('<<<<<< assembling multi-rigid mass >>>>>>');
MAKEMASS   % assemble inertia matrices of nodes 1,3 and 5

%==============================================================
%==============================================================
% Before assembling the flex elements, move rows and columns 7,8
% of M1 to 1st and 2nd place so that the state vector for
% the first node becomes [phi1,theta1,x1,y1,z1,d1,d2,d3]'
%==============================================================
M1=M1([7:8 1:6],:);M1=M1(:,[7:8 1:6]);
%==============================================================
% and remove rows and columns 21,22,23 thus eliminating wheels dof's
```

```
%====================================================================
M3new=M3(1:6,1:6);
%====================================================================

%====================================================================
%
%         FINITE ELEMENT MODEL
%
%====================================================================
%disp('<<<<<<.assembling mass and stiffness matrices >>>>>>');
%====================================================================
lele=L/nele;
[mele,kele]=beamele(pA,lele,E,G,Jx,Jy,Jz,A);
kele=kele+1.e-6*mele;
[mflex]=fem(mele,nele);[kflex]=fem(kele,nele);
%====================================================================
%
% Start assembling all flexible parts together
%
%====================================================================
KK=zeros(nfree);MM=zeros(nfree);T=eye(ndof);
KKflex=zeros(nflex);MMflex=zeros(nflex);
KKflexbus=zeros(ntotflex);MMflexbus=zeros(ntotflex);
%====================================================================
% A) assemble flexibility of bus
%    (can accept more than one nele elements)
%====================================================================
T(1:6,1:6)=TR1;T(ndof-5:ndof,ndof-5:ndof)=TL2;
nbeg=1;nend=ndof;
KKflexbus(nbeg:nend,nbeg:nend)=T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=T'*mflex*T;

T(1:6,1.6) = TR2;T(ndof-5:ndof,ndof-5:ndof) = TL3;
nbeg=ndof-5;nend=2*ndof-6;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

T(1:6,1:6) = TR3;T(ndof-5:ndof,ndof-5:ndof) = TL2;
nbeg=2*ndof-11;nend=3*ndof-12;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

T(1:6,1:6)=TR2;T(ndof-5:ndof,ndof-5:ndof)=TL5;
nbeg=3*ndof-17;nend=ntotflex;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

KKflex(1:ntotflex,1:ntotflex)=KKflexbus;
MMflex(1:ntotflex,1:ntotflex)=MMflexbus;
%====================================================================
% augment to include end payloads and wheels
KK(3:nfree-5,3:nfree-5)=KKflex;
MM(3:nfree-5,3:nfree-5)=MMflex;
%====================================================================
% Lump additional mass into finite element model
%====================================================================
% a) first lump spacecraft mass at nodes 1 to 5
MM(1:8,1:8)=MM(1:8,1:8)+M1;

nbeg=ndof-3;nend=ndof+2;
```

```
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M2;

nbeg=2*ndof-9;nend=2*ndof-4;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M3new;

nbeg=3*ndof-15;nend=3*ndof-10;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M4;

nbeg=4*ndof-21;nend=ntot;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M5;
%=============================================================
% arrange mass and stiffness matrices in a form ready for
% integration, i.e. move rigid body dof's after flex. dof's:
% state vector =
%  (flex. coord's + phi1,theta1,phi5,theta5,rate1,rate2,rate3)'
%          payload    (     #1    ) (     #2    )
%                                   torque wheel    (#1)   (#2)   (#3)
%=============================================================
MMint=zeros(nfree);KKint=zeros(nfree);pu=length(MMint);
%=============================================================
% move payloads dof's after flex dof's
%=============================================================
nbeg=[3:(nflex+2) 1 2 (nflex+3) (nflex+4)];nend=nbeg;
MMintn=MM(nbeg,nend);
KKintn=KK(nbeg,nend);
%=============================================================
% add wheels dof's
KKint=[KKintn                          zeros(length(KKintn),3);
       zeros(3,length(KKintn))         zeros(3)];
MMint=[MMintn                          zeros(length(MMintn),3);
       zeros(3,length(MMintn))         M3diag2];

nbeg=[(ntotflex/2+1):(ntotflex/2+3)];
nend=[(nfree-2):nfree];
MMint(nbeg,nend)=M3(4:6,7:9);

nbeg=[(nfree-2):nfree];
nend=[(ntotflex/2+1):(ntotflex/2+3)];
MMint(nbeg,nend)=M3(4:6,7:9)';
%=============================================================
% clamp payload #1 only on full model
%=============================================================
if nele==1,
 clamp=[1:30 33:pu];
elseif nele==2,
 clamp=[1:54 57:pu];
end
KKintc=KKint(clamp,clamp);
MMintc=MMint(clamp,clamp);
puc=length(MMintc);
%=============================================================
% clamp payload #2 on clamped model
%=============================================================
if nele==1,
 clampc=[1:30 33:puc];
elseif nele==2,
 clampc=[1:54 57:puc];
end
KKintcc=KKintc(clampc,clampc);
MMintcc=MMintc(clampc,clampc);
pucc=length(MMintcc);
%=============================================================
Tratt=[pw1 pw2 pw3;qw1 qw2 qw3;rw1 rw2 rw3];
invTratt=inv(Tratt);
Mee=MMint(1:nflex,1:nflex);Kee=KKint(1:nflex,1:nflex);
if p1_locked==0,
   Mrr=MMint(nflex+1:pu,nflex+1:pu);
   Mer=MMint(1:nflex,nflex+1:pu);Mre=Mer';
   Mew=MMint(1:nflex,pu-2:pu);Mwe=Mew';
   Mww=MMint(pu-2:pu,pu-2:pu);
 elseif p1_locked==1,
   Mrr=MMintc(nflex+1:puc,nflex+1:puc);
   Mer=MMintc(1:nflex,nflex+1:puc);Mre=Mer';
   Mew=MMintc(1:nflex,puc-2:puc);Mwe=Mew';
   Mww=MMintc(puc-2:puc,puc-2:puc);
 end
 Rtrans = 0*ones(nflex,3);
 Rtrans(2*ndof-8:2*ndof-6,:) = - Tratt;
 TTwheels= Rtrans - Mew*inv(Mww);
 [U,S,V]=svd(TTwheels);
 % TTwheels = U(1:30,1:3)*S(1:3,:)*V';
 invTT = V*inv(S(1:3,:))*U(1:30,1:3)';
 %=============================================================
 %=============================================================
 % Now introduce damping and Gyroscopic matrices
 % Also introduce geometric stiffness matrix due to g load
 %=============================================================
 DA=zeros(nfree);GYRO=zeros(nfree);DAMP=zeros(nfree);
 GYRODAMP=zeros(nfree);DAMPREDUCED=zeros(nflex);
 %=============================================================
 % The torque wheels contribute with zero stiffness. But they
 % contribute with a gyroscopic matrix GY. Assemble GY.
 %=============================================================
 % Computation of gyroscopic matrix  GYRO
 APw1=spin1*Jwa1*Aw1;BPw1=spin1*Jwa1*Bw1;CPw1=spin1*Jwa1*Cw1;
 APw2=spin2*Jwa2*Aw2;BPw2=spin2*Jwa2*Bw2;CPw2=spin2*Jwa2*Cw2;
 APw3=spin3*Jwa3*Aw3;BPw3=spin3*Jwa3*Bw3;CPw3=spin3*Jwa3*Cw3;
 GYRO(2*ndof-8,2*ndof-7)=APw1+APw2+APw3;
 GYRO(2*ndof-8,2*ndof-6)=BPw1+BPw2+BPw3;
 GYRO(2*ndof-7,2*ndof-6)=CPw1+CPw2+CPw3;
 GYRO(2*ndof-7,2*ndof-8)=-GYRO(2*ndof-8,2*ndof-7);
 GYRO(2*ndof-6,2*ndof-8)=-GYRO(2*ndof-8,2*ndof-6);
 GYRO(2*ndof-6,2*ndof-7)=-GYRO(2*ndof-7,2*ndof-6);
 gyroblock=GYRO(2*ndof-8:2*ndof-6,2*ndof-8:2*ndof-6);

 GYROpic=zeros(KKintn);
 GYROpic(2*ndof-8:2*ndof-6,2*ndof-8:2*ndof-6)=gyroblock;
 GYROint=[GYROpic                        zeros(length(KKintn),3);
          zeros(3,length(KKintn))      zeros(3)];
 GYROintc=GYROint(clamp,clamp);
 %=============================================================
 %                     END ASSEMBLY
 %=============================================================


 % PART B: solve eigenproblem

                     if solveig==1,
 %disp('<<<<<< solving eigenproblem >>>>>>');
 %=============================================================
 % order the state vectors
 %=============================================================
 TP = 0*eye(nflex);TTP=0*eye(nfree);
 n3 = numnp;
 for i=1:n3,                % for each of the nodes :
 TP(i,6*i-5)      = 1;      %x
 TP(i+n3,6*i-4)   = 1;      %y
 TP(i+2*n3,6*i-3)= 1;       %z
 TP(i+3*n3,6*i-2)= 1;       %angle theta1
 TP(i+4*n3,6*i-1)= 1;       %angle theta2
 TP(i+5*n3,6*i)   = 1;      %angle theta3
 end
```

```
TTP(1:nflex,1:nflex)=TP;
TTP(nflex+1:nfree,nflex+1:nfree)=eye(7);
TTPc=TTP(clamp,clamp);
%==============================================================
                        if pl_locked==0,
%==============================================================
% payloads free
KKint=KKint+1.e-7*MMint;              % stiffen RB modes
[eval,evec,s,poles,hertz]=eigenproblem(KKint,-MMint);
hertz=hertz-0.00005*ones(pu,1);hertz(rigidbody+1:pu);
freque=hertz(rigidbody+1:pu);
evec=TTP*evec;
%==============================================================
Aomegas=diag(eval);Adampin=-2*zeta*sqrt(-Aomegas);
Amodal=[zeros(Aomegas) eye(Aomegas);Aomegas  Adampin];
polesmod=eig(Amodal);polesmod=sort(polesmod);

[evecnorm,lamw]=normalizza(MMint,KKint);
modal_mass=evecnorm'*MMint*evecnorm;
modal_stif=evecnorm'*KKint*evecnorm;
OMEGA=sqrt(lamw);radsec=sort(diag(OMEGA));
DAMPMAT=2*zeta*(inv(evecnorm))'*OMEGA*inv(evecnorm);
GYRODAMPint=GYROint+DAMPMAT;
zo=rigidbody+1:length(lamw);
nrig=rigidbody;
%==============================================================
%  select rigid body modes
%==============================================================
RX=RNODE(1,:)';RY=RNODE(2,:)';RZ=RNODE(3,:)';
x  = [ones(n3,1)      ;0*ones(5*n3,1);0*ones(7,1)];
y  = [0*ones(n3,1)    ;ones(n3,1)     ;0*ones(4*n3,1);0*ones(7,1)];
z  = [0*ones(2*n3,1) ;ones(n3,1)     ;0*ones(3*n3,1);0*ones(7,1)];

t1 = [0*ones(n3,1);-RY;RZ;ones(n3,1);0*ones(2*n3,1);0*ones(7,1)];
t2 = [-RX;0*ones(n3,1);-RZ;0*ones(n3,1);...
      ones(n3,1);0*ones(1*n3,1);0*ones(7,1)];
t3 = [-RZ;RY;0*ones(3*n3,1);ones(n3,1);0*ones(7,1)];

pt1= [0*ones(6*n3,1) ;1;0*ones(6,1)];
pf1= [0*ones(6*n3,1) ;0;1;0*ones(5,1)];
pt5= [0*ones(6*n3,1) ;0;0;1;0*ones(4,1)];
pf5= [0*ones(6*n3,1) ;0;0;0;1;0*ones(3,1)];
pw1= [0*ones(6*n3,1) ;0;0;0;0;1;0*ones(2,1)];
pw2= [0*ones(6*n3,1) ;0;0;0;0;0;1;0];
pw3= [0*ones(6*n3,1) ;0;0;0;0;0;0;1];

X = [x y z t1 t2 t3 pt1 pf1 pt5 pf5 pw1 pw2 pw3];evec(:,1:nrig)=X;
%==============================================================
element1=-inv(MMint)*KKint;
element2=-inv(MMint)*GYRODAMPint;
Ac=[zeros(MMint)  eye(MMint);element1  element2];
puAc=length(Ac);
poles2=eig(Ac);
 states=puAc/2;
 inputs=7+2;
%==============================================================
                        elseif pl_locked==1,
%==============================================================
% payload #1 clamped
KKintc=KKintc+1.e-7*MMintc;              % stiffen RB modes
[evalc,evecc0,sc,polesc,hertzc]=eigenproblem(KKintc,-MMintc);
hertzc=hertzc-0.00005*ones(puc,1);hertzc(rigidbody+1-2:puc);
freque=hertzc(rigidbody+1-2:puc);
evecc=TTPc*evecc0;
evalc=sort(evalc);evalc=evalc([sc:-1:1],1);
evecc0=evecc0(:,[sc:-1:1]);
```

```
%=============================================================
Aomegasc=diag(evalc);Adampinc=-2*zeta*sqrt(-Aomegasc);
Amodalc=[zeros(Aomegasc) eye(Aomegasc);Aomegasc  Adampinc];
polesmodc=eig(Amodalc);polesmodc=sort(polesmodc);

[evecnorm,lamw]=normalizza(MMintc,KKintc);
modal_mass=evecnorm'*MMintc*evecnorm;
modal_stif=evecnorm'*KKintc*evecnorm;
% hertzc =  sort(sqrt(diag(modal_stif))/tupi) =
%         =  sort(sqrt(diag((lamw))))/tupi
OMEGA=sqrt(lamw);radsec=sort(diag(OMEGA));
DAMPMAT=2*zeta*(inv(evecnorm))'*OMEGA*inv(evecnorm);
GYRODAMPintc=GYROintc+DAMPMAT;
zo=rigidbody-1:length(lamw);
nrig=rigidbody-2;Llam=length(lamw);
%=============================================================
%  select rigid body modes
%=============================================================
RX=RNODE(1,:)';RY=RNODE(2,:)';RZ=RNODE(3,:)';
x  = [ones(n3,1)      ;0*ones(5*n3,1);0*ones(5,1)];
y  = [0*ones(n3,1)    ;ones(n3,1)     ;0*ones(4*n3,1);0*ones(5,1)];
z  = [0*ones(2*n3,1)  ;ones(n3,1)     ;0*ones(3*n3,1);0*ones(5,1)];

t1 = [0*ones(n3,1);-RY;RZ;ones(n3,1);0*ones(2*n3,1);0*ones(5,1)];
t2 = [-RX;0*ones(n3,1);-RZ;0*ones(n3,1);...
      ones(n3,1);0*ones(1*n3,1);0*ones(5,1)];
t3 = [-RZ;RY;0*ones(3*n3,1);ones(n3,1);0*ones(5,1)];

pt5= [0*ones(6*n3,1) ;1;0;0;0;0];
pf5= [0*ones(6*n3,1) ;0;1;0;0;0];
pw1= [0*ones(6*n3,1) ;0;0;1;0;0];
pw2= [0*ones(6*n3,1) ;0;0;0;1;0];
pw3= [0*ones(6*n3,1) ;0;0;0;0;1];

X = [x y z t1 t2 t3 pt5 pf5 pw1 pw2 pw3];evecc(:,1:11)=X;
%=============================================================
element1=-inv(MMintc)*KKintc;
element2=-inv(MMintc)*GYRODAMPintc;
Ac=[zeros(MMintc)  eye(MMintc);element1  element2];
puAc=length(Ac);
poles2c=eig(Ac);
 states=puAc/2;
 inputs=5+2;
freqall=sort(abs(poles2c(1:2:70))/tupi);
precfreq=freqall(nrig);     % precession frequency in Hz
%=============================================================
                     end
%=============================================================
                     end


% PART C: Build state space model

                    if statespace==1,
%=============================================================
% Assemble forcing Matrix
%=============================================================
% input control torques
% apply unit torques and forces:
%   - at theta and phi at node 1 and 5          = 4
%   - at torque wheels                          = 3
%   - differential torque between nodes 3 and 4 = 2 (active strut)
%   - torque disturbances at the central node   = 3
%=============================================================
%=============================================================
% Measurement matrix  (angular measurements in degrees)
% measure:
%    - theta and phi inertial at nodes 1 and 5              = 4
%    - rigid body rates at central node                     = 3
%    - vertical accelerations at bus nodes                  = 3*nodi
%    - angular gradient (bending) between nodes 3 and 4     = 2
%      (this is a numerical approximation to obtain the outputs
%       of the strain gauges)
%==============================================================
                     if pl_locked==0,
%==============================================================
 bc = 0*ones(states,inputs);
% a) payloads input torques in [Nm]
 bc(ntot-3,1) = 1;
 bc(ntot-2,2) = 1;
 bc(ntot-1,3) = 1;
 bc(ntot,4) = 1;
% b) torque wheels input torques  in [Nm]
 bc(ntot+1,5) = 1;
 bc(ntot+2,6) = 1;
 bc(ntot+3,7) = 1;
% c) differential torque between nodes 3 and 4
 bc(2*ndof-7,8) = -1;bc(3*ndof-13,8) = 1;      % out-of-plane
 bc(2*ndof-6,9) = -1;bc(3*ndof-12,9) = 1;      % in-plane

accel=3;
measure=7+accel+2;
cc = 0*ones(7+2,states);
cstar2=zeros(accel,states);

% a) rigid body inertial angles in [deg]
cc(1,2*ndof-8)=1*invrad;
cc(2,2*ndof-7)=1*invrad;
cc(3,2*ndof-6)=1*invrad;

% b) payloads inertial angles in [deg]
cc(4,ntot-3)=1*invrad;cc(4,ndof-8)=1*invrad;
cc(5,ntot-2)=1*invrad;cc(5,ndof-4)=1*invrad;
cc(6,ntot-1)=1*invrad;cc(6,ntot-6)=1*invrad;
cc(7,ntot)=1*invrad;cc(7,ntot-4)=1*invrad;

% c) angular gradient (bending) between nodes 3 and 4
cc(8,2*ndof-7)=-1*invrad/lele;cc(8,3*ndof-13)=1*invrad/lele;
cc(9,2*ndof-6)=-1*invrad/lele;cc(9,3*ndof-12)=1*invrad/lele;

% d) accelerations (3 components) at node 4
if nele==1,
  pick1=[19:21];
 %pick1=[1:3 7:9 13:15 19:21 25:27];
elseif nele==2,
  pick1=[37:39];
 %pick1=[1:3 7:9 13:15 19:21 25:27 31:33 37:39 43:45 49:51];
end
cstar2(1:accel,pick1)=eye(accel);
%==============================================================
Bc = [zeros(MMint);inv(MMint)]*bc;
Cc=[cc                 zeros(cc);
    zeros(cc)          cc;
    cstar2*element1  cstar2*element2];
[rowb,colb]=size(Bc);[rowc,colc]=size(Cc);
Dc=zeros(rowc,colb);
Dc = [zeros(rowc-accel,colb);
      cstar2*inv(MMint)*bc];
%==============================================================
%==============================================================
                     elseif pl_locked==1,
%==============================================================
  bc = 0*ones(states,inputs+3);
 % a) payloads input torques in [Nm]
  bc(ntot-3,1) = 1;
  bc(ntot-2,2) = 1;
 % b) torque wheels input torques in [Nm]
  bc(ntot-1,3) = 1;
  bc(ntot,4) = 1;
  bc(ntot+1,5) = 1;
  bc(2*ndof-8:2*ndof-6,3:5)  = -Tratt;
 % c) differential torque between nodes 3 and 4
  bc(2*ndof-7,6) = -1;bc(3*ndof-13,6) = 1;
  bc(2*ndof-6,7) = -1;bc(3*ndof-12,7) = 1;
 % d) bus attitude disturbance torques in [Nm]
  bc(2*ndof-8:2*ndof-6,8:10)  = eye(3);

 accel=3;
 measure=5+accel+2;
 cc = 0*ones(5+2,states);
 cstar2=zeros(accel,states);

 % a) rigid body inertial angles (attitude of the bus) in [deg]
 cc(1,2*ndof-8)=1*invrad;
 cc(2,2*ndof-7)=1*invrad;
 cc(3,2*ndof-6)=1*invrad;

 % b) payloads inertial angles in [deg]
 cc(4,ntot-3)=1*invrad;cc(4,ntot-6)=1*invrad;
 cc(5,ntot-2)=1*invrad;cc(5,ntot-4)=1*invrad;

 % c) angular gradient (bending) between nodes 3 and 4
 % horizontal strain on strut
 cc(6,2*ndof-7)=-1*invrad/lele;cc(6,3*ndof-13)=1*invrad/lele;
 % vertical strain on strut
 cc(7,2*ndof-6)=-1*invrad/lele;cc(7,3*ndof-12)=1*invrad/lele;

 % d) accelerations (3 components) at node 4
 if nele==1,
   pick1=[19:21];
  %pick1=[1:3 7:9 13:15 19:21 25:27];
 elseif nele==2,
   pick1=[37:39];
  %pick1=[1:3 7:9 13:15 19:21 25:27 31:33 37:39 43:45 49:51];
 end
 cstar2(1:accel,pick1)=eye(accel);
 %=============================================================
 Bc = [zeros(MMintc);inv(MMintc)]*bc;
 Cc=[cc                zeros(cc);
     zeros(cc)    .    cc;
     cstar2*element1   cstar2*element2];
 [rowb,colb]=size(Bc);[rowc,colc]=size(Cc);
 Dc=zeros(rowc,colb);
 Dc = [zeros(rowc-accel,colb);
       cstar2*inv(MMintc)*bc];
 %==============================================================
                       end

 %==============================================================
 if bodeplotsOL==1,
 %==============================================================
 % find minimal realization
 %==============================================================
 disp('>>>>>>looking for minimal realization<<<<<<<<<<<<');
 [Amin,Bmin,Cmin,Dmin]=minreal(Ac,Bc,Cc,Dc);
 Ac=Amin;Bc=Bmin;Cc=Cmin;Dc=Dmin;
 %==============================================================
 disp('>>>>>>computing data for the open-loop Bode plots<<<<<<<<<<<<');
```

```
[mag1,fase1]=bode(Ac,Bc,Cc,Dc,1,wbig*tupi);
[mag2,fase2]=bode(Ac,Bc,Cc,Dc,2,wbig*tupi);
%[mag3,fase3]=bode(Ac,Bc,Cc,Dc,3,wbig*tupi);
%[mag4,fase4]=bode(Ac,Bc,Cc,Dc,4,wbig*tupi);
%[mag5,fase5]=bode(Ac,Bc,Cc,Dc,5,wbig*tupi);
%[mag6,fase6]=bode(Ac,Bc,Cc,Dc,6,wbig*tupi);
%[mag7,fase7]=bode(Ac,Bc,Cc,Dc,7,wbig*tupi);
%[mag8,fase8]=bode(Ac,Bc,Cc,Dc,8,wbig*tupi);
%[mag9,fase9]=bode(Ac,Bc,Cc,Dc,9,wbig*tupi);
%[mag10,fase10]=bode(Ac,Bc,Cc,Dc,10,wbig*tupi);
%=================================================================
end
%=================================================================
if salva==1,

    save zero_g45,clear

end

if salva1==1,

    save zero_g1,clear

end
%=================================================================
if compsigma==1,
disp('>>>>>>computing loop singular values<<<<<<<<<<<<');
[singolar,omegas]=sigma(Ac,Bc(:,[1 2]),Cc,Dc(:,[1 2]));
end
if compzero==1,zeros=tzero(Ac,Bc,Cc,Dc);end
%=================================================================
                      end

%=================================================================
%SYSTEM=pck(Ac,Bc,Cc,Dc);minfo(SYSTEM),poles=spoles(SYSTEM);
%poles=poles-1.e-5;rifd(poles)
%=================================================================
% put system in modal form

if modal_redux==1,
%=================================================================
Tsim=0*ones(70);
Tsim=[evecc0 0*ones(35);0*ones(35) evecc0];
invTsim=inv(Tsim);
Amod=invTsim*Ac*Tsim;Bmod=invTsim*Bc;Cmod=Cc*Tsim;Dmod=Dc;
flexy=Llam-nrig;cut=puc-flexy;
SYSTEM0=pck(Amod,Bmod,Cmod,Dmod);poles0=spoles(SYSTEM0);
%=================================================================
% now consider only the flexible modes (i.e., partition out
% rigid body modes)
%  Note that there is no contribution of gyroscopic forces, and
%  that the equations of motion are linearized about a payload
%  configuration.
%=================================================================
Arigid=[0*ones(cut) eye(cut);0*ones(cut) 0*ones(cut)];
Brigid=[0*ones(cut,inputs+3);Bmod(states+flexy+1:2*states,:)];
Crigid=[Cmod(:,flexy+1:states) Cmod(:,states+flexy+1:2*states)];
%=================================================================
Aflex21=Amod(states+1:states+flexy,1:flexy);
Aflex22=Amod(states+1:states+flexy,states+1:states+flexy);
Aflex=[0*ones(flexy) eye(flexy);Aflex21 Aflex22];
Bflex=[0*ones(flexy,inputs+3);Bmod(states+1:states+flexy,:)];
Cflex=[Cmod(:,1:flexy) Cmod(:,states+1:states+flexy)];
Dflex=Dmod;
%=================================================================
% Balanced Reduction
disp(' ');
disp('   starting balanced modal reduction    ');
disp(' ');
%===============================================================
% do balanced truncation on flexible part and correct for DC
% stiffness of truncated modes
%===============================================================
[Abal,Bbal,Cbal,Hankel_sv,Tsimil] = balreal(Aflex,Bflex,Cflex);
% for the 0-90 0-90 configuration, keep the first 30 flex states
% (use :max(Hankel_sv)/130)
elim=find(Hankel_sv<max(Hankel_sv)/130);
Lelim=length(elim);elim=elim(Lelim:-1:1);
[Ar,Br,Cr,Dr]=modred(Abal,Bbal,Cbal,Dflex,elim);
nR=flexy-Lelim/2
%SYSTEM3=pck(Ar,Br,Cr,Dr);

% append back to rigid model
% new state = X = (etar etar_dot | eta_flex  eta_flex_dot);

AdynR=[Arigid 0*ones(2*nrig,2*nR);0*ones(2*nR,2*nrig) Ar];
BdynR=[Brigid;Br];
CdynR=[Crigid Cr];
DdynR=Dr;
newstates=nR+cut;
%SYSTEMR=pck(AdynR,BdynR,CdynR,DdynR);
%===============================================================
%semilogy(Hankel_sv,'x'),grid;
%plot(spoles(SYSTEM),'x'),hold,plot(spoles(SYSTEM3),'o'),hold
%===============================================================
% check transfer functions
%===============================================================
%disp(' ');
%disp('   compute full and reduced order transfer functions    ');
%disp(' ');
[mag1,fase1]=bode(Ac,Bc,Cc,Dc,1,wbig*tupi);
[mag2,fase2]=bode(Ac,Bc,Cc,Dc,2,wbig*tupi);
[magRp,faseRp]=bode(AdynR,BdynR,CdynR,DdynR,1,wbig*tupi);
[magR,faseR]=bode(AdynR,BdynR,CdynR,DdynR,2,wbig*tupi);
%paragone(mag2,magR,fase2,faseR,5,wbig,wbig);
%loglog(wbig,mag2(:,j),'--'),hold,loglog(wbig,magR(:,j)),hold,grid
%loglog(wbig,mag1(:,j),'--'),hold,loglog(wbig,magRp(:,j)),hold,grid
%===============================================================
end

% criterion on close-spaceness of frequencies
lam=sort(sqrt(diag(lamw)));lam=lam(Llam:-1:1);
vec=[];vac=[];vic=[];
for i=1:flexy,
 j=i+1;
 lam=sqrt(diag(lamw));
 num=max([lam(i);lam(j)]);
 den=abs(lam(i)-lam(j));
 test=zeta*num/den;
 vec=[vec;test];vac=[vac;num];vic=[vic;den];
end
%===============================================================
end

if timsim==1,
%===============================================================
% compute time response (neglecting non-linear terms)
%===============================================================
disp(' ');
disp('   computing linear response    ');
disp(' ');
```

```
HBwth=1;LBwth=0;
if HBwth==1,
 KPfi=310.4901;KDfi=4.28;
 KPtheta=452.5788;KDtheta=6.2387;
end
if LBwth==1,
 KPfi=0.0310;KDfi=0.0428;
 KPtheta=0.0453;KDtheta=0.0624;
end
KP=[KPfi 0;0 KPtheta];
KD=[KDfi 0;0 KDtheta];

% Initial Conditions for full order model
x0=0*ones(70,1);
x0(31:32,1)=[0;-150]*pi/180;
x0((31+35):(32+35),1)=[0;0]*pi/180;
tet0=x0(32,1);fi0=x0(31,1);
tetf=-30*pi/180;fif=0*pi/180;
%===========================================================
if modal_redux==1,
%===========================================================
% Initial Conditions for reduced order model
%===========================================================
% using x = T*x', transform state vector from:
%
% x = (qe qr | qe_dot qr_dot) to: x' = (qr qr_dot | qe qe_dot)'
%
%===========================================================
Tm=0*ones(2*states);
if pl_locked==0,nrigy=7;elseif pl_locked==1,nrigy=5;end;
subT1=0*ones(2*nrig,states);
subT2=0*ones(2*nrig,states);
subT3=0*ones(2*flexy,states);
subT4=0*ones(2*flexy,states);
subT1(1:nrig,flexy+1:states)=eye(nrig);
subT2(nrig+1:2*nrig,flexy+1:states)=eye(nrig);
subT3(1:flexy,1:flexy)=eye(flexy);
subT4(flexy+1:2*flexy,1:flexy)=eye(flexy);
Tm=[subT1 subT2;subT3 subT4];
x0R=Tm*invTsim*x0;
x0red=...
x0R([1:2*nrig 2*nrig+1:(2*nrig+nR) (2*nrig+flexy+1):(2*nrig+flexy+nR)]);
%===========================================================
end
%===========================================================
% DECENTRALIZED CONTROL SCHEME
% close the loop with high BW PD control at the payload
%  and low BW PD control at the torque wheels
% a stochastic disturbance acts at the central node
%===========================================================
%===========================================================
% wheel torques: low BW control and gyroscopic cancellation
%  see also Wie,Weiss and Arapostathis' paper
%===========================================================
kgain = 2*(2*pi*hertzc(12)/10)^2;
dgain = 2*0.707*(2*pi*hertzc(12)/10);
Jgain=diag((diag((JMACE))));
KPw1= kgain*Jgain;KDw1=dgain*Jgain;
Kwh=0*ones(3,nflex);Kwh(:,16:18)=eye(3);
KPw=KPw1*Kwh;KDw=KDw1*Kwh;
GAIN=0*ones(inputs+3,2*states);
GAIN(1,31)=-310.4901;GAIN(1,31+35)=-4.28;
GAIN(2,32)=-452.5788;GAIN(2,32+35)=-6.2387;
%  low BW attitude control loop
GAIN(3:5,33:35)=-KPw1;GAIN(3:5,33+35:35+35)=-KDw1;
```

```
ACL=Ac+Bc*GAIN;polescl=eig(ACL);polescl=sort(polescl);
plot(polescl,'x')
%===============================================================
tim=[0:0.01:10]';Lt=length(tim);Mt=max(tim);
timtstop=2.88;Jtet=0.0430;Jfi=0.0295;
taut=[];tauf=[];pos=[];vel=[];accel=[];
for k=0:0.01:Mt,
   trajectory=7;
    [q1dest,q1desdt,q1desddt]=rampvel(k,timtstop,tet0,tetf);
   trajectory=5;
    [q1desf,q1desdf,q1desddf,torquef]=traject(k,fi0,fif,Jfi);
   torquet=Jtet*q1desddt+452.5788*q1dest+6.2387*q1desdt;
   torquef=Jfi*q1desddf+310.4901*q1desf+4.28*q1desdf;
   taut=[taut;torquet];tauf=[tauf;torquef];
   pos=[pos;q1dest];vel=[vel;q1desdt];accel=[accel;q1desddt];
end
rand('normal');

tau1=rand(Lt,1);tau2=rand(Lt,1);tau3=rand(Lt,1);
tau_dist=[tau1  tau2  tau3];
tau_cont=[taut];
TAU=[tau_cont tau_dist];
influx=[2];
%===============================================================
[ys,xs]=lsim(ACL,Bc(:,influx),Cc,Dc(:,influx),TAU(:,1),tim,x0);
%===============================================================

subplot(211)
plot(tim,xs(:,32)*invrad),grid
title('angle vs. time')
subplot(212)
plot(tim,xs(:,32+35)*invrad),grid
title('angular velocity vs. time')
subplot(111)
%===============================================================
% now you can compare the linear trajectory tracking (ys, xs vs. tim)
% with the non-linear response (x vs. t from RUNMACE.m)
% load simulazione5
%plot(t,(x(:,32)+x(:,30))*invrad,'--'),hold,plot(tim,ys(:,5)),hold
%===============================================================
%[ysR,xsR]=lsim(AdynR,BdynR(:,2),CdynR,DdynR(:,2),taut,tim,x0red);
%j=12;plot(tim,ys(:,j)),hold,plot(tim,ysR(:,j),'o'),hold
%===============================================================
end

torquet=Jtet*accel-452.5788*(xs(:,32)-q1dest)...
         -6.2387*(xs(:,32+35)-q1desdt);

gravpar=0;
global trajectory timtstop gravpar
global rAoAA1 rAFF1 rFplusF1 rPAA1 rBPP1 rpayPP1
global rAoAA5 rAFF5 rFplusF5 rPAA5 rBPP5 rpayPP5
animation(tim,4,5,xs,1)


return

 plot(tim,xs(:,[1 7 13 19 25]))

 plot(tim,xs(:,[2 8 14 20 26]))

 plot(tim,xs(:,[3 9 15 21 27]))

 plot(tim,xs(:,[4 10 16 22 28])*invrad)

 plot(tim,xs(:,[5 11 17 23 29])*invrad)
plot(tim,xs(:,[6 12 18 24 30])*invrad)
```

```
%=================================================================
%=================================================================
% Author: Marco B. Quadrelli
% Date  : July 23, 1991
% Last revision: October 28, 1991
%=================================================================
% This subroutine appends the geometric
% input data for the 3D MACE test article in the presence of
% suspension wires
%  the payloads are mounted and pointing downwards!
%=================================================================
%=================================================================
inch=0.0254;
%=============================================================
[sf1,cf1,st1,ct1,ct1cf1,st1sf1,st1cf1,sf1ct1,st1ct1,sf1cf1,...
        ct1sf1,cf1st1,ct1st1,cf1sf1]=sintax(theta1,phi1);
[sf5,cf5,st5,ct5,ct5cf5,st5sf5,st5cf5,sf5ct5,st5ct5,sf5cf5,...
        ct5sf5,cf5st5,ct5st5,cf5sf5]=sintax(theta5,phi5);
%===============================================================
%  derive transformation matrices between frames
%===============================================================
%  transformation matrix from Fpivot to Fpayload (F=vectrix)
[R2p11,R2p21,R2p31,R2p1]=rotation(phi1,0,theta1);
[R2p15,R2p25,R2p35,R2p5]=rotation(phi5,0,theta5);
CAF1=R2p11;CAF5=R2p15;
CPA1=R2p31;CPA5=R2p35;
%===============================================================
%===============================================================
%        EXTERNAL DIMENSIONS
%===============================================================
L  = 0.22886;      % length of Lexan member
Ljl= 0.15264;      % length of long joint (node + collars)
Ljs= 0.10807;      % length of short joint(node + collar)
Lw = 4.6736;       % length of suspension wire
Lwrig= .05;    % length of rigid support of suspension wire
% dimensions of gimbal support
asup = 0.05;bsup = 0.05;csup= 0.05;
% dimensions of gimbals  (ag = height;bg = radius of cylinder)
ag1=0.2;bg1=0.05;
ag2=0.1;bg2=0.05;
%===============================================================
% s = short joint (external joints);l = long joint(internal joints)
hcollar=0.04458;diamcE=0.0444;diamcI=0.0254;
acube=0.0635;
ajl = 0.0635+2*hcollar;bjl = 0.0635;cjl = 0.0635;
ajs = 0.0635+1*hcollar;bjs = 0.0635;cjs = 0.0635;
% collar
Acollar=(pi/4)*(diamcE^2 - diamcI^2);
RcolRcg=[0.0575;0;0];RcolLcg=[-0.0575;0;0];
%===============================================================
%  sensors box at node 3
asen = 2.5*inch;bsen = 2.5*inch;csen = 2.5*inch;
%  momentum wheels box
aw = 0.2;bw = 0.2;cw = 0.2;
%  identical payloads at the ends
height = 3.5*inch;radius = 2.5*inch;
%  each torque wheel
Rw = 0.08;heightw=0.10;
%===============================================================
%     ELASTICITY PARAMETERS
%  FOR FINITE ELEMENT BEAM, JOINTS, SUSPENSION WIRES AND RIGID
%  BARS connecting the suspension wires to the bus
%===============================================================
Do = .0254;Di = .0190;Dw = 0.00278;Dwrig = 0.005;
Ro = Do/2;Ri = Di/2;
A  = (Ro^2 - Ri^2)*pi;     % Area for flexible beam
```

```
Aj = bjs*cjs;                  % Area for rigid joint
Aw = (pi/4)*(Dw)^2;            % Area for suspension wire
Awrig=(pi/4)*(Dwrig)^2;        % Area for rigid bar of suspension wire
Jx = (pi/32)*(Do^4 - Di^4);Jp=Jx;
Jy = (pi/4)*(Ro^4 - Ri^4);
Jz = Jy;
Jyw = (pi/4)*((Dw/2)^4);Jywrig = (pi/4)*((Dwrig/2)^4);
Jzw = Jyw;Jzwrig = Jywrig;
Jxw = Jyw + Jzw;Jxwrig = Jywrig + Jzwrig;Iw=Jxw;Jpw=Jxw;
%===========================================================
zeta=0.01;                     % damping ratio
E  = 2.3e+9;Ej = 7.31e+10;     % E for flexible beam and joint
Ew = 2.1e+11;                  % E for flexible suspension wire (steel)
nu = 0.37;nuj= 0.33;           % Poisson's ratio for beam and joint
nuw= 0.27;                     % Poisson's ratio for suspension wire
G  = (E/2)/(1+nu);Gj = (Ej/2)/(1+nuj);Gw = (Ew/2)/(1+nuw);
Ewrig=100000*Ew;Gwrig=100000*Gw;
pA = 1189.77*A;          % mass/length of beam
pAj= 2766.91*Aj;        % mass/length of joint
densw=7800;            % density of wire (steel)
pAw= 7800*Aw;          % mass/length of suspension wire
pAwrig=100000*pAw;
pAcollar=2711.57;
pAbase=pAj;
const=[E*Jy;E*Jz;E*A;G*Jp;Jp;pA;A;Jy;Jz]';
constw=[Ew*Jyw;Ew*Jzw;Ew*Aw;Gw*Jpw;Jpw;pAw;Aw;Jyw;Jzw]';
%===========================================================
%     MASSES
%===========================================================
%Mwheels  = 10.0;Mjoints = 0.4;Mjointl = 0.6;Msensors = 1.5;
% values updated on April 30, 1991
Mbase=1.70030;Mmcase=.5726;Mmarmature=0.1982;Mwheelone=1.09726;
Mfixed=Mmcase;Mrotat=Mmarmature+Mwheelone;Q=Mfixed+Mrotat;
Mwheels  = Mbase+3.*(Mmcase+Mmarmature+Mwheelone);
Mnodecube=0.2705;
Mcollar=0.11137;
Mjoints = Mnodecube+Mcollar;
Mjointl = Mnodecube+2*Mcollar;

Mdummy    = 7.09984;   % dummy payload at node 1
Msensors2 = 0.72968;   % sensors box at node 2 (triax. accel.)
Msensors3 = 1.07100;   % sensors box at node 3 (rate gyro package)
Msensors4 = 0.66060;   % sensors box at node 4 (triax. accel.)

Mcarriage=1.3132;
Mcable=pAw*Lw;
Mpayload=1.29798; % mass of rate gyro + payload can;
Msupport=2.991823;
Mgimbal1=2.922707;  % (composed of inner stage + encoder + frame)
Mgimbal2=1.061621;  % (composed of outer stage + payload support)
Mgimbal=Mgimbal1+Mgimbal2;
Mw=Mwheels/3;Mp=Mpayload;Mjs=Mjoints;Mjl=Mjointl;
Mtot3 = Mwheels+Msensors3+Mjointl;
Mtot  = Mpayload+Mgimbal1+Mgimbal2+Mjoints;

reduce=1;

Mbody1= reduce*(Mjoints+Msupport);
Mbody2= reduce*(Mgimbal1);
Mbody3= reduce*(Mgimbal2+Mpayload);
Mnode=Mbody1+Mbody2+Mbody3;


Mnode1=Mnode;
Mnode2=(Mjointl+Msensors2);
Mnode3=Mtot3;
```

```
 Mnode4=(Mjoint1+Msensors4);
 Mnode5=Mnode;
 if p1_locked==0,
      Mnode1s=Mnode;
 elseif p1_locked==1,
      Mnode1s=Mdummy+Mjoint1;
 end

 Massrigid = Mnode1s+Mnode2+Mnode3+Mnode4+Mnode5;
 Massflex  = 4*pA*L;
 MassMACE  = Massrigid + Massflex;
 NodemassMAP=[Mnode1s  Mnode2  Mnode3  Mnode4  Mnode5];
 %===========================================================
 %===========================================================
 %===========================================================
 %  from node 1 (or 5 ) to pivot  (center of gimbal no.1)
 %===========================================================
 rp1=rdum;rp5=rdum;
 if gimbal1==2,
 rp1 = [0;-0.1143;-0.00315];rp5 = [0;-0.1143;-0.00315];
 elseif gimbal1==1,
 rp1=...
 input('enter coordinates of point A wrt. point F (node 1):   ');
 rp5=...
 input('enter coordinates of point A wrt. point F (node 5):   ');
 end
 rAFF1=rp1;rAFF5=rp5;
 %===========================================================
 % off-set between centers of rotation of the two gimbals
 %===========================================================
 rPAA1=zeros(3,1);rPAA5=zeros(3,1);
 if center==2,
 rPAA1=[0;0;0];rPAA5=[0;0;0];
 elseif center==1,
 rPAA1=...
 input('enter coordinates of 2nd center wrt. the 1st [gimbal 1]:  ');
 rPAA5=...
 input('enter coordinates of 2nd center wrt. the 1st [gimbal 5]:  ');
 end

 %===========================================================
 % vector from pivot to payload center of mass (the frames at the
 % pivots 1 and 5 are parallel and in the same directions)
 % note: at zero angle, the payload is along p1
 if CG==1,
 rpayPP1=[0;0;0];rpayPP5=[0;0;0];
 elseif CG==2,
 rpayPP1=[0.17145;0;0];rpayPP5=[0.17145;0;0];
 end
 po1= rpayPP1(1);po5=rpayPP5(1);
 p1=CAF1*CPA1*rpayPP1;
 p5=CAF5*CPA5*rpayPP5;
 %p1 = po1*[cos(phi1)*cos(theta1);cos(phi1)*sin(theta1);sin(phi1)];
 %p5 = po5*[cos(phi5)*cos(theta5);cos(phi5)*sin(theta5);sin(phi5)];
 %===========================================================
 %  from node 1 (or 5) to center of mass of gimbal
 rg1 = [0;-0.1143;0];rg5 = [0;-0.1143;0];
 %===========================================================
 if ruote==1,
 rw = [0.07632;-0.10974;0];% radius vector of wheels com.wrt.left point
 rwroot=[0.07632;-0.07433;0]; %vector of root wrt. point L
 elseif ruote==2,
 rw = [0.07632;+0.10974;0];
 rwroot=[0.07632;+0.07433;0];
 end
 %===========================================================
% radius vector of com. of each wheel wrt.L point
w1=0.16;w2=0.16;w3=0.16; %distance of com of each wheel wrt. root point
rww1=w1*[pw1;qw1;rw1];
rww2=w2*[pw2;qw2;rw2];
rww3=w3*[pw3;qw3;rw3];
rwww1=rwroot+rww1;rwww2=rwroot+rww2;rwww3=rwroot+rww3;
rwww=(1/3)*(rwww1+rwww2+rwww3);
rs = [0.07632;-0.06985;0];          % same but for the sensor box
rj3= [0.07632;0;0];                 % same but for the central joint
rR3=[0.07632;0;0];
rRL3= [0.15264;0;0];          % same but for the right wrt.left point
rcomw1=[w1;0;0];rcomw2=[w2;0;0];rcomw3=[w3;0;0];
%=============================================================
%  center of mass of all node 3 wrt. left point
rcm3 = (1/Mtot3)*(rj3*Mjoint1+Msensors3*rs+Mwheels*rw);
rcmw = rcm3-rwww;rcmj = rcm3-rj3;rcms = rcm3-rs;
rcmrut=rcm3-rwroot;
% offset of c.g. of central node wrt. point on the bus centerline
rH3=rcm3-rR3;
%=============================================================
%  from node 1 (or 5) to center of mass of joint
rj1 = [0;0;0];rj5 = [0;0;0];
rjFF1=rj1;rjFF5=rj5;
%  from node 1 or 5 to com of gimbal support
rSFF1=[-0.08;-0.12315;0];rSFF5=[0.08;-0.12315;0];
rG1AA1=[0;0;0];rG1AA5=[0;0;0];
rG2PP1=[0;0;0];rG2PP5=[0;0;0];

% com's of gimbal motors are not centered
%rG1AA1=[-0.001;0;0];rG1AA5=[0.001;0;0];
%rG2PP1=[0;0.001;0];rG2PP5=[0;0.001;0];
%=============================================================
% center of mass of joint and gimbal mount (wrt. node 1 and 5)
rcmgj1 = (Mgimbal*rg1+Mjoints*rj1)/(Mgimbal+Mjoints);
rcmgj5 = (Mgimbal*rg5+Mjoints*rj5)/(Mgimbal+Mjoints);
%=============================================================
rga1 = rg1-rcmgj1;rga5 = rg5-rcmgj5;
rja1 = rj1-rcmgj1;rja5 = rj5-rcmgj5;
%=============================================================
%  vector from node 1 (or 5) to payload c.o.m.
rpay1 = rp1+p1;rpay5 = rp5+p5;
rR1 = rcmgj1;
rQ1 = (Mpayload*rpay1+(Mgimbal+Mjoints)*rcmgj1)/Mtot;
rQ5 = (Mpayload*rpay5+(Mgimbal+Mjoints)*rcmgj5)/Mtot;
rL5 = rcmgj5;
%=============================================================
rot1=rp1-rcmgj1;rot5=rp5-rcmgj5;
%=============================================================
rL2 = [-0.07632;0;0];rR2 = [0.07632;0;0];
rL3 = [-0.07632;0;0];rR3 = [0.07632;0;0];
%=============================================================
if pollo==1,
rSFF1=rdum;rG1AA1=rdum;rjFF1=rdum;rAFF1=rdum;rG1AA1=rdum;
rSFF5=rdum;rG1AA5=rdum;rjFF5=rdum;rAFF5=rdum;rG1AA5=rdum;
end
%=============================================================
%=============================================================
%============================================================
% compute the vectors from FE node on the bus to hinged node at the
%  bottom of the suspension wire (these hinged nodes are L,M,N);
%============================================================
rL1 = [0;+0.04205;0];       % [-0.054;+0.04205;0];
rM3 = [0;+0.04205;0];
rN5 = [0;+0.04205;0];       % [+0.054;+0.04205;0];
%============================================================
%=============================================================
%===========================================================
% center of mass of body 1 and other relevant vectors
%==========================================================
rFplusF1=(1/Mbody1)*(Mjoints*rjFF1+Msupport*rSFF1);
rFplusF5=(1/Mbody1)*(Mjoints*rjFF5+Msupport*rSFF5);
%==========================================================
%  center of mass of body 2  in A frame
%==========================================================
rAoAA1=rG1AA1;
rAoAA5=rG1AA5;
%==========================================================
%  center of mass of body 3  in P frame
%==========================================================
rBPP1=(1/Mbody3)*(rpayPP1*Mpayload+rG2PP1*Mgimbal2);
rpayBP1=rpayPP1-rBPP1;rG2BP1=rG2PP1-rBPP1;
rBPP5=(1/Mbody3)*(rpayPP5*Mpayload+rG2PP5*Mgimbal2);
rpayBP5=rpayPP5-rBPP5;rG2BP5=rG2PP5-rBPP5;
%==========================================================
% vector from FEM node to (body 1 + body 2) com  in F frame
%==========================================================
rAoF1=rAFF1+CAF1*rAoAA1;
rAoF5=rAFF5+CAF5*rAoAA5;
%==========================================================
% vector from FEM node to payload com in F frame
%==========================================================
rPF1=rAFF1+CAF1*rPAA1+CPA1*CAF1*rBPP1;
rPF5=rAFF5+CAF5*rPAA5+CPA5*CAF5*rBPP5;
%==========================================================
% center of mass of (body 1+ body 2 +body 3) wrt FEM node in F frame
%==========================================================
rcom1=(1/Mnode)*(Mbody1*rFplusF1+Mbody2*rAoF1+Mbody3*rPF1);
rcom5=(1/Mnode)*(Mbody1*rFplusF5+Mbody2*rAoF5+Mbody3*rPF5);
%==========================================================
%  transport vectors to A frame
%==========================================================
rAoFF1=rAoF1;
rAoFF5=rAoF5;
rAoAF1=rAoFF1-rAFF1;rAoAF5=rAoFF5-rAFF5;
rjAoF1=rjFF1-rAoFF1;rjAoF5=rjFF5-rAoFF5;
rjAA1=CAF1*rjAoF1;rjAA5=CAF5*rjAoF5;
rG1AoA1=rG1AA1-rAoAA1;rG1AoA5=rG1AA5-rAoAA5;
rSAA1=CAF1*(rSFF1-rAFF1);rSAA5=CAF5*(rSFF5-rAFF5);
%==========================================================
% compute coordinates of centers of mass of the whole body
% with respect to com of node 3; nodes begin from the left.
%==========================================================
rbar=[L;0;0];
rmon=rR3;rmon(2)=-rcm3(2)*0;
rnode4=(rmon+rbar+2*rR2);

rnode5body1=(rnode4+rbar+2*rR2+rFplusF5);
rnode5body2=(rnode4+rbar+2*rR2+rAoFF5);
rnode5body3=(rnode4+rbar+2*rR2+rPF5);

rnode2=-rnode4;
rnode1body1=(rnode2-rbar-2*rR2+rFplusF1);
rnode1body2=(rnode2-rbar-2*rR2+rAoFF1);
rnode1body3=(rnode2-rbar-2*rR2+rPF1);

rcombar1=rnode2-rR2-(rbar/2.);
rcombar2=rcm3-(rbar/2.);
rcombar3=-(rcombar2);
rcombar4=-(rcombar1);
lengthMACE=4*rbar(1)+8*rR2(1);    % from node 1 to node 5 along X
%==========================================================
% compute constants of inertia matrices of end nodes
```

```
%===============================================================
[scugni1]=addyadic(1,rFplusF1);
[scugni5]=addyadic(1,rFplusF5);
[buga1]=transport(rFplusF1);
[buga5]=transport(rFplusF5);

A11=rAoAA1(1);
B11=rAoAA1(2)*cf1-rAoAA1(3)*sf1;
C11=rAoAA1(2)*sf1+rAoAA1(3)*cf1;
A15=rAoAA5(1);
B15=rAoAA5(2)*cf5-rAoAA5(3)*sf5;
C15=rAoAA5(2)*sf5+rAoAA5(3)*cf5;

A21=C11+rAFF1(3);
B21=B11+rAFF1(2);
C21=A11+rAFF1(1);
A25=C15+rAFF5(3);
B25=B15+rAFF5(2);
C25=A15+rAFF5(1);

A31=rPAA1(1);
B31=rPAA1(2)*cf1-rPAA1(3)*sf1;
C31=rPAA1(2)*sf1+rPAA1(3)*cf1;
A35=rPAA5(1);
B35=rPAA5(2)*cf5-rPAA5(3)*sf5;
C35=rPAA5(2)*sf5+rPAA5(3)*cf5;

A41=rBPP1(1)*ct1-rBPP1(2)*st1;
B41=rBPP1(1)*st1*cf1+rBPP1(2)*ct1*cf1-rBPP1(3)*sf1;
C41=rBPP1(1)*st1*sf1+rBPP1(2)*ct1*sf1+rBPP1(3)*cf1;
A45=rBPP5(1)*ct5-rBPP5(2)*st5;
B45=rBPP5(1)*st5*cf5+rBPP5(2)*ct5*cf5-rBPP5(3)*sf5;
C45=rBPP5(1)*st5*sf5+rBPP5(2)*ct5*sf5+rBPP5(3)*cf5;
A51=rAFF1(3)+C31+C41;
B51=rAFF1(2)+B31+B41;
C51=C41*sf1+B41*cf1;
D51=rAFF1(1)+A31+A41;
E51=C31+C41;
F51=B31+B41;
G51=A41*cf1;
H51=A41*sf1;
L51=C41*cf1-B41*sf1;
M51=A31+A41;
N51=C41+C11;
O51=C41*sf1+2*B41*cf1;
T51=B31+B41;
W51=2*C41*sf1+B41*cf1;
A55=rAFF5(3)+C35+C45;
B55=rAFF5(2)+B35+B45;
C55=C45*sf5+B45*cf5;
D55=rAFF5(1)+A35+A45;
E55=C35+C45;
F55=B35+B45;
G55=A45*cf5;
H55=A45*sf5;
L55=C45*cf5-B45*sf5;
M55=A35+A45;
N55=C45+C35;
O55=C45*sf5+2*B45*cf5;
T55=B35+B45;
W55=2*C45*sf5+B45*cf5;
%=============================================================
%===============================================================
%        SECOND MOMENTS OF INERTIA
%===============================================================
% moments of inertia of gimbal, joint and payload wrt. their
%  centers of mass;    D-  stands for Dyadic
%[J1g1,J2g1,J3g1]=cylinder(Mgimbal1,bg1,ag1);
%[J1g2,J2g2,J3g2]=cylinder(Mgimbal2,bg2,ag2);
%[J1sup,J2sup,J3sup]=inertial(Msupport,asup,bsup,csup);
%==============================================================
%[J1jl,J2jl,J3jl]=inertial(Mjointl,ajl,bjl,cjl);
%[J1js,J2js,J3js]=inertial(Mjoints,ajs,bjs,cjs);
%J1=5.0026e-05;J23=1.1574e-04;Ja=1.8145e-04;Jfi=8.0042e-05;
Jcubex=3.04e-4;Jcubey=Jcubex;Jcubez=Jcubex;
Jcollarx=.5*Mcollar*((.5*diamcE)^2+(.5*diamcI)^2);
Jcollary=.5*Mcollar*((.5*diamcE)^2+(.5*diamcI)^2+(hcollar^2)/3);
Jcollarz=Jcollary;
[Daddcol]=addyadic(Mcollar,RcolRcg);
J1js=Jcubex+Jcollarx;J2js=Jcubey+Jcollary;J3js=Jcubez+Jcollarz;
J1jl=Jcubex+2*Jcollarx;J2jl=Jcubey+2*Jcollary;J3jl=Jcubez+2*Jcollarz;
[DJjl] =principal(J1jl,J2jl,J3jl);
[DJjs] =principal(J1js,J2js,J3js);
DJjl=DJjl+2*Daddcol;DJjs=DJjl+Daddcol;
%==============================================================
% inertias of sensors boxes
J1sen=0.00111;J2sen=0.000788;J3sen=0.000696;  % at node 3
J2sen2=7.78e-5;J1sen2=0.000267;J3sen2=J2sen2; % at node 2
J2sen4=8.00e-5;J1sen4=0.000274;J3sen4=J2sen4; % at node 4
%==============================================================
J1pl=2.07e-3;J2pl=1.96e-3;J3pl=1.96e-3;  % rate gyro + payload can
%==============================================================
Jbasex=2.61e-3;Jbasey=4.473-3;Jbasez=2.68e-3;
[DJbase]=principal(Jbasex,Jbasey,Jbasez);
[J1w,J2w,J3w]=inertial(Mwheels,aw,bw,cw);
% axial and transverse inertia of each torque wheel
Jwa1=5.28e-3+3.38e-4;Jwa2=Jwa1;Jwa3=Jwa1;
Jwt1=2.6927e-3+6.67e-4;Jwt2=Jwt1;Jwt3=Jwt1;
%==============================================================
Jbar1=((pA*L)/2.)*(Ro^2-Ri^2);
Jbar2=((pA*L)/12.)*(L^3)+Jbar1/2.;
Jbar3=Jbar2;
%==============================================================
%  these are the central inertia dyadics from the IDEAS model
%  as of April 19, 1991
%==============================================================
DJsupport=[0.0114607       -0.005298906      1.955414e-11;
          -0.005298906      0.02323236      -1.637091e-11;
           1.955414e-11    -1.637091e-11     0.02756642];
DJgimbal1=[0.005146256     -2.557536e-5      9.972862e-5;
          -2.557536e-5      0.01187599      -1.573758e-5;
           9.972862e-5     -1.573758e-5      0.01350181];
DJgimbal2=[0.003440874     -2.585639e-13    -1.756260e-15;
          -2.585639e-13     0.001280201      3.759716e-06;
          -1.756260e-15     3.759716e-06     0.002916165];
[DJgimbal21]=reorient(DJgimbal2,CPA1');
[DJgimbal25]=reorient(DJgimbal2,CPA5');
Jdumx=5.32e-3;Jdumy=2.97e-2;Jdumz=2.90e-2;
[DJdummy]=principal(Jdumx,Jdumy,Jdumz);
[Dadddum]=addyadic(Mdummy,[0;-0.0635;0]);
DJatdum=DJjs+DJdummy+Dadddum;
%==============================================================
% Inertias of triaxial accelerometer packages at node 2 and node 4
DJacc2=principal(J1sen2,J2sen2,J3sen2);
DJacc4=principal(J1sen4,J2sen4,J3sen4);
%==============================================================
[DJsen]=principal(J1sen,J2sen,J3sen);
[DJpl] =principal(J1pl,J2pl,J3pl);
[DJw]  =principal(J1w,J2w,J3w);
[DJw1] =principal(Jwa1,Jwt1,Jwt1);
[DJw2] =principal(Jwa2,Jwt2,Jwt2);
[DJw3] =principal(Jwa3,Jwt3,Jwt3);
```

```
[DJbar]=principal(Jbar1,Jbar2,Jbar3);
%=============================================================
%  inertia of bodies of end nodes 2 and 4 about their com's
%=============================================================
[Daddnodo2]=addyadic(Msensors2,[0;0.06553;0]);
[Daddnodo4]=addyadic(Msensors4,[0;0.06553;0]);
DJ2=Daddnodo2+DJacc2+DJj1;
DJ4=Daddnodo4+DJacc4+DJj1;
%=============================================================
%  inertia of bodies of end nodes 1 or 5 about their com's
%=============================================================
[Daddjs1]=addyadic(Mjoints,rFplusF1-rjFF1);
[Daddjs5]=addyadic(Mjoints,rFplusF5-rjFF5);
[Daddsup1]=addyadic(Msupport,rFplusF1-rSFF1);
[Daddsup5]=addyadic(Msupport,rFplusF5-rSFF5);
[Daddg11] =addyadic(Mgimbal1,rAoAA1-rG1AA1);
[Daddg15] =addyadic(Mgimbal1,rAoAA5-rG1AA5);
[Daddg21]=addyadic(Mgimbal2,-rG2PP1+rBPP1);
[Daddg25]=addyadic(Mgimbal2,-rG2PP5+rBPP5);
[Daddpay1]=addyadic(Mpayload,-rpayPP1+rBPP1);
[Daddpay5]=addyadic(Mpayload,-rpayPP5+rBPP5);

DJbody11=DJsupport+DJjs+Daddjs1+Daddsup1;
DJbody15=DJsupport+DJjs+Daddjs5+Daddsup5;
DJbody21=DJgimbal1+Daddg11;
DJbody25=DJgimbal1+Daddg15;
DJbody31=DJp1+DJgimbal21+Daddg21+Daddpay1;
DJbody35=DJp1+DJgimbal25+Daddg25+Daddpay5;
%=============================================================
%=============================================================
% Inertia of node 3 including the effect of torque wheels
%=============================================================
% wrt. the com. of node 3
DJw1Tr=RwL1*DJw1*RwL1';
DJw2Tr=RwL2*DJw2*RwL2';
DJw3Tr=RwL3*DJw3*RwL3';
[Daddw1]=addyadic(Mw+Mmcase+Mmarmature,rww1-rcmrut);
[Daddw2]=addyadic(Mw+Mmcase+Mmarmature,rww2-rcmrut);
[Daddw3]=addyadic(Mw+Mmcase+Mmarmature,rww3-rcmrut);
[Daddsen3]=addyadic(Msensors3,rcms);
[Daddbase]=addyadic(Mbase,rcmrut);
[Daddj13]=addyadic(Mjoint1,rcmj);
DJsenjoi13=...
  DJj1+DJsen+Daddsen3+Daddw1+Daddw2+Daddw3+DJbase+Daddbase;
Dnode3L3=DJsenjoi13+DJw1Tr+DJw2Tr+DJw3Tr;
%=============================================================
% compute inertia dyadic of MACE about its principal axes
% assumed they originate at node 3
%=============================================================
[Daddnode1body1]=addyadic(Mbody1,rnode1body1);
[Daddnode1body2]=addyadic(Mbody2,rnode1body2);
[Daddnode1body3]=addyadic(Mbody3,rnode1body3);
[Daddnode5body1]=addyadic(Mbody1,rnode5body1);
[Daddnode5body2]=addyadic(Mbody2,rnode5body2);
[Daddnode5body3]=addyadic(Mbody3,rnode5body3);
[Daddnode2]=addyadic(Mnode2,rnode2);
[Daddnode4]=addyadic(Mnode4,rnode4);
[Daddbar1]=addyadic(pA*L,rcombar1);
[Daddbar2]=addyadic(pA*L,rcombar2);
[Daddbar3]=addyadic(pA*L,rcombar3);
[Daddbar4]=addyadic(pA*L,rcombar4);
JMACEcom=Dnode3L3+DJbody11+DJbody21+DJbody15+DJbody25+2*DJj1+4*DJbar;
JMACEadd=Daddnode1body1+Daddnode5body1+Daddnode1body2+...
         Daddnode5body2+Daddnode1body3+Daddnode5body3+...
         Daddnode2+Daddnode4+...
         Daddbar1+Daddbar2+Daddbar3+Daddbar4;
```

```
JMACE=JMACEcom+JMACEadd;
DJ3equiv=Dnode3L3+2*DJj1+4*DJbar+Daddnode4+Daddnode2+...
        Daddbar1+Daddbar2+Daddbar3+Daddbar4;
%==================================================================
% length of each of the two flex cantilevered beams at the sides
% and their elastic constant
%==================================================================
reflen=2*L+3*rR2(1);
flexibx= (G*Jx)/(reflen);
flexiby= (E*Jy)/((reflen));
flexibz= (E*Jz)/((reflen));
%==================================================================
%=========================================================================
% date   : July 24, 1991
% by     : Marco Quadrelli
% note   : normalization with respect to the mass matrix of the
%          modal matrix of an n-th order system
% input  : M and K
% output: evecnorm
%=========================================================================
% Ref.   : Kane,Levinson, "Dynamics: Theory and Applications", 1985
%=========================================================================
function [evecnorm,lamw] = normalizza(M,K)

n = length(M);

% find Cholesky decomposition of M
v = chol(M);

p=inv(v);
W = p'*K*p;

[Cw,lamw] = eig(W,'nobalance');

evecnorm = zeros(n);

for i = 1:n,

        B      = p*Cw(:,i);
        N      = sqrt(B'*M*B);
        A(:,i) = B./N;
        evecnorm(:,i)=A(:,i);
        %evecnorm = [evecnorm A(:,i)];

end
```

```
%=============================================================
%=============================================================
% Author: Marco B. Quadrelli
% Date  : July 23, 1991
% Last revision: October 25, 1991
%=============================================================
% This subroutine appends the geometric
% input data for the 3D MACE test article in the presence of
% suspension wires
%=============================================================
% Assembles the mass matrix of the end nodes (payload + gimbals)
%=============================================================

%=============================================================
% CONFIGURATION DEPENDENT MASS MATRICES
%=============================================================
M1(1:8,1:8)=zeros(8);M5(1:8,1:8)=zeros(8);
%=============================================================
% Assemble mass matrix at node 1  and 5 (see my notes)
% state:  x  y  z  tet1  tet2  tet3  phi1  theta1
% state:  x  y  z  tet1  tet2  tet3  phi5  theta5
%=============================================================
%       Compute P Q R S matrices for M1 and M5
%=============================================================
PP1=zeros(6);QQ1=zeros(6);P1=zeros(7);Q1=zeros(7);R1=zeros(8);S1=zeros(8);
ma1=zeros(6);mb1=zeros(7);mc1=zeros(8);manew1=zeros(8);mbnew1=zeros(8);
PP5=zeros(6);QQ5=zeros(6);P5=zeros(7);Q5=zeros(7);R5=zeros(8);S5=zeros(8);
ma5=zeros(6);mb5=zeros(7);mc5=zeros(8);manew5=zeros(8);mbnew5=zeros(8);
%=============================================================
PP1(1:3,1:3)=eye(3);PP1(4:6,4:6)=scugni1;
PP1(1:3,4:6)=buga1;PP1(4:6,1:3)=PP1(1:3,4:6)';
PP5(1:3,1:3)=eye(3);PP5(4:6,4:6)=scugni5;
PP5(1:3,4:6)=buga5;PP5(4:6,1:3)=PP5(1:3,4:6)';

P1(1,1)=1;P1(2,2)=1;P1(3,3)=1;
P1(4,4)=A21^2+B21^2;
P1(5,5)=A21^2+C21^2;
P1(6,6)=B21^2+C21^2;
P1(7,7)=C11^2+B11^2;
P1(1,5)=A21;P1(5,1)=P1(1,5);
P1(1,6)=-B21;P1(6,1)=P1(1,6);
P1(2,4)=-A21;P1(4,2)=P1(2,4);
P1(2,6)=C21;P1(6,2)=P1(2,6);
P1(2,7)=-C11;P1(7,2)=P1(2,7);
P1(3,4)=B21;P1(4,3)=P1(3,4);
P1(3,5)=-C21;P1(5,3)=P1(3,5);
P1(3,7)=B11;P1(7,3)=P1(3,7);
P1(4,5)=-B21*C21;P1(5,4)=P1(4,5);
P1(4,6)=-A21*C21;P1(6,4)=P1(4,6);
P1(4,7)=A21*C11+B11*B21;P1(7,4)=P1(4,7);
P1(5,6)=-A21*B21;P1(6,5)=P1(5,6);
P1(5,7)=-C21*B11;P1(7,5)=P1(5,7);
P1(6,7)=-C11*C21;P1(7,6)=P1(6,7);

P5(1,1)=1;P5(2,2)=1;P5(3,3)=1;
P5(4,4)=A25^2+B25^2;
P5(5,5)=A25^2+C25^2;
P5(6,6)=B25^2+C25^2;
P5(7,7)=C15^2+B15^2;
P5(1,5)=A25;P5(5,1)=P5(1,5);
P5(1,6)=-B25;P5(6,1)=P5(1,6);
P5(2,4)=-A25;P5(4,2)=P5(2,4);
P5(2,6)=C25;P5(6,2)=P5(2,6);
P5(2,7)=-C15;P5(7,2)=P5(2,7);
P5(3,4)=B25;P5(4,3)=P5(3,4);
P5(3,5)=-C25;P5(5,3)=P5(3,5);
```

```
P5(3,7)=B15;P5(7,3)=P5(3,7);
P5(4,5)=-B25*C25;P5(5,4)=P5(4,5);
P5(4,6)=-A25*C25;P5(6,4)=P5(4,6);
P5(4,7)=A25*C15+B15*B25;P5(7,4)=P5(4,7);
P5(5,6)=-A25*B25;P5(6,5)=P5(5,6);
P5(5,7)=-C25*B15;P5(7,5)=P5(5,7);
P5(6,7)=-C15*C25;P5(7,6)=P5(6,7);

QQ1(4:6,4:6)=DJbody11;
QQ5(4:6,4:6)=DJbody15;

Q1(4:6,4:6)=DJbody21;Q1(7,7)=DJbody21(1,1);
Q1(4,7)=DJbody21(1,1);Q1(7,4)=Q1(4,7);
Q1(5,7)=DJbody21(2,1);Q1(7,5)=Q1(5,7);
Q1(6,7)=DJbody21(1,3);Q1(7,6)=Q1(6,7);

Q5(4:6,4:6)=DJbody25;Q5(7,7)=DJbody25(1,1);
Q5(4,7)=DJbody25(1,1);Q5(7,4)=Q5(4,7);
Q5(5,7)=DJbody25(2,1);Q5(7,5)=Q5(5,7);
Q5(6,7)=DJbody25(1,3);Q5(7,6)=Q5(6,7);

S1(4:6,4:6)=DJbody31;S1(7,7)=DJbody31(1,1);
S1(8,8)=DJbody31(2,2)*sf1^2+DJbody31(3,3)*cf1^2-2*DJbody31(2,3)*sf1*cf1;
S1(4,7)=DJbody31(1,1);S1(7,4)=S1(4,7);
S1(5,7)=DJbody31(2,1);S1(7,5)=S1(5,7);
S1(6,7)=DJbody31(1,3);S1(7,6)=S1(6,7);
S1(4,8)=DJbody31(3,1)*cf1-DJbody31(2,1)*sf1;S1(8,4)=S1(4,8);
S1(5,8)=DJbody31(2,3)*cf1-DJbody31(2,2)*sf1;S1(8,5)=S1(5,8);
S1(6,8)=DJbody31(3,3)*cf1-DJbody31(3,2)*sf1;S1(8,6)=S1(6,8);
S1(7,8)=DJbody31(3,1)*cf1-DJbody31(2,1)*sf1;S1(8,7)=S1(7,8);

S5(4:6,4:6)=DJbody35;S5(7,7)=DJbody35(1,1);
S5(8,8)=DJbody35(2,2)*sf5^2+DJbody35(3,3)*cf5^2-2*DJbody35(2,3)*sf5*cf5;
S5(4,7)=DJbody35(1,1);S5(7,4)=S5(4,7);
S5(5,7)=DJbody35(2,1);S5(7,5)=S5(5,7);
S5(6,7)=DJbody35(1,3);S5(7,6)=S5(6,7);
S5(4,8)=DJbody35(3,1)*cf5-DJbody35(2,1)*sf5;S5(8,4)=S5(4,8);
S5(5,8)=DJbody35(2,3)*cf5-DJbody35(2,2)*sf5;S5(8,5)=S5(5,8);
S5(6,8)=DJbody35(3,3)*cf5-DJbody35(3,2)*sf5;S5(8,6)=S5(6,8);
S5(7,8)=DJbody35(3,1)*cf5-DJbody35(2,1)*sf5;S5(8,7)=S5(7,8);

R1(1,1)=1;R1(2,2)=1;R1(3,3)=1;
R1(4,4)=A51^2+B51^2;
R1(5,5)=A51^2+D51^2;
R1(6,6)=D51^2+B51^2;
R1(7,7)=F51^2+E51^2;
R1(8,8)=C51^2+G51^2+H51^2;
R1(1,5)=A51;R1(5,1)=R1(1,5);
R1(1,6)=-B51;R1(6,1)=R1(1,6);
R1(1,8)=-C51;R1(8,1)=R1(1,8);
R1(2,4)=-A51;R1(4,2)=R1(2,4);
R1(2,6)=D51;R1(6,2)=R1(2,6);
R1(2,7)=-E51;R1(7,2)=R1(2,7);
R1(2,8)=G51;R1(8,2)=R1(2,8);
R1(3,4)=B51;R1(4,3)=R1(3,4);
R1(3,5)=-D51;R1(5,3)=R1(3,5);
R1(3,7)=F51;R1(7,3)=R1(3,7);
R1(3,8)=H51;R1(8,3)=R1(3,8);
R1(4,5)=-B51*D51;R1(5,4)=R1(4,5);
R1(4,6)=-A51*D51;R1(6,4)=R1(4,6);
R1(4,7)=A51*E51+B51*F51;R1(7,4)=R1(4,7);
R1(4,8)=-G51*A51+H51*B51;R1(8,4)=R1(4,8);
R1(5,6)=-A51*B51;R1(6,5)=R1(5,6);
R1(5,7)=-D51*F51;R1(7,5)=R1(5,7);
R1(5,8)=-A51*C51-D51*H51;R1(8,5)=R1(5,8);
R1(6,7)=-D51*E51;R1(7,6)=R1(6,7);
```

```
 R1(6,8)=C51*B51+G51*D51;R1(8,6)=R1(6,8);
 R1(7,8)=-E51*G51+H51*F51;R1(8,7)=R1(7,8);

 R5(1,1)=1;R5(2,2)=1;R5(3,3)=1;
 R5(4,4)=A55^2+B55^2;
 R5(5,5)=A55^2+D55^2;
 R5(6,6)=D55^2+B55^2;
 R5(7,7)=F55^2+E55^2;
 R5(8,8)=C55^2+G55^2+H55^2;
 R5(1,5)=A55;R5(5,1)=R5(1,5);
 R5(1,6)=-B55;R5(6,1)=R5(1,6);
 R5(1,8)=-C55;R5(8,1)=R5(1,8);
 R5(2,4)=-A55;R5(4,2)=R5(2,4);
 R5(2,6)=D55;R5(6,2)=R5(2,6);
 R5(2,7)=-E55;R5(7,2)=R5(2,7);
 R5(2,8)=G55;R5(8,2)=R5(2,8);
 R5(3,4)=B55;R5(4,3)=R5(3,4);
 R5(3,5)=-D55;R5(5,3)=R5(3,5);
 R5(3,7)=F55;R5(7,3)=R5(3,7);
 R5(3,8)=H55;R5(8,3)=R5(3,8);
 R5(4,5)=-B55*D55;R5(5,4)=R5(4,5);
 R5(4,6)=-A55*D55;R5(6,4)=R5(4,6);
 R5(4,7)=A55*E55+B55*F55;R5(7,4)=R5(4,7);
 R5(4,8)=-G55*A55+H55*B55;R5(8,4)=R5(4,8);
 R5(5,6)=-A55*B55;R5(6,5)=R5(5,6);
 R5(5,7)=-D55*F55;R5(7,5)=R5(5,7);
 R5(5,8)=-A55*C55-D55*H55;R5(8,5)=R5(5,8);
 R5(6,7)=-D55*E55;R5(7,6)=R5(6,7);
 R5(6,8)=C55*B55+G55*D55;R5(8,6)=R5(6,8);
 R5(7,8)=-E55*G55+H55*F55;R5(8,7)=R5(7,8);
%===============================================================
%===============================================================
% assemble mass matrices
%===============================================================
balubi=1.e-9*eye(6);baluba=1.e-9*eye(7);balubo=1.e-9*eye(8);
PP1=PP1+balubi;PP5=PP5+balubi;QQ1=QQ1+balubi;QQ5=QQ5+balubi;
P1=P1+baluba;P5=P5+baluba;Q1=Q1+baluba;Q5=Q5+baluba;
R1=R1+balubo;R5=R5+balubo;S1=S1+balubo;S5=S5+balubo;
%===============================================================
for i=1:6,
  for j=1:6,
    ma1(i,j)=Mbody1*PP1(i,j)+QQ1(i,j);
    ma5(i,j)=Mbody1*PP5(i,j)+QQ5(i,j);
  end
end
for i=1:7,
  for j=1:7,
    mb1(i,j)=Mbody2*P1(i,j)+Q1(i,j);
    mb5(i,j)=Mbody2*P5(i,j)+Q5(i,j);
  end
end
for i=1:8,
  for j=1:8,
    mc1(i,j)=Mbody3*R1(i,j)+S1(i,j);
    mc5(i,j)=Mbody3*R5(i,j)+S5(i,j);
  end
end
manew1(1:6,1:6)=ma1(1:6,1:6);
manew5(1:6,1:6)=ma5(1:6,1:6);
mbnew1(1:7,1:7)=mb1(1:7,1:7);
mbnew5(1:7,1:7)=mb5(1:7,1:7);
for i=1:8,
  for j=1:8,
    M1(i,j)=manew1(i,j)+mbnew1(i,j)+mc1(i,j);
    M5(i,j)=manew5(i,j)+mbnew5(i,j)+mc5(i,j);
  end
```

```
end
%================================================================
if pl_locked==1,
   stit=diag(M1);M1=zeros(8);
   M1(1:3,1:3)=(Mdummy+Mjoints)*eye(3);
   M1(4:6,4:6)=DJatdum;
   M1(7:8,7:8)=diag(stit(7:8));
end

%================================================================
% Assemble mass matrix at node 2,3,4
% state: x  y  z  tet1  tet2  tet3  omega1  omega2  omega3
% and refer it to the central node
%================================================================
M3(1:9,1:9)=zeros(9);
[M3diag1]=principal(Mnode3,Mnode3,Mnode3);M3(1:3,1:3)=M3diag1;
[M3diag2]=principal(Jwa1,Jwa2,Jwa3);M3(7:9,7:9)=M3diag2;
M3(4:6,4:6)=Dnode3L3;
M3(4,7)=Jwa1*pw1;M3(5,7)=Jwa1*qw1;M3(6,7)=Jwa1*rw1;
M3(4,8)=Jwa2*pw2;M3(5,8)=Jwa2*qw2;M3(6,8)=Jwa2*rw2;
M3(4,9)=Jwa3*pw3;M3(5,9)=Jwa3*qw3;M3(6,9)=Jwa3*rw3;
M3(7:9,4:6)=M3(4:6,7:9)';
%================================================================
[M2]=massmx(Mnode2,DJ2(1,1),DJ2(2,2),DJ2(3,3));
[M4]=massmx(Mnode4,DJ4(1,1),DJ4(2,2),DJ4(3,3));
```

```
%==============================================================
%==============================================================
%  Program RUOTE.m
%
%  date : July, 23 1991
% author: M.B.Quadrelli
% for   : MACE.m, MACENEW.m, MACESUSP.m,MACE0g.m
%
% computes: precession frequency and entries for gyroscopic matrix
%
%==============================================================
%==============================================================
% components of angular momentum of MACE along b1 b2 and b3
% when wheels are spinning
%==============================================================
% Note:
% total angular momentum of spacecraft (in MACE body axes):
%          Hcomp =
%                = ANGMOM + Sum[i=1:3](DJwi * WHEELSPEED(i)) =
%                = ANGMOM + PITO*WHEELSPEED;
% where WHEELSPEED(i) = i-th wheel ang.velocity relative to  body
% spinMACE = vector of angular velocity in body axes
% ANGMOM   = vector of angular momentum in body axes =
%          = JMACE*spinMACE;
%
%==============================================================
spinMACE=[spinMACE1;spinMACE2;spinMACE3];
ANGMOM=JMACE*spinMACE;
%==============================================================
%==============================================================
%PITO=[Jwa1*pw1  Jwa2*pw2  Jwa3*pw3;
%      Jwa1*qw1  Jwa2*qw2  Jwa3*qw3;
%      Jwa1*rw1  Jwa2*rw2  Jwa3*rw3];
% direction cosines of spin axis of _ith wheel in MACE body axis
a_1=[pw1;qw1;rw1];a_2=[pw2;qw2;rw2];a_3=[pw3;qw3;rw3];
PITO=[pw1  pw2  pw3;qw1  qw2  qw3;rw1  rw2  rw3];
%==============================================================
if choice2==1,
%==============================================================
Hrel=inv(PITO)*(Hcomp-ANGMOM);
WHEELSPEED(1)=Hrel(1)/Jwa1;
WHEELSPEED(2)=Hrel(2)/Jwa2;
WHEELSPEED(3)=Hrel(3)/Jwa3;
spin1=WHEELSPEED(1);spin2=WHEELSPEED(2);spin3=WHEELSPEED(3);
%==============================================================
elseif choice2==2,
%==============================================================
WHEELSPEED=[spin1;spin2;spin3];
%==============================================================
end
%==============================================================
%disp('wheel speeds  #1, #2  and #3 [rpm]');
%disp((WHEELSPEED*rpmm')');
vecspin1=[spin1;0;0];vecspin2=[spin2;0;0];vecspin3=[spin3;0;0];
%==============================================================
% vector angular momentum of ith wheel about its a_ith axis
Hs1=DJw1*vecspin1;Hs2=DJw2*vecspin2;Hs3=DJw3*vecspin3;
Hs_1=Hs1(1);Hs_2=Hs2(1);Hs_3=Hs3(1);
%==============================================================
% resultant angular momentum of the 3 wheels about node_3 body axes
H1=RwL1*Hs1;H2=RwL2*Hs2;H3=RwL3*Hs3;
Hs=H1+H2+H3;Hstot=resultant(Hs);
%==============================================================
% result for N Kelvin's gyrostats
Sum=Hs_1*a_1+Hs_2*a_2+Hs_3*a_3;
Aequiv=(1/Hstot)*Sum;    % direction of equivalent vector h
```

```
%==============================================================
% total angular momentum of MACE about XYZ axes
Htotalvec=ANGMOM+Sum;HtotXY=sqrt(Htotalvec(1)^2+Htotalvec(2)^2);
Htotal=resultant(Htotalvec);
%==============================================================
%==============================================================
% the nutational frequency with flexibility included could be
% something like this:
%                  omega=Hstot/sqrt(pippo)
%where pippo= Dnode3L3(i,i)^2 + (Hstot^2)/(flexibility*Dnode3L3(i,i))
%==============================================================
% estimate nutation (actually, precession)  frequency
%==============================================================
NUTATION_RIGID1=(1/(2*pi))*Hstot/sqrt((JMACE(2,2)*JMACE(3,3)));
NUTATION_RIGID2=(1/(2*pi))*Hstot/sqrt((JMACE(1,1)*JMACE(3,3)));
NUTATION_RIGID3=(1/(2*pi))*Hstot/sqrt((JMACE(1,1)*JMACE(2,2)));
Nutation_rigid=[NUTATION_RIGID1;NUTATION_RIGID2;NUTATION_RIGID3];
Nutrigidres=resultant(Nutation_rigid);
%==============================================================
Nutation_1=...
abs((1/(2*pi))*sqrt(((a_1)'*Dnode3L3*(a_1))/(det(Dnode3L3)))*(Hs_1));
Nutation_2=...
abs((1/(2*pi))*sqrt(((a_2)'*Dnode3L3*(a_2))/(det(Dnode3L3)))*(Hs_2));
Nutation_3=...
abs((1/(2*pi))*sqrt(((a_3)'*Dnode3L3*(a_3))/(det(Dnode3L3)))*(Hs_3));
Vector_of_nut_freq=[Nutation_1;Nutation_2;Nutation_3];
Nutation_frequency=resultant(Vector_of_nut_freq);
%==============================================================
stima1=(1/tupi)*(Hs_1/Jwt1);
stima2=(1/tupi)*(Hs_2/Jwt2);
stima3=(1/tupi)*(Hs_3/Jwt3);
stima=[stima1;stima2;stima3];
stimares=resultant(stima);
%==============================================================
NUTATION_FLEXx =...
abs((1/(2*pi))*Hs_1/sqrt((Dnode3L3(2,2)*Dnode3L3(3,3))));
NUTATION_FLEXy =...
abs((1/(2*pi))*Hs_2/sqrt((Dnode3L3(1,1)*Dnode3L3(3,3))));
NUTATION_FLEXz =...
abs((1/(2*pi))*Hs_3/sqrt((Dnode3L3(1,1)*Dnode3L3(2,2))));
Nutation_flex=[NUTATION_FLEXx;NUTATION_FLEXy;NUTATION_FLEXz];
Nutflexres=resultant(Nutation_flex);
%==============================================================
% add flexibility effect
%==============================================================
pippo= Dnode3L3(1,1)^2 + (Hstot^2)/(flexiby*Dnode3L3(1,1));
omeganutation=Hstot/sqrt(pippo);
OMEGANUT=...
(1/tupi)*sqrt((tupi*Nutation_frequency)^2 +flexibx/Dnode3L3(1,1)...
+ flexiby/Dnode3L3(2,2) + flexibz/Dnode3L3(3,3));
%==============================================================
first=Nutation_frequency*tupi;
second=0.5*flexiby/JMACE(3,3);
omegaprec=(1/tupi)*sqrt(first^2+second);
%==============================================================
show=[Nutation_rigid  Vector_of_nut_freq  Nutation_flex  stima];
showres=[Nutrigidres  Nutation_frequency  Nutflexres   stimares];
%==============================================================
%==============================================================
function pippol(time,tmax,inct,xs,index)

% pippol(tim,Mt,40,xs)
%
% This function plots the time history of the inclination angles
% the links of a deflecting beam. A plot is also shown of the
% successive spatial positions assumed by the beam. In order to
% prevent the plotted positions from overlapping or lying too close
% together,a maximum time tmax can be used as well as an increment
% inct counting the number of increments taken between successive
% positions shown.

% time - the time vector corresponding to different rows in
%        the xs matrix
% theta - xs from MACE0g.m
% L    - the vector containing the lengths of the various links.
% inct - the integer increment used to plot the deflection history.
%        when inct equals one, all positions are shown.
% index- choose projection

% Author: Marco Quadrelli
% Date  : Jan-2-1992

if gravpar==0,
PHI1=xs(:,31);TETA1=xs(:,32);
elseif gravpar==1,
PHI1=xs(:,46);TETA1=xs(:,47);
end
Lt=length(time);
RADp11=rAFF1(1).*ones(Lt,1)+rPAA1(1).*ones(Lt,1)...
       +rBPP1(1).*cos(TETA1)-rBPP1(2).*sin(TETA1);
RADp12=rAFF1(2).*ones(Lt,1)+rPAA1(2).*cos(PHI1)-rPAA1(3).*sin(PHI1)+...
       rBPP1(1).*sin(TETA1).*cos(PHI1)+...
       rBPP1(2).*cos(TETA1).*cos(PHI1)-...
       rBPP1(3).*sin(PHI1);
RADp13=rAFF1(3).*ones(Lt,1)+rPAA1(2).*sin(PHI1)+rPAA1(3).*cos(PHI1)+...
       rBPP1(1).*sin(TETA1).*sin(PHI1)+...
       rBPP1(2).*cos(TETA1).*sin(PHI1)+...
       rBPP1(3).*cos(PHI1);
RADp1=[RADp11 RADp12 RADp13];


itmax=sum(time<=tmax);
RADp11=RADp1(1:inct:itmax,1);
RADp12=RADp1(1:inct:itmax,2);
RADp13=RADp1(1:inct:itmax,3);

theta=xs(1:inct:itmax,:);[mm,nn]=size(theta);xx=zeros(mm,6);
xx1=[theta(:,[1 7 13 19 25]) zeros(mm,1)];
yy =[theta(:,[2 8 14 20 26]) zeros(mm,1)];
zz =[theta(:,[3 9 15 21 27]) zeros(mm,1)];
t1 =theta(:,[4 10 16 22 28]);
t2 =theta(:,[5 11 17 23 29]);
t3 =theta(:,[6 12 18 24 30]);
L1=0.2289*ones(mm,1);rR2=0.0763*ones(mm,1);QL1=L1+2*rR2;

xx(:,1)=xx1(:,1);
xx(:,2)=QL1+xx1(:,1)+xx1(:,2);
xx(:,3)=QL1+xx(:,2)+xx1(:,3);
xx(:,4)=QL1+xx(:,3)+xx1(:,4);
xx(:,5)=QL1+xx(:,4)+xx1(:,5);
xx(:,6)=xx(:,6)+xx(:,5)+RADp11;
yy(:,6)=yy(:,6)+RADp12;
zz(:,6)=zz(:,6)+RADp13;

%xx=cumsum(xx')'; yy=cumsum(yy')';
```

```
xx=[zeros(mm,1),xx]; yy=[zeros(mm,1),yy]; zz=[zeros(mm,1),zz];
clg; axis; hold off

% subplot(221),
%axis('square');
      if index==1,
if gravpar==0,
r=2;axis([-r/100,r,-r/6,r/6]);
elseif gravpar==1,
r=2;axis([-r/100,r,-r,r]);
end
for i=1:mm,
plot(xx(i,:),yy(i,:),'-'); hold on;
plot(xx(i,:),yy(i,:),'o'); hold on;
end
plot(xx(mm,:),yy(mm,:),'-'); hold on,plot(xx(mm,:),yy(mm,:),'o');grid
hold off; axis('normal');
title('y vs. x');

      elseif index==2,
% subplot(222),
%axis('square');
r=2;axis([-r/400,r/2000,-r/8,r/8]);
for i=1:mm,
plot(zz(i,:),yy(i,:),'-'); hold on;plot(zz(i,:),yy(i,:),'o'); hold on;
end
plot(zz(mm,:),yy(mm,:),'-'); hold on,plot(zz(mm,:),yy(mm,:),'o');grid
hold off; axis('normal');
title('y vs. z');

      elseif index==3,
% subplot(223),
%axis('square');
r=2;axis([r/100,r,-r/400,r/2000]);
for i=1:mm,
plot(xx(i,:),zz(i,:),'-'); hold on;plot(xx(i,:),zz(i,:),'o'); hold on;
end
plot(xx(mm,:),zz(mm,:),'-'); hold on,plot(xx(mm,:),zz(mm,:),'o');grid
hold off; axis('normal');
title('z vs. x');

% subplot(111),title('DYNAMICS OF MACE')

     end
```

```
function [glob] = fem(element,number)
[totdof,totdof]=size(element);
nn=totdof/2;
g=(number+1)*nn;
glob=0*eye(g);
a=[eye(totdof),0*ones(totdof,g-totdof)];
for c=[1:1:number]
glob=a'*element*a+glob;
a=[0*ones(totdof,nn)  a(:,[1:(g-nn)])];
end
```

```
function [M1]=massconf(m,J,f1,t1,rAoAA,rAFF,rFplusF,rPAA,rBPP);
%==============================================================
%  author: Marco Quadrelli
%  date   : November 13, 1991
%  done for: MACE
%  revised: November 13,1991
%  Usage:   builds the configuration dependent inertia matrices
%==============================================================
st1=sin(t1);sf1=sin(f1);ct1=cos(t1);cf1=cos(f1);


m1=m(1);
m2=m(2);
m3=m(3);
J1=J(:,1:3);
J2=J(:,4:6);
J3=J(:,7:9);
%==============================================================
[scugnil]=addyadic(1,rFplusF);
[bugal]=transport(rFplusF);

A11=rAoAA(1);
B11=rAoAA(2)*cf1-rAoAA(3)*sf1;
C11=rAoAA(2)*sf1+rAoAA(3)*cf1;
A21=C11+rAFF(3);
B21=B11+rAFF(2);
C21=A11+rAFF(1);
A31=rPAA(1);
B31=rPAA(2)*cf1-rPAA(3)*sf1;
C31=rPAA(2)*sf1+rPAA(3)*cf1;
A41=rBPP(1)*ct1-rBPP(2)*st1;
B41=rBPP(1)*st1*cf1+rBPP(2)*ct1*cf1-rBPP(3)*sf1;
C41=rBPP(1)*st1*sf1+rBPP(2)*ct1*sf1+rBPP(3)*cf1;
A51=rAFF(3)+C31+C41;
B51=rAFF(2)+B31+B41;
C51=C41*sf1+B41*cf1;
D51=rAFF(1)+A31+A41;
E51=C31+C41;
F51=B31+B41;
G51=A41*cf1;
H51=A41*sf1;
L51=C41*cf1-B41*sf1;
M51=A31+A41;
N51=C41+C11;
O51=C41*sf1+2*B41*cf1;
T51=B31+B41;
W51=2*C41*sf1+B41*cf1;
%==============================================================
% Assemble mass matrix at node 1  and 5 (see my notes)
% state:  x  y  z  tet1  tet2  tet3  phi1  theta1
% state:  x  y  z  tet1  tet2  tet3  phi5  theta5
%==============================================================
%       Compute P Q R S matrices for M1 and M5
%==============================================================
PP1=zeros(6);QQ1=zeros(6);P1=zeros(7);Q1=zeros(7);R1=zeros(8);S1=zeros(8);
ma1=zeros(6);mb1=zeros(7);mc1=zeros(8);manew1=zeros(8);mbnew1=zeros(8);
%==============================================================
PP1(1:3,1:3)=eye(3);PP1(4:6,4:6)=scugnil;
PP1(1:3,4:6)=bugal;PP1(4:6,1:3)=PP1(1:3,4:6)';

P1(1,1)=1;P1(2,2)=1;P1(3,3)=1;
P1(4,4)=A21^2+B21^2;
P1(5,5)=A21^2+C21^2;
P1(6,6)=B21^2+C21^2;
P1(7,7)=C11^2+B11^2;
P1(1,5)=A21;P1(5,1)=P1(1,5);
```

```
P1(1,6)=-B21;P1(6,1)=P1(1,6);
P1(2,4)=-A21;P1(4,2)=P1(2,4);
P1(2,6)=C21;P1(6,2)=P1(2,6);
P1(2,7)=-C11;P1(7,2)=P1(2,7);
P1(3,4)=B21;P1(4,3)=P1(3,4);
P1(3,5)=-C21;P1(5,3)=P1(3,5);
P1(3,7)=B11;P1(7,3)=P1(3,7);
P1(4,5)=-B21*C21;P1(5,4)=P1(4,5);
P1(4,6)=-A21*C21;P1(6,4)=P1(4,6);
P1(4,7)=A21*C11+B11*B21;P1(7,4)=P1(4,7);
P1(5,6)=-A21*B21;P1(6,5)=P1(5,6);
P1(5,7)=-C21*B11;P1(7,5)=P1(5,7);
P1(6,7)=-C11*C21;P1(7,6)=P1(6,7);

QQ1(4:6,4:6)=J1;

Q1(4:6,4:6)=J2;Q1(7,7)=J2(1,1);
Q1(4,7)=J2(1,1);Q1(7,4)=Q1(4,7);
Q1(5,7)=J2(2,1);Q1(7,5)=Q1(5,7);
Q1(6,7)=J2(1,3);Q1(7,6)=Q1(6,7);

S1(4:6,4:6)=J3;S1(7,7)=J3(1,1);
S1(8,8)=J3(2,2)*sf1^2+J3(3,3)*cf1^2-2*J3(2,3)*sf1*cf1;
S1(4,7)=J3(1,1);S1(7,4)=S1(4,7);
S1(5,7)=J3(2,1);S1(7,5)=S1(5,7);
S1(6,7)=J3(1,3);S1(7,6)=S1(6,7);
S1(4,8)=J3(3,1)*cf1-J3(2,1)*sf1;S1(8,4)=S1(4,8);
S1(5,8)=J3(2,3)*cf1-J3(2,2)*sf1;S1(8,5)=S1(5,8);
S1(6,8)=J3(3,3)*cf1-J3(3,2)*sf1;S1(8,6)=S1(6,8);
S1(7,8)=J3(3,1)*cf1-J3(2,1)*sf1;S1(8,7)=S1(7,8);

R1(1,1)=1;R1(2,2)=1;R1(3,3)=1;
R1(4,4)=A51^2+B51^2;
R1(5,5)=A51^2+D51^2;
R1(6,6)=D51^2+B51^2;
R1(7,7)=F51^2+E51^2;
R1(8,8)=C51^2+G51^2+H51^2;
R1(1,5)=A51;R1(5,1)=R1(1,5);
R1(1,6)=-B51;R1(6,1)=R1(1,6);
R1(1,8)=-C51;R1(8,1)=R1(1,8);
R1(2,4)=-A51;R1(4,2)=R1(2,4);
R1(2,6)=D51;R1(6,2)=R1(2,6);
R1(2,7)=-E51;R1(7,2)=R1(2,7);
R1(2,8)=G51;R1(8,2)=R1(2,8);
R1(3,4)=B51;R1(4,3)=R1(3,4);
R1(3,5)=-D51;R1(5,3)=R1(3,5);
R1(3,7)=F51;R1(7,3)=R1(3,7);
R1(3,8)=H51;R1(8,3)=R1(3,8);
R1(4,5)=-B51*D51;R1(5,4)=R1(4,5);
R1(4,6)=-A51*D51;R1(6,4)=R1(4,6);
R1(4,7)=A51*E51+B51*F51;R1(7,4)=R1(4,7);
R1(4,8)=-G51*A51+H51*B51;R1(8,4)=R1(4,8);
R1(5,6)=-A51*B51;R1(6,5)=R1(5,6);
R1(5,7)=-D51*F51;R1(7,5)=R1(5,7);
R1(5,8)=-A51*C51-D51*H51;R1(8,5)=R1(5,8);
R1(6,7)=-D51*E51;R1(7,6)=R1(6,7);
R1(6,8)=C51*B51+G51*D51;R1(8,6)=R1(6,8);
R1(7,8)=-E51*G51+H51*F51;R1(8,7)=R1(7,8);
%==============================================================
%==============================================================
% assemble mass matrices
%==============================================================
balubi=1.e-9*eye(6);baluba=1.e-9*eye(7);balubo=1.e-9*eye(8);
PP1=PP1+balubi;QQ1=QQ1+balubi;
P1=P1+baluba;Q1=Q1+baluba;
R1=R1+balubo;S1=S1+balubo;
```

```
%=====================================================================
for i=1:6,
  for j=1:6,
    ma1(i,j)=m1*PP1(i,j)+QQ1(i,j);
  end
end
for i=1:7,
  for j=1:7,
    mb1(i,j)=m2*P1(i,j)+Q1(i,j);
  end
end
for i=1:8,
  for j=1:8,
    mc1(i,j)=m3*R1(i,j)+S1(i,j);
  end
end
manew1(1:6,1:6)=ma1(1:6,1:6);
mbnew1(1:7,1:7)=mb1(1:7,1:7);
for i=1:8,
  for j=1:8,
    M1(i,j)=manew1(i,j)+mbnew1(i,j)+mc1(i,j);
  end
end
```

```
function xprime = MACE0g(t,x)
%
%--------------------------------------------------------------
%                MACE 3-D model in zero-g
%
% Author       : Marco B. Quadrelli
% Date         : October 17, 1991
% Last revision: November 26, 1991
%--------------------------------------------------------------
% Notes : This version considers the
%         (support + inner stage + outer stage + payload)
%         as a chain of three bodies connected by two 1dof
%         revolute joints.
%         The three bodies are:
%         - joint + support
%         - gimbal 1
%         - gimbal 2 + payload.
%         To date, 29-Aug-1991, the program works only when
%         nelew=1 and nelef=-1(no flexible appendage yet), but
%         nele can be chosen to be any number.
%         The central node is allowed to store a finite amount of
%         internal angular momentum.
%         All units in S.I.
%--------------------------------------------------------------
%
%          torque  -> /   \                0  -> hinged payload
%          wheels    /  \ / \             /
%        1=======2=======3=======4=======5/
%       /                              \
%     /                                 \--> spacecraft bus
%   0
%==============================================================
% the state vector is:
%_______________________________________________________________
%                  description                   +    d.o.f.
%________________________________________________+______________
%  X = [  (x,y,z,theta1,theta2,theta3)_1         |          1:ndof-6
%         (x,y,z,theta1,theta2,theta3)_2         |      ndof-5:ndof
%         (x,y,z,theta1,theta2,theta3)_3         |      ndof+1:2*ndof-6
%         (x,y,z,theta1,theta2,theta3)_4         |    2*ndof-5:3*ndof-12
%         (x,y,z,theta1,theta2,theta3)_5         |   3*ndof-11:ntotflex
%         (phi,theta)_#1                         |     ntot-3:ntot-2
%         (phi,theta)_#2                         |     ntot-1:ntot
%         (omega1,omega2,omega3)_wheels          |     ntot+1:ntot+3
%         -------------------------------|------------------------
%         and all the first derivatives  |    ntot+19:2*(ntot+18)]
%         -------------------------------|------------------------
%==============================================================
L  = 0.22886;     % length of Lexan member
rjFF1=[0;0;0];rjFF5=[0;0;0];


 u4=x(2*ndof-8+states);        %theta1_dot (central node)
 u5=x(2*ndof-7+states);        %theta2_dot      "
 u6=x(2*ndof-6+states);        %theta3_dot       "
 if p1_locked==0,
  u7=x(ntot+16);           %omega1 for 1st wheel
  u8=x(ntot+17);           %omega2 for 2nd wheel
  u9=x(ntot+18);           %omega3 for 3rd wheel
 elseif p1_locked==1,
  u7=x(ntot+14);           %omega1 for 1st wheel
  u8=x(ntot+15);           %omega2 for 2nd wheel
  u9=x(ntot+16);           %omega3 for 3rd wheel
 end
 t11d=x(ndof-8+states);        %theta1_dot node 1
 t21d=x(ndof-7+states);        %theta2_dot   "
 t31d=x(ndof-6+states);      %theta3_dot   "
 t15d=x(3*ndof-8+states);    %theta1_dot node 5
 t25d=x(3*ndof-7+states);    %theta2_dot   "
 t35d=x(ntotflex+states);    %theta3_dot   "

if p1_locked==0,

 f1=x(ntot-3);              %fi payload #1
 t1=x(ntot-2);              %theta    "
 f5=x(ntot-1);              %fi payload #5
 t5=x(ntot);                %theta    "
 f1d=x(ntot-3+states);      %fi_dot payload #1
 t1d=x(ntot-2+states);      %theta_dot     "
 f5d=x(ntot-1+states);      %fi_dot payload #5
 t5d=x(ntot+states);        %theta_dot    "

 t1_inertial   = t1+x(6);
 f1_inertial   = f1+x(4);
 t5_inertial   = t5+x(30);
 f5_inertial   = f5+x(28);
 t1d_inertial   = t1+x(6+states);
 f1d_inertial   = f1+x(4+states);
 t5d_inertial   = t5+x(30+states);
 f5d_inertial   = f5+x(28+states);

elseif p1_locked==1,

 f5=x(ntot-1);              %fi payload #5
 t5=x(ntot);                %theta    "
 f5d=x(ntot-1+states);      %fi_dot payload #5
 t5d=x(ntot+states);        %theta_dot    "

 t5_inertial    = t5+x(30);
 f5_inertial    = f5+x(28);
 t5d_inertial   = t5+x(30+states);
 f5d_inertial   = f5+x(28+states);


end


f1=0;t1=0;
phi1=f1;theta1=t1;phi5=f5;theta5=t5;
%==============================================================
Fnl=zeros(states,1);Fnltrue=zeros(states,1);
FnlKane=zeros(states,1);Fnldes=zeros(states,1);
Fact=zeros(states,1);Factuator=zeros(inputs,1);

                if p1_locked==0,

%  q1 = out-of-plane angle phi    q2 = in-plane angle theta
[q1des1,q1desd1,q1desdd1,tau_reffi]=traject(t,fi0,fif,0.0295);
[q2des1,q2desd1,q2desdd1,tau_refte]=traject(t,tet0,tetf,0.0430);

[q1des5,q1desd5,q1desdd5,tau_reffi]=traject(t,fi0,fif,0.0295);
[q2des5,q2desd5,q2desdd5,tau_refte]=traject(t,tet0,tetf,0.0430);

                elseif p1_locked==1,

%  q1 = out-of-plane angle phi    q2 = in-plane angle theta
[q1des5,q1desd5,q1desdd5,tau_reffi]=traject(t,fi0,fif,0.0295);
[q2des5,q2desd5,q2desdd5,tau_refte]=traject(t,tet0,tetf,0.0430);

                end

%==============================================================
[sf1,cf1,st1,ct1,ct1cf1,st1sf1,st1cf1,sf1ct1,st1ct1,sf1cf1,...
        ct1sf1,cf1st1,ct1st1,cf1sf1]=sintax(theta1,phi1);
[sf5,cf5,st5,ct5,ct5cf5,st5sf5,st5cf5,sf5ct5,st5ct5,sf5cf5,...
        ct5sf5,cf5st5,ct5st5,cf5sf5]=sintax(theta5,phi5);
%==============================================================
% derive transformation matrices between frames
%==============================================================
% transformation matrix from Fpivot to Fpayload (F=vectrix)
[R2p11,R2p21,R2p31,R2p1]=rotation(phi1,0,theta1);
[R2p15,R2p25,R2p35,R2p5]=rotation(phi5,0,theta5);
CAF1=R2p11;CAF5=R2p15;
CPA1=R2p31;CPA5=R2p35;
%==============================================================
po1= rpayPP1(1);po5=rpayPP5(1);
p1=CAF1*CPA1*rpayPP1;
p5=CAF5*CPA5*rpayPP5;
%==============================================================
% vector from FEM node to (body 1 + body 2) com  in F frame
%==============================================================
rAoF1=rAFF1+CAF1*rAoAA1;
rAoF5=rAFF5+CAF5*rAoAA5;
%==============================================================
% vector from FEM node to payload com in F frame
%==============================================================
rPF1=rAFF1+CAF1*rPAA1+CPA1*CAF1*rBPP1;
rPF5=rAFF5+CAF5*rPAA5+CPA5*CAF5*rBPP5;
%==============================================================
% center of mass of (body 1+ body 2 +body 3) wrt FEM node in F frame
%==============================================================
rcom1=(1/Mnode1)*(Mbody1*rFplusF1+Mbody2*rAoF1+Mbody3*rPF1);
rcom5=(1/Mnode5)*(Mbody1*rFplusF5+Mbody2*rAoF5+Mbody3*rPF5);
%==============================================================
%==============================================================
%        SECOND MOMENTS OF INERTIA
%==============================================================
[DJgimbal21]=reorient(DJgimbal2,CPA1');
[DJgimbal25]=reorient(DJgimbal2,CPA5');
%==============================================================
% inertia of bodies of end nodes 1 or 5 about their com's
DJbody11=DJsupport+DJjs+Daddjs1+Daddsup1;
DJbody15=DJsupport+DJjs+Daddjs5+Daddsup5;
DJbody21=DJgimbal1+Daddg11;
DJbody25=DJgimbal1+Daddg15;
DJbody31=DJpl+DJgimbal21+Daddg21+Daddpay1;
DJbody35=DJpl+DJgimbal25+Daddg25+Daddpay5;
%==============================================================
ma1=[Mbody1;Mbody2;Mbody3];ma5=ma1;
J1=[DJbody11  DJbody21  DJbody31];
J5=[DJbody15  DJbody25  DJbody35];
%==============================================================
%==============================================================
% assemble mass matrices
%==============================================================
%
% Note: Mass # 1 has been substituted with a dummy gimbal
%       on October 30, 1991
%==============================================================
M1(1:8,1:8)=zeros(8);M5(1:8,1:8)=zeros(8);
%==============================================================
% Assemble mass matrix at node 1  and 5 (see my notes)
% state:  x  y  z  tet1  tet2  tet3  phi1  theta1
% state:  x  y  z  tet1  tet2  tet3  phi5  theta5
%==============================================================
rAoAA=rAoAA1;rAFF=rAFF1;rFplusF=rFplusF1;rPAA=rPAA1;rBPP=rBPP1;
[M1]=massconf(ma1,J1,phi1,theta1,rAoAA,rAFF,rFplusF,rPAA,rBPP);
rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
```

```
[M5]=massconf(ma5,J5,phi5,theta5,rAoAA,rAFF,rFplusF,rPAA,rBPP);
%===============================================================
if p1_locked==1,
   stit=diag(M1);M1=zeros(8);
   M1(1:3,1:3)=(Mdummy+Mjoints)*eye(3);
   M1(4:6,4:6)=DJatdum;
   M1(7:8,7:8)=diag(stit(7:8));
end
%===============================================================
%===============================================================
%===============================================================
%===============================================================
% Before assembling the flex elements, move rows and columns 7,8
% of M1 to 1st and 2nd place so that the state vector for
% the first node becomes [phi1,theta1,x1,y1,z1,d1,d2,d3]'
%===============================================================
M1=M1([7:8 1:6],:);M1=M1(:,[7:8 1:6]);
%===============================================================


%===============================================================
%
%          FINITE ELEMENT MODEL
%
%===============================================================
%
% Start assembling all flexible parts together
%
%===============================================================
KK=zeros(nfree);MM=zeros(nfree);T=eye(ndof);
KKflex=zeros(nflex);MMflex=zeros(nflex);
KKflexbus=zeros(ntotflex);MMflexbus=zeros(ntotflex);
MMint=zeros(nfree);KKint=zeros(nfree);pu=length(MMint);
DA=zeros(nfree);GYRO=zeros(nfree);DAMP=zeros(nfree);
GYRODAMP=zeros(nfree);DAMPREDUCED=zeros(nflex);
%===============================================================
% A) assemble flexibility of bus
%    (can accept more than one nele elements)
%===============================================================
T(1:6,1:6)=TR1;T(ndof-5:ndof,ndof-5:ndof)=TL2;
nbeg=1;nend=ndof;
KKflexbus(nbeg:nend,nbeg:nend)=T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=T'*mflex*T;

T(1:6,1:6) = TR2;T(ndof-5:ndof,ndof-5:ndof) = TL3;
nbeg=ndof-5;nend=2*ndof-6;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

T(1:6,1:6) = TR3;T(ndof-5:ndof,ndof-5:ndof) = TL2;
nbeg=2*ndof-11;nend=3*ndof-12;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

T(1:6,1:6)=TR2;T(ndof-5:ndof,ndof-5:ndof)=TL5;
nbeg=3*ndof-17;nend=ntotflex;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

KKflex(1:ntotflex,1:ntotflex)=KKflexbus;
```

```
MMflex(1:ntotflex,1:ntotflex)=MMflexbus;
%=============================================================
% augment to include end payloads and wheels
KK(3:nfree-5,3:nfree-5)=KKflex;
MM(3:nfree-5,3:nfree-5)=MMflex;
%=============================================================
% Lump additional mass into finite element model
%=============================================================
% a) first lump spacecraft mass at nodes 1 to 5
MM(1:8,1:8)=MM(1:8,1:8)+M1;

nbeg=ndof-3;nend=ndof+2;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M2;

nbeg=2*ndof-9;nend=2*ndof-4;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M3new;

nbeg=3*ndof-15;nend=3*ndof-10;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M4;

nbeg=4*ndof-21;nend=ntot;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M5;
%=============================================================
% arrange mass and stiffness matrices in a form ready for
% integration, i.e. move rigid body dof's after flex. dof's:
% state vector =
%  (flex. coord's + phi1,theta1,phi5,theta5,rate1,rate2,rate3)'
%         payload    (     #1    ) (     #2    )
%                                torque wheel    (#1)  (#2)  (#3)
%=============================================================
MMint=zeros(nfree);KKint=zeros(nfree);pu=length(MMint);
DA=zeros(nfree);GYRO=zeros(nfree);DAMP=zeros(nfree);
GYRODAMP=zeros(nfree);DAMPREDUCED=zeros(nflex);
%=============================================================
% move payloads dof's after flex dof's
%=============================================================
nbeg=[3:(nflex+2) 1 2 (nflex+3) (nflex+4)];nend=nbeg;
MMintn=MM(nbeg,nend);
KKintn=KK(nbeg,nend);
%=============================================================
% add wheels dof's
KKint=[KKintn                          zeros(length(KKintn),3);
       zeros(3,length(KKintn))         zeros(3)];
MMint=[MMintn                          zeros(length(MMintn),3);
       zeros(3,length(MMintn))         M3diag2];

nbeg=[(ntotflex/2+1):(ntotflex/2+3)];
nend=[(nfree-2):nfree];
MMint(nbeg,nend)=M3(4:6,7:9);

nbeg=[(nfree-2):nfree];
nend=[(ntotflex/2+1):(ntotflex/2+3)];
MMint(nbeg,nend)=M3(4:6,7:9)';
%=============================================================
% clamp payload #1 only on full model
%=============================================================
clamp=[1:30 33:pu];
KKintc=KKint(clamp,clamp);
MMintc=MMint(clamp,clamp);
puc=length(MMintc);
%=============================================================
Mee=MMint(1:nflex,1:nflex);Kee=KKint(1:nflex,1:nflex);
if pl_locked==0,
  Mrr=MMint(nflex+1:pu,nflex+1:pu);
  Mer=MMint(1:nflex,nflex+1:pu);Mre=Mer';
  Mew=MMint(1:nflex,pu-2:pu);Mwe=Mew';
  Mww=MMint(pu-2:pu,pu-2:pu);
 elseif pl_locked==1,
  Mrr=MMintc(nflex+1.puc,nflex+1:puc);
  Mer=MMintc(1:nflex,nflex+1:puc);Mre=Mer';
  Mew=MMintc(1:nflex,puc-2:puc);Mwe=Mew';
  Mww=MMintc(puc-2:puc,puc-2:puc);
 end
 %==============================================================
 % Now introduce damping and Gyroscopic matrices
 % Also introduce geometric stiffness matrix due to g load
 %==============================================================
 DA=zeros(nfree);GYRO=zeros(nfree);DAMP=zeros(nfree);
 GYRODAMP=zeros(nfree);DAMPREDUCED=zeros(nflex);
 %==============================================================
 % The torque wheels contribute with zero stiffness. But they
 % contribute with a gyroscopic matrix GY. Assemble GY.
 %==============================================================
 spin1=u7;spin2=u8;spin3=u9;
 APw1=spin1*Jwa1*Aw1;BPw1=spin1*Jwa1*Bw1;CPw1=spin1*Jwa1*Cw1;
 APw2=spin2*Jwa2*Aw2;BPw2=spin2*Jwa2*Bw2;CPw2=spin2*Jwa2*Cw2;
 APw3=spin3*Jwa3*Aw3;BPw3=spin3*Jwa3*Bw3;CPw3=spin3*Jwa3*Cw3;
 [gyroblock]=skew([APw1+APw2+APw3;BPw1+BPw2+BPw3;CPw1+CPw2+CPw3]);
 GYRO(2*ndof-8:2*ndof-6,2*ndof-8:2*ndof-6)=gyroblock;
 GYROpic=zeros(KKintn);
 GYROpic(2*ndof-8:2*ndof-6,2*ndof-8:2*ndof-6)=gyroblock;
 GYROint=[GYROpic                     zeros(length(KKintn),3);
          zeros(3,length(KKintn))     zeros(3)];
 GYROintc=GYROint(clamp,clamp);
 %==============================================================
 % add proportional damping
 %==============================================================
 %alfa=.10;beta=1.e-5;
 %DAMPREDUCED=alfa*MMflex+beta*KKflex;
 %DAMPint=[DAMPREDUCED                 zeros(length(KKflex),7);
 %         zeros(7,length(KKflex))     zeros(7)];
 %==============================================================
 if pl_locked==0,
        GYRODAMPint=GYROint+DAMPMAT;
 elseif pl_locked==1,
        GYRODAMPintc=GYROintc+DAMPMAT;
 end
 %==============================================================
 %                       END ASSEMBLY
 %==============================================================



 % PART F: append vector of non-linear terms due to payload motion,
 %         i.e., Coriolis and centripetal terms


 %==============================================================
 %         build the non-linear terms in the equations
 %         of motion for MACENEW.m
 % NOTE:   both payloads must be at least hinged
 %==============================================================
 %==============================================================
 % compute the non-linear terms for the central node
 %==============================================================
 SUM1=u4*pw1+u5*qw1+u6*rw1+2*u7;
 SUM2=u4*pw2+u5*qw2+u6*rw2+2*u8;
 SUM3=u4*pw3+u5*qw3+u6*rw3+2*u9;
 SUM41=u5*Aw1+u6*Bw1;
 SUM42=u5*Aw2+u6*Bw2;
 SUM43=u5*Aw3+u6*Bw3;
```

```
SUM51=-u4*Aw1+u6*Cw1;
SUM52=-u4*Aw2+u6*Cw2;
SUM53=-u4*Aw3+u6*Cw3;
SUM61=-u4*Bw1-u5*Cw1;
SUM62=-u4*Bw2-u5*Cw2;
SUM63=-u4*Bw3-u5*Cw3;
ADD11=(Jwa1-Jwt1)*SUM41*SUM1;
ADD12=(Jwa2-Jwt2)*SUM42*SUM2;
ADD13=(Jwa3-Jwt3)*SUM43*SUM3;
ADD21=(Jwa1-Jwt1)*SUM51*SUM1;
ADD22=(Jwa2-Jwt2)*SUM52*SUM2;
ADD23=(Jwa3-Jwt3)*SUM53*SUM3;
ADD31=(Jwa1-Jwt1)*SUM61*SUM1;
ADD32=(Jwa2-Jwt2)*SUM62*SUM2;
ADD33=(Jwa3-Jwt3)*SUM63*SUM3;
SUM4=ADD11+ADD12+ADD13;
SUM5=ADD21+ADD22+ADD23;
SUM6=ADD31+ADD32+ADD33;
%===========================================================
% for the central node (Coriolis and centripetal)
%===========================================================
if flexibility==0,

 Fnl(2*ndof-8:2*ndof-6,1)=zeros(3,1);

elseif flexibility==1,

 Fnl(2*ndof-8,1)=...
      DJ(1,3)*u4*u5+DJ(2,3)*(u5^2-u6^2)+u5*u6*(DJ(3,3)-DJ(2,2))...
      -DJ(1,2)*u4*u6+SUM4;
 Fnl(2*ndof-7,1)=...
      DJ(1,2)*u5*u6+DJ(1,3)*(u6^2-u4^2)+u4*u6*(DJ(1,1)-DJ(3,3))...
      -DJ(2,3)*u4*u5+SUM5;
 Fnl(2*ndof-6,1)=...
      DJ(2,3)*u4*u6+DJ(1,2)*(u4^2-u5^2)+u4*u5*(DJ(2,2)-DJ(1,1))...
      -DJ(1,3)*u5*u6+SUM6;

end
%===========================================================
% for the end nodes (Coriolis and centripetal), non-linear terms in
% payload angle and rates
%===========================================================

                        if pl_locked==0,
%===========================================================
% payload #1
%===========================================================
  dai1=[1:ndof-6  ntot-3:ntot-2];
rAoAA=rAoAA1;rAFF=rAFF1;rFplusF=rFplusF1;rPAA=rPAA1;rBPP=rBPP1;
[Fnltrue(dai1,1),verf1,verfg1]=...
fzt(ma1,J1,t11d,t21d,t31d,f1,t1,f1d,t1d,rAoAA,rAFF,rFplusF,rPAA,rBPP);

[Fnl(dai1,1),verf1,verfg1]=...
fz(ma1,J1,0,0,0,f1,t1,f1d,t1d,rAoAA,rAFF,rFplusF,rPAA,rBPP);

[FnlKane(dai1,1),Fnlrot1,Fnltrans1]=...
Fnonlin(ma1,J1,t11d,t21d,t31d,f1,t1,f1d,t1d,rAoAA,rAFF,rFplusF,rPAA,rBPP);
%===========================================================
% payload #2
%===========================================================
  dai2=[3*ndof-11:ntotflex  ntot-1:ntot];
rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
[Fnltrue(dai2,1),verf5,verfg5] =...
fzt(ma5,J5,t15d,t25d,t35d,f5,t5,f5d,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);

[Fnl(dai2,1),verf5,verfg5] =...
```

```
fz(ma5,J5,0,0,0,f5,t5,f5d,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);

[FnlKane(dai2,1),Fnlrot5,Fnltrans5]=...
Fnonlin(ma5,J5,t15d,t25d,t35d,f5,t5,f5d,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);
%=================================================================
                           elseif p1_locked==1,
%=================================================================
% payload #2
%=================================================================
  dai2=[3*ndof-11:ntotflex  ntot-3:ntot-2];
rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
[Fnltrue(dai2,1),verf5,verfg5] =...
fzt(ma5,J5,t15d,t25d,t35d,f5,t5,f5d,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);

[Fnl(dai2,1),verf5,verfg5] =...
fz(ma5,J5,0,0,0,f5,t5,f5d,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);

[FnlKane(dai2,1),Fnlrot5,Fnltrans5]=...
Fnonlin(ma5,J5,t15d,t25d,t35d,f5,t5,f5d,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);
%=================================================================
                           end

%=================================================================
FnlKane(2*ndof-8:2*ndof-6,1)=Fnl(2*ndof-8:2*ndof-6,1);
%=================================================================


% PART G: append vector of external forces: non-linear + actuation
%          (not linearized)

% compute desired non-linear terms

                          if p1_locked==0,
%=================================================================
% payload #1
%=================================================================
f1des=q1des1;
f1ddes=q1desd1;
f1dddes=q1desdd1;
t1des=q2des1;
t1ddes=q2desd1;
t1dddes=q2desdd1;

  dai1=[1:ndof-6  ntot-3:ntot-2];
rAoAA=rAoAA1;rAFF=rAFF1;rFplusF=rFplusF1;rPAA=rPAA1;rBPP=rBPP1;
[Fnldes(dai1,1),verf1des,verfg1des]=...
fz(ma1,J1,0,0,0,f1des,t1des,f1ddes,t1ddes,rAoAA,rAFF,rFplusF,rPAA,rBPP);

qdes1=[q1des1;q2des1];
qdesd1=[q1desd1;q2desd1];
qdesdd1=[q1desdd1;q2desdd1];
q1=x(31:32,1);
qdot1=x((31+states):(32+states),1);
qtilde1=q1-qdes1;
qtilded1=qdot1-qdesd1;
%=================================================================
% payload #2
%=================================================================
f5des=q1des5;
f5ddes=q1desd5;
f5dddes=q1desdd5;
t5des=q2des5;
t5ddes=q2desd5;
t5dddes=q2desdd5;

  dai2=[3*ndof-11:ntotflex  ntot-1:ntot];
```

```
 rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
 [Fnldes(dai2,1),verf5des,verfg5des] =...
 fz(ma5,J5,0,0,0,f5des,t5des,f5ddes,t5ddes,rAoAA,rAFF,rFplusF,rPAA,rBPP);

 qdes5=[q1des5;q2des5];
 qdesd5=[q1desd5;q2desd5];
 qdesdd5=[q1desdd5;q2desdd5];
 q5=x(33:34,1);
 qdot5=x((33+states):(34+states),1);
 qtilde5=q5-qdes5;
 qtilded5=qdot5-qdesd5;

 rAoAA=rAoAA1;rAFF=rAFF1;rFplusF=rFplusF1;rPAA=rPAA1;rBPP=rBPP1;
 [M1des]=...
   massconf(ma1,J1,q1des1,q2des1,rAoAA,rAFF,rFplusF,rPAA,rBPP);
 rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
 [M5des]=...
   massconf(ma5,J5,q1des5,q2des5,rAoAA,rAFF,rFplusF,rPAA,rBPP);
 M1d=M1des(7:8,7:8);
 M5d=M5des(7:8,7:8);
 %===============================================================
                          elseif pl_locked==1,
 %===============================================================
 % payload #2
 %===============================================================
 f5des=q1des5;
 f5ddes=q1desd5;
 f5dddes=q1desdd5;
 t5des=q2des5;
 t5ddes=q2desd5;
 t5dddes=q2desdd5;

   dai2=[3*ndof-11:ntotflex  ntot-3:ntot-2];
 rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
 [Fnldes(dai2,1),verf5des,verfg5des] =...
 fz(ma5,J5,0,0,0,f5des,t5des,f5ddes,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);

 qdes5=[q1des5;q2des5];
 qdesd5=[q1desd5;q2desd5];
 qdesdd5=[q1desdd5;q2desdd5];
 q5=x(31:32,1);
 qdot5=x((31+states):(32+states),1);
 qtilde5=q5-qdes5;
 qtilded5=qdot5-qdesd5;

 rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
 [M5des]=...
   massconf(ma5,J5,q1des5,q2des5,rAoAA,rAFF,rFplusF,rPAA,rBPP);
 M5d=M5des(7:8,7:8);
 %===============================================================
                          end

 %===============================================================
 %      Factuator(1,1) =   out-of-plane torque
 %      Factuator(2,1) =   in-plane torque
 %===============================================================

                     if pl_locked==0,
 %===============================================================
 KPfi=0;KDfi=0;KPtheta=1000;KDtheta=100;
 Factuator(1,1)=-KPfi*qtilde1(1)-KDfi*qtilded1(1);
 Factuator(2,1)=-KPtheta*qtilde1(2)-KDtheta*qtilded1(2);
 Factuator(3,1)=-KPfi*qtilde5(1)-KDfi*qtilded5(1);
 Factuator(4,1)=-KPtheta*qtilde5(2)-KDtheta*qtilded5(2);
 %===============================================================
 Fact(31,1)=Factuator(1,1);
```

```
Fact(32,1)=Factuator(2,1);
Fact(33,1)=Factuator(3,1);
Fact(34,1)=Factuator(4,1);
Fact(35,1)=0;
Fact(36,1)=0;
Fact(37,1)=0;

vec1=x((states+1):(2*states),1);        % vector of velocities
invM=inv(MMint);
vectors1=Fact-Fnltrue;
vectors2=-KKint*x(1:states,1)-GYRODAMPint*x((states+1):(2*states),1);
vector=vectors1+vectors2;
vec2=invM*vector;

% inertial accelerations
 t1dd_inertial   = vec2(6)+vec2(32);
 f1dd_inertial   = vec2(4)+vec2(31);
 t5dd_inertial   = vec2(30)+vec2(34);
 f5dd_inertial   = vec2(28)+vec2(33);

xprime = [vec1;vec2];
%==============================================================

                  elseif p1_locked==1,

%==============================================================
% inertial angular velocities, as seen by the rate gyro
invel(1)=qtilded5(1)+x(28+states);
invel(2)=qtilded5(2)+x(30+states);
%==============================================================
% Note: gains computed assuming the payload to be an independent
%       rigid body
% i.e. neglect Mre*qdd_e in 'r' equations
%
% out-of-plane low-bandwidth control
%  bandwidth = 1/10 of 1st flex mode (at 1.6328 Hz)=.16328*tupi rad/s
%  inertia about X = 0.0295 Kgm2
%  zeta = 0.7071
%                KPfi=0.0310;KDfi=0.0428;
% out-of-plane high-bandwidth control
%  bandwidth = 10 times 1st flex mode (at 1.6328 Hz)=16.328*tupi rad/s
%  inertia about X = 0.0295 Kgm2
%  zeta = 0.7071
%                KPfi=310.4901;KDfi=4.28;
% in-plane low-bandwidth control
%  bandwidth = 1/10 of 1st flex mode (at 1.6328 Hz)=.16328*tupi rad/s
%  inertia about Z = 0.0430 Kgm2
%  zeta = 0.7071
%                KPtheta=0.0453;KDtheta=0.0624;
% in-plane high-bandwidth control
%  bandwidth = 10 times 1st flex mode (at 1.6328 Hz)=16.328*tupi rad/s
%  inertia about Z = 0.0430 Kgm2
%  zeta = 0.7071
%                KPtheta=452.5788;KDtheta=6.2387;
% gains:
%
%  KP=J*(2*pi*bandwidth)^2              KD=2*zeta*sqrt(J*KP)
%==============================================================
if HBwth==1,
 KPfi=310.4901;KDfi=4.28;
 KPtheta=452.5788;KDtheta=6.2387;
end
if LBwth==1,
 KPfi=0.0310;KDfi=0.0428;
 KPtheta=0.0453;KDtheta=0.0624;
end
```

```
KP=[KPfi 0;0 KPtheta];
KD=[KDfi 0;0 KDtheta];

tau=[tau_reffi;tau_refte];
Jiner=[0.0295  0;0  0.0430];
FDBK=-KP*qtilde5-KD*qtilded5;      % feedback controller (relative)
FFWD1=tau;                         % feedforward controller #1
FFWD2=Fnl(7:8);                    % feedforward controller #2
FFWD=FFWD1+FFWD2;
%==============================================================
% Include non-linear friction/stiction in bearings
%==============================================================
%coef1=20;       %[oz.in]
%coef2=0.001;    %[oz.in/rad/sec]
%Fstiction(1,1)=coef1*sign(x(31+states))+coef2*(x(31+states));
%Fstiction(2,1)=coef1*sign(x(32+states))+coef2*(x(32+states));
%==============================================================
%==============================================================
%  List of runs
%==============================================================
% this was implemented on the file simulazione.mat (Nov-15-91)
%     i.e. bang-bang torque plus PD term
% note that we are feeding back the relative angle and rate
%Factuator(1,1)=0;
%Factuator(2,1)=tau_refte-KPtheta*(q5(2)-qref)-KDtheta*(qdot5(2)-qrefd);
%==============================================================
% this was implemented on the file simulazione0.mat (Nov-18-91)
%     i.e. simple position control using PD and trajectory=6
% note that we are feeding back the relative angle and rate
%Factuator(1,1)=0;
%Factuator(2,1)=-KPtheta*qtilde5(2)-KDtheta*qdot5(2);
%==============================================================
% this was implemented on the file simulazione2.mat (Nov-20-91)
%     i.e. bang-bang torque plus PD term
% note that we are feeding back the relative angle and rate
%Factuator(1,1)=0;
%Factuator(2,1)=tau_ref;%-KPtheta*(q5(2)-qref)-KDtheta*(qdot5(2)-qrefd);
%==============================================================
% this was implemented on the file simulazione3.mat (Nov-26-91)
% simple position control on trajectory=#7 using PD law
% note that we are feeding back the relative angle and inertial rate
%Factuator(1,1)=0;
%Factuator(2,1)=-KPtheta*qtilde5(2)-KDtheta*invel(2);
%==============================================================
% this was implemented on the file simulazione4.mat (Nov-29-91)
% simple position control on trajectory=#7 using PD law
% note that we are feeding back the relative angle and rate
%Factuator(1,1)=0;
%Factuator(2,1)=tau_refte-KPtheta*qtilde5(2)-KDtheta*invel(2);
%==============================================================
% this was implemented on the file simulazione7.mat (Dec-22-91)
% simple trajectory control on trajectory=#7 using
% high-bandwidth PD law and feedforward
% note that we are feeding back the relative angle and rate
% and we cancel non-linearities

Factuator(1,1)=0;
Factuator(2,1)=tau_refte-KPtheta*qtilde5(2)-KDtheta*qtilded5(2)+...
               Fnl(32,1);

%==============================================================
%Factuator(1:2,1)=FDBK+FFWD;
%==============================================================
% wheel torques: low BW control and gyroscopic coupling cancellation
%  see also Wie,Weiss and Arapostathis,:
%  "Eigenaxis Rotational Maneuver via Quaternion Feedback"
% JMACE is computed for 0-90 0-90 configuration
%===================================================================
kgain = 2*(2*pi*1.638/10)^2;
dgain = 2*0.707*(2*pi*1.638/10);
JMACE =[0.3971    0.0008    0.0002;
        0.0008   15.3131   -0.0024;
        0.0002   -0.0024   14.0386];
Jgain=diag((diag((JMACE))));
KPw1= kgain*Jgain;KDw1=dgain*Jgain;
Kwh=0*ones(3,nflex);Kwh(:,16:18)=eye(3);
KPw=KPw1*Kwh;KDw=KDw1*Kwh;
%===================================================================
Tratt=[pw1 pw2 pw3;qw1 qw2 qw3;rw1 rw2 rw3];
invTratt=inv(Tratt);
Rtrans = 0*ones(nflex,3);
Rtrans(2*ndof-8:2*ndof-6,:) = -Tratt;
TTwheels= Rtrans - Mew*inv(Mww);
[U,S,V]=svd(TTwheels);
% TTwheels = U(1:30,1:3)*S(1:3,:)*V';
invTT = V*inv(S(1:3,:))*U(1:30,1:3)';

FDBKw = -KPw*x(1:nflex,1)-...
         KDw*x((states+1):(states+nflex),1);
tauW = FDBKw +...
  invTT*GYRODAMPintc(1:nflex,1:nflex)*x((states+1):(states+nflex),1);

tauw1=tauW(1);
tauw2=tauW(2);
tauw3=tauW(3);
taubody=-Tratt*[tauw1;tauw2;tauw3];
%===================================================================
Fact(16:18,1)=taubody;
%Fact(16,1)=-tauw1*pw1-tauw2*pw2-tauw3*pw3;
%Fact(17,1)=-tauw1*qw1-tauw2*qw2-tauw3*qw3;
%Fact(18,1)=-tauw1*rw1-tauw2*rw2-tauw3*rw3;
Fact(31,1)=Factuator(1,1);
Fact(32,1)=Factuator(2,1);
%rand('normal');
Fact(33,1)=tauw1;    % rand(sin(1));
Fact(34,1)=tauw2;    % rand(sin(1));
Fact(35,1)=tauw3;    % rand(sin(1));

vec1=x((states+1):(2*states),1);       % vector of velocities
invM=inv(MMintc);
vectors1=Fact-FnlKane;
vectors2=-KKintc*x(1:states,1)-GYRODAMPintc*x((states+1):(2*states),1);
vector=vectors1+vectors2;
vec2=invM*vector;

% inertial accelerations
 t5dd_inertial   = vec2(30)+vec2(32);
 f5dd_inertial   = vec2(28)+vec2(31);

xprime = [vec1;vec2];
%===================================================================

                    end

%===================================================================

if see_inside==1,
  t,format long,[x(1:states) FnlKane Fact],format
end
```

```
function [q1des,q1desd,q1desdd,tau_ref]=traject(t,angle0,anglef,J)

% builds the trajectories which the payload in MACE must track
% The payload is assumed to be an independent rigid body, with
%  no coupling to the flexibility
% done by Marco Quadrelli
% all variables in rad.
tupi=2*pi;

if trajectory==1,                         % sharp step
    q1des=anglef;q1desd=0;q1desdd=0;tau_ref=q1desdd*J;
elseif trajectory==2,                     % cosine
    q1des=-anglef*(1-cos(tupi*t));
    q1desd=-anglef*tupi*(sin(tupi*t));
    q1desdd=tupi*tupi*(-anglef)*cos(tupi*t);
    tau_ref=q1desdd*J;
elseif trajectory==3,                     % sine
    q1des=anglef*sin(tupi*t);
    q1desd=anglef*cos(tupi*t);
    q1desdd=-anglef*sin(tupi*t);
    tau_ref=q1desdd*J;
elseif trajectory==4,                     % exponential step
    q1des=anglef*(1-exp(-4*t));
    q1desd=4*anglef*exp(-4*t);
    q1desdd=-anglef*16*exp(-4*t);
    tau_ref=q1desdd*J;
elseif trajectory==5,                     % quintic polynomial
    teta0=angle0;tetad0=0;tetadd0=0;
    tetaf=anglef;tetadf=0;tetaddf=0;
    t0=0;tf=timtstop;T=timtstop;
    [A1]=quintic(t0,tf,teta0,tetad0,tetadd0,tetaf,tetadf,tetaddf);
    a0=A1(1);a1=A1(2);a2=A1(3);a3=A1(4);a4=A1(5);a5=A1(6);
    if t <= T,
      q1des  =a0+a1*t+a2*(t^2)+a3*(t^3)+a4*(t^4)+a5*(t^5);
      q1desd =a1+2*a2*t+3*a3*(t^2)+4*a4*(t^3)+5*a5*(t^4);
      q1desdd=2*a2+6*a3*t+12*a4*(t^2)+20*a5*(t^3);
    elseif t > T,
      q1des=anglef;q1desd=0;q1desdd=0;
    end
    tau_ref=q1desdd*J;

elseif trajectory==6,                     % smooth (sine) step
    q1des0=angle0;
    q1desT=anglef;
    T=timtstop;
    maxvel=((q1desT-q1des0)/(T));
    if t <= T,
      q1des=q1des0+maxvel*(t-(T/tupi)*sin(tupi*t./T));
      q1desd=maxvel*(1-cos(tupi*t./T));
      q1desdd=maxvel*(T/tupi)*sin(tupi*t./T);
    elseif t > T,
      q1des=anglef;q1desd=0;q1desdd=0;
    end
    tau_ref=q1desdd*J;

elseif trajectory==7,                     % reference MACE maneuver

    T=timtstop;
    [q1des,q1desd,q1desdd]=rampvel(t,T,angle0,anglef);
    tau_ref=q1desdd*J;

elseif trajectory==8,                     % Lissajous figure

% specify frequency Liss in Hz
    q1des=sin(tupi*Liss*t);
    q1desd=tupi*Liss*cos(tupi*Liss*t);
    q1desdd=-((tupi*Liss)^2)*sin(tupi*Liss*t);
    tau_ref=q1desdd*J;

elseif trajectory==9,                  % Bang-Bang smooth slew

    tau_max=0.2150*.9;    % take only 90% of Max. value used in slew
    alfa=.25/3;           % shaping parameter
    [smoothing]=smooth(t,alfa,timtstop);
    if anglef >= angle0,
      tau_ref = +tau_max*smoothing;
    elseif anglef <= angle0,
      tau_ref = -tau_max*smoothing;
    end
    % assume reference bang-bang maneuver is computed neglecting flexibility

t1=T/2;
t2=T;

    q1desdd=(tau_ref/J);
    q1desd=q1desdd*t;
    q1des=.5*q1desdd*(t^2)+angle0;

end


return


trajectory=7;timtstop=2.88;
Q=[];P=[];R=[];
for v=0:0.01:4,
[q1des,q1desd,q1desdd,tau_ref]=traject(v,-150*pi/180,-30*pi/180,0.0430);
Q=[Q q1des];P=[P q1desd];R=[R q1desdd];
end
v=0:0.01:4;v=v';
subplot(221),plot(v,Q'*180/pi),grid,title('position vs. time')
subplot(222),plot(v,P'*180/pi),grid,title('velocity vs. time')
subplot(223),plot(v,R'*180/pi),grid,title('acceleration vs. time')
subplot(111)
plot(v,Q*180/pi),grid
```

```
function [q1des,q1desd,q1desdd]=rampvel(t,T,q1des0,q1desf)
%=============================================================
% builds the velocity profile of a trajectory which must
% reach 50 deg/s within the first 10 deg, then coast at
% constant rate and eventually slow down to zero rate.
% Total excursion: 120 degrees.
% Uses quintic polynomials in time: they are 3 matched at two
% different instants of time, t1, t2. Note that if you change the
% end angular position, the new trajectory time (time_new)
% is given by:
%
% (theta_initial - theta_final_old)/(theta_initial - theta_final_new) =
%  = time_old/time_new
%
% and therefore q1desf and T change accordingly.
%
% by: Marco Quadrelli on December 20, 1991
%=============================================================
tupi=2*pi;t0=0;t1=(1/6)*T;t2=T-(1/6)*T;t3=T;limit=50*pi/180;
q1des1=q1des0+10*pi/180;q1des2=q1des0+10*pi/180+limit*(t2-t1);
q1desd0=0;q1desdf=0;
[A1]=quintic(t0,t1,q1des0,0,0,q1des1,limit,0);
[A2]=quintic(t2,t3,q1des2,limit,0,q1desf,0,0);
A3(1)=q1des2;A3(2)=limit-A1(2);
A3(3)=-A1(3);A3(4)=-A1(4);A3(5)=-A1(5);A3(6)=-A1(6);
%=============================================================
if t < t1,
%=============================================================
% step # 1: velocity ramp to 50 deg/sec from t0 to t1
q1des  = A1(1)+A1(2)*t+A1(3)*(t^2)+A1(4)*(t^3)+A1(5)*(t^4)+A1(6)*(t^5);
q1desd = A1(2)+2*A1(3)*(t)+3*A1(4)*(t^2)+4*A1(5)*(t^3)+5*A1(6)*(t^4);
q1desdd= 2*A1(3)+6*A1(4)*(t)+12*A1(5)*(t^2)+20*A1(6)*(t^3);
%=============================================================
elseif t >= t1,
%=============================================================
    if t < t2,
%=============================================================
% step # 2: velocity at 50 deg/sec from t1 to t2
q1des  = q1des1+limit*(t-t1);q1desd = limit;q1desdd= 0;
%=============================================================
    elseif t >= t2,
%=============================================================
        if t < t3,
%=============================================================
% step # 3: velocity ramp from 50 deg/sec to zero from t2 to t3
t=t-t2;
q1des  = A3(1)+A3(2)*t+A3(3)*(t^2)+A3(4)*(t^3)+A3(5)*(t^4)+A3(6)*(t^5);
q1desd = A3(2)+2*A3(3)*(t)+3*A3(4)*(t^2)+4*A3(5)*(t^3)+5*A3(6)*(t^4);
q1desdd= 2*A3(3)+6*A3(4)*(t)+12*A3(5)*(t^2)+20*A3(6)*(t^3);
%=============================================================
        elseif t >= t3,
%=============================================================
q1desd=0;q1desdd=0;q1des=q1desf;
%=============================================================
        end
    end
end


return


trajectory=7;timtstop=2.8856;Q=[];P=[];R=[];
for v=0:0.01:4,
[q1des,q1desd,q1desdd]=rampvel(v,timtstop,-150*pi/180,-30*pi/180);
Q=[Q;q1des];P=[P;q1desd];R=[R;q1desdd];
```

```
end
v=[0:0.01:4]';
subplot(221),plot(v,Q*180/pi),grid,title('position vs. time')
subplot(222),plot(v,P*180/pi),grid,title('velocity vs. time')
subplot(223),plot(v,R*180/pi),grid,title('acceleration vs. time')
subplot(111)
plot(v,Q*180/pi),grid
```

```
function [A1]=quintic(t0,tfi,teta0,tetad0,tetadd0,tetaf,tetadf,tetaddf)
%==============================================================
% builds the coefficients of a quintic polynomial in time.
% by: Marco Quadrelli on November 22, 1991
%==============================================================
tf=tfi-t0;
a0=teta0;
a1=tetad0;
a2=tetadd0/2;

a3=(20*(tetaf-teta0)-tfi*(8*tetadf+12*tetad0)...
   -(tfi^2)*(3*tetadd0-tetaddf))/(2*(tfi^3));

a4=(30*(teta0-tetaf)+tfi*(14*tetadf+16*tetad0)...
   +(tfi^2)*(3*tetadd0-2*tetaddf))/(2*(tfi^4));

a5=(12*(tetaf-teta0)-tfi*(6*tetadf+6*tetad0)...
   -(tfi^2)*(tetadd0-tetaddf))/(2*(tfi^5));


%tf=tfi-t0;
%Q=[tf^3  tf^4  tf^5;
%   3*tf^2  4*tf^3  5*tf^4;
%   6*tf  12*tf^2  20*tf^3];
%Qinv=inv(Q);
%a345=Qinv*[tetaf-teta0-tetad0*tf-.5*tetadd0*tf^2;
%           tetadf-tetad0-tetadd0*tf;
%           tetaddf-tetadd0];
%a3=a345(1);a4=a345(2);a5=a345(3);

A1=[a0 a1 a2 a3 a4 a5];
```

```
function [Fnl,verifica,verificag]=...
     fz(m,J,t1d,t2d,t3d,f,t,fd,td,rAoAA,rAFF,rFplusF,rPAA,rBPP);
%===============================================================
%  author: Marco Quadrelli
%  date  : May 20, 1991
%  done for: MACE, MACENEW and MACESUSP
%  revised: November 13,1991
%===============================================================
%===============================================================
% Usage:
%         builds the non-linear terms in the equations
%         of motion for MACE.m
%         The addition of the g_ij terms makes the equations
%         fully non-linear in theta_dot, phi_dot, theta1_dot,
%         theta2_dot and theta3_dot
%===============================================================
st=sin(t);sf=sin(f);ct=cos(t);cf=cos(f);
tds=td^2;fds=fd^2;
t1ds=t1d^2;t2ds=t2d^2;t3ds=t3d^2;

m1=m(1);
m2=m(2);
m3=m(3);
J1=J(:,1:3);
J2=J(:,4:6);
J3=J(:,7:9);
A1=rAoAA(1);
B1=rAoAA(2)*cf-rAoAA(3)*sf;
C1=rAoAA(2)*sf+rAoAA(3)*cf;
A2=C1+rAFF(3);
B2=B1+rAFF(2);
C2=A1+rAFF(1);
A3=rPAA(1);
B3=rPAA(2)*cf-rPAA(3)*sf;
C3=rPAA(2)*sf+rPAA(3)*cf;
A4=rBPP(1)*ct-rBPP(2)*st;
B4=rBPP(1)*st*cf+rBPP(2)*ct*cf-rBPP(3)*sf;
C4=rBPP(1)*st*sf+rBPP(2)*ct*sf+rBPP(3)*cf;
A5=rAFF(3)+C3+C4;
B5=rAFF(2)+B3+B4;
C5=C4*sf+B4*cf;
D5=rAFF(1)+A3+A4;
E5=C3+C4;
F5=B3+B4;
G5=A4*cf;
H5=A4*sf;
L5=C4*cf-B4*sf;
M5=A3+A4;
N5=C3+C4;
O5=C4*sf+2*B4*cf;
T5=B3+B4;
W5=2*C4*sf+B4*cf;
AA1=C2;BB1=A2;CC1=B2;
RT1=D5;RT2=B5;RT3=A5;

%===============================================================
% Computation of angular velocities
%===============================================================
WFN=[t1d;t2d;t3d];
WAN=[t1d+fd;t2d;t3d];
WPN=[t1d+fd;t2d-td*sf;t3d+td*cf];
%===============================================================
% Computation of partial velocities
%===============================================================
% rotational
```

```
wfn1=[0;0;0];
wfn2=[0;0;0];
wfn3=[0;0;0];
wfn4=[1;0;0];
wfn5=[0;1;0];
wfn6=[0;0;1];
wfn=[wfn1;wfn2;wfn3;wfn4;wfn5;wfn6];

wan1=[0;0;0];
wan2=[0;0;0];
wan3=[0;0;0];
wan4=[1;0;0];
wan5=[0;1;0];
wan6=[0;0;1];
wan7=[1;0;0];
wan=[wan1;wan2;wan3;wan4;wan5;wan6;wan7];

wpn1=[0;0;0];
wpn2=[0;0;0];
wpn3=[0;0;0];
wpn4=[1;0;0];
wpn5=[0;1;0];
wpn6=[0;0;1];
wpn7=[1;0;0];
wpn8=[0;-sf;cf];
wpn=[wpn1;wpn2;wpn3;wpn4;wpn5;wpn6;wpn7;wpn8];

% translational

vfn1=[1;0;0];
vfn2=[0;1;0];
vfn3=[0;0;1];
vfn4=[0;-rFplusF(3);rFplusF(2)];
vfn5=[rFplusF(3);0;-rFplusF(1)];
vfn6=[-rFplusF(2);rFplusF(1);0];
vfn=[vfn1;vfn2;vfn3;vfn4;vfn5;vfn6];

van1=[1;0;0];
van2=[0;1;0];
van3=[0;0;1];
van4=[0;-A2;B2];
van5=[A2;0;-C2];
van6=[-B2;C2;0];
van7=[0;-C1;B1];
van=[van1;van2;van3;van4;van5;van6;van7];

vbn1=[1;0;0];
vbn2=[0;1;0];
vbn3=[0;0;1];
vbn4=[0;-A5;B5];
vbn5=[A5;0;-D5];
vbn6=[-B5;D5;0];
vbn7=[0;-E5;F5];
vbn8=[-C5;G5;H5];
vbn=[vbn1;vbn2;vbn3;vbn4;vbn5;vbn6;vbn7;vbn8];

%================================================================
% compute angular acceleration remainder terms and the torques
%  note that I have deleted the contribution of the flexibility
%
%================================================================
alfafnt=[0;0;0];
alfaant=[0;0;0];
alfapnt=[0;-fd*td*cf;-fd*td*sf];

[qwq1]=dotdyad(J1,WFN);
```

```
[qwq2]=dotdyad(J2,WAN);
[qwq3]=dotdyad(J3,WPN);

[qaq1]=dotdyad(J1,alfafnt);
[qaq2]=dotdyad(J2,alfaant);
[qaq3]=dotdyad(J3,alfapnt);

[cross1]=cross(WFN,qwq1);
[cross2]=cross(WAN,qwq2);
[cross3]=cross(WPN,qwq3);

T1tstar=cross1+qaq1;
T2tstar=cross2+qaq2;
T3tstar=cross3+qaq3;

%================================================================
% compute linear acceleration remainder terms and the forces
%  note that I have deleted the contribution of the flexibility
%
%================================================================
afnt=[0;0;0];
aant=[0;-B1*fds;-C1*fds];
abnt=[-A4*tds;-fds*F5-tds*cf*ct-2*A4*fd*td*sf;...
      -fds*E5-tds*C5*sf+2*A4*td*fd*cf];
F1tstar=m1*afnt;
F2tstar=m2*aant;
F3tstar=m3*abnt;
%================================================================
%================================================================

%================================================================
%  Compute non-linear terms depending on theta1_dot,
%  theta2_dot and theta3_dot
%================================================================
if keepgs==1,

% body 1

g11=m1*(t1d*t2d*rFplusF(2)-t2ds*rFplusF(1)-t3ds*rFplusF(1)+...
        rFplusF(3)*t1d*t3d);
g12=m1*(t2d*t3d*rFplusF(3)-t3ds*rFplusF(2)-t1ds*rFplusF(2)+...
        rFplusF(1)*t1d*t2d);
g13=m1*(t1d*t3d*rFplusF(1)-t1ds*rFplusF(3)-t2ds*rFplusF(3)+...
        rFplusF(2)*t2d*t3d);
g14=J1(3,1)*t1d*t2d+J1(3,2)*t2ds+J1(3,3)*t2d*t3d-J1(2,1)*t1d*t3d-...
    J1(2,2)*t2d*t3d-J1(2,3)*t3ds;
g15=J1(1,1)*t1d*t3d+J1(1,3)*t3ds+J1(1,2)*t2d*t3d-J1(3,2)*t1d*t2d-...
    J1(3,3)*t1d*t3d-J1(3,1)*t1ds;
g16=J1(2,2)*t1d*t2d+J1(2,1)*t1ds+J1(2,3)*t1d*t3d-J1(1,1)*t1d*t2d-...
    J1(1,3)*t2d*t3d-J1(1,2)*t2ds;

% body 2

g21=m2*(t1d*t2d*CC1-t2ds*AA1-t3ds*AA1+BB1*t1d*t3d);
g22=m2*(t2d*t3d*BB1-t3ds*CC1-t1ds*CC1+AA1*t1d*t2d);
g23=m2*(t1d*t3d*AA1-t1ds*BB1-t2ds*BB1+CC1*t2d*t3d);
g24=J2(3,1)*t1d*t2d+J2(3,2)*t2ds+J2(3,3)*t2d*t3d-J2(2,1)*t1d*t3d-...
    J2(2,2)*t2d*t3d-J2(2,3)*t3ds;
g25=J2(1,1)*t1d*t3d+J2(1,3)*t3ds+J2(1,2)*t2d*t3d-J2(3,2)*t1d*t2d-...
    J2(3,3)*t1d*t3d-J2(3,1)*t1ds;
g26=J2(2,2)*t1d*t2d+J2(2,1)*t1ds+J2(2,3)*t1d*t3d-J2(1,1)*t1d*t2d-...
    J2(1,3)*t2d*t3d-J2(1,2)*t2ds;
g27=g24;

% body 3
```

```
g31=m3*(t1d*t2d*RT2-t2ds*RT1-t3ds*RT1+RT3*t1d*t3d);
g32=m3*(t2d*t3d*RT3-t3ds*RT2-t1ds*RT2+RT1*t1d*t2d);
g33=m3*(t1d*t3d*RT1-t1ds*RT3-t2ds*RT3+RT2*t2d*t3d);
g34=J3(3,1)*t1d*t2d+J3(3,2)*t2ds+J3(3,3)*t2d*t3d-J3(2,1)*t1d*t3d-...
    J3(2,2)*t2d*t3d-J3(2,3)*t3ds;
g35=J3(1,1)*t1d*t3d+J3(1,3)*t3ds+J3(1,2)*t2d*t3d-J3(3,2)*t1d*t2d-...
    J3(3,3)*t1d*t3d-J3(3,1)*t1ds;
g36=J3(2,2)*t1d*t2d+J3(2,1)*t1ds+J3(2,3)*t1d*t3d-J3(1,1)*t1d*t2d-...
    J3(1,3)*t2d*t3d-J3(1,2)*t2ds;

elseif keepgs==0,

     g11=0;g12=0;g13=0;g14=0;g15=0;g16=0;
     g21=0;g22=0;g23=0;g24=0;g25=0;g26=0;g27=0;
     g31=0;g32=0;g33=0;g34=0;g35=0;g36=0;

end

%===========================================================


                 if flexibility==1,


%===========================================================

%===========================================================
%  translational equations
%===========================================================

% body 1

F11=g11;
F12=g12;
F13=g13;

% body 2

F21=m2*(B1*t2d*fd+C1*t3d*fd)+g21;
F22=m2*(-B1*fd^2-2*B1*fd*t1d+A1*fd*t2d)+g22;
F23=m2*(-C1*fd^2-2*C1*t1d*fd+A1*fd*t3d)+g23;

% body 3

F31=m3*(t2d*fd*F5+t3d*fd*E5+t1d*td*L5+2*td*t2d*H5-2*td*t3d*G5-...
    A4*tds)+g31;
F32=m3*(-fd^2*F5-2*F5*fd*t1d+fd*t2d*M5-2*td*fd*H5-td^2*cf*C5-...
    td*t1d*H5+td*t2d*C4*cf-td*t3d*O5)+g32;
F33=m3*(-fd^2*E5-2*fd*t1d*E5+t3d*fd*M5+2*fd*td*G5+td*t1d*G5-...
    td^2*sf*C5+td*t2d*W5-B4*sf*td*t3d)+g33;

%===========================================================
%  rotational equations
%===========================================================

% body 1

F14=(-rFplusF(3)*g12+rFplusF(2)*g13)+g14;
F15=(rFplusF(3)*g11-rFplusF(1)*g13)+g15;
F16=(-rFplusF(2)*g11+rFplusF(1)*g12)+g16;


% body 2

F24=-A2*F22+B2*F23-J2(2,1)*fd*t3d+J2(3,1)*fd*t2d+g24;
F25=A2*F21-C2*F23+(J2(1,1)-J2(3,3))*fd*t3d-2*J2(3,1)*fd*t1d-...
```

```
    J2(3,1)*fd^2-J2(3,2)*t2d*fd+g25;
F26=-B2*F21+C2*F22+2*J2(2,1)*t1d*fd+J2(2,1)*fd^2+...
    (J2(2,2)-J2(1,1))*t2d*fd+J2(2,3)*t3d*fd+g26;
F27=-C1*F22+B1*F23+J2(3,1)*fd*t2d-J2(2,1)*fd*t3d+g27;


% body 3

F34=-A5*F32+B5*F33-J3(1,3)*(td*fd*sf+t1d*td*sf-t2d*fd+td*fd*sf)+...
     J3(3,2)*(td^2*sf^2-2*td*t2d*sf)+(J3(3,3)-J3(2,2))*...
     (-td*t3d*sf+td*t2d*cf-td^2*sf*cf)-J3(2,1)*(t1d*td*cf+...
     fd*t3d+2*td*fd*cf)-J3(2,3)*(td^2*cf^2+2*td*t3d*cf) + g34;
F35=A5*F31-D5*F33-J3(2,3)*td*fd*sf+(J3(1,1)-J3(3,3))*...
    (t1d*td*cf+fd*t3d+td*fd*cf)+J3(1,2)*(td*t2d*cf-td*t3d*sf-...
    td^2*sf*cf)+J3(1,3)*(td^2*cf^2+2*td*t3d*cf)-J3(3,1)*...
    (fd^2+2*fd*t1d)-J3(3,2)*(t2d*fd-td*t1d*sf-td*fd*sf)-...
    J3(2,2)*td*fd*cf + g35;
F36=-B5*F31+D5*F32-J3(3,3)*td*fd*sf+J3(2,1)*(fd^2+2*fd*t1d)...
    +(J3(2,2)-J3(1,1))*(t2d*fd-td*t1d*sf-td*fd*sf)+J3(2,3)*...
    (td*t1d*cf+fd*t3d+td*fd*cf)-J3(1,2)*(td^2*sf^2-2*td*t2d*sf)...
    -J3(1,3)*(td*t2d*cf-td*t3d*sf-td^2*sf*cf)-...
    J3(3,2)*td*fd*cf + g36;

%============================================================
%  payload angle equations
%============================================================

F37=-E5*F32+F5*F33-J3(1,3)*td*fd*sf+J3(1,3)*...
     (t2d*fd-t1d*td*sf-td*fd*sf)+J3(3,2)*(td^2*sf^2-2*td*t2d*sf)...
     +(J3(3,3)-J3(2,2))*(td*t2d*cf-td*t3d*sf-td^2*sf*cf)-J3(2,1)...
     *(td*t1d*cf+fd*t3d+td*fd*cf)-J3(2,3)*(td^2*cf^2+2*td*t3d*cf)-...
     J3(1,2)*td*fd*cf + g34;
F38=-C5*F31+G5*F32+H5*F33+...
     sf*(J3(2,2)*td*fd*cf+J3(2,3)*td*fd*sf)-...
     cf*(J3(3,2)*td*fd*cf+J3(3,3)*td*fd*sf)-...
  sf*( (J3(1,1)-J3(3,3))*(td*t1d*cf+fd*t3d+td*fd*cf)+J3(1,2)*...
   (td*t2d*cf-td*t3d*sf-td^2*sf*cf)+J3(1,3)*(td^2*cf^2+2*td*t3d*cf)-...
   J3(3,1)*(fd^2+2*t1d*fd)-J3(3,2)*(t2d*fd-td*t1d*sf-td*fd*sf)+g35 )+...
  cf*( J3(2,1)*(fd^2+2*fd*t1d)+(J3(2,2)-J3(1,1))*...
   (t2d*fd-td*t1d*sf-td*fd*sf)+J3(2,3)*(td*t1d*cf+fd*t3d+td*fd*cf)...
   -J3(1,2)*(td^2*sf^2-2*td*t2d*sf)-J3(1,3)*...
   (td*t2d*cf-td*t3d*sf-td^2*sf*cf)+g36 );

%===================================== =======================
%  assemble generalized inertia forces due to non-linear terms
%============================================================
Fnl(1,1)=F11+F21+F31;
Fnl(2,1)=F12+F22+F33;
Fnl(3,1)=F13+F23+F33;
Fnl(4,1)=F14+F24+F34;
Fnl(5,1)=F15+F25+F35;
Fnl(6,1)=F16+F26+F36;
Fnl(7,1)=F27+F37;
Fnl(8,1)=F38;
%============================================================


                    elseif flexibility==0,


%============================================================

%============================================================
%  translational equations
%============================================================
% body 1

F11=m1*g11*0;
F12=m1*g12*0;
F13=m1*g13*0;

% body 2

F21=m2*(B1*t2d*fd+C1*t3d*fd)*0;
F22=m2*(-B1*fd^2);
F23=m2*(-C1*fd^2);

% body 3

F31=m3*(-A4*td^2);
F32=m3*(-fd^2*F5-2*td*fd*H5-td^2*cf*C5);
F33=m3*(-fd^2*E5+2*fd*td*G5+td^2*sf*C5);

%=============================================================
%  rotational equations
%=============================================================

% body 1

F14=0;
F15=0;
F16=0;


% body 2

F24=-A2*F22+B2*F23;
F25=A2*F21-C2*F23+J2(3,1)*fd^2;
F26=-B2*F21+C2*F22+J2(2,1)*fd^2;
F27=-C1*F22+B1*F23;


% body 3

F34=-A5*F32+B5*F33-J3(1,3)*(td*fd*sf+td*fd*sf)+...
     J3(3,2)*(td^2*sf^2)+(J3(3,3)-J3(2,2))*...
     (-td^2*sf*cf)-J3(2,1)*(2*td*fd*cf)-J3(2,3)*(td^2*cf^2);
F35=A5*F31-D5*F33-J3(2,3)*td*fd*sf+(J3(1,1)-J3(3,3))*...
    (td*fd*cf)+J3(1,2)*(td^2*sf*cf)+J3(1,3)*(td^2*cf^2)-J3(3,1)*...
    (fd^2)-J3(3,2)*(-td*fd*sf)-J3(2,2)*td*fd*cf;
F36=-B5*F31+D5*F32-J3(3,3)*td*fd*sf+J3(2,1)*(fd^2)...
    +(J3(2,2)-J3(1,1))*(-td*fd*sf)+J3(2,3)*...
    (td*fd*cf)-J3(1,2)*(td^2*sf^2)...
    -J3(1,3)*(-td^2*sf*cf)-J3(3,2)*td*fd*cf;

%=============================================================
%  payload angle equations
%=============================================================

F37=-E5*F32+F5*F33-J3(1,3)*td*fd*sf+J3(1,3)*...
     (-td*fd*sf)+J3(3,2)*(td^2*sf^2)...
     +(J3(3,3)-J3(2,2))*(-td^2*sf*cf)-J3(2,1)...
     *(td*fd*cf)-J3(2,3)*(td^2*cf^2)-J3(1,2)*td*fd*cf;
F38=-C5*F31+G5*F32+H5*F33+...
     sf*(J3(2,2)*td*fd*cf+J3(2,3)*td*fd*sf)-...
     cf*(J3(3,2)*td*fd*cf+J3(3,3)*td*fd*sf)-...
     sf*( (J3(1,1)-J3(3,3))*(td*fd*cf)+J3(1,2)*...
        (-td^2*sf*cf)+J3(1,3)*(td^2*cf^2)-...
        J3(3,1)*(fd^2)-J3(3,2)*(-td*fd*sf) )+...
     cf*( J3(2,1)*(fd^2)+(J3(2,2)-J3(1,1))*...
        (-td*fd*sf)+J3(2,3)*(td*fd*cf)...
        -J3(1,2)*(td^2*sf^2)-J3(1,3)*(-td^2*sf*cf) );

%==============================================================
%  assemble generalized inertia forces due to non-linear terms
%==============================================================
Fnl(1,1)=F11+F21+F31;
Fnl(2,1)=F12+F22+F33;
Fnl(3,1)=F13+F23+F33;
Fnl(4,1)=F14+F24+F34;
Fnl(5,1)=F15+F25+F35;
Fnl(6,1)=F16+F26+F36;
Fnl(7,1)=F27+F37;
Fnl(8,1)=F38;
%==============================================================
%==============================================================

                             end

%==============================================================
verifica=[F11 F12 F13 F14 F15 F16 0 0;
          F21 F22 F23 F24 F25 F26 F27 0;
          F31 F32 F33 F34 F35 F36 F37 F38]';

verificag=[g11  g21  g31;
           g12  g22  g32;
           g13  g23  g33;
           g14  g24  g34;
           g15  g25  g35;
           g16  g26  g36;
           0    g27  0];
```

```
function [FNL,FNLrot,FNLtrans]=...
     Fnonlin(m,J,t1d,t2d,t3d,f,t,fd,td,rAoAA,rAFF,rFplusF,rPAA,rBPP);
%===============================================================
%  author: Marco Quadrelli
%  date  : December 10, 1991
%  done for: MACE, MACENEW and MACESUSP
%  revised: December 10,1991
%===============================================================
%===============================================================
% Usage:
%         builds the non-linear terms in the equations
%         of motion for MACE.m using Kane's method
%         The inclusion of the g_ij terms makes the equations
%         fully non-linear in theta_dot, phi_dot, theta1_dot,
%         theta2_dot and theta3_dot
%===============================================================
st=sin(t);sf=sin(f);ct=cos(t);cf=cos(f);
tds=td^2;fds=fd^2;
t1ds=t1d^2;t2ds=t2d^2;t3ds=t3d^2;

m1=m(1);
m2=m(2);
m3=m(3);
J1=J(:,1:3);
J2=J(:,4:6);
J3=J(:,7:9);
A1=rAoAA(1);
B1=rAoAA(2)*cf-rAoAA(3)*sf;
C1=rAoAA(2)*sf+rAoAA(3)*cf;
A2=C1+rAFF(3);
B2=B1+rAFF(2);
C2=A1+rAFF(1);
A3=rPAA(1);
B3=rPAA(2)*cf-rPAA(3)*sf;
C3=rPAA(2)*sf+rPAA(3)*cf;
A4=rBPP(1)*ct-rBPP(2)*st;
B4=rBPP(1)*st*cf+rBPP(2)*ct*cf-rBPP(3)*sf;
C4=rBPP(1)*st*sf+rBPP(2)*ct*sf+rBPP(3)*cf;
A5=rAFF(3)+C3+C4;
B5=rAFF(2)+B3+B4;
C5=C4*sf+B4*cf;
D5=rAFF(1)+A3+A4;
E5=C3+C4;
F5=B3+B4;
G5=A4*cf;
H5=A4*sf;
L5=C4*cf-B4*sf;
M5=A3+A4;
N5=C3+C4;
O5=C4*sf+2*B4*cf;
T5=B3+B4;
W5=2*C4*sf+B4*cf;
AA1=C2;BB1=A2;CC1=B2;
RT1=D5;RT2=B5;RT3=A5;

%===============================================================
% Computation of angular velocities
%===============================================================
WFN=[t1d;t2d;t3d];
WAN=[t1d+fd;t2d;t3d];
WPN=[t1d+fd;t2d-td*sf;t3d+td*cf];
%===============================================================
% Computation of partial velocities
%===============================================================
% rotational
```

```
wfn1=[0;0;0];
wfn2=[0;0;0];
wfn3=[0;0;0];
wfn4=[1;0;0];
wfn5=[0;1;0];
wfn6=[0;0;1];
wfn=[wfn1';wfn2';wfn3';wfn4';wfn5',wfn6';0 0 0;0 0 0];

wan1=[0;0;0];
wan2=[0;0;0];
wan3=[0;0;0];
wan4=[1;0;0];
wan5=[0;1;0];
wan6=[0;0;1];
wan7=[1;0;0];
wan=[wan1';wan2';wan3';wan4';wan5';wan6';wan7';0 0 0];

wpn1=[0;0;0];
wpn2=[0;0;0];
wpn3=[0;0;0];
wpn4=[1;0;0];
wpn5=[0;1;0];
wpn6=[0;0;1];
wpn7=[1;0;0];
wpn8=[0;-sf;cf];
wpn=[wpn1';wpn2';wpn3';wpn4';wpn5';wpn6';wpn7';wpn8'];

% translational

vfn1=[1;0;0];
vfn2=[0;1;0];
vfn3=[0;0;1];
vfn4=[0;-rFplusF(3);rFplusF(2)];
vfn5=[rFplusF(3);0;-rFplusF(1)];
vfn6=[-rFplusF(2);rFplusF(1);0];
vfn=[vfn1';vfn2';vfn3';vfn4';vfn5';vfn6';0 0 0;0 0 0];

van1=[1;0;0];
van2=[0;1;0];
van3=[0;0;1];
van4=[0;-A2;B2];
van5=[A2;0;-C2];
van6=[-B2;C2;0];
van7=[0;-C1;B1];
van=[van1';van2';van3';van4';van5';van6';van7';0 0 0];

vbn1=[1;0;0];
vbn2=[0;1;0];
vbn3=[0;0;1];
vbn4=[0;-A5;B5];
vbn5=[A5;0;-D5];
vbn6=[-B5;D5;0];
vbn7=[0;-E5;F5];
vbn8=[-C5;G5;H5];
vbn=[vbn1';vbn2';vbn3';vbn4';vbn5';vbn6';vbn7';vbn8'];

%=============================================================
FNLrot=zeros(8,1);FNLtrans=zeros(8,1);

[qwq1]=dotdyad(J1,WFN);
[qwq2]=dotdyad(J2,WAN);
[qwq3]=dotdyad(J3,WPN);

[cross1]=cross(WFN,qwq1);
[cross2]=cross(WAN,qwq2);
[cross3]=cross(WPN,qwq3);
```

```
[cross4]=cross(WFN,rFplusF);[cross5]=cross(WFN,cross4);

[cross6]=cross(WFN,rAFF);[cross7]=cross(WFN,cross6);
[cross8]=cross(WAN,rAoAA);[cross9]=cross(WAN,cross8);

[cross10]=cross(WFN,rAFF);[cross11]=cross(WFN,cross10);
[cross12]=cross(WAN,rPAA);[cross13]=cross(WAN,cross12);
[cross14]=cross(WAN,rBPP);[cross15]=cross(WPN,cross14);

for i=1:8,
   [term1(i)]=dot(wfn(i,:),cross1);
   [term2(i)]=dot(wan(i,:),cross2);
   [term3(i)]=dot(wpn(i,:),cross3);
   FNLrot(i)=term1(i)+term2(i)+term3(i);
end
for i=1:8,
   [term1(i)]=dot(vfn(i,:),cross4);
   [term2(i)]=dot(van(i,:),cross6);
   [term3(i)]=dot(van(i,:),cross8);
   [term4(i)]=dot(vbn(i,:),cross10);
   [term5(i)]=dot(vbn(i,:),cross12);
   [term6(i)]=dot(vbn(i,:),cross14);
   FNLtrans(i)=term1(i)+term2(i)+term3(i)+term4(i)+term5(i)+term6(i);
end

FNL=FNLtrans+FNLrot;
```

```
%------------------------------------------------------------
%                MACE 3-D model
%
% Author       : Marco B. Quadrelli
% Date         : June 25, 1991
% Last revision: October 28, 1991
%------------------------------------------------------------
% Notes : This version considers the
%         (support + inner stage + outer stage + payload)
%         as a chain of three bodies connected by two 1dof
%         revolute joints.
%         The three bodies are:
%         - joint + support
%         - gimbal 1
%         - gimbal 2 + payload.
%         This version, equal to MACENEW.m - version of July 18,1991 -
%         also models the suspension wires as beam-cable elements.
%         The wire is hinged at the top and has only 1 translation
%         allowed (suspension carriage vertical displacement) and it
%         is hinged at the bottom at its connection to the rigid
%         element of the bus.
%         The suspension carriage and its local control loop
%         impose a vertical displacement from above, so as to
%         bring the carriage to its reference position when it has
%         been removed from it.
%         To date, 29-Aug-1991, the program works only when
%         nelew=1 and nelef=-1(no flexible appendage yet), but
%         nele can be chosen to be any number.
%         The suspension wires are modelled as (1) flexible beam element
%         with a length equal to the whole length and hinged at the end
%         connections.
%         The central node is allowed to store a finite amount of
%         internal angular momentum.
%         All units in S.I.
%------------------------------------------------------------
%
%
%                                              -> laboratory ceiling
% ______6_______________7_______________8___________
%       |               |               |
%       |               |               |
%       |               |               |
%       |               |               |    ->  suspension wires
%       |               |               |
%       |  torque   -> / | \            |  0   -> hinged payload
%       | wheels      /  \|/ \          | /
%       1=======2=======3=======4=======5/
%      /        .                  \
%    /                              \--> spacecraft bus
%   0
%============================================================
% the state vector is: (after imposing constraints)
%  note: this is for the case nelew=1 always
%____________________________________________________________
%                  description                   +    d.o.f.
%________________________________________________+___________
%  X = [  (x,y,z,theta1,theta2,theta3)_1   |       1:ndof-6
%         (x,y,z,theta1,theta2,theta3)_2   |    ndof-5:ndof
%         (x,y,z,theta1,theta2,theta3)_3   |    ndof+1:2*ndof-6
%         (x,y,z,theta1,theta2,theta3)_4   |  2*ndof-5:3*ndof-12
%         (x,y,z,theta1,theta2,theta3)_5   | 3*ndof-11:ntotflex
%         (y,theta1,theta3)_6              |    ntot-3:ntot-1
%         (y,theta1,theta3)_7              |      ntot:ntot+2
%         (y,theta1,theta3)_8              |    ntot+3:ntot+5
%         (theta1,theta3)_h1   :hinge_1    |    ntot+6:ntot+7
%         (theta1,theta3)_h2   :hinge_2    |    ntot+8:ntot+9
%          (theta1,theta3)_h3   :hinge_3    |     ntot+10:ntot+11
%          (phi,theta)_#1                   |     ntot+12:ntot+13
%          (phi,theta)_#2                   |     ntot+14:ntot+15
%          (omega1,omega2,omega3)_wheels    |     ntot+16:ntot+18
%          ---------------------------------|------------------------
%          and all the first derivatives    |     ntot+19:2*(ntot+18)
%          ---------------------------------|------------------------
%(nwires*4 internal states of suspension)]|2*(ntot+18):2*(ntot+18)+12
%
%==============================================================
%==============================================================
%          ENTER INPUT DATA
%==============================================================
%==============================================================
%--------------------------------------------------------------
% Define parameters and size of arrays
%--------------------------------------------------------------
tupi=2*pi;rad=pi/180;invrad=1/rad;invrpm=(2*pi)/60;rpmm=1/invrpm;
grav=9.8065;wbig=(logspace(-1,3,400));
rdum = [0;0;0];
global trajectory timtstop gravpar
global rAoAA1 rAFF1 rFplusF1 rPAA1 rBPP1 rpayPP1
global rAoAA5 rAFF5 rFplusF5 rPAA5 rBPP5 rpayPP5
%==============================================================
%nele = input(' number of elements per flexible Lexan beam [1 or 2]: ');
%nelew = input(' number of elements per flexible cable-beam [1]: ');
%nelef = input(' number of elements for flexible appendage [1]: ');
nele=1;
nelew=1;nelef=-1;

npay1=2;          % # of rigid body dof from payload 1
npay2=2;          % # of rigid body dof from payload 2
nwheels=3;        % # of rigid body dof from torque wheels

nnod= nele+1;     % # of nodes per Lexan beam
ndof= 6*nnod;     % # of dof allowed per Lexan beam
ntot= 24*nele+10;% # of dof allowed on bus + 2x2 payloads rotations
ntotflex= ntot-4;% # of flexible dof allowed on bus only
totaldof= ntot+3;% total # of on bus including 3 dof of torque wheels
nodi=ntotflex/6; % total # of nodes in spacecraft bus

nnodw= nelew+1;  % # of nodes per suspension cable-beam
ndofw= 6*nnodw;  % # of dof allowed per·suspension cable-beam
nodiw=ndofw/6;   % total # of nodes in suspension wire
ntotw=ndofw;     % # of dof allowed per suspension cable-beam
nwires=3;        % # of suspension wires
ntotsusp=nwires*ntotw;% # of dof allowed by the suspension

nelef=-1;
nnodf= nelef+1;    % # of nodes per flexible appendage beam
ndoff= 6*nnodf;    % # of dof allowed per flexible appendage beam
nodif=ndoff/6;     % total # of nodes in flexible appendage beam

nhingew=3;       % # of rigid body dof from hinges per susp.wire
nhinges=nwires*nhingew;% # of rigid body dof from hinges at suspension

% rigidbody = total # of rigid body dof allowed by the structure:
%        bus + 2*payloads + 3*wheels          in free space
rigidbody=6+npay1+npay2+nwheels;

numnp=nodi+3*nodiw-3+nodif;  % # of nodal points of all structure
nflex=numnp*6+nhinges;

% nfree = total # of dof allowed by the structure:
%        bus + nwires*wires + 2*payloads + 3*wheels
% in free space (before imposing constraints)
```

```
nfree=nflex + npay1 + npay2 + nwheels;

%================================================================
% 1st rotation at node 1: angle phi is Cw about a1 -- out-of-plane XY
% starts from the Z axis of the bus
% 2nd rotation at node 1: angle theta is Cw about p3-- in-plane XY
% starts from the X axis of the bus
% 1st rotation at node 5: angle phi is CCw about a1 -- out-of-plane XY
% starts from the Z axis of the bus
% 2nd rotation at node 5: angle theta is CCw about p3-- in-plane XY
% starts from the X axis of the bus
%================================================================
%choice=...
%menu('choose configuration','gimbals locked','gimbals free',...
%     'gimbal no.1 locked and gimbal no.5 free');
choice=2;

phi1  =...
input('out-of-plane angle phi1 [from XY plane towards +Z] (deg): ')*rad;
theta1=...
input('in-plane angle theta1 [from XZ plane towards +Y] (deg): ')*rad;
phi5  =..
input('out-of-plane angle phi5 [from XY plane towards +Z] (deg): ')*rad;
theta5=...
input('in-plane angle theta5 [from XZ plane towards +Y] (deg): ')*rad;

%phi1=f1;theta1=t1;phi5=f5;theta5=t5;

p1_locked=input('if payload #1 free enter [0],if locked or no moving mass :enter [1]:   ');
%p1_locked=1;

bounce=input('enter bounce frequency [Hz]:    ');
active_suspension=...
 input('suspension active [enter 1], if passive [enter 0]:    ');

%choice2=...
%menu('enter',...
%'components of angular momentum in MACE body axes','wheels speeds');
choice2=1;

if choice2==1,
Hcomp=[0;1;0];
%Hcomp=...
%input(...
%'enter components of angular momentum in body axes [N.m.sec]:    ');
elseif choice2==2,
spin1= input(' steady angular velocity of wheel 1 [rpm]:  ');
spin2= input(' steady angular velocity of wheel 2 [rpm]:  ');
spin3= input(' steady angular velocity of wheel 3 [rpm]:  ');
%in rad/s
spin1=invrpm*spin1;spin2=invrpm*spin2;spin3=invrpm*spin3;
end
spinMACE1=0;spinMACE2=0;spinMACE3=0;
spinMACE1=invrpm*spinMACE1;spinMACE2=invrpm*spinMACE2;
spinMACE3=invrpm*spinMACE3;

%choicew=...
%menu('select one','non-linear geom. stiffness for wire','string
%stiffness');
choicew=1;

smorza=2;
damping=1;
%damping=menu('choose type of damping','modal','prop to K',...
%               'prop to M','old type','decoupling');
```

```
%CG=menu('is each payload','CG mounted ?','non CG mounted ?');
CG=2;
%gimbal1=menu('new coordinates of reference point of gimbal 1?',...
%              'yes,change','no,use default');
gimbal1=2;
%center=menu('gimbals rotation axes are centered',...
%             'no','yes');
center=2;
%pollo=menu('pivots lie on bus centerline','yes','no');
pollo=2;

statespace=menu('do you want state space model?','yes','no');
if statespace==1,
 bodeplots=menu('do you want Bode plots?','open-loop','closed-loop',...
                'no Bode plots');
 if bodeplots==1,
   bodeplotsOL=1;bodeplotsCL1=0;
 elseif bodeplots==2,
   bodeplotsOL=0;bodeplotsCL1=1;
 elseif bodeplots==3,
   bodeplotsOL=0;bodeplotsCL1=0;
 end
end


salva=menu('save results in one_g.mat ?','yes','no');
salva1=menu('save results in one_g1.mat ?','yes','no');

timsim=input('do you want linear time simulation, yes[1], no[0]:   ');
if timsim==1,
trajectory=...
menu('choose trajectory',...
'constant step (sudden step to end)',...
'cosine',...
'sine',...
'exponential (shaped step to end)',...
'quintic polynomial for rest-to-rest',...
'shaped (smooth) step in specified time',...
'ref. MACE large angle slew (requires min.3 sec.of simulation)',...
'Lissajous Figure',...
'ref. smoothed bang-bang slew');
end

%loop=menu('select suspension control loop',...
%    'with displacement + acceleration feedback',...
%     'with displacement + velocity + acceleration feedback');
loop=1;                 %loop=2 is not the case of MACE!
%K_var=...
%input('variable gain of accelerometer(min=0.2;recd=0.2;max=10):  ');
%Kdisp=input('gain of LVDT (min=1;nom.=4;max=11[Volt/inch]:   ');
Kdisp=4;K_var=.2;
%=================================================================
%  these are the 2 angles made by each wheel rotation axis
%  with the X axis of the bus
%             alfa_i - in plane ;  beta_i out-of-plane
%  which is fixed in the L frame -X Y Z- (in radians)
%=================================================================
%ruote=menu('wheels box symm. axis pointing towards:','-Y','+Y');
ruote=2;
if ruote==1,
alfa1=90*rad;beta1=-35.3*rad;
alfa2=-30*rad;beta2=-35.3*rad;
alfa3=-150*rad;beta3=-35.3*rad;
elseif ruote==2,
alfa1=90*rad;beta1=35.3*rad;
alfa2=-30*rad;beta2=35.3*rad;
```

```
alfa3=-150*rad;beta3=35.3*rad;
end
%===========================================================
%  transformation matrix between Fwheel-i and FB
[RwL11,RwL21,RwL31,RwLL1]=rotation(beta1,0,-alfa1);
[RwL12,RwL22,RwL32,RwLL2]=rotation(beta2,0,-alfa2);
[RwL13,RwL23,RwL33,RwLL3]=rotation(beta3,0,-alfa3);
%============================================================
[pw1,qw1,rw1,lw1,mw1,nw1,fw1,gw1,hw1,Aw1,Bw1,Cw1,RwL1]=...
WitoBody(alfa1,beta1);
[pw2,qw2,rw2,lw2,mw2.nw2,fw2,gw2,hw2,Aw2,Bw2,Cw2,RwL2]=...
WitoBody(alfa2,beta2);
[pw3,qw3,rw3,lw3,mw3,nw3,fw3,gw3,hw3,Aw3,Bw3,Cw3,RwL3]=...
WitoBody(alfa3,beta3);
%============================================================
if choice==1,
totaldegof = totaldof-4;effectdof=totaldegof-3;
rigid=6;rigidbody=9;flex=rigidbody+1;
rigidbodynw=6;flexnw=rigidbodynw+1;
j0=15;final=2*(totaldof-4);
nhinge=0;
elseif choice==2,
totaldegof = ntot+3;effectdof=totaldegof-3;
rigid=10;rigidbody=13;flex=rigidbody+1;
rigidbodynw=10;flexnw=rigidbodynw+1;
j0=7;final=2*totaldof;
nhinge=4;
elseif choice==3,
totaldegof = ntot+3-2;effectdof=totaldegof-3;
rigid=8;rigidbody=11;flex=rigidbody+1;
rigidbodynw=8;flexnw=rigidbodynw+1;
j0=11;final=2*(totaldof-2);
nhinge=2;
end
%============================================================
%==============================================================
%    End of interactive input data
%==============================================================
%============================================================

LEGGIGEO    % enter vectors and geometric information

%============================================================
%============================================================

RUOTE       % compute precession frequency and gyroscopic parameters

%============================================================
%============================================================
% Transformation matrices relating inertial velocities at nodes
% placed a given distance apart (R=right,L=left of node)
%============================================================
[TR1]=transform([0.07632;0;0]);[TL5]=transform([-0.07632;0;0]);
[TL2]=transform(rL2);[TR2]=transform(rR2);TL4 = TL2;TR4 = TR2;
%==============================================================
% if central node F.E. node is at the cg
%[TL3]=transform(-rcm3);[TR3]=transform(rRL3-rcm3);
%==============================================================
% if central node F.E. node is on bus centerline
[TL3]=transform(rL3);[TR3]=transform(rR3);
%==============================================================
%==============================================================
% assemble mass matrices
%============================================================

%disp('<<<<<< assembling multi-rigid mass >>>>>>');
```

```
MAKEMASS   % assemble inertia matrices of nodes 1,3 and 5


%===============================================================
%===============================================================
% Before assembling the flex elements, move rows and columns 7,8
% of M1 to 1st and 2nd place so that the state vector for
% the first node becomes [phi1,theta1,x1,y1,z1,d1,d2,d3]'
%===============================================================
M1=M1([7:8 1:6],:);M1=M1(:,[7:8 1:6]);
%===============================================================
% and remove rows and columns 21,22,23 thus eliminating wheels dof's
%===============================================================
M3new=M3(1:6,1:6);
%===============================================================

%===============================================================
%
%          FINITE ELEMENT MODEL
%
%===============================================================
%disp('<<<<<< assembling mass and stiffness matrices >>>>>>');
%===============================================================
lele=L/nele;
[mele,kele]=beamele(pA,lele,E,G,Jx,Jy,Jz,A);
kele=kele+1.e-6*mele;
[mflex]=fem(mele,nele);[kflex]=fem(kele,nele);
%===============================================================
%
% Start assembling all flexible parts together
%
%===============================================================
KK=zeros(nfree);MM=zeros(nfree);T=eye(ndof);
KKflex=zeros(nflex);MMflex=zeros(nflex);
KKflexbus=zeros(ntotflex);MMflexbus=zeros(ntotflex);
%===============================================================
% A) assemble flexibility of bus
%    (can accept more than one nele elements)
%===============================================================
T(1:6,1:6)=TR1;T(ndof-5:ndof,ndof-5:ndof)=TL2;
nbeg=1;nend=ndof;
KKflexbus(nbeg:nend,nbeg:nend)=T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=T'*mflex*T;

T(1:6,1:6) = TR2;T(ndof-5:ndof,ndof-5:ndof) = TL3;
nbeg=ndof-5;nend=2*ndof-6;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

T(1:6,1:6) = TR3;T(ndof-5:ndof,ndof-5:ndof) = TL2;
nbeg=2*ndof-11;nend=3*ndof-12;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

T(1:6,1:6)=TR2;T(ndof-5:ndof,ndof-5:ndof)=TL5;
nbeg=3*ndof-17;nend=ntotflex;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

KKflex(1:ntotflex,1:ntotflex)=KKflexbus;
```

```
MMflex(1:ntotflex,1:ntotflex)=MMflexbus;
%============================================================
% B) add flexibility due to suspension wires
%============================================================
% Assemble finite element model for suspension wires
% compute melew and kelew of the wire elements in their
% local frames. (-rig refers to the short rigid element)
%============================================================
lelew=Lw/nelew;

ADDENDUM3d

%outputs of ADDENDUM3d:
%            KKw1,KKw2,KKw3,MMw1,MMw2,MMw3,
%            all 12x12, all stiffened by gravity
%nbegattach  = bottom attachment to bus
%nendcarriage= top attachment to carriage

nbegattach=1:3;
nendcarriage=(ntotflex+1):(ntotflex+ntotw-3);
nbeg=[nbegattach nendcarriage];
nend=[nbegattach nendcarriage];
KKflex(nbeg,nend)=KKflex(nbeg,nend)+KKw1;
MMflex(nbeg,nend)=MMflex(nbeg,nend)+MMw1;

nbegattach=(2*ndof-11):(2*ndof-9);
nendcarriage=(ntotflex+ntotw-2):(ntotflex+2*ntotw-6);
nbeg=[nbegattach nendcarriage];
nend=[nbegattach nendcarriage];
KKflex(nbeg,nend)=KKflex(nbeg,nend)+KKw2;
MMflex(nbeg,nend)=MMflex(nbeg,nend)+MMw2;

nbegattach=(ntotflex-5):(ntotflex-3);
nendcarriage=(ntotflex+2*ntotw-5):(ntotflex+3*ntotw-9);
nbeg=[nbegattach nendcarriage];
nend=[nbegattach nendcarriage];
KKflex(nbeg,nend)=KKflex(nbeg,nend)+KKw3;
MMflex(nbeg,nend)=MMflex(nbeg,nend)+MMw3;
%============================================================
% augment to include end payloads and wheels
KK(3:nfree-5,3:nfree-5)=KKflex;
MM(3:nfree-5,3:nfree-5)=MMflex;
%============================================================
% Lump additional mass into finite element model
%============================================================
% a) first lump spacecraft mass at nodes 1 to 5
MM(1:8,1:8)=MM(1:8,1:8)+M1;

nbeg=ndof-3;nend=ndof+2;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M2;

nbeg=2*ndof-9;nend=2*ndof-4;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M3new;

nbeg=3*ndof-15;nend=3*ndof-10;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M4;

nbeg=4*ndof-21;nend=ntot-2;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M5(1:6,1:6);

% dof's theta and phi for payload at node 5 are placed after the
% flex elements in KK
nbeg=[nflex+3 nflex+4];nend=[nflex+3 nflex+4];
MM(nbeg,nend)=MM(nbeg,nend)+M5(7:8,7:8);

nbeg=[(4*ndof-21):(4*ndof-21+5)];nend=[nflex+3 nflex+4];
```

```
MM(nbeg,nend)=MM(nbeg,nend)+M5(1:6,7:8);

nbeg=[nflex+3 nflex+4];nend=[(4*ndof-21):(4*ndof-21+5)];
MM(nbeg,nend)=MM(nbeg,nend)+M5(7:8,1:6);

%=============================================================
% arrange mass and stiffness matrices in a form ready for
% integration, i.e. move rigid body dof's after flex. dof's:
% state vector =
%  (flex. coord's + phi1,theta1,phi5,theta5,rate1,rate2,rate3)'
%          payload   (     #1    ) (     #2    )
%                              torque wheel    (#1)   (#2)   (#3)
%=============================================================
MMint=zeros(nfree);KKint=zeros(nfree);paw=length(MMint);
%=============================================================
% move payloads dof's after flex+hinges dof's
%=============================================================
nbeg=[3:(nflex+2) 1 2 (nflex+3) (nflex+4)];nend=nbeg;
MMintn=MM(nbeg,nend);
KKintn=KK(nbeg,nend);
%=============================================================
% move hinges dof's after flex dof's and before payloads dof's
%=============================================================
nbeg=[1:ntotflex ntot:ntot+5 ntot+9:ntot+14 ntot+18:ntot+23...
ntotflex+1:ntot-1 ntot+6:ntot+8 ntot+15:ntot+17 ntot+24:nfree-3];
if nelew ==2,
nbeg=[1:ntotflex ntotflex+4:ntotflex+ntotw-3...
  ntotflex+ntotw+1:ntotflex+2*ntotw-6...
  ntotflex+2*ntotw-2:ntotflex+3*ntotw-9...
  ntotflex+1:ntotflex+3 ntotflex+ntotw-2:ntotflex+ntotw...
  ntotflex+2*ntotw-5:ntotflex+2*ntotw-3 ntotflex+3*ntotw-8:nfree-3];
end
nend=nbeg;
MMintnw=MMintn(nbeg,nend);
KKintnw=KKintn(nbeg,nend);
puw=length(MMintnw);
%=============================================================
% add wheels dof's
KKint=[KKintnw                        zeros(length(KKintnw),3);
       zeros(3,length(KKintnw))       zeros(3)];
MMint=[MMintnw                        zeros(length(MMintnw),3);
       zeros(3,length(MMintnw))       M3diag2];

nbeg=[(ntotflex/2+1):(ntotflex/2+3)];
nend=[(nfree-2):nfree];
MMint(nbeg,nend)=M3(4:6,7:9);

nbeg=[(nfree-2):nfree];
nend=[(ntotflex/2+1):(ntotflex/2+3)];
MMint(nbeg,nend)=M3(4:6,7:9)';
%=============================================================
% hinge suspension wires at the top (equivalent to removing
%  translational dof's) but keep vertical translation
%  and do not remove torsion from the wires yet
%=============================================================
clamp1=[1:ntot-4 ntot-2 ntot:ntot+2 ntot+4 ntot+6:ntot+8 ntot+10...
        ntot+12:nfree];
if nelew==2,
clamp1=[1:ntotflex+ntotw-ndof  ntotflex+ntotw-10...
   ntotflex+ntotw-8:ntotflex+2*ntotw-18   ntotflex+2*ntotw-16...
   ntotflex+2*ntotw-14:ntotflex+3*ntotw-24    ntotflex+3*ntotw-22...
   ntotflex+3*ntotw-20:nfree];
end
KKintncl=KKint(clamp1,clamp1);
MMintncl=MMint(clamp1,clamp1);
pucl=length(MMintncl);
```

```
%===============================================================
% add carriage mass along the vertical direction
%===============================================================
MMintncl(ntot-3,ntot-3)=MMintncl(ntot-3,ntot-3)+Mcarriage;
MMintncl(ntot+1,ntot+1)=MMintncl(ntot+1,ntot+1)+Mcarriage;
MMintncl(ntot+5,ntot+5)=MMintncl(ntot+5,ntot+5)+Mcarriage;
%===============================================================
%  remove torsion from the wires and hinges dof's
%===============================================================
clamp2=[1:ntot-2 ntot:ntot+2 ntot+4:ntot+6 ntot+8:ntot+9...
        ntot+11:ntot+12 ntot+14:ntot+15 ntot+17:ntot+24];
if nelew==2,
clamp2a=...
[1:ntotflex+ntotw-14  ntotflex+ntotw-12:2:ntotflex+ntotw-8...
ntotflex+ntotw-6:ntotflex+ntotw-2...
ntotflex+2*ntotw-18:2:ntotflex+2*ntotw-14...
ntotflex+2*ntotw-12:ntotflex+2*ntotw-8];
clamp2b=...
[ntotflex+2*ntotw-6:2:ntotflex+2*ntotw ntotflex+2*ntotw+1...
ntotflex+2*ntotw+3:ntotflex+2*ntotw+5...
ntotflex+3*ntotw-12:ntotflex+3*ntotw-11 ntotflex+3*ntotw-9:pucl];
clamp2=[clamp2a  clamp2b];
end
KKintnc2=KKintncl(clamp2,clamp2);
MMintnc2=MMintncl(clamp2,clamp2);
puc2=length(MMintnc2);
%===============================================================
% clamp payload #1 only on full model
%===============================================================
if nele==1,
clamp5=[1:45 48:puc2];
elseif nele==2,
clamp5=[1:69 72:puc2];
end
%===============================================================
% add air spring for purposes of static analysis
%===============================================================
Kairl=(Mnodel + .5*Mnode2 + Mcarriage + Mcable)*(.47*tupi)^2;
Kair3=(Mnode3 + .5*(Mnode2+Mnode4) + Mcarriage + Mcable)*(.57*tupi)^2;
Kair5=(Mnode5 + .5*Mnode4 + Mcarriage + Mcable)*(.47*tupi)^2;
Kair=(MassMACE/3 + Mcarriage + Mcable)*(bounce*tupi)^2;
KKintnc2(ntot-3,ntot-3)=KKintnc2(ntot-3,ntot-3)+Kair;
KKintnc2(ntot,ntot)=KKintnc2(ntot,ntot)+Kair;
KKintnc2(ntot+3,ntot+3)=KKintnc2(ntot+3,ntot+3)+Kair;
%===============================================================
KKintnc5=KKintnc2(clamp5,clamp5);
MMintnc5=MMintnc2(clamp5,clamp5);
puc5=length(MMintnc5);
%===============================================================
KKnog2=KKintnc2;
KKnog5=KKintnc5;
%===============================================================
% options:
%===============================================================
% hinge suspension wires completely at the top (no translation)
%===============================================================
clamp3=[1:ntot-4 ntot-2 ntot-1 ntot+1 ntot+2 ntot+4:ntot+18];
KKintnc3=KKintnc2(clamp3,clamp3);
MMintnc3=MMintnc2(clamp3,clamp3);
KKnog3=KKnog2(clamp3,clamp3);
puc3=length(MMintnc3);
%===============================================================
% lock the hinges at payload # 1
%===============================================================
clamp4=[1:ntot+8 ntot+11:ntot+15];
KKintnc4=KKintnc3(clamp4,clamp4);
```

```
MMintnc4=MMintnc3(clamp4,clamp4);
KKnog4=KKnog3(clamp4,clamp4);
puc4=length(MMintnc4);
%=============================================================
% Now introduce damping and Gyroscopic matrices
% Also introduce geometric stiffness matrix due to g load
%=============================================================
DA=zeros(nfree);GYRO=zeros(nfree);DAMP=zeros(nfree);
GYRODAMP=zeros(nfree);DAMPREDUCED=zeros(nflex);
%=============================================================
% The torque wheels contribute with zero stiffness. But they
% contribute with a gyroscopic matrix GY. Assemble GY.
%=============================================================
% Computation of gyroscopic matrix  GYRO
APw1=spin1*Jwa1*Aw1;BPw1=spin1*Jwa1*Bw1;CPw1=spin1*Jwa1*Cw1;
APw2=spin2*Jwa2*Aw2;BPw2=spin2*Jwa2*Bw2;CPw2=spin2*Jwa2*Cw2;
APw3=spin3*Jwa3*Aw3;BPw3=spin3*Jwa3*Bw3;CPw3=spin3*Jwa3*Cw3;
GYRO(2*ndof-8,2*ndof-7)=APw1+APw2+APw3;
GYRO(2*ndof-8,2*ndof-6)=BPw1+BPw2+BPw3;
GYRO(2*ndof-7,2*ndof-6)=CPw1+CPw2+CPw3;
GYRO(2*ndof-7,2*ndof-8)=-GYRO(2*ndof-8,2*ndof-7);
GYRO(2*ndof-6,2*ndof-8)=-GYRO(2*ndof-8,2*ndof-6);
GYRO(2*ndof-6,2*ndof-7)=-GYRO(2*ndof-7,2*ndof-6);
gyroblock=GYRO(2*ndof-8:2*ndof-6,2*ndof-8:2*ndof-6);

GYROpic=zeros(KKintnw);
GYROpic(2*ndof-8:2*ndof-6,2*ndof-8:2*ndof-6)=gyroblock;
GYROint=[GYROpic                         zeros(length(KKintnw),3);
         zeros(3,length(KKintnw))        zeros(3)];
GYROintnc1=GYROint(clamp1,clamp1);
GYROintnc2=GYROintnc1(clamp2,clamp2);
GYROintnc5=GYROintnc2(clamp5,clamp5);
%=============================================================
%=============================================================
% add stiffening matrix representing the effect due to gravity only
% in the multi-rigid chain and in the central node
% NOTE: (see my notes) this is used in the linearized analysis
%=============================================================
Kgrav1=zeros(5);Kgrav5=zeros(5);Kgrav3=zeros(3);
d1=po1;d5=po5;a1=rAoAA1;a5=rAoAA5;
b1=rFplusF1;c1=rAFF1;b5=rFplusF5;c5=rAFF5;
A101=a1(1);
A201=a1(2)*cf1-a1(3)*sf1;
A301=a1(2)*sf1+a1(3)*cf1;
A105=a5(1);
A205=a5(2)*cf5-a5(3)*sf5;
A305=a5(2)*sf5+a5(3)*cf5;
D101=d1*ct1;
D1t1=d1*st1;
D201=d1*st1*cf1;
D2f1=d1*st1*sf1;
D2t1=d1*ct1*cf1;
D301=D2f1;
D3f1=D201;
D3t1=d1*ct1*sf1;
D105=d5*ct5;
D1t5=d5*st5;
D205=d5*st5*cf5;
D2f5=d5*st5*sf5;
D2t5=d5*ct5*cf5;
D305=D2f5;
D3f5=D205;
D3t5=d5*ct5*sf5;
%=============================================================
e111=Mbody2*a1(2)*cf1-Mbody2*a1(3)*sf1+Mbody3*d1*st1*cf1;
e121=Mbody3*d1*sf1*ct1;
```

```
el31=Mbody3*d1*sf1*ct1;
el41=Mbody3*d1*st1*cf1;
el15=Mbody2*a5(2)*cf5-Mbody2*a5(3)*sf5+Mbody3*d5*st5*cf5;
el25=Mbody3*d5*sf5*ct5;
el35=Mbody3*d5*sf5*ct5;
el45=Mbody3*d5*st5*cf5;
Kg1=[el11 el21;el31 el41];Kg5=[el15 el25;el35 el45];
Kgrav1(4:5,4:5)=Kg1;Kgrav5(4:5,4:5)=Kg5;
%============================================================
Kgrav1(1,1)=...
   Mbody1*b1(2)+Mbody2*(c1(2)+A201)+Mbody3*(c1(2)+D201)-Mnode1*rL1(2);
Kgrav1(1,4)=Mbody2*A201+Mbody3*D3f1;Kgrav1(4,1)=Kgrav1(1,4);
Kgrav1(1,5)=Mbody3*D3t1;Kgrav1(5,1)=Kgrav1(1,5);
Kgrav1(3,3)=Kgrav1(1,1);
Kgrav1(3,5)=Mbody3*D1t1;Kgrav1(5,3)=Kgrav1(3,5);

Kgrav5(1,1)=...
   Mbody1*b5(2)+Mbody2*(c5(2)+A205)+Mbody3*(c5(2)+D205)-Mnode5*rN5(2);
Kgrav5(1,4)=Mbody2*A205+Mbody3*D3f5;Kgrav5(4,1)=Kgrav5(1,4);
Kgrav5(1,5)=Mbody3*D3t5;Kgrav5(5,1)=Kgrav5(1,5);
Kgrav5(3,3)=Kgrav5(1,1);
Kgrav5(3,5)=Mbody3*D1t5;Kgrav5(5,3)=Kgrav5(3,5);

Kgrav3(1,1)=Mnode3*(rH3(2)-rM3(2));
Kgrav3(3,3)=Kgrav3(1,1);
%============================================================
nbeg=[4:6 puc2-6:puc2-5];nend=[4:6 puc2-6:puc2-5];
KKintnc2(nbeg,nend)=KKintnc2(nbeg,nend)-grav*Kgrav1;
nbeg=[ntot-6:ntot-4  puc2-4:puc2-3];nend=[ntot-6:ntot-4  puc2-4:puc2-3];
KKintnc2(nbeg,nend)=KKintnc2(nbeg,nend)-grav*Kgrav5;
nbeg=[(2*ndof-8):(2*ndof-6)];nend=[(2*ndof-8):(2*ndof-6)];
KKintnc2(nbeg,nend)=KKintnc2(nbeg,nend)-grav*Kgrav3;

KKintnc3=KKintnc2(clamp3,clamp3);
KKintnc4=KKintnc3(clamp4,clamp4);
KKintnc5=KKintnc2(clamp5,clamp5);
%============================================================
%nbeg=[4:6 puc3-6:puc3-5];nend=[4:6 puc3-6:puc3-5];
%KKintnc3(nbeg,nend)=KKintnc3(nbeg,nend)-grav*Kgrav1;
%nbeg=[ntot-6:ntot-4 puc3-4:puc3-3];nend=[ntot-6:ntot-4 puc3-4:puc3-3];
%KKintnc3(nbeg,nend)=KKintnc3(nbeg,nend)-grav*Kgrav5;
%nbeg=[4:6];nend=[4:6];
%KKintnc4(nbeg,nend)=KKintnc4(nbeg,nend)-grav*Kgrav1(1:3,1:3);
%nbeg=[ntot-6:ntot-4 puc4-4:puc4-3];nend=[ntot-6:ntot-4 puc4-4:puc4-3];
%KKintnc4(nbeg,nend)=KKintnc4(nbeg,nend)-grav*Kgrav5;
%KKintnc3(nbeg,nend)=KKintnc3(nbeg,nend)-grav*Kgrav3;
%KKintnc4(nbeg,nend)=KKintnc4(nbeg,nend)-grav*Kgrav3;
%KKintnc5=KKintnc2(clamp5,clamp5);
%============================================================
%                     END ASSEMBLY
%============================================================


% PART B: solve eigenproblem

%disp('<<<<<< solving eigenproblem >>>>>>');
%============================================================
% order the state vectors
%============================================================
TP = 0*eye(nflex);TTP=0*eye(nfree);
n3 = numnp;
for i=1:n3,                 % for each of the nodes :
TP(i,6*i-5)      = 1;       %x
TP(i+n3,6*i-4)   = 1;       %y
TP(i+2*n3,6*i-3)= 1;        %z
TP(i+3*n3,6*i-2)= 1;        %angle theta1
```

```
 TP(i+4*n3,6*i-1)= 1;     %angle theta2
 TP(i+5*n3,6*i)  = 1;     %angle theta3
 end
 TTP(1:nflex,1:nflex)=TP;
 TTP(nflex+1:nfree,nflex+1:nfree)=eye(7);
 TTPnc1=TTP(clamp1,clamp1);
 %=================================================================
 % solve eigenproblem
 %=================================================================
 % payloads free
 KKintnc2=KKintnc2+1.e-7*MMintnc2;              % stiffen RB modes
 [evalnc,evecnc,snc,polesnc,hertznc]=eigenproblem(KKintnc2,-MMintnc2);
 hertznc=hertznc-0.00005*ones(puc2,1);
 TTPnc2=TTPnc1(clamp2,clamp2);
 evecnc=TTPnc2*evecnc;

 [evecnorm,lamw]=normalizza(MMintnc2,KKintnc2);
 modal_mass=evecnorm'*MMintnc2*evecnorm;
 modal_stif=evecnorm'*KKintnc2*evecnorm;
 OMEGA=sqrt(lamw);
 DAMPMAT=2*zeta*(inv(evecnorm))'*OMEGA*inv(evecnorm);
 GYRODAMPintnc2=GYROintnc2+DAMPMAT;

 %=================================================================
 % payload #1 clamped
 KKintnc5=KKintnc5+1.e-7*MMintnc5;              % stiffen RB modes
 [evalnc5,evecnc5,snc5,polesnc5,hertznc5]=...
           eigenproblem(KKintnc5,-MMintnc5);
 hertznc5=hertznc5-0.00005*ones(puc5,1);
 TTPnc5=TTPnc2(clamp5,clamp5);
 evecnc5=TTPnc5*evecnc5;

 [evecnorm,lamw]=normalizza(MMintnc5,KKintnc5);
 modal_mass=evecnorm'*MMintnc5*evecnorm;
 modal_stif=evecnorm'*KKintnc5*evecnorm;
 OMEGA=sqrt(lamw);
 DAMPMAT=2*zeta*(inv(evecnorm))'*OMEGA*inv(evecnorm);
 GYRODAMPintnc5=GYROintnc5+DAMPMAT;

 [hertznc(4:puc2-2) hertznc5(4:puc5)];
 %=================================================================
 % payloads free but no translation alloved at the carriages
 %KKintnc3=KKintnc3+1.e-7*MMintnc3;             % stiffen RB modes
 %[evalnc3,evecnc3,snc3,polesnc3,hertzn3]=...
 %eigenproblem(KKintnc3,-MMintnc3);
 %hertzn3=hertzn3-0.00005*ones(puc3,1)
 %TTPnc3=TTPnc2(clamp3,clamp3);
 %evecnc3=TTPnc3*evecnc3;
 %=================================================================
 % payload #1 clamped but no translation alloved at the carriages
 %KKintnc4=KKintnc4+1.e-7*MMintnc4;             % stiffen RB modes
 %[evalnc4,evecnc4,snc4,polesnc4,hertzn4]=...
 %eigenproblem(KKintnc4,-MMintnc4);
 %hertzn4=hertzn4-0.00005*ones(puc4,1)
 %TTPnc4=TTPnc3(clamp4,clamp4);
 %evecnc4=TTPnc4*evecnc4;
 %=================================================================
 %=================================================================
 %  in modal coordinates
 %=================================================================
 Aomegas=diag(evalnc);
 Adampin=-2*zeta*sqrt(-Aomegas);
 Amodal=[zeros(Aomegas) eye(Aomegas);Aomegas  Adampin];
 polesmod=eig(Amodal);
 polesmod=sort(polesmod);
 %=================================================================
                        if p1_locked==0,
%=================================================================
element1=-inv(MMintnc2)*KKintnc2;
element2=-inv(MMintnc2)*GYRODAMPintnc2;
Ac=[zeros(MMintnc2)  eye(MMintnc2);element1  element2];
puAc=length(Ac);
polesc2=eig(Ac);
states=puAc/2;
%=================================================================
                        elseif p1_locked==1,
%=================================================================
element15=-inv(MMintnc5)*KKintnc5;
element25=-inv(MMintnc5)*GYRODAMPintnc5;
Ac5=[zeros(MMintnc5)  eye(MMintnc5);element15  element25];
puAc5=length(Ac5);
polesc25=eig(Ac5);
states=puAc5/2;
%=================================================================
                        end




% PART C: Build state space model

                     if statespace==1,

%disp('<<<<<<< building state space model >>>>>>>');
%=================================================================
% Amplifier data
%=================================================================
Kgimbal=.6493;      %in [Nm/V]
Kwheels=.09;        %in [Nm/V]

Kaccelr=1.376;      %in [V/(m/s2)]
Kstraingage=4483.5; %in [V/microstrain]
Krategyro=40*1.e-3; %in [V/(deg/sec)]
%=================================================================
% Assemble forcing Matrix
%=================================================================
% input control torques
% apply unit torques and forces:
%   - at theta and phi at node 1 and 5        = 4
%   - at torque wheels                        = 3
%   - vertical force at suspension carriages  = 3 (excluded)
%   - active strut unit input
%=================================================================
%=================================================================
% Measurement matrix  (angular measurements in degrees)
% measure:
%   - theta and phi inertial at nodes 1 and 5            = 4
%   - rigid body rates at central node                   = 3
%   - vertical accelerations at suspension carriages     = 3 (excluded)
%   - vertical accelerations at bus nodes                = 3*nodi
%   - angular gradient (bending) between nodes 3 and 4  = 2
%     (this is a numerical approximation to obtain the outputs
%      of the strain gauges)
%=================================================================
                        if p1_locked==0,
%=================================================================

 inputs=7+2;
 bc = 0*ones(states,inputs);
 % a) payloads input torques in [Nm]
  bc(ntot+12,1) = 1;
  bc(ntot+13,2) = 1;
  bc(ntot+14,3) = 1;
  bc(ntot+15,4) = 1;
 % b) torque wheels input torques  in [Nm]
  bc(ntot+16,5) = 1;
  bc(ntot+17,6) = 1;
  bc(ntot+18,7) = 1;
 % c) differential torque between nodes 3 and 4
  bc(2*ndof-7,8) = -1;bc(3*ndof-13,8) = 1;      % out-of-plane
  bc(2*ndof-6,9) = -1;bc(3*ndof-12,9) = 1;      % in-plane

 % d) forces at carriages  in [N]
 % bc(ntot-3,10)  = 1;
 % bc(ntot,11)    = 1;
 % bc(ntot+3,12)  = 1;

 accel=3;
 measure=7+accel+2;
 cc = 0*ones(7+2,states);
 cstar2=zeros(accel,s  ates);
 %carro=3;cstar1=zeros(carro,states);

 % a) rigid body inertial angles in [deg]
 cc(1,2*ndof-8)=1*invrad;
 cc(2,2*ndof-7)=1*invrad;
 cc(3,2*ndof-6)=1*invrad;

 % b) payloads inertial angles in [deg]
 cc(4,ntot+12)=1*invrad;cc(4,ndof-8)=1*invrad;
 cc(5,ntot+13)=1*invrad;cc(5,ndof-4)=1*invrad;
 cc(6,ntot+14)=1*invrad;cc(6,ntot-6)=1*invrad;
 cc(7,ntot+15)=1*invrad;cc(7,ntot-4)=1*invrad;

 % c) angular gradient (bending) between nodes 3 and 4
 cc(8,2*ndof-7)=-1*invrad/lele;cc(8,3*ndof-13)=1*invrad/lele;
 cc(9,2*ndof-6)=-1*invrad/lele;cc(9,3*ndof-12)=1*invrad/lele;

 % c) carriages accelerations in [m/sec2]
 %cstar1(1,ntot-3)=1;      % at node 6
 %cstar1(2,ntot)  =1;      % at node 7
 %cstar1(3,ntot+3)=1;      % at node 8

 % d) accelerations (3 components) at node 4
 if nele==1,
   pick1=[19:21];
  %pick1=[1:3 7:9 13:15 19:21 25:27];
 elseif nele==2,
   pick1=[37:39];
  %pick1=[1:3 7:9 13:15 19:21 25:27 31:33 37:39 43:45 49:51];
 end
 cstar2(1:accel,pick1)=eye(accel);
 %============================================================
 nmod=20;                   % number of retained modes
 gyromod=evecnc'*GYROintnc2*evecnc;
 newAdampin=Adampin(1:nmod,1:nmod)-gyromod(1:nmod,1:nmod);
 Amodal=[zeros(nmod) eye(nmod);Aomegas(1:nmod,1:nmod) newAdampin];
 Bmodal0=evecnc'*bc;Bmodal0=Bmodal0(1:nmod,:);
 Bmodal=[0*Bmodal0;Bmodal0];
 Cmodal0=cc*evecnc;Cmodal0=Cmodal0(:,1:nmod);
 Cmodal=[Cmodal0 zeros(Cmodal0)];
 %============================================================
 Bc = [zeros(MMintnc2);inv(MMintnc2)]*bc;
 %Cc=[cc                  zeros(cc);
 %    zeros(cc)           cc;
 %    cstar1*element1  cstar1*element2;
 %    cstar2*element1  cstar2*element2];
 Cc=[cc               zeros(cc);
     zeros(cc)        cc;
     cstar2*element1  cstar2*element2];
 [rowb,colb]=size(Bc);[rowc,colc]=size(Cc);
 Dc=zeros(rowc,colb);
 %Dc = [zeros(rowc-accel,colb);
 %       cstar1*inv(MMintnc2)*bc;
 %       cstar2*inv(MMintnc2)*bc];
 Dc = [zeros(rowc-accel,colb);
       cstar2*inv(MMintnc2)*bc];
 %=================================================================
 %================================================================
                         elseif pl_locked==1,
 %================================================================

  inputs=5+2;
  bc = 0*ones(states,inputs+3);
 % a) payloads input torques in [Nm]
  bc(ntot+12,1) = 1;
  bc(ntot+13,2) = 1;
 % b) torque wheels input torques  in [Nm]
  bc(ntot+14,3) = 1;
  bc(ntot+15,4) = 1;
  bc(ntot+16,5) = 1;
 % c) differential torque between nodes 3 and 4
  bc(2*ndof-7,6) = -1;bc(3*ndof-13,6) = 1;
  bc(2*ndof-6,7) = -1;bc(3*ndof-12,7) = 1;
 % d) forces at carriages  in [N]
 % bc(ntot-3,6)  = 1;
 % bc(ntot,7)    = 1;
 % bc(ntot+3,8) = 1;
 % d) bus attitude disturbance torques in [Nm]
  bc(2*ndof-8,8)  = 1;
  bc(2*ndof-7,9)  = 1;
  bc(2*ndof-6,10) = 1;

accel=3;
measure=5+accel+2;
cc = 0*ones(5+2,states);
cstar2=zeros(accel,states);%carro=3;
%cstar1=zeros(carro,states);

% a) rigid body inertial angles in [deg]
cc(1,2*ndof-8)=1*invrad;
cc(2,2*ndof-7)=1*invrad;
cc(3,2*ndof-6)=1*invrad;

% b) payloads inertial angles in [deg]
cc(4,ntot+12)=1*invrad;cc(4,ntot+4)=1*invrad;cc(4,ntot+10)=1*invrad;
cc(4,ntot-6)=1*invrad;
cc(5,ntot+13)=1*invrad;cc(5,ntot+5)=1*invrad;cc(5,ntot+11)=1*invrad;
cc(5,ntot-4)=1*invrad;

% c) angular gradient (bending) between nodes 3 and 4
% horizontal strain on strut
cc(6,2*ndof-7)=-1*invrad/lele;cc(6,3*ndof-13)=1*invrad/lele;
% vertical strain on strut
cc(7,2*ndof-6)=-1*invrad/lele;cc(7,3*ndof-12)=1*invrad/lele;

% c) carriages accelerations in [m/sec2]
%cstar1(1,ntot-3)=1;      % at node 6
%cstar1(2,ntot)  =1;      % at node 7
%cstar1(3,ntot+3)=1;      % at node 8
```

```
% d) accelerations (3 components) at node 4
if nele==1,
  pick1=[19:21];
 %pick1=[1:3 7:9 13:15 19:21 25:27];
elseif nele==2,
  pick1=[37:39];
 %pick1=[1:3 7:9 13:15 19:21 25:27 31:33 37:39 43:45 49:51];
end
cstar2(1:accel,pick1)=eye(accel)·
%==================================  ==============================
Ac=Ac5;
Bc = [zeros(MMintnc5);inv(MMintnc5)]*bc;
%Cc=[cc                zeros(cc);
%    zeros(cc)         cc;
%    cstar1*element15  cstar1*element25;
%    cstar2*element15  cstar2*element25];
Cc=[cc               zeros(cc);
    zeros(cc)        cc;
    cstar2*element15  cstar2*element25];
[rowb,colb]=size(Bc);[rowc,colc]=size(Cc);
Dc=zeros(rowc,colb);
%Dc = [zeros(rowc-carro-accel,colb);
%      cstar1*inv(MMintnc5)*bc;
%      cstar2*inv(MMintnc5)*bc];
Dc = [zeros(rowc-accel,colb);
      cstar2*inv(MMintnc5)*bc];
%=================================================================
                              end
%=================================================================


if bodeplotsOL==1,
%=================================================================
% find minimal realization
%=================================================================
disp('>>>>>>looking for minimal realization<<<<<<<<<<<<');
[Amin,Bmin,Cmin,Dmin]=minreal(Ac,Bc,Cc,Dc);
Ac=Amin;Bc=Bmin;Cc=Cmin;Dc=Dmin;
%=================================================================
disp('>>>>>>computing data for the open-loop Bode plots<<<<<<<<<<<<');
[mag1,fase1]=bode(Ac,Bc,Cc,Dc,1,wbig*tupi);
[mag2,fase2]=bode(Ac,Bc,Cc,Dc,2,wbig*tupi);
%[mag3,fase3]=bode(Ac,Bc,Cc,Dc,3,wbig*tupi);
%[mag4,fase4]=bode(Ac,Bc,Cc,Dc,4,wbig*tupi);
%[mag5,fase5]=bode(Ac,Bc,Cc,Dc,5,wbig*tupi);
%[mag6,fase6]=bode(Ac,Bc,Cc,Dc,6,wbig*tupi);
%[mag7,fase7]=bode(Ac,Bc,Cc,Dc,7,wbig*tupi);
%[mag8,fase8]=bode(Ac,Bc,Cc,Dc,8,wbig*tupi);
%[mag9,fase9]=bode(Ac,Bc,Cc,Dc,9,wbig*tupi);
%[mag10,fase10]=bode(Ac,Bc,Cc,Dc,10,wbig*tupi);
%=================================================================
end
%=================================================================


               if active_suspension==1,

% PART D/E: build SUSPENSION CONTROLLER DYNAMICS and append
%           the controller dynamics to the open-loop dynamics


%=================================================================
disp('<<<<< working on building the controller and >>>>>>>>>');
disp('<<<<< working on the state augmentation      >>>>>>>>>');
%=================================================================
% take vertical acceleration(s) at the suspension carriage(s) from
% the output vector Y of open-loop system
freadv=[rowc-2  rowc-1  rowc];
% column(s) of Bc and Dc relative to the vertical force(s) at the
% suspension carriage(s)
floadv=[colb-2  colb-1  colb];
% columns of Bc and Dc relative to all the other inputs
frestv=[1:colb];
%==============================================================
% input to the controller = P * (output from plant) =
% positive acceleration of each of the  3 suspension carriages
%==============================================================
Pacc=zeros(3,rowc);Pacc(1:3,rowc-2:rowc)=eye(3);

P=Pacc(1,:);freqd=freadv(1);fload=floadv(1);
Kair=Kair;
SUSPCONTROL
APPENDYN3d
Ac=Abig;Bc=Bbig;Cc=Cbig;Dc=Dbig;

P=Pacc(2,:);freqd=freadv(2);fload=floadv(2);
Kair=Kair;
SUSPCONTROL
APPENDYN3d
Ac=Abig;Bc=Bbig;Cc=Cbig;Dc=Dbig;

P=Pacc(3,:);freqd=freadv(3);fload=floadv(3);
Kair=Kair;
SUSPCONTROL
APPENDYN3d
Ac=Abig;Bc=Bbig;Cc=Cbig;Dc=Dbig;
%==============================================================
%  compute closed-loop poles with suspension dynamics only
%==============================================================
polesbig=eig(Abig);
states=states+6;
%==============================================================
%[z1,p1,kk1]=ss2zp(Abig,Bbig,Cbig,Dbig,1);
%[z2,p2,kk2]=ss2zp(Abig,Bbig,Cbig,Dbig,2);
%[z3,p3,kk3]=ss2zp(Abig,Bbig,Cbig,Dbig,3);
%==============================================================
if bodeplotsCL1==1,
%==============================================================
% find minimal realization
%==============================================================
disp('>>>>>>looking for minimal realization<<<<<<<<<<<<');
[Amin,Bmin,Cmin,Dmin]=minreal(Ac,Bc,Cc,Dc);
Ac=Amin;Bc=Bmin;Cc=Cmin;Dc=Dmin;
%==============================================================
disp('>>>>>>computing data for the closed-loop Bode plots<<<<<<<<<<<<');
[mag1c,fase1c]=bode(Ac,Bc,Cc,Dc,1,wbig*tupi);
[mag2c,fase2c]=bode(Ac,Bc,Cc,Dc,2,wbig*tupi);
[mag3c,fase3c]=bode(Ac,Bc,Cc,Dc,3,wbig*tupi);
[mag4c,fase4c]=bode(Ac,Bc,Cc,Dc,4,wbig*tupi);
[mag5c,fase5c]=bode(Ac,Bc,Cc,Dc,5,wbig*tupi);
[mag6c,fase6c]=bode(Ac,Bc,Cc,Dc,6,wbig*tupi);
[mag7c,fase7c]=bode(Ac,Bc,Cc,Dc,7,wbig*tupi);
%[mag8c,fase8c]=bode(Ac,Bc,Cc,Dc,8,wbig*tupi);
%[mag9c,fase9c]=bode(Ac,Bc,Cc,Dc,9,wbig*tupi);
%[mag10c,fase10c]=bode(Ac,Bc,Cc,Dc,10,wbig*tupi);
end
%==============================================================
                      end      % (of no active suspension loop)

                      end      % (of no statespace loop)
```

```
%==================================================================
if timsim==1,
%==================================================================
% compute time response (neglecting non-linear terms) for
% the case pl_locked==1
%==================================================================
disp(' ');
disp('   computing linear response   ');
disp(' ');

HBwth=1;LBwth=0;
if HBwth==1,
 KPfi=310.4901;KDfi=4.28;
 KPtheta=452.5788;KDtheta=6.2387;
end
if LBwth==1,
 KPfi=0.0310;KDfi=0.0428;
 KPtheta=0.0453;KDtheta=0.0624;
end
KP=[KPfi 0;0 KPtheta];
KD=[KDfi 0;0 KDtheta];

% Initial Conditions for full order model
 if active_suspension==1,
x0=zeros(112,1);
x0(46:47,1)=[0;-150]*pi/180;
x0((46+50):(47+50),1)=[0;0]*pi/180;
tet0=x0(47,1);fi0=x0(46,1);
tetf=-30*pi/180;fif=0*pi/180;
 elseif  active_suspension==0,
x0=zeros(100,1);
x0(46:47,1)=[0;-150]*pi/180;
x0((46+50):(47+50),1)=[0;0]*pi/180;
tet0=x0(47,1);fi0=x0(46,1);
tetf=-30*pi/180;fif=0*pi/180;
 end
%==================================================================
% DECENTRALIZED CONTROL SCHEME
% close the loop with high BW PD control at the payload
%  and low BW PD control at the torque wheels
%==================================================================
GAIN=zeros(5,2*states);
GAIN(1,46)=-310.4901;GAIN(1,46+states)=-4.28;
GAIN(2,47)=-452.5788;GAIN(2,47+states)=-6.2387;
GAIN(3,48)=-0.0453;GAIN(3,48+states)=-0.0624;
GAIN(4,49)=-0.0453;GAIN(4,49+states)=-0.0624;
GAIN(5,50)=-0.0453;GAIN(5,50+states)=-0.0624;
ACL=Ac+Bc(:,1:5)*GAIN;
%==================================================================
tim=[0:0.01:4]';Lt=length(tim);Mt=max(tim);
timtstop=2.88;Jtet=0.0430;Jfi=0.0295;
taut=[];tauf=[];pos=[];vel=[];accel=[];
for k=0:0.01:Mt,
   trajectory=7;
    [q1dest,q1desdt,q1desddt]=rampvel(k,timtstop,tet0,tetf);
   trajectory=5;
    [q1desf,q1desdf,q1desddf,torquef]=traject(k,fi0,fif,Jfi);
   torquet=Jtet*q1desddt+452.5788*q1dest+6.2387*q1desdt;
   torquef=Jfi*q1desddf+310.4901*q1desf+4.28*q1desdf;
   taut=[taut;torquet];tauf=[tauf;torquef];
   pos=[pos;q1dest];vel=[vel;q1desdt];accel=[accel;q1desddt];
end
rand('normal');
tau1=rand(Lt,1);tau2=rand(Lt,1);tau3=rand(Lt,1);
tau_dist=[tau1  tau2  tau3];
tau_cont=[tauf*0 taut];
```

```
TAU=[tau_cont tau_dist*0];
%=============================================================
influx=[1:2 8:10];
[ys,xs]=lsim(ACL,Bc(:,influx),Cc,Dc(:,influx),TAU,tim,x0);
plot(tim,xs(:,47)*invrad)
plot(tim,ys(:,5))          % inertial angle
animation(tim,Mt,40,xs,1)
%=============================================================
% now you can compare the linear trajectory tracking (ys, xs vs. tim)
% with the non-linear response (x vs. t from RUNMACE.m)
% load simulazione8
%j=47;plot(t,x(:,j)*invrad),hold,plot(tim,xs(:,j)*invrad),hold
%=============================================================
end


return




%=============================================================



% PART F: append vector of non-linear terms due to payload motion,
%         i.e., Coriolis and centripetal terms


%=============================================================
%disp('>>>>>>computing vector of non-linear terms<<<<<<<<<<<<');
%=============================================================
%        build the non-linear terms in the equations
%        of motion for MACENEW.m
% NOTE:  both payloads must be hinged!
%=============================================================

if pl_locked==0,
  pu=states;totstate=2*(ntot+18)+12;suspstates=nwires*4;
  x=zeros(2*states,1);Fnl=zeros(states,1);
elseif pl_locked==1,
  pu=states;totstate=2*(ntot+18)+12-2;suspstates=nwires*4;
  x=zeros(2*states,1);Fnl=zeros(states,1);
end

if nele==1,
 u4=x(2*ndof-8+pu);      %theta1_dot (central node)
 u5=x(2*ndof-7+pu);      %theta2_dot      "
 u6=x(2*ndof-6+pu);      %theta3_dot      "
 if pl_locked==0,
  u7=x(ntot+16);          %omega1 for 1st wheel
  u8=x(ntot+17);          %omega2 for 2nd wheel
  u9=x(ntot+18);          %omega3 for 3rd wheel
 elseif pl_locked==1,
  u7=x(ntot+14);          %omega1 for 1st wheel
  u8=x(ntot+15);          %omega2 for 2nd wheel
  u9=x(ntot+16);          %omega3 for 3rd wheel
 end
 t11d=x(ndof-8+pu);      %theta1_dot node 1
 t21d=x(ndof-7+pu);      %theta2_dot   "
 t31d=x(ndof-6+pu);      %theta3_dot   "
 t15d=x(3*ndof-8+pu);    %theta1_dot node 5
 t25d=x(3*ndof-7+pu);    %theta2_dot   "
 t35d=x(ntotflex+pu);    %theta3_dot   "
elseif nele==2,
   u4=x(28+pu);             %theta1_dot (central node)
   u5=x(29+pu);             %theta2_dot      "
   u6=x(30+pu);             %theta3_dot       "
   if pl_locked==0,
    u7=x(ntot+16);           %omega1 for 1st wheel
    u8=x(ntot+17);           %omega2 for 2nd wheel
    u9=x(ntot+18);           %omega3 for 3rd wheel
   elseif pl_locked==1,
    u7=x(ntot+14);           %omega1 for 1st wheel
    u8=x(ntot+15);           %omega2 for 2nd wheel
    u9=x(ntot+16);           %omega3 for 3rd wheel
   end
   t11d=x(4+pu)             %theta1_dot node 1
   t21d=x(5+pu);            %theta2_dot    "
   t31d=x(6+pu);            %theta3_dot    "
   t15d=x(52+pu);           %theta1_dot node 5
   t25d=x(53+pu);           %theta2_dot    "
   t35d=x(54+pu);           %theta3_dot    "
  end

  if pl_locked==0,
   f1=x(ntot+12);          %fi payload #1
   t1=x(ntot+13);          %theta    "
   f5=x(ntot+14);          %fi payload #5
   t5=x(ntot+15);          %theta    "
   f1d=x(ntot+12+pu);      %fi_dot payload #1
   t1d=x(ntot+13+pu);      %theta_dot      "
   f5d=x(ntot+14+pu);      %fi_dot payload #5
   t5d=x(ntot+15+pu);      %theta_dot     "
  elseif pl_locked==1,
   f5=x(ntot+12);          %fi payload #5
   t5=x(ntot+13);          %theta    "
   f5d=x(ntot+12+pu);      %fi_dot payload #5
   t5d=x(ntot+13+pu);      %theta_dot     "
  end
  %================================================================


  % PART G: append vector of external forces: gravity + actuation
  %          (not linearized)

  %================================================================
  %disp('>>>>>>computing vector of external forces<<<<<<<<<<<<');
  %================================================================
  Fgravity=zeros(states,1);
  Fgravitysag=zeros(states,1);

                   · if pl_locked==0,

  if nele==1,

   Fgravity([1:ndof-6  ntot+12:ntot+13],1)=...
   [0;
   -Mnode1*grav;
   0;
   grav*(Mbody1*(b1(3))+Mbody2*(c1(3)+A301)+Mbody3*(c1(3)+D301));
   0;
   -grav*(Mbody1*(b1(1))+Mbody2*(c1(1)+A101)+Mbody3*(c1(1)+D101));
   grav*(Mbody2*(A301)+Mbody3*(D301));
   -grav*(Mbody3*(D101*cf1))];

   Fgravity([3*ndof-11:ntotflex  ntot+14:ntot+15],1) =...
   [0;
   -Mnode5*grav;
   0;
   grav*(Mbody1*(b5(3))+Mbody2*(c5(3)+A305)+Mbody3*(c5(3)+D305));
```

```
 0;
 -grav*(Mbody1*(b5(1))+Mbody2*(c5(1)+A105)+Mbody3*(c5(1)+D105));
 grav*(Mbody2*(A305)+Mbody3*(D305));
 -grav*(Mbody3*(D105*cf5))];

 Fgravity(2*ndof-10,1)=-grav*Mnode3;
 Fgravity((2*ndof-8):(2*ndof-6),1)=...
        -grav*Mnode3*[-rH3(3);0;rH3(1)];
 Fgravity(ndof-4,1)=-grav*Mnode2;
 Fgravity(2*ndof-4,1)=-grav*Mnode4;

elseif nele==2,

 Fgravity([1:6  ntot+12:ntot+13],1)=...
 [0;
 -Mnode1*grav;
 0;
 grav*(Mbody1*(b1(3))+Mbody2*(c1(3)+A301)+Mbody3*(c1(3)+D301));
 0;
 -grav*(Mbody1*(b1(1))+Mbody2*(c1(1)+A101)+Mbody3*(c1(1)+D101));
 grav*(Mbody2*(A301)+Mbody3*(D301));
 -grav*(Mbody3*(D101*cf1))];


 Fgravity([49:ntotflex  ntot+14:ntot+15],1) =...
 [0;
 -Mnode5*grav;
 0;
 grav*(Mbody1*(b5(3))+Mbody2*(c5(3)+A305)+Mbody3*(c5(3)+D305));
 0;
 -grav*(Mbody1*(b5(1))+Mbody2*(c5(1)+A105)+Mbody3*(c5(1)+D105));
 grav*(Mbody2*(A305)+Mbody3*(D305));
 -grav*(Mbody3*(D105*cf5))];

 Fgravity(26,1)=-grav*Mnode3;
 Fgravity(28:30,1)=...
        -grav*Mnode3*[-rH3(3);0;rH3(1)];
 Fgravity(14,1)=-grav*Mnode2;
 Fgravity(38,1)=-grav*Mnode4;

end

                      elseif p1_locked==1,

if nele==1,

 Fgravity([1:ndof-6],1)=...
 [0;
 -Mnode1*grav;
 0;
 grav*(Mbody1*(b1(3))+Mbody2*(c1(3)+A301)+Mbody3*(c1(3)+D301));
 0;
 -grav*(Mbody1*(b1(1))+Mbody2*(c1(1)+A101)+Mbody3*(c1(1)+D101))];


 Fgravity([3*ndof-11:ntotflex  ntot+12:ntot+13],1) =...
 [0;
 -Mnode5*grav;
 0;
 grav*(Mbody1*(b5(3))+Mbody2*(c5(3)+A305)+Mbody3*(c5(3)+D305));
 0;
 -grav*(Mbody1*(b5(1))+Mbody2*(c5(1)+A105)+Mbody3*(c5(1)+D105));
 grav*(Mbody2*(A305)+Mbody3*(D305));
 -grav*(Mbody3*(D105*cf5))];

 Fgravity(2*ndof-10,1)=-grav*Mnode3;
```

```
  Fgravity((2*ndof-8):(2*ndof-6),1)=...
        -grav*Mnode3*[-rH3(3);0;rH3(1)];
  Fgravity(ndof-4,1)=-grav*Mnode2;
  Fgravity(2*ndof-4,1)=-grav*Mnode4;

 elseif nele==2,

  Fgravity([1:6],1)=...
  [0;
  -Mnode1*grav;
  0;
  grav*(Mbody1*(b1(3))+Mbody2*(c1(3)+A301)+Mbody3*(c1(3)+D301));
  0;
  -grav*(Mbody1*(b1(1))+Mbody2*(c1(1)+A101)+Mbody3*(c1(1)+D101))];


  Fgravity([49:ntotflex  ntot+12:ntot+13],1) =...
  [0;
  -Mnode5*grav;
  0;
  grav*(Mbody1*(b5(3))+Mbody2*(c5(3)+A305)+Mbody3*(c5(3)+D305));
  0;
  -grav*(Mbody1*(b5(1))+Mbody2*(c5(1)+A105)+Mbody3*(c5(1)+D105));
  grav*(Mbody2*(A305)+Mbody3*(D305));
  -grav*(Mbody3*(D105*cf5))];

  Fgravity(26,1)=-grav*Mnode3;
  Fgravity(28:30,1)=...
        -grav*Mnode3*[-rH3(3);0;rH3(1)];
  Fgravity(ndof-4,1)=-grav*Mnode2;
  Fgravity(38,1)=-grav*Mnode4;

 end
                                    end

%================================================================
% Compute the consistent nodal loads representing the distributed
% effect of gravity (acting as distributed lo d on vertical plane only)
%================================================================
Fcons=zeros(12,1);
Fcons(2,1)=-pA*grav*lele*(1/2);
Fcons(6,1)=-pA*grav*lele*(1/12)*lele;
Fcons(8,1)=-pA*grav*lele*(1/2);
Fcons(12,1)=pA*grav*lele*(1/12)*lele;
%================================================================
Fg=zeros(ntotflex,1);
if nele==1,
Fg(1:12,1)=Fg(1:12,1)+Fcons;Fg(7:18,1)=Fg(7:18,1)+Fcons;
Fg(13:24,1)=Fg(13:24,1)+Fcons;Fg(19:30,1)=Fg(19:30,1)+Fcons;
Fgravitysag(1:30)=Fgravitysag(1:30)+Fg;
Fgravitysag(2)=-grav*Mnode1;
Fgravitysag(ndof-4)=-grav*Mnode2;
Fgravitysag(2*ndof-10)=-grav*Mnode3;
Fgravitysag(2*ndof-4)=-grav*Mnode4;
Fgravitysag(3*ndof-10)=-grav*Mnode5;
elseif nele==2,
Fg(1:12,1)=Fg(1:12,1)+Fcons;Fg(7:18,1)=Fg(7:18,1)+Fcons;
Fg(13:24,1)=Fg(13:24,1)+Fcons;Fg(19:30,1)=Fg(19:30,1)+Fcons;
Fg(25:36,1)=Fg(25:36,1)+Fcons;Fg(31:42,1)=Fg(31:42,1)+Fcons;
Fg(37:48,1)=Fg(37:48,1)+Fcons;Fg(43:54,1)=Fg(43:54,1)+Fcons;
Fgravitysag(1:54)=Fgravitysag(1:54)+Fg;
Fgravitysag(2)=-grav*Mnode1;
Fgravitysag(ndof-4)=-grav*Mnode2;
Fgravitysag(2*ndof-10)=-grav*Mnode3;
Fgravitysag(2*ndof-4)=-grav*Mnode4;
Fgravitysag(3*ndof-10)=-grav*Mnode5;
```

```
end
%=============================================================
% add gravity load to carriages
%=============================================================
Fgravity(ntot-3,1)=-grav*Mcarriage;
Fgravity(ntot,1)  =-grav*Mcarriage;
Fgravity(ntot+3,1)=-grav*Mcarriage;
%=============================================================
% Compute new equilibrium of nodal coordinates of bus centerline
% from gravity_forces (this will be the input state map to the
% non-linear time simulation routine)
%
%       X_present = inv(KK)*(Fg + KK*X_initial)
%
%  where:  Fg        = vector of NODAL loads due to gravity only
%          X_initial = vector of initial displacements as defined
%                      in MAKEMAP (set to zero for the moment)
%=============================================================

          if pl_locked==0,
    KKintnc2=KKintnc2-1.e-7*MMintnc2;
 if nele==1,
  Xnew  =inv(KKintnc2(1:49,1:49))*Fgravitysag(1:49),
  Xnewx =Xnew([1 7 13 19 25],1);
  Xnewy =Xnew([2 8 14 20 26],1);
  Xnewz =Xnew([3 9 15 21 27],1);
  Xnewt1=Xnew([4 10 16 22 28],1);
  Xnewt2=Xnew([5 11 17 23 29],1);
  Xnewt3=Xnew([6 12 18 24 30],1);
  Xneww=Xnew([31  34  37],1);
 elseif nele==2,
  Xnew  =inv(KKintnc2(1:73,1:73))*Fgravitysag(1:73);
  Xnewx =Xnew([1 7 13 19 25 31 37 43 49],1);
  Xnewy =Xnew([2 8 14 20 26 32 38 44 50],1);
  Xnewz =Xnew([3 9 15 21 27 33 39 45 51],1);
  Xnewt1=Xnew([4 10 16 22 28 34 40 46 52],1);
  Xnewt2=Xnew([5 11 17 23 29 35 41 47 53],1);
  Xnewt3=Xnew([6 12 18 24 30 36 42 48 54],1);
 end
                 elseif pl_locked==1,
             KKintnc5=KKintnc5-1.e-7*MMintnc5;
 if nele==1,
  Xnew  =inv(KKintnc5(1:47,1:47))*Fgravitysag(1:47);
  Xnewx =Xnew([1 7 13 19 25],1);
  Xnewy =Xnew([2 8 14 20 26],1);
  Xnewz =Xnew([3 9 15 21 27],1);
  Xnewt1=Xnew([4 10 16 22 28],1);
  Xnewt2=Xnew([5 11 17 23 29],1);
  Xnewt3=Xnew([6 12 18 24 30],1);
  Xneww=Xnew([31  34  37],1);
 elseif nele==2,
  Xnew  =inv(KKintnc5(1:68,1:68))*Fgravitysag(1:68);
  Xnewx =Xnew([1 7 13 19 25 31 37 43 49],1);
  Xnewy =Xnew([2 8 14 20 26 32 38 44 50],1);
  Xnewz =Xnew([3 9 15 21 27 33 39 45 51],1);
  Xnewt1=Xnew([4 10 16 22 28 34 40 46 52],1);
  Xnewt2=Xnew([5 11 17 23 29 35 41 47 53],1);
  Xnewt3=Xnew([6 12 18 24 30 36 42 48 54],1);
 end
                         end

delstatic=grav/(tupi*bounce)^2;
Xvert=Xnewy+delstatic*ones(numnp-3,1);
Sag=[Xnewx  Xnewy  Xnewz  Xnewt1  Xnewt2  Xnewt3];
NormSagY=max(abs(Xnewy))./lengthMACE;
%format long e,Sag,NormSagY,format
```

```
%plot(Xnewy),grid,
%title('static gravity sag of MACE in [mm]'),
%xlabel('FE nodes')

L1=0.2289;rR2w=0.0763;QL1=L1+2*rR2w;
Xnewx1(1)=Xnewx(1);
Xnewx1(2)=QL1+Xnewx(1)+Xnewx(2);
Xnewx1(3)=QL1+Xnewx1(2)+Xnewx(3);
Xnewx1(4)=QL1+Xnewx1(3)+Xnewx(4);
Xnewx1(5)=QL1+Xnewx1(4)+Xnewx(5);
clg; axis; hold off;r=2;axis([-r/100,r,0.07,r/20]);
plot(Xnewx1,Xvert,'-'); hold on;
plot(Xnewx1,Xvert,'o'); hold on;
grid;hold off; axis('normal');title('y vs. x');
%===============================================================

if salva==1,

     save one_g,clear

end

if salva1==1,

     save one_g1,clear

end
```

```
function xprime = MACESIMUL1g(t,x)
%
%------------------------------------------------------------
%                MACE 3-D model in one-g
%
% Author       : Marco B. Quadrelli
% Date         : October 17, 1991
% Last revision: November 26, 1991
%------------------------------------------------------------
% Notes : This version considers the
%         (support + inner stage + outer stage + payload)
%         as a chain of three bodies connected by two 1dof
%         revolute joints.
%         The three bodies are:
%         - joint + support
%         - gimbal 1
%         - gimbal 2 + payload.
%         This version, equal to MACENEW.m - version of July 18,1991 -
%         also models the suspension wires as beam-cable elements.
%         The wire is hinged at the top and has only 1 translation
%         allowed (suspension carriage vertical displacement) and it
%         is hinged at the bottom at its connection to the rigid
%         element of the bus.
%         The suspension carriage and its local control loop
%         impose a vertical displacement from above, so as to
%         bring the carriage to its reference position when it has
%         been removed from it.
%         To date, 29-Aug-1991, the program works only when
%         nelew=1 and nelef=-1(no flexible appendage yet), but
%         nele can be chosen to be any number.
%         The suspension wires are modelled as (1) flexible beam element
%         with a length equal to the whole length and hinged at the end
%         connections.
%         The central node is allowed to store a finite amount of
%         internal angular momentum.
%         All units in S.I.
%------------------------------------------------------------
%
%
%                                                  -> laboratory ceiling
% ______6_______________7_______________8__________
%       |               |               |
%       |               |               |
%       |               |               |
%       |               |               |   ->  suspension wires
%       |               |               |
%       |  torque  -> / | \             |  0  -> hinged payload
%       | wheels     /  \|/ \           | /
%       1=======2=======3=======4=======5/
%      /                                \
%     /                                  \--> spacecraft bus
%    0
%============================================================
% the state vector is: (after imposing constraints)
%  note: this is for the case nelew=1 always
%
%_____________________________________________________________
%                description                   +    d.o.f.
%                                              +
%______________________________________________+______________
%  X = [  (x,y,z,theta1,theta2,theta3)_1       |      1:ndof-6
%         (x,y,z,theta1,theta2,theta3)_2       |   ndof-5:ndof
%         (x,y,z,theta1,theta2,theta3)_3       |   ndof+1:2*ndof-6
%         (x,y,z,theta1,theta2,theta3)_4       |  2*ndof-5:3*ndof-12
%         (x,y,z,theta1,theta2,theta3)_5       | 3*ndof-11:ntotflex
%         (y,theta1,theta3)_6                  |   ntot-3:ntot-1
%         (y,theta1,theta3)_7                  |     ntot:ntot+2
%         (y,theta1,theta3)_8                  |   ntot+3:ntot+5
%          (theta1,theta3)_h1   :hinge_1    |      ntot+6:ntot+7
%          (theta1,theta3)_h2   :hinge_2    |      ntot+8:ntot+9
%          (theta1,theta3)_h3   :hinge_3    |     ntot+10:ntot+11
%          (phi,theta)_#1                   |     ntot+12:ntot+13
%          (phi,theta)_#2                   |     ntot+14:ntot+15
%          (omega1,omega2,omega3)_wheels    |     ntot+16:ntot+18
%          ---------------------------------|------------------------
%          and all the first derivatives    |     ntot+19:2*(ntot+18)
%          ---------------------------------|------------------------
%(nwires*4 internal states of suspension)]|2*(ntot+18):2*(ntot+18)+12
%
%==============================================================
L  = 0.22886;      % length of Lexan member
Lw = 4.6736;       % length of suspension wire
lele=L/nele;
rjFF1=[0;0;0];rjFF5=[0;0;0];

 u4=x(2*ndof-8+states);       %theta1_dot (central node)
 u5=x(2*ndof-7+states);       %theta2_dot      "
 u6=x(2*ndof-6+states);       %theta3_dot      "
 if pl_locked==0,
  u7=x(ntot+16);           %omega1 for 1st wheel
  u8=x(ntot+17);           %omega2 for 2nd wheel
  u9=x(ntot+18);           %omega3 for 3rd wheel
 elseif pl_locked==1,
  u7=x(ntot+14);           %omega1 for 1st wheel
  u8=x(ntot+15);           %omega2 for 2nd wheel
  u9=x(ntot+16);           %omega3 for 3rd wheel
 end
 t11d=x(ndof-8+states);       %theta1_dot node 1
 t21d=x(ndof-7+states);       %theta2_dot   "
 t31d=x(ndof-6+states);       %theta3_dot   "
 t15d=x(3*ndof-8+states);     %theta1_dot node 5
 t25d=x(3*ndof-7+states);     %theta2_dot   '
 t35d=x(ntotflex+states);     %theta3_dot   "

if pl_locked==0,

 f1=x(ntot+12);           %fi payload #1
 t1=x(ntot+13);           %theta    "
 f5=x(ntot+14);           %fi payload #5
 t5=x(ntot+15);           %theta    "
 f1d=x(ntot+12+pu);       %fi_dot payload #1
 t1d=x(ntot+13+pu);       %theta_dot     "
 f5d=x(ntot+14+pu);       %fi_dot payload #5
 t5d=x(ntot+15+pu);       %theta_dot    "

 t1_inertial   = t1+x(6);
 f1_inertial   = f1+x(4);
 t5_inertial   = t5+x(30);
 f5_inertial   = f5+x(28);
 t1d_inertial   = t1+x(6+states);
 f1d_inertial   = f1+x(4+states);
 t5d_inertial   = t5+x(30+states);
 f5d_inertial   = f5+x(28+states);

elseif pl_locked==1,

 f5=x(ntot+12);           %fi payload #5
 t5=x(ntot+13);           %theta    "
 f5d=x(ntot+12+pu);       %fi_dot payload #5
 t5d=x(ntot+13+pu);       %theta_dot    "

 t5_inertial    = t5+x(30);
 f5_inertial    = f5+x(28);
 t5d_inertial   = t5+x(30+states);
 f5d_inertial    = f5+x(28+states);

 end


 f1=0;t1=0;
 phi1=f1;theta1=t1;phi5=f5;theta5=t5;
 %===========================================================
 Fnl=zeros(states,1);Fnltrue=zeros(states,1);
 FnlKane=zeros(states,1);Fnldes=zeros(states,1);
 Fact=zeros(states,1);Factuator=zeros(inputs,1);
 Fgravity=zeros(states,1);Fgravitysag=zeros(states,1);
 Fcons=zeros(12,1);Fg=zeros(ntotflex,1);

                  if p1_locked==0,

 %  q1 = out-of-plane angle phi    q2 = in-plane angle theta
 [q1des1,q1desd1,q1desdd1,tau_reffi]=traject(t,fi0,fif,0.0295);
 [q2des1,q2desd1,q2desdd1,tau_refte]=traject(t,tet0,tetf,0.0430);

 [q1des5,q1desd5,q1desdd5,tau_reffi]=traject(t,fi0,fif,0.0295);
 [q2des5,q2desd5,q2desdd5,tau_refte]=traject(t,tet0,tetf,0.0430);

                  elseif p1_locked==1,

 %  q1 = out-of-plane angle phi    q2 = in-plane angle theta
 [q1des5,q1desd5,q1desdd5,tau_reffi]=traject(t,fi0,fif,0.0295);
 [q2des5,q2desd5,q2desdd5,tau_refte]=traject(t,tet0,tetf,0.0430);

                  end
%===========================================================
[sf1,cf1,st1,ct1,ct1cf1,st1sf1,st1cf1,sf1ct1,st1ct1,sf1cf1,...
        ct1sf1,cf1st1,ct1st1,cf1sf1]=sintax(theta1,phi1);
[sf5,cf5,st5,ct5,ct5cf5,st5sf5,st5cf5,sf5ct5,st5ct5,sf5cf5,...
        ct5sf5,cf5st5,ct5st5,cf5sf5]=sintax(theta5,phi5);
%===========================================================
%  derive transformation matrices between frames
%===========================================================
%  transformation matrix from Fpivot to Fpayload (F=vectrix)
[R2p11,R2p21,R2p31,R2p1]=rotation(phi1,0,theta1);
[R2p15,R2p25,R2p35,R2p5]=rotation(phi5,0,theta5);
CAF1=R2p11;CAF5=R2p15;
CPA1=R2p31;CPA5=R2p35;
%===========================================================
po1= rpayPP1(1);po5=rpayPP5(1);
p1=CAF1*CPA1*rpayPP1;
p5=CAF5*CPA5*rpayPP5;
%===========================================================
% vector from FEM node to (body 1 + body 2) com  in F frame
%===========================================================
rAoF1=rAFF1+CAF1*rAoAA1;
rAoF5=rAFF5+CAF5*rAoAA5;
%===========================================================
% vector from FEM node to payload com in F frame
%===========================================================
rPF1=rAFF1+CAF1*rPAA1+CPA1*CAF1*rBPP1;
rPF5=rAFF5+CAF5*rPAA5+CPA5*CAF5*rBPP5;
%===========================================================
% center of mass of (body 1+ body 2 +body 3) wrt FEM node in F frame
%===========================================================
rcom1=(1/Mnode1)*(Mbody1*rFplusF1+Mbody2*rAoF1+Mbody3*rPF1);
rcom5=(1/Mnode5)*(Mbody1*rFplusF5+Mbody2*rAoF5+Mbody3*rPF5);
%===========================================================
%===========================================================
%        SECOND MOMENTS OF INERTIA
%===========================================================
[DJgimbal21]=reorient(DJgimbal2,CPA1');
[DJgimbal25]=reorient(DJgimbal2,CPA5');
%=================================================================
% inertia of bodies of end nodes 1 or 5 about their com's
DJbody11=DJsupport+DJjs+Daddjs1+Daddsup1;
DJbody15=DJsupport+DJjs+Daddjs5+Daddsup5;
DJbody21=DJgimbal1+Daddg11;
DJbody25=DJgimbal1+Daddg15;
DJbody31=DJpl+DJgimbal21+Daddg21+Daddpay1;
DJbody35=DJpl+DJgimbal25+Daddg25+Daddpay5;
%=================================================================
ma1=[Mbody1;Mbody2;Mbody3];ma5=ma1;
J1=[DJbody11  DJbody21  DJbody31];
J5=[DJbody15  DJbody25  DJbody35];
%=================================================================
%==================================================================
% assemble mass matrices
%=================================================================
%
% Note: Mass # 1 has been substituted with a dummy gimbal
%       on October 30, 1991
%==================================================================
M1(1:8,1:8)=zeros(8);M5(1:8,1:8)=zeros(8);
%=================================================================
% Assemble mass matrix at node 1  and 5 (see my notes)
% state:  x  y  z  tet1  tet2  tet3  phi1  theta1
% state:  x  y  z  tet1  tet2  tet3  phi5  theta5
%=================================================================
rAoAA=rAoAA1;rAFF=rAFF1;rFplusF=rFplusF1;rPAA=rPAA1;rBPP=rBPP1;
[M1]=massconf(ma1,J1,phi1,theta1,rAoAA,rAFF,rFplusF,rPAA,rBPP);
rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
[M5]=massconf(ma5,J5,phi5,theta5,rAoAA,rAFF,rFplusF,rPAA,rBPP);
%=================================================================
if p1_locked==1,
   stit=diag(M1);M1=zeros(8);
   M1(1:3,1:3)=(Mdummy+Mjoints)*eye(3);
   M1(4:6,4:6)=DJatdum;
   M1(7:8,7:8)=diag(stit(7:8));
end
%=================================================================
%=================================================================
%=================================================================
%=================================================================
% Before assembling the flex elements, move rows and columns 7,8
% of M1 to 1st and 2nd place so that the state vector for
% the first node becomes [phi1,theta1,x1,y1,z1,d1,d2,d3]'
%=================================================================
M1=M1([7:8 1:6],:);M1=M1(:,[7:8 1:6]);
%=================================================================


%=================================================================
%
%         FINITE ELEMENT MODEL
%
%=================================================================
%
% Start assembling all flexible parts together
%
%=================================================================
KK=zeros(nfree);MM=zeros(nfree);T=eye(ndof);
KKflex=zeros(nflex);MMflex=zeros(nflex);
KKflexbus=zeros(ntotflex);MMflexbus=zeros(ntotflex);
MMint=zeros(nfree);KKint=zeros(nfree);paw=length(MMint);
DA=zeros(nfree);GYRO=zeros(nfree);DAMP=zeros(nfree);
GYRODAMP=zeros(nfree);DAMPREDUCED=zeros(nflex);
```

```
%============================================================
% A) assemble flexibility of bus
%    (can accept more than one nele elements)
%============================================================
T(1:6,1:6)=TR1;T(ndof-5:ndof,ndof-5:ndof)=TL2;
nbeg=1;nend=ndof;
KKflexbus(nbeg:nend,nbeg:nend)=T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=T'*mflex*T;

T(1:6,1:6) = TR2;T(ndof-5:ndof,ndof-5:ndof) = TL3;
nbeg=ndof-5;nend=2*ndof-6;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

T(1:6,1:6) = TR3;T(ndof-5:ndof,ndof-5:ndof) = TL2;
nbeg=2*ndof-11;nend=3*ndof-12;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

T(1:6,1:6)=TR2;T(ndof-5:ndof,ndof-5:ndof)=TL5;
nbeg=3*ndof-17;nend=ntotflex;
KKflexbus(nbeg:nend,nbeg:nend)=...
KKflexbus(nbeg:nend,nbeg:nend)+T'*kflex*T;
MMflexbus(nbeg:nend,nbeg:nend)=...
MMflexbus(nbeg:nend,nbeg:nend)+T'*mflex*T;

KKflex(1:ntotflex,1:ntotflex)=KKflexbus;
MMflex(1:ntotflex,1:ntotflex)=MMflexbus;
%============================================================
% B) add flexibility due to suspension wires
%============================================================
% Assemble finite element model for suspension wires
% compute melew and kelew of the wire elements in their
% local frames. (-rig refers to the short rigid element)
%============================================================
lelew=Lw/nelew;
%============================================================
%  call : ADDENDUM3d (in DATAFIX.m)
%outputs of ADDENDUM3d:
%              KKw1,KKw2,KKw3,MMw1,MMw2,MMw3,
%              all 12x12, all stiffened by gravity
%nbegattach  = bottom attachment to bus
%nendcarriage= top attachment to carriage

nbegattach=1:3;
nendcarriage=(ntotflex+1):(ntotflex+ntotw-3);
nbeg=[nbegattach nendcarriage];
nend=[nbegattach nendcarriage];
KKflex(nbeg,nend)=KKflex(nbeg,nend)+KKw1;
MMflex(nbeg,nend)=MMflex(nbeg,nend)+MMw1;

nbegattach=(2*ndof-11):(2*ndof-9);
nendcarriage=(ntotflex+ntotw-2):(ntotflex+2*ntotw-6);
nbeg=[nbegattach nendcarriage];
nend=[nbegattach nendcarriage];
KKflex(nbeg,nend)=KKflex(nbeg,nend)+KKw2;
MMflex(nbeg,nend)=MMflex(nbeg,nend)+MMw2;

nbegattach=(ntotflex-5):(ntotflex-3);
nendcarriage=(ntotflex+2*ntotw-5):(ntotflex+3*ntotw-9);
nbeg=[nbegattach nendcarriage];
nend=[nbegattach nendcarriage];
```

```
KKflex(nbeg,nend)=KKflex(nbeg,nend)+KKw3;
MMflex(nbeg,nend)=MMflex(nbeg,nend)+MMw3;
%==============================================================
% augment to include end payloads and wheels
KK(3:nfree-5,3:nfree-5)=KKflex;
MM(3:nfree-5,3:nfree-5)=MMflex;
%==============================================================
% Lump additional mass into finite element model
%==============================================================
% a) first lump spacecraft mass at nodes 1 to 5
MM(1:8,1:8)=MM(1:8,1:8)+M1;

M2=M4;
nbeg=ndof-3;nend=ndof+2;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M2;

nbeg=2*ndof-9;nend=2*ndof-4;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M3new;

nbeg=3*ndof-15;nend=3*ndof-10;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M4;

nbeg=4*ndof-21;nend=ntot-2;
MM(nbeg:nend,nbeg:nend)=MM(nbeg:nend,nbeg:nend)+M5(1:6,1:6);

% dof's theta and phi for payload at node 5 are placed after the
% flex elements in KK
nbeg=[nflex+3 nflex+4];nend=[nflex+3 nflex+4];
MM(nbeg,nend)=MM(nbeg,nend)+M5(7:8,7:8);

nbeg=[(4*ndof-21):(4*ndof-21+5)];nend=[nflex+3 nflex+4];
MM(nbeg,nend)=MM(nbeg,nend)+M5(1:6,7:8);

nbeg=[nflex+3 nflex+4];nend=[(4*ndof-21):(4*ndof-21+5)];
MM(nbeg,nend)=MM(nbeg,nend)+M5(7:8,1:6);
%==============================================================
% arrange mass and stiffness matrices in a form ready for
% integration, i.e. move rigid body dof's after flex. dof's:
% state vector =
%  (flex. coord's + phi1,theta1,phi5,theta5,rate1,rate2,rate3)'
%         payload   (    #1    ) (    #2    )
%                                torque wheel   (#1)  (#2)  (#3)
%==============================================================
MMint=zeros(nfree);KKint=zeros(nfree);paw=length(MMint);
DA=zeros(nfree);GYRO=zeros(nfree);DAMP=zeros(nfree);
GYRODAMP=zeros(nfree);DAMPREDUCED=zeros(nflex);
%==============================================================
% move payloads dof's after flex dof's
%==============================================================
nbeg=[3:(nflex+2) 1 2 (nflex+3) (nflex+4)];nend=nbeg;
MMintn=MM(nbeg,nend);
KKintn=KK(nbeg,nend);
%==============================================================
% move hinges dof's after flex dof's and before payloads dof's
%==============================================================
nbeg=[1:ntotflex ntot:ntot+5 ntot+9:ntot+14 ntot+18:ntot+23...
ntotflex+1:ntot-1 ntot+6:ntot+8 ntot+15:ntot+17 ntot+24:nfree-3];
nend=nbeg;
MMintnw=MMintn(nbeg,nend);
KKintnw=KKintn(nbeg,nend);
puw=length(MMintnw);
%==============================================================
% add wheels dof's
KKint=[KKintnw                          zeros(length(KKintnw),3);
       zeros(3,length(KKintnw))         zeros(3)];
MMint=[MMintnw                          zeros(length(MMintnw),3);
```

```
        zeros(3,length(MMintnw))      M3diag2];

 nbeg=[(ntotflex/2+1):(ntotflex/2+3)];
 nend=[(nfree-2):nfree];
 MMint(nbeg,nend)=M3(4:6,7:9);

 nbeg=[(nfree-2):nfree];
 nend=[(ntotflex/2+1):(ntotflex/2+3)];
 MMint(nbeg,nend)=M3(4:6,7:9)';
 %=============================================================
 % hinge suspension wires at the top (equivalent to removing
 %  translational dof's) but keep vertical translation
 %  and do not remove torsion from the wires yet
 %=============================================================
 clamp1=[1:ntot-4 ntot-2 ntot:ntot+2 ntot+4 ntot+6:ntot+8 ntot+10...
         ntot+12:nfree];
 KKintnc1=KKint(clamp1,clamp1);
 MMintnc1=MMint(clamp1,clamp1);
 puc1=length(MMintnc1);
 %=============================================================
 % add carriage mass along the vertical direction
 %=============================================================
 MMintnc1(ntot-3,ntot-3)=MMintnc1(ntot-3,ntot-3)+Mcarriage;
 MMintnc1(ntot+1,ntot+1)=MMintnc1(ntot+1,ntot+1)+Mcarriage;
 MMintnc1(ntot+5,ntot+5)=MMintnc1(ntot+5,ntot+5)+Mcarriage;
 %=============================================================
 %  remove torsion from the wires and hinges dof's
 %=============================================================
 clamp2=[1:ntot-2 ntot:ntot+2 ntot+4:ntot+6 ntot+8:ntot+9...
         ntot+11:ntot+12 ntot+14:ntot+15 ntot+17:ntot+24];
 KKintnc2=KKintnc1(clamp2,clamp2);
 MMintnc2=MMintnc1(clamp2,clamp2);
 puc2=length(MMintnc2);
 %=============================================================
 % clamp payload #1 only on full model
 %=============================================================
 clamp5=[1:45 48:puc2];
 %=============================================================
 % add air spring for purposes of static and dynamic analysis
 %=============================================================
 KKintnc2(ntot-3,ntot-3)=KKintnc2(ntot-3,ntot-3)+Kair;
 KKintnc2(ntot,ntot)=KKintnc2(ntot,ntot)+Kair;
 KKintnc2(ntot+3,ntot+3)=KKintnc2(ntot+3,ntot+3)+Kair;
 %=============================================================
 KKintnc5=KKintnc2(clamp5,clamp5);
 MMintnc5=MMintnc2(clamp5,clamp5);
 puc5=length(MMintnc5);
 %=============================================================
 %=============================================================
 % Now introduce damping and Gyroscopic matrices
 % Also introduce geometric stiffness matrix due to g load
 %=============================================================
 DA=zeros(nfree);GYRO=zeros(nfree);DAMP=zeros(nfree);
 GYRODAMP=zeros(nfree);DAMPREDUCED=zeros(nflex);
 %=============================================================
 % The torque wheels contribute with zero stiffness. But they
 % contribute with a gyroscopic matrix GY. Assemble GY.
 %=============================================================
 spin1=u7;spin2=u8;spin3=u9;
 APw1=spin1*Jwa1*Aw1;BPw1=spin1*Jwa1*Bw1;CPw1=spin1*Jwa1*Cw1;
 APw2=spin2*Jwa2*Aw2;BPw2=spin2*Jwa2*Bw2;CPw2=spin2*Jwa2*Cw2;
 APw3=spin3*Jwa3*Aw3;BPw3=spin3*Jwa3*Bw3;CPw3=spin3*Jwa3*Cw3;
 [gyroblock]=skew([APw1+APw2+APw3;BPw1+BPw2+BPw3;CPw1+CPw2+CPw3]);
 GYRO(2*ndof-8:2*ndof-6,2*ndof-8:2*ndof-6)=gyroblock;
 GYROpic=zeros(KKintnw);
 GYROpic(2*ndof-8:2*ndof-6,2*ndof-8:2*ndof-6)=gyroblock;
```

```
GYROint=[GYROpic                       zeros(length(KKintnw),3);
         zeros(3,length(KKintnw))      zeros(3)];
GYROintnc1=GYROint(clamp1,clamp1);
GYROintnc2=GYROintnc1(clamp2,clamp2);
GYROintnc5=GYROintnc2(clamp5,clamp5);
%=================================================================
% add proportional damping
% note: use K not stiffened by gravity
%=================================================================
%alfa=.10;beta=1.e-5;
%DAMPREDUCED=alfa*MMintnw+beta*KKintnw;
%DAMPint=[DAMPREDUCED                   zeros(length(KKintnw),3);
%         zeros(3,length(KKintnw))      zeros(3)];
%DAMPintnc1=DAMPint(clamp1,clamp1);
%DAMPintnc2=DAMPintnc1(clamp2,clamp2);
%DAMPintnc5=DAMPintnc2(clamp5,clamp5);
%=================================================================
%GYRODAMPintnc2=GYROintnc2+DAMPintnc2;
%GYRODAMPintnc5=GYROintnc5+DAMPintnc5;
%=================================================================
%=================================================================
if pl_locked==0,
        GYRODAMPintnc2=GYROintnc2+DAMPMAT;
elseif pl_locked==1,
        GYRODAMPintnc5=GYROintnc5+DAMPMAT;
end
%=================================================================
%                    END ASSEMBLY
%=================================================================



% PART F: append vector of non-linear terms due to payload motion,
%         i.e., Coriolis and centripetal terms


%=================================================================
%        build the non-linear terms in the equations
%        of motion for MACENEW.m
% NOTE:  both payloads must be at least hinged
%=================================================================
%=================================================================
% compute the non-linear terms for the central node
%=================================================================
SUM1=u4*pw1+u5*qw1+u6*rw1+2*u7;
SUM2=u4*pw2+u5*qw2+u6*rw2+2*u8;
SUM3=u4*pw3+u5*qw3+u6*rw3+2*u9;
SUM41=u5*Aw1+u6*Bw1;
SUM42=u5*Aw2+u6*Bw2;
SUM43=u5*Aw3+u6*Bw3;
SUM51=-u4*Aw1+u6*Cw1;
SUM52=-u4*Aw2+u6*Cw2;
SUM53=-u4*Aw3+u6*Cw3;
SUM61=-u4*Bw1-u5*Cw1;
SUM62=-u4*Bw2-u5*Cw2;
SUM63=-u4*Bw3-u5*Cw3;
ADD11=(Jwa1-Jwt1)*SUM41*SUM1;
ADD12=(Jwa2-Jwt2)*SUM42*SUM2;
ADD13=(Jwa3-Jwt3)*SUM43*SUM3;
ADD21=(Jwa1-Jwt1)*SUM51*SUM1;
ADD22=(Jwa2-Jwt2)*SUM52*SUM2;
ADD23=(Jwa3-Jwt3)*SUM53*SUM3;
ADD31=(Jwa1-Jwt1)*SUM61*SUM1;
ADD32=(Jwa2-Jwt2)*SUM62*SUM2;
ADD33=(Jwa3-Jwt3)*SUM63*SUM3;
```

```
SUM4=ADD11+ADD12+ADD13;
SUM5=ADD21+ADD22+ADD23;
SUM6=ADD31+ADD32+ADD33;
%=============================================================
% for the central node (Coriolis and centripetal)
%=============================================================
if flexibility==0,

 Fnl(2*ndof-8:2*ndof-6,1)=zeros(3,1);

elseif flexibility==1,

 Fnl(2*ndof-8,1)=...
      DJ(1,3)*u4*u5+DJ(2,3)*(u5^2-u6^2)+u5*u6*(DJ(3,3)-DJ(2,2))...
      -DJ(1,2)*u4*u6+SUM4;
 Fnl(2*ndof-7,1)=...
      DJ(1,2)*u5*u6+DJ(1,3)*(u6^2-u4^2)+u4*u6*(DJ(1,1)-DJ(3,3))...
      -DJ(2,3)*u4*u5+SUM5;
 Fnl(2*ndof-6,1)=...
      DJ(2,3)*u4*u6+DJ(1,2)*(u4^2-u5^2)+u4*u5*(DJ(2,2)-DJ(1,1))...
      -DJ(1,3)*u5*u6+SUM6;

end
%=============================================================
% for the end nodes (Coriolis and centripetal), non-linear terms in
% payload angle and rates
%=============================================================

                         if pl_locked==0,
%=============================================================
% payload #1
%=============================================================
  dai1=[1:ndof-6  ntot+12:ntot+13];
rAoAA=rAoAA1;rAFF=rAFF1;rFplusF=rFplusF1;rPAA=rPAA1;rBPP=rBPP1;
[Fnl(dai1,1),verf1,verfg1]=...
fz(ma1,J1,t11d,t21d,t31d,f1,t1,f1d,t1d,rAoAA,rAFF,rFplusF,rPAA,rBPP);

[FnlKane(dai1,1),Fnlrot1,Fnltrans1]=...
Fnonlin(ma1,J1,t11d,t21d,t31d,f1,t1,f1d,t1d,rAoAA,rAFF,rFplusF,rPAA,rBPP);
%=============================================================
% payload #2
%=============================================================
  dai2=[3*ndof-11:ntotflex  ntot+14:ntot+15];
rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
[Fnl(dai2,1),verf5,verfg5] =...
fz(ma5,J5,t15d,t25d,t35d,f5,t5,f5d,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);

[FnlKane(dai2,1),Fnlrot5,Fnltrans5]=...
Fnonlin(ma5,J5,t15d,t25d,t35d,f5,t5,f5d,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);
%=============================================================
                         elseif pl_locked==1,
%=============================================================
% payload #2
%=============================================================
  dai2=[3*ndof-11:ntotflex  ntot+12:ntot+13];
rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
[Fnl(dai2,1),verf5,verfg5] =...
fz(ma5,J5,t15d,t25d,t35d,f5,t5,f5d,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);

[FnlKane(dai2,1),Fnlrot5,Fnltrans5]=...
Fnonlin(ma5,J5,t15d,t25d,t35d,f5,t5,f5d,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);
%=============================================================
                         end
%=============================================================
% PART G: append vector of external forces: non-linear + actuation
%         (not linearized)

% compute desired non-linear terms

                        if pl_locked==0,
%================================================================
% payload #1
%================================================================
f1des=q1des1; -
f1ddes=q1desd1;
f1dddes=q1desdd1;
t1des=q2des1;
t1ddes=q2desd1;
t1dddes=q2desdd1;

  dai1=[1:ndof-6  ntot+12:ntot+13];
rAoAA=rAoAA1;rAFF=rAFF1;rFplusF=rFplusF1;rPAA=rPAA1;rBPP=rBPP1;
[Fnldes(dai1,1),verf1des,verfg1des]=...
fz(ma1,J1,0,0,0,f1des,t1des,f1ddes,t1ddes,rAoAA,rAFF,rFplusF,rPAA,rBPP);

qdes1=[q1des1;q2des1];
qdesd1=[q1desd1;q2desd1];
qdesdd1=[q1desdd1;q2desdd1];
q1=x(ntot+12:ntot+13,1);
qdot1=x((ntot+12+states):(ntot+13+states),1);
qtilde1=q1-qdes1;
qtilded1=qdot1-qdesd1;
%================================================================
% payload #2
%================================================================
f5des=q1des5;
f5ddes=q1desd5;
f5dddes=q1desdd5;
t5des=q2des5;
t5ddes=q2desd5;
t5dddes=q2desdd5;

  dai2=[3*ndof-11:ntotflex  ntot+14:ntot+15];
rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
[Fnldes(dai2,1),verf5des,verfg5des] =...
fz(ma5,J5,0,0,0,f5des,t5des,f5ddes,t5ddes,rAoAA,rAFF,rFplusF,rPAA,rBPP);

qdes5=[q1des5;q2des5];
qdesd5=[q1desd5;q2desd5];
qdesdd5=[q1desdd5;q2desdd5];
q5=x(ntot+14:ntot+15,1);
qdot5=x((ntot+14+states):(ntot+15+states),1);
qtilde5=q5-qdes5;
qtilded5=qdot5-qdesd5;

rAoAA=rAoAA1;rAFF=rAFF1;rFplusF=rFplusF1;rPAA=rPAA1;rBPP=rBPP1;
[M1des]=...
  massconf(ma1,J1,q1des1,q2des1,rAoAA,rAFF,rFplusF,rPAA,rBPP);
rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
[M5des]=...
  massconf(ma5,J5,q1des5,q2des5,rAoAA,rAFF,rFplusF,rPAA,rBPP);
M1d=M1des(7:8,7:8);
M5d=M5des(7:8,7:8);
%================================================================
                        elseif pl_locked==1,
%================================================================
% payload #2
%================================================================
f5des=q1des5;
f5ddes=q1desd5;
```

```
f5dddes=q1desdd5;
t5des=q2des5;
t5ddes=q2desd5;
t5dddes=q2desdd5;

  dai2=[3*ndof-11:ntotflex  ntot+12:ntot+13];
rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
[Fnldes(dai2,1),verf5des,verfg5des] =...
fz(ma5,J5,0,0,0,f5des,t5des,f5ddes,t5d,rAoAA,rAFF,rFplusF,rPAA,rBPP);

qdes5=[q1des5;q2des5];
qdesd5=[q1desd5;q2desd5];
qdesdd5=[q1desdd5;q2desdd5];
q5=x(ntot+12:ntot+13,1);
qdot5=x((ntot+12+states):(ntot+13+states),1);
qtilde5=q5-qdes5;
qtilded5=qdot5-qdesd5;

rAoAA=rAoAA5;rAFF=rAFF5;rFplusF=rFplusF5;rPAA=rPAA5;rBPP=rBPP5;
[M5des]=...
  massconf(ma5,J5,q1des5,q2des5,rAoAA,rAFF,rFplusF,rPAA,rBPP);
M5d=M5des(7:8,7:8);
%===============================================================
                          end


% PART G: append vector of external forces: gravity + actuation
%         (not linearized)
%===============================================================
d1=po1;d5=po5;a1=rAoAA1;a5=rAoAA5;
b1=rFplusF1;c1=rAFF1;b5=rFplusF5;c5=rAFF5;
A101=a1(1);
A201=a1(2)*cf1-a1(3)*sf1;
A301=a1(2)*sf1+a1(3)*cf1;
A105=a5(1);
A205=a5(2)*cf5-a5(3)*sf5;
A305=a5(2)*sf5+a5(3)*cf5;
D101=d1*ct1;
D1t1=d1*st1;
D201=d1*st1*cf1;
D2f1=d1*st1*sf1;
D2t1=d1*ct1*cf1;
D301=D2f1;
D3f1=D201;
D3t1=d1*ct1*sf1;
D105=d5*ct5;
D1t5=d5*st5;
D205=d5*st5*cf5;
D2f5=d5*st5*sf5;
D2t5=d5*ct5*cf5;
D305=D2f5;
D3f5=D205;
D3t5=d5*ct5*sf5;
%===============================================================

                  if p1_locked==0,

 Fgravity([1:ndof-6  ntot+12:ntot+13],1)=...
 [0;
 -Mnode1*grav;
 0;
 grav*(Mbody1*(b1(3))+Mbody2*(c1(3)+A301)+Mbody3*(c1(3)+D301));
 0;
 -grav*(Mbody1*(b1(1))+Mbody2*(c1(1)+A101)+Mbody3*(c1(1)+D101));
 grav*(Mbody2*(A301)+Mbody3*(D301));
 -grav*(Mbody3*(D101*cf1))];
 Fgravity([3*ndof-11:ntotflex  ntot+14:ntot+15],1) =...
 [0;
 -Mnode5*grav;
 0;
 grav*(Mbody1*(b5(3))+Mbody2*(c5(3)+A305)+Mbody3*(c5(3)+D305));
 0;
 -grav*(Mbody1*(b5(1))+Mbody2*(c5(1)+A105)+Mbody3*(c5(1)+D105));
 grav*(Mbody2*(A305)+Mbody3*(D305));
 -grav*(Mbody3*(D105*cf5))];

 Fgravity(2*ndof-10,1)=-grav*Mnode3;
 Fgravity((2*ndof-8):(2*ndof-6),1)=...
       -grav*Mnode3*[-rH3(3);0;rH3(1)];
 Fgravity(ndof-4,1)=-grav*Mnode2;
 Fgravity(2*ndof-4,1)=-grav*Mnode4;

                      elseif p1_locked==1,


 Fgravity([1:ndof-6],1)=...
 [0;
 -Mnode1*grav;
 0;
 grav*(Mbody1*(b1(3))+Mbody2*(c1(3)+A301)+Mbody3*(c1(3)+D301));
 0;
 -grav*(Mbody1*(b1(1))+Mbody2*(c1(1)+A101)+Mbody3*(c1(1)+D101))];


 Fgravity([3*ndof-11:ntotflex  ntot+12:ntot+13],1) =...
 [0;
 -Mnode5*grav;
 0;
 grav*(Mbody1*(b5(3))+Mbody2*(c5(3)+A305)+Mbody3*(c5(3)+D305));
 0;
 -grav*(Mbody1*(b5(1))+Mbody2*(c5(1)+A105)+Mbody3*(c5(1)+D105));
 grav*(Mbody2*(A305)+Mbody3*(D305));
 -grav*(Mbody3*(D105*cf5))];

 Fgravity(2*ndof-10,1)=-grav*Mnode3;
 Fgravity((2*ndof-8):(2*ndof-6),1)=...
       -grav*Mnode3*[-rH3(3);0;rH3(1)];
 Fgravity(ndof-4,1)=-grav*Mnode2;
 Fgravity(2*ndof-4,1)=-grav*Mnode4;

                             end

%=============================================================
% Compute the consistent nodal loads representing the distributed
% effect of gravity (acting as distributed load on vertical plane only)
%=============================================================
Fcons(2,1)=-pA*grav*lele*(1/2);
Fcons(6,1)=-pA*grav*lele*(1/12)*lele;
Fcons(8,1)=-pA*grav*lele*(1/2);
Fcons(12,1)=pA*grav*lele*(1/12)*lele;
Fg(1:12,1)=Fg(1:12,1)+Fcons;Fg(7:18,1)=Fg(7:18,1)+Fcons;
Fg(13:24,1)=Fg(13:24,1)+Fcons;Fg(19:30,1)=Fg(19:30,1)+Fcons;
Fgravity(1:30)=Fgravity(1:30)+Fg;
%=============================================================
% add gravity load to carriages
%=============================================================
Fgravity(ntot-3,1)=-grav*Mcarriage;
Fgravity(ntot,1)  =-grav*Mcarriage;
Fgravity(ntot+3,1)=-grav*Mcarriage;
%=============================================================
%=================================================================
%Factuator(1,1) =   out-of-plane torque
%Factuator(2,1) =   in-plane torque
%=================================================================
%  for smoothed bang-bang slew compute the reference torque
%  a-la Junkins, Bang & Rhaman
%  p=[];for t=0:0.01:1,[Q]=smooth(t,alfa,.5);p=[p Q];end
%tau_max=0.2150*.9;   % take only 90% of reference value used in slew
%alfa=.25/3;
%[smoothing]=smooth(t,alfa,timtstop);
%if tetf >= tet0,
%  tau_ref = +tau_max*smoothing;
%elseif tetf <= tet0,
%  tau_ref = -tau_max*smoothing;
%end
%=================================================================

                    if pl_locked==0,
%=================================================================
KPfi=0;KDfi=0;KPtheta=1000;KDtheta=100;
Factuator(1,1)=-KPfi*qtilde1(1)-KDfi*qdot1(1);
Factuator(2,1)=-KPtheta*qtilde1(2)-KDtheta*qdot1(2);
Factuator(3,1)=-KPfi*qtilde5(1)-KDfi*qdot5(1);
Factuator(4,1)=-KPtheta*qtilde5(2)-KDtheta*qdot5(2);
%=================================================================
Fact(ntot+12,1)=Factuator(1,1);
Fact(ntot+13,1)=Factuator(2,1);
Fact(ntot+14,1)=Factuator(3,1);
Fact(ntot+15,1)=Factuator(4,1);
Fact(ntot+16,1)=0;
Fact(ntot+17,1)=0;
Fact(ntot+18,1)=0;

vec1=x((puc2+1):(2*puc2),1);        % vector of velocities
invM=inv(MMintnc2);
vectors1=Fact+Fgravity-Fnl;         % no linearized stiffening effect!
vectors2=-KKintnc2*x(1:puc2,1)-GYRODAMPintnc2*x((puc2+1):(2*puc2),1);
vector=vectors1+vectors2;
vec2=invM*vector;


% inertial accelerations
 t1dd_inertial    = vec2(6)+vec2(ntot+13);
 f1dd_inertial    = vec2(4)+vec2(ntot+12);
 t5dd_inertial  . = vec2(30)+vec2(ntot+15);
 f5dd_inertial    = vec2(28)+vec2(ntot+14);

xprime = [vec1;vec2];
%=================================================================

                    elseif pl_locked==1,

%=================================================================
% inertial angular velocities, as seen by the rate gyro
invel(1)=qtilded5(1)+x(28+states);
invel(2)=qtilded5(2)+x(30+states);
%=================================================================
% Note: gains computed assuming the payload to be an independent
%        rigid body
% i.e. neglect Mre*qdd_e in 'r' equations
%
% out-of-plane low-bandwidth control
%  bandwidth = 1/10 of 1st flex mode (at 1.6328 Hz)=.16328*tupi rad/s
```

```
 %  inertia about X = 0.0295 Kgm2
 %  zeta = 0.7071
 %                KPfi=0.0310;KDfi=0.0428;
 % out-of-plane high-bandwidth control
 %  bandwidth = 10 times 1st flex mode (at 1.6328 Hz)=16.328*tupi rad/s
 %  inertia about X = 0.0295 Kgm2
 %  zeta = 0.7071
 %                KPfi=310.4901;KDfi=4.28;
 % in-plane low-bandwidth control
 %  bandwidth =-1/10 of 1st flex mode (at 1.6328 Hz)=.16328*tupi rad/s
 %  inertia about Z = 0.0430 Kgm2
 %  zeta = 0.7071
 %                KPtheta=0.0453;KDtheta=0.0624;
 % in-plane high-bandwidth control
 %  bandwidth = 10 times 1st flex mode (at 1.6328 Hz)=16.328*tupi rad/s
 %  inertia about Z = 0.0430 Kgm2
 %  zeta = 0.7071
 %                KPtheta=452.5788;KDtheta=6.2387;
 % gains:
 %
 %  KP=J*(2*pi*bandwidth)^2             KD=2*zeta*sqrt(J*KP)
 %=============================================================
 if HBwth==1,
  KPfi=310.4901;KDfi=4.28;
  KPtheta=452.5788;KDtheta=6.2387;
 end
 if LBwth==1,
  KPfi=0.0310;KDfi=0.0428;
  KPtheta=0.0453;KDtheta=0.0624;
 end
 KP=[KPfi 0;0 KPtheta];
 KD=[KDfi 0;0 KDtheta];

 tau=[tau_reffi;tau_refte];
 Jiner=[0.0295  0;0  0.0430];
 FDBK=-KP*qtilde5-KD*qtilded5;        % feedback controller (relative)
 FFWD1=tau;                           % feedforward controller #1
 FFWD2=Fnl(7:8);                      % feedforward controller #2
 FFWD3=-Fgravity(ntot+12:ntot+13,1);% feedforward controller #3
 FFWD=FFWD1+FFWD3;
 %=============================================================
 % this was implemented on the file simulazione.mat (Nov-15-91)
 %     i.e. bang-bang torque plus PD term
 % note that we are feeding back the relative angle and rate
 %Factuator(1,1)=-KPfi*qtilde5(1)-KDfi*invel(1);
 %Factuator(2,1)=tau_refte-KPtheta*(q5(2)-qref)-KDtheta*(qdot5(2)-qrefd);
 %=============================================================
 % this was implemented on the file simulazione0.mat (Nov-18-91)
 %     i.e. simple position control using PD and trajectory=6
 % note that we are feeding back the relative angle and rate
 %Factuator(1,1)=-KPfi*qtilde5(1)-KDfi*invel(1);
 %Factuator(2,1)=-KPtheta*qtilde5(2)-KDtheta*qdot5(2);
 %=============================================================
 % this was implemented on the file simulazione2.mat (Nov-20-91)
 %     i.e. bang-bang torque plus PD term
 % note that we are feeding back the relative angle and rate
 %Factuator(1,1)=-KPfi*qtilde5(1)-KDfi*invel(1);
 %Factuator(2,1)=tau_ref;%-KPtheta*(q5(2)-qref)-KDtheta*(qdot5(2)-qrefd);
 %=============================================================
 % this was implemented on the file simulazione3.mat (Nov-26-91)
 % simple position control on trajectory=#7 using PD law
 % note that we are feeding back the relative angle and inertial rate
 %Factuator(1,1)=-KPfi*qtilde5(1)-KDfi*invel(1);
 %Factuator(2,1)=-KPtheta*qtilde5(2)-KDtheta*invel(2);
 %=============================================================
 % this was implemented on the file simulazione4.mat (Nov-29-91)
 % simple position control on trajectory=#7 using PD law
 % note that we are feeding back the relative angle and rate
 %Factuator(1,1)=-KPfi*qtilde5(1)-KDfi*invel(1);
 %Factuator(2,1)=tau_refte-KPtheta*qtilde5(2)-KDtheta*invel(2);
 %===============================================================
 % this was implemented on the file simulazione6.mat (Dec-16-91)
 % simple trajectory control on trajectory=#7 using
 % high-bandwidth PD law and feedforward
 % note that we are feeding back the relative angle and rate
 % and canceling non-linear gravity terms
 %Factuator(1,1)=0;
 %Factuator(2,1)=tau_refte-KPtheta*qtilde5(2)-KDtheta*qtilded5(2)...
 %                -Fgravity(ntot+13,1);
 %===============================================================
 % this was implemented on the file simulazione8.mat (Dec-20-91)
 % simple trajectory control on trajectory=#7 using
 % high-bandwidth PD law and feedforward
 % note that we are feeding back the relative angle and rate
 % NO canceling non-linear gravity terms
 Factuator(1,1)=0;
 Factuator(2,1)=tau_refte-KPtheta*qtilde5(2)-KDtheta*qtilded5(2);

 %===============================================================
 Fact(ntot+12,1)=Factuator(1,1);
 Fact(ntot+13,1)=Factuator(2,1);
 Fact(ntot+14,1)=0;
 Fact(ntot+15,1)=0;
 Fact(ntot+16,1)=0;

vec1=x((puc5+1):(2*puc5),1);        % vector of velocities
invM=inv(MMintnc5);
vectors1=Fact+Fgravity-FnlKane;     % no linearized stiffening effect!
vectors2=-KKintnc5*x(1:puc5,1)-GYRODAMPintnc5*x((puc5+1):(2*puc5),1);
vector=vectors1+vectors2;
vec2=invM*vector;


% inertial accelerations
 t5dd_inertial    = vec2(39)+vec2(45)+vec2(ntot+13),
 f5dd_inertial    = vec2(38)+vec2(44)+vec2(ntot+12);

xprime = [vec1;vec2];
%===============================================================

                    end

%===============================================================

if see_inside==1,
  t,format long,[x(1:states) FnlKane],format
end
```

```
%------------------------------------------------------------------
% date    : August 16,1991
% author : M.B. Quadrelli
% revised: October 15, 1991
% done for: cable.m and MACESUSP.m
%------------------------------------------------------------------
%=============================================================
% Assemble controller dynamics of the suspension device
% (see my notes).
%  The low-pass filter in the acceleration feedback loop is
%  a 4 pole Cauer-elliptic filter (use the Matlab 'ellip' function).
%  The low-pass filter in the acceleration feedback loop is
%  a lead-lag filter with two poles.
%  Assume 100 Hz sampling frequency
%  All data in S.I.
%=============================================================
% Active electromechanical part (containing controller dynamics)
%=============================================================
samplerate=100;
npoles=4;                        % # of filter poles
Rp=0.5;                          % db of ripple in the passband
Rs=20;                           % stopband Rs db down
cutoff=35;                       % cutoff frequency in [Hz]
Wn=cutoff/(0.5*samplerate);      %normalized cutoff frequency in [Hz]
numf=1;
denf=[1 (127/tupi)+(254/tupi) (127/tupi)*(254/tupi)];
[Af,Bf,Cf,Df]=tf2ss(numf,denf);
[zerif,polif,kf]=ss2zp(Af,Bf,Cf,Df,1);
%=============================================================
%[Af,Bf,Cf,Df]=ellip(npoles,Rp,Rs,Wn);
%=============================================================
% consider nominal (averages) values of gains as from CSA presentation
%=============================================================
K_L_M=1.4;          % gain of linear motor            [lbf/Amp]
K_P_A=0.62;         % gain of power amplifier         [Amps/Volt]
K_LVDT=Kdisp;       % gain of LVDT (min=1; max=11     [Volt/inch])
K_VEL=4.17;         % velocity feedback gain          [Volts/(in*sec)]
K_ACCEL_0=0.7;      % constant gain of accelerometer [Volts/g]
K_ACCEL=K_ACCEL_0*K_var;     % gain of accelerometer
K_excit=0.868;      % external excitation signal      [lbf/Volt]
K_aux=0.49;         % maximum stable gain             [Volts/Volts]
%=============================================================
% Passive pneumatic part (spring and damper in series)
% note: the damper is critically damped
%=============================================================
Pgauge=1;                    % because the piston leaks to the air
Kpress=1+1/Pgauge;
spec_heat=1.4;dens=1.225;    % air data
area=1.484/1550;             % area of piston
vol=30*0.0037854;            % volume of tank
Kair=spec_heat*dens*grav*area*Kpress;            % air spring K
Wnat=sqrt(spec_heat*grav*area/vol)*sqrt(Kpress); % air spring Wn
Bdamp=2*Kair/Wnat;                               % air damper b
%=============================================================
% Note: I will use a larger value for Kair (plunge mode = 0.5 Hz)
%=============================================================
Kair=(1/3)*MassMACE*(0.5*tupi)^2;
%=============================================================
% Assemble A,B,C,D matrices for control loop of suspension
% using Matlab functions  (see my notes)
% Note: I decided not to include the damping in the passive
% pneumatic part because it will mean to introduce a new state.
%=============================================================
if loop==1,
%=============================================================
% numerators and denominators of single block transfer functions
% in displacement-acceleration loop
n1=K_ACCEL*numf;d1=denf;
n2=K_LVDT;d2=1;
n3=K_excit;d3=1;
n4=Kair;d4=1;
n5=K_L_M*K_P_A;d5=1;
n6=1;d6=1;
n7=1;d7=[1 0 0];
n8=1;d8=1;
nblocks=8;
%=============================================================
iu=[6];                 % inputs to suspension controller
iy=[8];                 % output from suspension controller
%=============================================================
% assemble connectivity matrix
% note that at the summing point the voltage/displacement loop
% enters with a + sign, while the voltage/acceleration loop
% enters with a - sign to provide the mass cancellation effect;
% the reference signal of 0.868 lbf/Volt enters with a + sign too.
%=============================================================
Q=[1   6   0   0
   2   7   0   0
   3   0   0   0
   4   7   0   0
   5  -1   2   3
   6   0   0   0
   7   6   0   0
   8   5   4   0];
%=============================================================
[a,b,c,d] = tf2ss(n1,d1);
[at,bt,ct,dt] = tf2ss(n2,d2);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n3,d3);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n4,d4);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n5,d5);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n6,d6);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n7,d7);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n8,d8);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
%=============================================================
elseif loop==2,
%=============================================================
% numerators and denominators of single block transfer functions
% in displacement-acceleration loop
n1=1;d1=1;
n2=1;d2=[1 0];
n3=1;d3=[1 0];
n4=K_LVDT;d4=1;
n5=K_VEL;d5=1;
n6=K_ACCEL;d6=1;
n7=K_aux*numf;d7=denf;
n8=K_excit;d8=1;
n9=K_L_M*K_P_A;d9=1;
n10=Kair;d10=1;
n11=1;d11=1;
nblocks=11;
%=============================================================
iu=[1];                 % inputs to suspension controller
iy=[11];                % output from suspension controller
%=============================================================
% assemble connectivity matrix
% note that at the summing point the voltage/displacement loop
% enters with a + sign, the voltage/velocity loop enters with
% a + sign, while the voltage/acceleration loop enters with
% a - sign to provide the mass cancellation effect;
% the reference signal of 0.868 lbf/Volt enters with a + sign too.
%=========================================================
Q=[1    0    0    0
   2    1    0    0
   3    2    0    0
   4    3    0    0
   5    2    0    0
   6    1    0    0
   7    4    5    0
   8    0    0    0
   9   -6    7    8
   10   3    0    0
   11   9   10    0];
%=========================================================
% assemble state space model
[a,b,c,d] = tf2ss(n1,d1);
[at,bt,ct,dt] = tf2ss(n2,d2);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n3,d3);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n4,d4);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n5,d5);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n6,d6);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n7,d7);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n8,d8);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n9,d9);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n10,d10);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
[at,bt,ct,dt] = tf2ss(n11,d11);
[a,b,c,d]=append(a,b,c,d,at,bt,ct,dt);
%=========================================================
end
%=========================================================
[ac,bc,cc,dc]=connect(a,b,c,d,Q,iu,iy);
[acm,bcm,ccm,dcm]=minreal(ac,bc,cc,dc);
[zerosacm,polesacm,kacm]=ss2zp(acm,bcm,ccm,dcm,1);
%=========================================================
function [Kgip] = nodalgip3d(car1,car2,constw,lele,choice)
%===================================================================
% date: August 24,1991
%  by : Marco Quadrelli
% notes: done for MACESUSP.m
%===================================================================
Kgip = zeros(12);
if choice==1,
%===================================================================
% load with axial pretension
%===================================================================
k22 = car1*6/5 + car2*6/10;
k66 = (lele^2)*(car1*2/15+car2*1/30);
k88 = car1*6/5+car2*6/10;;
k1212=(lele^2)*(car1*2/15+car2*1/10);
k26 = lele*(car1/10+car2/10);
k28 = -k22;
k212=lele*(car1/10);k812=-k212;
k68 = -lele*(car1/10+car2/10);
k612 =-(lele^2)*(car1/30+car2*1/60);

Ka = [k22   k26   k28   k212;
      k26   k66   k68   k612;
      k28   k68   k88   k812,
      k212  k612  k812  k1212];

Kb = [k22   -k26   k28   -k212;
      -k26   k66   -k68    k612;
      k28   -k68   k88    -k812;
      -k212  k612  -k812  k1212];

Kgip = zeros(12);
Kgip([2 6 8 12],[2 6 8 12])=Kgip([2 6 8 12],[2 6 8 12])+Ka;
Kgip([3 5 9 11],[3 5 9 11])=Kgip([3 5 9 11],[3 5 9 11])+Kb;

Kgip=(1/lele)*Kgip;
%===================================================================
elseif choice==2,
%===================================================================
% string or cable-beam stiffness matrix
%===================================================================
carave=0.5*(car1+car2);
Kstring=(carave/lele)*[1 -1;-1  1];
Kgip([2 8],[2 8])=Kgip([2 8],[2 8])+Kstring;
Kgip([3 9],[3 9])=Kgip([3 9],[3 9])+Kstring;
%===================================================================
end
```

```
%----------------------------------------------------------------------
% date    : August 24,1991
% author  : M.B. Quadrelli
% notes   : this program  builds the 3D flex model of a suspension
%           cable.
%           It assembles the Mflex and Kflex for the beam-cable
%           elements and for the beam cable elements only.
%           The structural damping matrix is also assembled.
% revised: August 24, 1991
% Note: suffix "_w" refers to suspension wire
%       choicew= 1 = non-linear geom. stiffness for wire
%       choicew= 2 = string stiffness
%
%----------------------------------------------------------------------
%=============================================================
% compute:
% -the transformation matrices between the displacements of the
%  bus and wires nodes L, M and N;
% -the local to global frame direction cosine matrices for the
%  suspension wires FE (remember that the wire is at 90 deg ckwise);
%=============================================================
[RH1]=transform(rL1);[RH2]=transform(rM3);[RH3]=transform(rN5);
[RR1]=eye(6);[RR2]=eye(6);[RR3]=eye(6);
%=============================================================
[L2G0]=[0  1  0;-1  0  0;0  0  1];
L2G  = [L2G0  zeros(3);zeros(3) L2G0];
L2Gglob = [L2G  zeros(6);zeros(6) L2G];
%================================================================
[melew,kelew]=beamele(pAw,lelew,Ew,Gw,Jxw,Jyw,Jzw,Aw);
kelew=kelew+1.e-4*melew;
%=============================================================
%nelewrig=nelew;lelewrig=.15/nelewrig;
%[melewrig,kelewrig]=...
%beamele(pAwrig,lelewrig,Ewrig,Gwrig,Jxwrig,Jywrig,Jzwrig,Awrig);
%[mflexwrig]=fem(melewrig,nelewrig);[kflexwrig]=fem(kelewrig,nelewrig);
%mflexwrg = L2Gglob'*mflexwrig*L2Gglob;
%kflexwrg = L2Gglob'*kflexwrig*L2Gglob;
%=============================================================
% Set the rotational stiffness associated with the rotational dof's
% at the lower node (hinge) equal to zero and then assemble.
%=============================================================
kelew(:,[4:6])=zeros(12,3);kelew([4:6],:)=zeros(3,12);
%=============================================================
%=============================================================
% load with axial pretension
%=============================================================
mflexw=melew;kflexw=kelew;
W=(MassMACE/3);
carl=W*grav;car2=pAw*grav*lelew+carl;
[kelelx] = nodalgip3d(carl,car2,constw,lelew,choicew);
kelet1= kflexw+kelelx;
carl=W*grav+pAw*grav*lelew;car2=pAw*grav*lelew+carl;
[kele2x] = nodalgip3d(carl,car2,constw,lelew,choicew);
kelet2= kflexw+kele2x;
carl=W*grav+2*pAw*grav*lelew;car2=pAw*grav*lelew+carl;
[kele3x] = nodalgip3d(carl,car2,constw,lelew,choicew);
kelet3= kflexw+kele3x;
%=============================================================
% transform mflexw and kelet_i from Local to Global frame
% (Global = XYZ of spacecraft bus)
%=============================================================
mele1 = L2Gglob'*mflexw*L2Gglob;kele1 = L2Gglob'*kelet1*L2Gglob;
mele2 = L2Gglob'*mflexw*L2Gglob;kele2 = L2Gglob'*kelet2*L2Gglob;
mele3 = L2Gglob'*mflexw*L2Gglob;kele3 = L2Gglob'*kelet3*L2Gglob;
%=============================================================
%=============================================================
%
% Start assembling flexible part
%
% note: 1st flex node is at the bottom of the suspension,
%       last, at the top  (3 flex.elements/wire are enough)
%===============================================================
ntotwl = ndofw;    % (ndofw flex dof's)
%===============================================================
% Wire # 1
%===============================================================
KKw1 = 0*eye(ntotw1);MMw1 = 0*eye(ntotw1);Dcdw1= 0*eye(ntotw1);
Tw  = eye(12);
T3r=RH1;
if nelew==1,
nbeg=1;nend=12;
Tw(1:6,1:6) = T3r;Tw(7:12,7:12) = eye(6);
KKw1(nbeg:nend,nbeg:nend) = KKw1(nbeg:nend,nbeg:nend) + Tw'*kele1*Tw;
MMw1(nbeg:nend,nbeg:nend) = MMw1(nbeg:nend,nbeg:nend) + Tw'*mele1*Tw;
end
if nelew==2,
nbeg=1;nend=12;
Tw(1:6,1:6) = T3r;Tw(7:12,7:12) = eye(6);
KKw1(nbeg:nend,nbeg:nend) = KKw1(nbeg:nend,nbeg:nend) + Tw'*kele1*Tw;
MMw1(nbeg:nend,nbeg:nend) = MMw1(nbeg:nend,nbeg:nend) + Tw'*mele1*Tw;
nbeg=7;nend=18;
KKw1(nbeg:nend,nbeg:nend) = KKw1(nbeg:nend,nbeg:nend) + kele2;
MMw1(nbeg:nend,nbeg:nend) = MMw1(nbeg:nend,nbeg:nend) + mele2;
end
if nelew==3,
nbeg=1;nend=12;
Tw(1:6,1:6) = T3r;Tw(7:12,7:12) = eye(6);
KKw1(nbeg:nend,nbeg:nend) = KKw1(nbeg:nend,nbeg:nend) + Tw'*kele1*Tw;
MMw1(nbeg:nend,nbeg:nend) = MMw1(nbeg:nend,nbeg:nend) + Tw'*mele1*Tw;
nbeg=7;nend=18;
KKw1(nbeg:nend,nbeg:nend) = KKw1(nbeg:nend,nbeg:nend) + kele2;
MMw1(nbeg:nend,nbeg:nend) = MMw1(nbeg:nend,nbeg:nend) + mele2;
nbeg=13;nend=24;
KKw1(nbeg:nend,nbeg:nend) = KKw1(nbeg:nend,nbeg:nend) + kele3;
MMw1(nbeg:nend,nbeg:nend) = MMw1(nbeg:nend,nbeg:nend) + mele3;
end
%===============================================================
% Wire # 2
%===============================================================
KKw2 = 0*eye(ntotw1);MMw2 = 0*eye(ntotw1);Dcdw2= 0*eye(ntotw1);
Tw  = eye(12);
T3r=RH2;
if nelew==1,
nbeg=1;nend=12;
Tw(1:6,1:6) = T3r;Tw(7:12,7:12) = eye(6);
KKw2(nbeg:nend,nbeg:nend) = KKw2(nbeg:nend,nbeg:nend) + Tw'*kele1*Tw;
MMw2(nbeg:nend,nbeg:nend) = MMw2(nbeg:nend,nbeg:nend) + Tw'*mele1*Tw;
end
if nelew==2,
nbeg=1;nend=12;
Tw(1:6,1:6) = T3r;Tw(7:12,7:12) = eye(6);
KKw2(nbeg:nend,nbeg:nend) = KKw2(nbeg:nend,nbeg:nend) + Tw'*kele1*Tw;
MMw2(nbeg:nend,nbeg:nend) = MMw2(nbeg:nend,nbeg:nend) + Tw'*mele1*Tw;
nbeg=7;nend=18;
KKw2(nbeg:nend,nbeg:nend) = KKw2(nbeg:nend,nbeg:nend) + kele2;
MMw2(nbeg:nend,nbeg:nend) = MMw2(nbeg:nend,nbeg:nend) + mele2;
end
if nelew==3,
nbeg=1;nend=12;
Tw(1:6,1:6) = T3r;Tw(7:12,7:12) = eye(6);
KKw2(nbeg:nend,nbeg:nend) = KKw2(nbeg:nend,nbeg:nend) + Tw'*kele1*Tw;
MMw2(nbeg:nend,nbeg:nend) = MMw2(nbeg:nend,nbeg:nend) + Tw'*mele1*Tw;
```

```
nbeg=7;nend=18;
KKw2(nbeg:nend,nbeg:nend) = KKw2(nbeg:nend,nbeg:nend) + kele2;
MMw2(nbeg:nend,nbeg:nend) = MMw2(nbeg:nend,nbeg:nend) + mele2;
nbeg=13;nend=24;
KKw2(nbeg:nend,nbeg:nend) = KKw2(nbeg:nend,nbeg:nend) + kele3;
MMw2(nbeg:nend,nbeg:nend) = MMw2(nbeg:nend,nbeg:nend) + mele3;
end
%==============================================================
% Wire # 3
%==============================================================
KKw3 = 0*eye(ntotwl);MMw3 = 0*eye(ntotwl);Dcdw3= 0*eye(ntotwl);
Tw   = eye(12);
T3r=RH3;
if nelew==1,
nbeg=1;nend=12;
Tw(1:6,1:6) = T3r;Tw(7:12,7:12) = eye(6);
KKw3(nbeg:nend,nbeg:nend) = KKw3(nbeg:nend,nbeg:nend) + Tw'*kele1*Tw;
MMw3(nbeg:nend,nbeg:nend) = MMw3(nbeg:nend,nbeg:nend) + Tw'*mele1*Tw;
end
if nelew==2,
nbeg=1;nend=12;
Tw(1:6,1:6) = T3r;Tw(7:12,7:12) = eye(6);
KKw3(nbeg:nend,nbeg:nend) = KKw3(nbeg:nend,nbeg:nend) + Tw'*kele1*Tw;
MMw3(nbeg:nend,nbeg:nend) = MMw3(nbeg:nend,nbeg:nend) + Tw'*mele1*Tw;
nbeg=7;nend=18;
KKw3(nbeg:nend,nbeg:nend) = KKw3(nbeg:nend,nbeg:nend) + kele2;
MMw3(nbeg:nend,nbeg:nend) = MMw3(nbeg:nend,nbeg:nend) + mele2;
end
if nelew==3,
nbeg=1;nend=12;
Tw(1:6,1:6) = T3r;Tw(7:12,7:12) = eye(6);
KKw3(nbeg:nend,nbeg:nend) = KKw3(nbeg:nend,nbeg:nend) + Tw'*kele1*Tw;
MMw3(nbeg:nend,nbeg:nend) = MMw3(nbeg:nend,nbeg:nend) + Tw'*mele1*Tw;
nbeg=7;nend=18;
KKw3(nbeg:nend,nbeg:nend) = KKw3(nbeg:nend,nbeg:nend) + kele2;
MMw3(nbeg:nend,nbeg:nend) = MMw3(nbeg:nend,nbeg:nend) + mele2;
nbeg=13;nend=24;
KKw3(nbeg:nend,nbeg:nend) = KKw3(nbeg:nend,nbeg:nend) + kele3;
MMw3(nbeg:nend,nbeg:nend) = MMw3(nbeg:nend,nbeg:nend) + mele3;
end
%==============================================================
% check frequencies and mode shapes of each wire
%==============================================================
[evalw,evecw,snw,polesw,hertzw]=eigenproblem(KKw1,-MMw1);
%==============================================================
% So far, the M and K of each wire are in the global coordinates
% of each node, including the two extreme hinges.
% Now, append these K and M to the M and K of the
% spacecraft bus (see MACESUSP.m)
%==============================================================
%================================================================
%================================================================
% Author: Marco B. Quadrelli
% Date  : July 23, 1991
% Last revision: November 26, 1991
%================================================================
% This subroutine appends the geometric
% input data for the 3D MACE test article in the presence of
% suspension wires
%  the payloads are mounted and pointing downwards!
% Used by: RUNMACE0g.m and RUNMACE1g.m to relieve the
%          computational burden associated with these operations
%================================================================
%disp('  ');
%disp('static characteristics determined');
%disp('  ');
%go_on=menu('do you want to continue?','yes','no, abort');
%if go_on==1,
%   disp('Strike any key to continue ...') ; pause
%elseif go_on==2,
%   return
%end
%================================================================
%                 ENTER INPUT DATA
%-------------------------------------------------------------
% Define parameters and size of arrays
%-------------------------------------------------------------
tupi=2*pi;rad=pi/180;invrad=1/rad;invrpm=(2*pi)/60;rpmm=1/invrpm;
grav=9.8065;inch=0.0254;
rdum = [0;0;0];
%================================================================
%nele = input(' number of elements per flexible Lexan beam [1 or 2]: ');
nele=1;
%nelef = input(' number of elements for flexible appendage [1]: ');
nelew=1;

pl_locked=1;
%pl_locked=input('payload #1 free [0],locked (no moving mass) [1]:   ');

npay1=2;          % # of rigid body dof from payload 1
npay2=2;          % # of rigid body dof from payload 2
nwheels=3;        % # of rigid body dof from torque wheels

nnod= nele+1;     % # of nodes per Lexan beam
ndof= 6*nnod;     % # of dof allowed per Lexan beam
ntot= 24*nele+10;% # of dof allowed on bus + 2x2 payloads rotations
ntotflex= ntot-4;% # of flexible dof allowed on bus only
totaldof= ntot+3;% total # of on bus including 3 dof of torque wheels
nodi=ntotflex/6; % total # of nodes in spacecraft bus


                 if gravpar==0,

nelef=-1;
nnodf= nelef+1;     % # of nodes per flexible appendage beam
ndoff= 6*nnodf;     % # of dof allowed per flexible appendage beam
nodif=ndoff/6;      % total # of nodes in flexible appendage beam
numnp=nodi+nodif;   % # of nodal points of all structure
nflex=numnp*6;

% rigidbody = total # of rigid body dof allowed by the structure:
%        bus + 2*payloads + 3*wheels          in free space
% nfree = total # of dof allowed by the structure:
%        bus + 2*payloads + 3*wheels
% in free space (before imposing constraints)
   nfree=nflex + npay1 + npay2 + nwheels;
   rigidbody=6+npay1+npay2+nwheels;

 if p1_locked==0,
      states=nfree;totstate=2*(ntot+18)+12;inputs=9;
 elseif p1_locked==1,
      states=nfree-2;totstate=2*(ntot+18)+12-2;inputs=7;
 end
 pu=states;

               -  elseif gravpar==1,

 nnodw= nelew+1;  % # of nodes per suspension cable-beam
 ndofw= 6*nnodw;  % # of dof allowed per suspension cable-beam
 nodiw=ndofw/6;   % total # of nodes in suspension wire
 ntotw=ndofw;     % # of dof allowed per suspension cable-beam
 nwires=3;        % # of suspension wires
 ntotsusp=nwires*ntotw;% # of dof allowed by the suspension

 nelef=-1;
 nnodf= nelef+1;     % # of nodes per flexible appendage beam
 ndoff= 6*nnodf;     % # of dof allowed per flexible appendage beam
 nodif=ndoff/6;      % total # of nodes in flexible appendage beam

 nhingew=3;         % # of rigid body dof from hinges per susp.wire
 nhinges=nwires*nhingew;% # of rigid body dof from hinges at suspension

 % rigidbody = total # of rigid body dof allowed by the structure:
 %       bus + 2*payloads + 3*wheels          in free space
 rigidbody=6+npay1+npay2+nwheels;

 numnp=nodi+3*nodiw-3+nodif;  % # of nodal points of all structure
 nflex=numnp*6+nhinges;

 % nfree = total # of dof allowed by the structure:
 %       bus + nwires*wires + 2*payloads + 3*wheels
 % in free space (before imposing constraints)
 nfree=nflex + npay1 + npay2 + nwheels;
 nfreeq=nfree-12; % because I remove torsion and x,z translations

 if p1_locked==0,
      states=nfreeq;totstate=2*(ntot+18)+12;inputs=9;pu=52;
 elseif p1_locked==1,
      states=nfreeq-2;totstate=2*(ntot+18)+12-2;inputs=7;pu=50;
 end
                         end

 suspstates=nwires*4;
 %===================================================================
 %solveig=input('solve eigenproblem ?, yes [1], no [0]:   ');
 solveig=0;

 %choice2=...
 %menu('enter',...
 %'components of angular momentum in MACE body axes','wheels speeds');
 choice2=1;
 if choice2==1,
   Hcomp=[0;.1;0];
   %Hcomp=...
   %input(...
   %'enter components of angular momentum in body axes [N.m.sec]:   ');
 elseif choice2==2,
   spin1= input(' steady angular velocity of wheel 1 [rpm]:  ');
   spin2= input(' steady angular velocity of wheel 2 [rpm]:  ');
   spin3= input(' steady angular velocity of wheel 3 [rpm]:  ');
   %in rad/s
   spin1=invrpm*spin1;spin2=invrpm*spin2;spin3=invrpm*spin3;
```

```
end
spinMACE1=0;spinMACE2=0;spinMACE3=0;
spinMACE1=invrpm*spinMACE1;spinMACE2=invrpm*spinMACE2;
spinMACE3=invrpm*spinMACE3;

smorza=2;
damping=1;
%damping=menu('choose type of damping','modal','prop to K',...
%               'prop to M','old type','decoupling');
%CG=menu('is each payload','CG mounted ?','non CG mounted ?');
CG=2;
%gimbal1=menu('new coordinates of reference point of gimbal 1?',...
%               'yes,change','no,use default');
gimbal1=2;
%center=menu('gimbals rotation axes are centered',...
%              'no','yes');
center=2;
%pollo=menu('pivots lie on bus centerline','yes','no');
pollo=2;
statespace=0;

%bounce=input('enter bounce frequency [Hz]:   ');
bounce=0.57;
%active_suspension=...
% input('suspension active [enter 1], if passive [enter 0]:   ');
active_suspension=0;
%choicew=...
%menu('select one','non-linear geom. stiffness for wire','string
%stiffness');
choicew=1;
%loop=menu('select suspension control loop',...
%    'with displacement + acceleration feedback',...
%     'with displacement + velocity + acceleration feedback');
loop=1;                 %loop=2 is not the case of MACE!
%K_var=...
%input('variable gain of accelerometer(min=0.2;recd=0.2;max=10):  ');
%Kdisp=input('gain of LVDT (min=1;nom.=4;max=11[Volt/inch]:   ');
Kdisp=4;K_var=.2;
%==============================================================
%  these are the 2 angles made by each wheel rotation axis
%  with the X axis of the bus
%            alfa_i - in plane ;  beta_i out-of-plane
%  which is fixed in the L frame -X Y Z- (in radians)
%==============================================================
%ruote=menu('wheels box symm. axis pointing towards:','-Y','+Y');
ruote=2;
if ruote==1,
alfa1=90*rad;beta1=-35.3*rad;
alfa2=-30*rad;beta2=-35.3*rad;
alfa3=-150*rad;beta3=-35.3*rad;
elseif ruote==2,
alfa1=90*rad;beta1=35.3*rad;
alfa2=-30*rad;beta2=35.3*rad;
alfa3=-150*rad;beta3=35.3*rad;
end
%============================================================
%  transformation matrix between Fwheel-i and FB
[RwL11,RwL21,RwL31,RwLL1]=rotation(beta1,0,-alfa1);
[RwL12,RwL22,RwL32,RwLL2]=rotation(beta2,0,-alfa2);
[RwL13,RwL23,RwL33,RwLL3]=rotation(beta3,0,-alfa3);
%============================================================
[pw1,qw1,rw1,lw1,mw1,nw1,fw1,gw1,hw1,Aw1,Bw1,Cw1,RwL1]=...
   WitoBody(alfa1,beta1);
[pw2,qw2,rw2,lw2,mw2,nw2,fw2,gw2,hw2,Aw2,Bw2,Cw2,RwL2]=...
   WitoBody(alfa2,beta2);
[pw3,qw3,rw3,lw3,mw3,nw3,fw3,gw3,hw3,Aw3,Bw3,Cw3,RwL3]=...
```

```
   WitoBody(alfa3,beta3);
%=============================================================
%=============================================================
%        EXTERNAL DIMENSIONS
%=============================================================
L  = 0.22886;      % length of Lexan member
Ljl= 0.15264;      % length of long joint (node + collars)
Ljs= 0.10807;      % length of short joint(node + collar)
Lw = 4.6736;       % length of suspension wire
Lwrig= .05;    -% length of rigid support of suspension wire
% dimensions of gimbal support
asup = 0.05;bsup = 0.05;csup= 0.05;
% dimensions of gimbals  (ag = height;bg = radius of cylinder)
ag1=0.2;bg1=0.05;
ag2=0.1;bg2=0.05;
%=============================================================
% s = short joint (external joints);l = long joint(internal joints)
hcollar=0.04458;diamcE=0.0444;diamcI=0.0254;
acube=0.0635;
ajl = 0.0635+2*hcollar;bjl = 0.0635;cjl = 0.0635;
ajs = 0.0635+1*hcollar;bjs = 0.0635;cjs = 0.0635;
% collar
Acollar=(pi/4)*(diamcE^2 - diamcI^2);
RcolRcg=[0.0575;0;0];RcolLcg=[-0.0575;0;0];
%=============================================================
%  sensors box at node 3
asen = 2.5*inch;bsen = 2.5*inch;csen = 2.5*inch;
%  momentum wheels box
aw = 0.2;bw = 0.2;cw = 0.2;
%  identical payloads at the ends
height = 3.5*inch;radius = 2.5*inch;
%  each torque wheel
Rw = 0.08;heightw=0.10;
%=============================================================
%     ELASTICITY PARAMETERS
%  FOR FINITE ELEMENT BEAM, JOINTS, SUSPENSION WIRES AND RIGID
%  BARS connecting the suspension wires to the bus
%=============================================================
Do = .0254;Di = .0190;Dw = 0.00278;Dwrig = 0.005;
Ro = Do/2;Ri = Di/2;
A  = (Ro^2 - Ri^2)*pi;      % Area for flexible beam
Aj = bjs*cjs;               % Area for rigid joint
Aw = (pi/4)*(Dw)^2;         % Area for suspension wire
Awrig=(pi/4)*(Dwrig)^2;     % Area for rigid bar of suspension wire
Jx = (pi/32)*(Do^4 - Di^4);Jp=Jx;
Jy = (pi/4)*(Ro^4 - Ri^4);
Jz = Jy;
Jyw = (pi/4)*((Dw/2)^4);Jywrig = (pi/4)*((Dwrig/2)^4);
Jzw = Jyw;Jzwrig = Jywrig;
Jxw = Jyw + Jzw;Jxwrig = Jywrig + Jzwrig;Iw=Jxw;Jpw=Jxw;
%=============================================================
zeta=0.01;                  % damping ratio
E  = 2.3e+9;Ej = 7.31e+10; % E for flexible beam and joint
Ew = 2.1e+11;               % E for flexible suspension wire (steel)
nu = 0.37;nuj= 0.33;        % Poisson's ratio for beam and joint
nuw= 0.27;                  % Poisson's ratio for suspension wire
G  = (E/2)/(1+nu);Gj = (Ej/2)/(1+nuj);Gw = (Ew/2)/(1+nuw);
Ewrig=100000*Ew;Gwrig=100000*Gw;
pA = 1189.77*A;         % mass/length of beam
pAj= 2766.91*Aj;      % mass/length of joint
densw=7800;          % density of wire (steel)
pAw= 7800*Aw;        % mass/length of suspension wire
pAwrig=100000*pAw;
pAcollar=2711.57;
pAbase=pAj;
const=[E*Jy;E*Jz;E*A;G*Jp;Jp;pA;A;Jy;Jz]';
```

```
constw=[Ew*Jyw;Ew*Jzw;Ew*Aw;Gw*Jpw;Jpw;pAw;Aw;Jyw;Jzw]';
%================================================================
%     MASSES
%================================================================
%Mwheels  = 10.0;Mjoints = 0.4;Mjointl = 0.6;Msensors = 1.5;
% values updated on April 30, 1991
Mbase=1.70030;Mmcase=.5726;Mmarmature=0.1982;Mwheelone=1.09726;
Mfixed=Mmcase;Mrotat=Mmarmature+Mwheelone;Q=Mfixed+Mrotat;
Mwheels  = Mbase+3.*(Mmcase+Mmarmature+Mwheelone);
Mnodecube=0.2705;
Mcollar=0.11137·
Mjoints = Mnodecube+Mcollar;
Mjointl = Mnodecube+2*Mcollar;

Mdummy    = 7.09984;   % dummy payload at node 1
Msensors2 = 0.72968;   % sensors box at node 2 (triax. accel.)
Msensors3 = 1.07100;   % sensors box at node 3 (rate gyro package)
Msensors4 = 0.66060;   % sensors box at node 4 (triax. accel.)

Mcarriage=1.3132;
Mcable=pAw*Lw;
Mpayload=1.29798; % mass of rate gyro + payload can;
Msupport=2.991823;
Mgimbal1=2.922707;  % (composed of inner stage + encoder + frame)
Mgimbal2=1.061621;  % (composed of outer stage + payload support)
Mgimbal=Mgimbal1+Mgimbal2;
Mw=Mwheels/3;Mp=Mpayload;Mjs=Mjoints;Mjl=Mjointl;
Mtot3 = Mwheels+Msensors3+Mjointl;
Mtot  = Mpayload+Mgimbal1+Mgimbal2+Mjoints;

reduce=1;

Mbody1= reduce*(Mjoints+Msupport);
Mbody2= reduce*(Mgimbal1);
Mbody3= reduce*(Mgimbal2+Mpayload);
Mnode=Mbody1+Mbody2+Mbody3;


Mnode1=Mnode;
Mnode2=(Mjointl+Msensors2);
Mnode3=Mtot3;
Mnode4=(Mjointl+Msensors4);
Mnode5=Mnode;
if pl_locked==0,
     Mnode1s=Mnode;
elseif pl_locked==1,
     Mnode1s=Mdummy+Mjointl;
end

Massrigid = Mnode1s+Mnode2+Mnode3+Mnode4+Mnode5;
Massflex  = 4*pA*L;
MassMACE  = Massrigid + Massflex;
NodemassMAP=[Mnode1s  Mnode2  Mnode3  Mnode4  Mnode5];
%================================================================
%================================================================
%================================================================
%  from node 1 (or 5 ) to pivot  (center of gimbal no.1)
%================================================================
rp1=rdum;rp5=rdum;
if gimbal1==2,
rp1 = [0;-0.1143;-0.00315];rp5 = [0;-0.1143;-0.00315];
elseif gimbal1==1,
rp1=...
input('enter coordinates of point A wrt. point F (node 1):   ');
rp5=...
input('enter coordinates of point A wrt. point F (node 5):   ');
```

```
end
rAFF1=rp1;rAFF5=rp5;
%==============================================================
% off-set between centers of rotation of the two gimbals
%==============================================================
rPAA1=[0;0;0];rPAA5=[0;0;0];
if center==2,
rPAA1=[0;0;0];rPAA5=[0;0;0];
elseif center==1,
rPAA1=...
input('enter coordinates of 2nd center wrt. the 1st [gimbal 1]:  ');
rPAA5=...
input('enter coordinates of 2nd center wrt. the 1st [gimbal 5]:  ');
end

%==============================================================
% vector from pivot to payload center of mass (the frames at the
% pivots 1 and 5 are parallel and in the same directions)
% note: at zero angle, the payload is along p1
if CG==1,
rpayPP1=[0;0;0];rpayPP5=[0;0;0];
elseif CG==2,
rpayPP1=[0.17145;0;0];rpayPP5=[0.17145;0;0];
end
po1= rpayPP1(1);po5=rpayPP5(1);
%==============================================================
%  from node 1 (or 5) to center of mass of gimbal
rg1 = [0;-0.1143;0];rg5 = [0;-0.1143;0];
%==============================================================
if ruote==1,
rw = [0.07632;-0.10974;0];% radius vector of wheels com.wrt.left point
rwroot=[0.07632;-0.07433;0]; %vector of root wrt. point L
elseif ruote==2,
rw = [0.07632;+0.10974;0];
rwroot=[0.07632;+0.07433;0];
end
%==============================================================
% radius vector of com. of each wheel wrt.L point
w1=0.16;w2=0.16;w3=0.16; %distance of com of each wheel wrt. root point
rww1=w1*[pw1;qw1;rw1];
rww2=w2*[pw2;qw2;rw2];
rww3=w3*[pw3;qw3;rw3];
rwww1=rwroot+rww1;rwww2=rwroot+rww2;rwww3=rwroot+rww3;
rwww=(1/3)*(rwww1+rwww2+rwww3);
rs = [0.07632;-0.06985;0];        % same but for the sensor box
rj3= [0.07632;0;0];               % same but for the central joint
rR3=[0.07632;0;0];
rRL3= [0.15264;0;0];          % same but for the right wrt.left point
rcomw1=[w1;0;0];rcomw2=[w2;0;0];rcomw3=[w3;0;0];
%==============================================================
%  center of mass of all node 3 wrt. left point
rcm3 = (1/Mtot3)*(rj3*Mjoint1+Msensors3*rs+Mwheels*rw);
rcmw = rcm3-rwww;rcmj = rcm3-rj3;rcms = rcm3-rs;
rcmrut=rcm3-rwroot;
% offset of c.g. of central node wrt. point on the bus centerline
rH3=rcm3-rR3;
%==============================================================
%  from node 1 (or 5) to center of mass of joint
rj1 = [0;0;0];rj5 = [0;0;0];
rjFF1=rj1;rjFF5=rj5;
%  from node 1 or 5 to com of gimbal support
rSFF1=[-0.08;-0.12315;0];rSFF5=[0.08;-0.12315;0];
rG1AA1=[0;0;0];rG1AA5=[0;0;0];
rG2PP1=[0;0;0];rG2PP5=[0;0;0];

% com's of gimbal motors are not centered
```

```
%rG1AA1=[-0.001;0;0];rG1AA5=[0.001;0;0];
%rG2PP1=[0;0.001;0];rG2PP5=[0;0.001;0];
%==============================================================
% center of mass of joint and gimbal mount (wrt. node 1 and 5)
rcmgj1 = (Mgimbal*rg1+Mjoints*rj1)/(Mgimbal+Mjoints);
rcmgj5 = (Mgimbal*rg5+Mjoints*rj5)/(Mgimbal+Mjoints);
%==============================================================
rga1 = rg1-rcmgj1;rga5 = rg5-rcmgj5;
rja1 = rj1-rcmgj1;rja5 = rj5-rcmgj5;
%==============================================================
rL2 = [-0.07632;0;0];rR2 = [0.07632;0;0];
rL3 = [-0.07632;0;0];rR3 = [0.07632;0;0];
%==============================================================
if pollo==1,
rSFF1=rdum;rG1AA1=rdum;rjFF1=rdum;rAFF1=rdum;rG1AA1=rdum;
rSFF5=rdum;rG1AA5=rdum;rjFF5=rdum;rAFF5=rdum;rG1AA5=rdum;
end
%==============================================================
%=============================================================
% compute the vectors from FE node on the bus to hinged node at the
%  bottom of the suspension wire (these hinged nodes are L,M,N);
%=============================================================
rL1 = [0;+0.04205;0];      % [-0.054;+0.04205;0];
rM3 = [0;+0.04205;0];
rN5 = [0;+0.04205;0];      % [+0.054;+0.04205;0];
%=============================================================
%==============================================================
%==============================================================
% center of mass of body 1 and other relevant vectors
%=============================================================
rFplusF1=(1/Mbody1)*(Mjoints*rjFF1+Msupport*rSFF1);
rFplusF5=(1/Mbody1)*(Mjoints*rjFF5+Msupport*rSFF5);
%=============================================================
%  center of mass of body 2  in A frame
%=============================================================
rAoAA1=rG1AA1;
rAoAA5=rG1AA5;
%=============================================================
%  center of mass of body 3  in P frame
%=============================================================
%rBPP1=(1/Mbody3)*(rpayPP1*Mpayload+rG2PP1*Mgimbal2);
rBPP1=rpayPP1;
rpayBP1=rpayPP1-rBPP1;rG2BP1=rG2PP1-rBPP1;

%rBPP5=(1/Mbody3)*(rpayPP5*Mpayload+rG2PP5*Mgimbal2);
rBPP5=rpayPP5;
rpayBP5=rpayPP5-rBPP5;rG2BP5=rG2PP5-rBPP5;
%=============================================================
%==============================================================
%        SECOND MOMENTS OF INERTIA
%==============================================================
% moments of inertia of gimbal, joint and payload wrt. their
%  centers of mass;    D-  stands for Dyadic
%[J1g1,J2g1,J3g1]=cylinder(Mgimbal1,bg1,ag1);
%[J1g2,J2g2,J3g2]=cylinder(Mgimbal2,bg2,ag2);
%[J1sup,J2sup,J3sup]=inertial(Msupport,asup,bsup,csup);
%==============================================================
%[J1j1,J2j1,J3j1]=inertial(Mjoint1,aj1,bj1,cj1);
%[J1js,J2js,J3js]=inertial(Mjoints,ajs,bjs,cjs);
%J1=5.0026e-05;J23=1.1574e-04;Ja=1.8145e-04;Jfi=8.0042e-05;
Jcubex=3.04e-4;Jcubey=Jcubex;Jcubez=Jcubex;
Jcollarx=.5*Mcollar*((.5*diamcE)^2+(.5*diamcI)^2);
Jcollary=.5*Mcollar*((.5*diamcE)^2+(.5*diamcI)^2+(hcollar^2)/3);
Jcollarz=Jcollary;
[Daddcol]=addyadic(Mcollar,RcolRcg);
J1js=Jcubex+Jcollarx;J2js=Jcubey+Jcollary;J3js=Jcubez+Jcollarz;
```

```
J1jl=Jcubex+2*Jcollarx;J2jl=Jcubey+2*Jcollary;J3jl=Jcubez+2*Jcollarz;
[DJjl] =principal(J1jl,J2jl,J3jl);
[DJjs] =principal(J1js,J2js,J3js);
DJjl=DJjl+2*Daddcol;DJjs=DJjl+Daddcol;
%==============================================================
% inertias of sensors boxes
J1sen=0.00111;J2sen=0.000788;J3sen=0.000696;  % at node 3
J2sen2=7.78e-5;J1sen2=0.000267;J3sen2=J2sen2; % at node 2
J2sen4=8.00e-5;J1sen4=0.000274;J3sen4=J2sen4; % at node 4
%==============================================================
J1pl=2.07e-3;J2pl=1.96e-3;J3pl=1.96e-3;  % rate gyro + payload can
%==============================================================
Jbasex=2.61e-3;Jbasey=4.473-3;Jbasez=2.68e-3;
[DJbase]=principal(Jbasex,Jbasey,Jbasez);
[J1w,J2w,J3w]=inertial(Mwheels,aw,bw,cw);
% axial and transverse inertia of each torque wheel
Jwa1=5.28e-3+3.38e-4;Jwa2=Jwa1;Jwa3=Jwa1;
Jwt1=2.6927e-3+6.67e-4;Jwt2=Jwt1;Jwt3=Jwt1;
%==============================================================
Jbar1=((pA*L)/2.)*(Ro^2-Ri^2);
Jbar2=((pA*L)/12.)*(L^3)+Jbar1/2.;
Jbar3=Jbar2;
%==============================================================
%  these are the central inertia dyadics from the IDEAS model
%  as of April 19, 1991
%==============================================================
DJsupport=[0.0114607      -0.005298906      1.955414e-11;
          -0.005298906     0.02323236      -1.637091e-11;
           1.955414e-11   -1.637091e-11     0.02756642];
DJgimbal1=[0.005146256    -2.557536e-5      9.972862e-5;
          -2.557536e-5     0.01187599      -1.573758e-5;
           9.972862e-5    -1.573758e-5      0.01350181];
DJgimbal2=[0.003440874    -2.585639e-13    -1.756260e-15;
          -2.585639e-13    0.001280201      3.759716e-06;
          -1.756260e-15    3.759716e-06     0.002916165];
Jdumx=5.32e-3;Jdumy=2.97e-2;Jdumz=2.90e-2;
[DJdummy]=principal(Jdumx,Jdumy,Jdumz);
[Dadddum]=addyadic(Mdummy,[0;-0.0635;0]);
DJatdum=DJjs+DJdummy+Dadddum;
%==============================================================
% Inertias of triaxial accelerometer packages at node 2 and node 4
DJacc2=principal(J1sen2,J2sen2,J3sen2);
DJacc4=principal(J1sen4,J2sen4,J3sen4);
%==============================================================
[DJsen]=principal(J1sen,J2sen,J3sen);
[DJpl] =principal(J1pl,J2pl,J3pl);
[DJw]  =principal(J1w,J2w,J3w);
[DJw1] =principal(Jwa1,Jwt1,Jwt1);
[DJw2] =principal(Jwa2,Jwt2,Jwt2);
[DJw3] =principal(Jwa3,Jwt3,Jwt3);
[DJbar]=principal(Jbar1,Jbar2,Jbar3);
%==============================================================
%  inertia of bodies of end nodes 2 and 4 about their com's
%==============================================================
[Daddnodo2]=addyadic(Msensors2,[0;0.06553;0]);
[Daddnodo4]=addyadic(Msensors4,[0;0.06553;0]);
DJ2=Daddnodo2+DJacc2+DJjl;
DJ4=Daddnodo4+DJacc4+DJjl;
%==============================================================
%==============================================================
%  inertia of bodies of end nodes 1 or 5 about their com's
%==============================================================
[Daddjs1]=addyadic(Mjoints,rFplusF1-rjFF1);
[Daddjs5]=addyadic(Mjoints,rFplusF5-rjFF5);
[Daddsup1]=addyadic(Msupport,rFplusF1-rSFF1);
[Daddsup5]=addyadic(Msupport,rFplusF5-rSFF5);
```

```
[Daddgl1] =addyadic(Mgimbal1,rAoAA1-rG1AA1);
[Daddgl5] =addyadic(Mgimbal1,rAoAA5-rG1AA5);
[Daddg21]=addyadic(Mgimbal2,-rG2PP1+rBPP1);
[Daddg25]=addyadic(Mgimbal2,-rG2PP5+rBPP5);
[Daddpay1]=addyadic(Mpayload,-rpayPP1+rBPP1);
[Daddpay5]=addyadic(Mpayload,-rpayPP5+rBPP5);
%=============================================================
% Inertia of node 3 including the effect of torque wheels
%=============================================================
% wrt. the com. of node 3
DJw1Tr=RwL1*DJw1*RwL1';
DJw2Tr=RwL2*DJw2*RwL2';
DJw3Tr=RwL3*DJw3*RwL3';
[Daddw1]=addyadic(Mw+Mmcase+Mmarmature,rww1-rcmrut);
[Daddw2]=addyadic(Mw+Mmcase+Mmarmature,rww2-rcmrut);
[Daddw3]=addyadic(Mw+Mmcase+Mmarmature,rww3-rcmrut);
[Daddsen3]=addyadic(Msensors3,rcms);
[Daddbase]=addyadic(Mbase,rcmrut);
[Daddj13]=addyadic(Mjoint1,rcmj);
DJsenjoil3=...
  DJj1+DJsen+Daddsen3+Daddw1+Daddw2+Daddw3+DJbase+Daddbase;
Dnode3L3=DJsenjoil3+DJw1Tr+DJw2Tr+DJw3Tr;
%=============================================================
%=============================================================
% Transformation matrices relating inertial velocities at nodes
% placed a given distance apart (R=right,L=left of node)
%=============================================================
[TR1]=transform([0.07632;0;0]);[TL5]=transform([-0.07632;0;0]);
[TL2]=transform(rL2);[TR2]=transform(rR2);TL4 = TL2;TR4 = TR2;
%==============================================================
% if central node F.E. node is at the cg
%[TL3]=transform(-rcm3);[TR3]=transform(rRL3-rcm3);
%==============================================================
% if central node F.E. node is on bus centerline
[TL3]=transform(rL3);[TR3]=transform(rR3);
%==============================================================
%==============================================================
% assemble mass matrices
%=============================================================
%
% Note: Mass # 1 has been substituted with a dummy gimbal
%       on October 30, 1991
%==============================================================
% MASS MATRICES
%==============================================================
M3(1:9,1:9)=zeros(9);
[M2]=massmx(Mnode2,DJ2(1,1),DJ2(2,2),DJ2(3,3));
[M4]=massmx(Mnode4,DJ4(1,1),DJ4(2,2),DJ4(3,3));
%=============================================================
%=============================================================
%=============================================================
% Assemble mass matrix at node 3
% state: x  y  z  tet1  tet2  tet3  omega1  omega2  omega3
% and refer it to the central node (located on the bus centerline)
%=============================================================
[M3diag1]=principal(Mnode3,Mnode3,Mnode3);M3(1:3,1:3)=M3diag1;
[M3diag2]=principal(Jwa1,Jwa2,Jwa3);M3(7:9,7:9)=M3diag2;
M3(4:6,4:6)=Dnode3L3;
M3(4,7)=Jwa1*pw1;M3(5,7)=Jwa1*qw1;M3(6,7)=Jwa1*rw1;
M3(4,8)=Jwa2*pw2;M3(5,8)=Jwa2*qw2;M3(6,8)=Jwa2*rw2;
M3(4,9)=Jwa3*pw3;M3(5,9)=Jwa3*qw3;M3(6,9)=Jwa3*rw3;
M3(7:9,4:6)=M3(4:6,7:9)';
%=============================================================
% remove rows and columns 21,22,23 thus eliminating wheels dof's
%=============================================================
M3new=M3(1:6,1:6);
```

```
M3red=M3(4:9,4:9);
DJ=M3red(1:3,1:3);
%==============================================================

%==============================================================
%==============================================================
%
%         FINITE ELEMENT MODEL
%
%==============================================================
lele=L/nele;
[mele,kele]=beamele(pA,lele,E,G,Jx,Jy,Jz,A);
kele=kele+1.e-6*mele;
[mflex]=fem(mele,nele);[kflex]=fem(kele,nele);
%==============================================================

                  if gravpar==1,
%==============================================================
% Assemble finite element model for suspension wires
% compute melew and kelew of the wire elements in their
% local frames. (-rig refers to the short rigid element)
%==============================================================
lelew=Lw/nelew;

ADDENDUM3d

%outputs of ADDENDUM3d:
%            KKw1,KKw2,KKw3,MMw1,MMw2,MMw3,
%            all 12x12, all stiffened by gravity
%nbegattach  = bottom attachment to bus
%nendcarriage= top attachment to carriage

Kair1=(Mnode1 + .5*Mnode2 + Mcarriage + Mcable)*(.47*tupi)^2;
Kair3=(Mnode3 + .5*(Mnode2+Mnode4) + Mcarriage + Mcable)*(.57*tupi)^2;
Kair5=(Mnode5 + .5*Mnode4 + Mcarriage + Mcable)*(.47*tupi)^2;
Kair=(MassMACE/3 + Mcarriage + Mcable)*(bounce*tupi)^2;
%==============================================================
                       end

%==============================================================
% load damping matrices for payload # 1 locked
%==============================================================
if gravpar==0,
   load DUMP0g
elseif gravpar==1,
   load DUMP1g
end
%==============================================================
function[p,q,r,l,m,n,f,g,h,A,B,C,Rot]=WitoBody(phi,theta)
%  computes direction cosines of transformation between W-i and B
%  phi == beta ; theta == alfa
%p=cf*ct;q=cf*st;r=-sf;
%l=-st;m=ct;n=0;
%f=sf*ct;g=sf*st;h=cf;

cf=cos(phi);ct=cos(theta);sf=sin(phi);st=sin(theta);
rot1=[1  0  0;0  cf  sf;0  -sf  cf];
rot2=[ct  -st  0;st  ct  0;0  0  1];
ROT=rot2*rot1;
p=ROT(1,1);
q=ROT(1,2);
r=ROT(1,3);
l=ROT(2,1);
m=ROT(2,2);
n=ROT(2,3);
f=ROT(3,1);
g=ROT(3,2);
h=ROT(3,3);
Rot=ROT;
%p=ct;q=-cf*st;r=-sf*st;
%l=st;m=ct*cf;n=sf*ct;
%f=0;g=-sf;h=-cf;
%Rot=[p  q  r;l  m  n;f  g  h];

A=g*l-m*f;
B=h*l-n*f;
C=m*h-n*g;
% A,B and C have to be multiplied by 2*spin_i*Jwai
```

```
function [shiftmass] = addyadic(m,r)
shiftmass=[r(2)^2+r(3)^2      -r(1)*r(2)          -r(1)*r(3);
           -r(1)*r(2)          r(3)^2+r(1)^2       -r(2)*r(3);
           -r(1)*r(3)          -r(2)*r(3)          r(1)^2+r(2)^2];
shiftmass=m*shiftmass;
```

```
function [evecbig,evalbig,sbig,polesAAA]=autovalori(AAA)

[evecbig1,evalbig1]=eig(AAA);

evalbig = (diag(evalbig1));[evalpp,i]=sort(real(evalbig));
evalbig = evalbig(i);
sbig = size(evalbig);evalbig=evalbig([sbig:-1:1]);
evecbig=real(evecbig1(:,i));evecbig=evecbig(:,[sbig:-1:1]);

polesAAA=(evalbig)/(2*pi);
[polesAAA,ii]=sort(polesAAA);
```

```
function [mele,kele]=beamele(m,L,E,G,Jx,Jy,Jz,A)
% compute mass and stiffness matrix of a 6 dof element
% m = mass per unit length
% no rotary inertia and shear effect included
%
%            x1,x2,x3,x4,x5,x6,
%            x7,x8,x9,x10,x11,x12
%
%       ^ 2
%       | rot-5(cck)
%       |
%       +-------> 1, rot-4(cck)
%      /
%     / rot-6(cck)
%    3
%                                              ^ 8
%                                              | rot-11(cck)
%                                              |
%                                              +-------> 7, rot-10(cck)
%                                             /
%                                            / rot-12(cck)
%                                           9
% Jx = polar moment of inertia of the section of the element
% Jy = about vertical axis
% Jz = about out-of-plane axis
%=================================================================
% this is the mass matrix for a 12 dof element (Bernoulli-Euler)
% see Przemieniecki
mele(1:12,1:12)=zeros(12);
R1=(140*Jx)/A;
R2=(70*Jx)/A;
mm=4*L^2;nn=-3*L^2;pp=13*L;qq=22*L;

mele(1,1)=1/3;mele(1,7)=1/6;mele(7,7)=1/3;mele(7,1)=mele(1,7);
mele(4,4)=Jx/(3*A);mele(10,10)=mele(4,4);mele(10,4)=Jx/(6*A);
mele(4,10)=mele(10,4);
mele(2,2)=(13/35)+(6*Jz)/(A*L^2);mele(8,8)=mele(2,2);
mele(3,3)=(13/35)+(6*Jy)/(A*L^2);mele(9,9)=mele(3,3);
mele(5,5)=(L^2/105)+(2*Jy)/(15*A);mele(11,11)=mele(5,5);
mele(6,6)=(L^2/105)+(2*Jz)/(15*A);mele(12,12)=mele(6,6);
mele(5,3)=-(11*L)/210-Jy/(10*A*L);mele(3,5)=mele(5,3);
mele(8,6)=(13*L)/420-Jz/(10*A*L);mele(6,8)=mele(8,6);
mele(11,9)=-mele(5,3);mele(9,11)=mele(11,9);
mele(6,2)=(11*L)/210+Jz/(10*A*L);mele(2,6)=mele(6,2);
mele(9,5)=-(13*L)/420+Jy/(10*A*L);mele(5,9)=mele(9,5);
mele(12,8)=-mele(6,2);mele(8,12)=mele(12,8);
mele(8,2)=(9/70)-(6*Jz)/(5*A*L^2);mele(2,8)=mele(8,2);
mele(9,3)=(9/70)-(6*Jy)/(5*A*L^2);mele(3,9)=mele(9,3);
mele(11,5)=-L^2/140-Jy/(30*A);mele(5,11)=mele(11,5);
mele(12,6)=-L^2/140-Jz/(30*A);mele(6,12)=mele(12,6);
mele(11,3)=(13*L)/420-Jy/(10*A*L);mele(3,11)=mele(11,3);
mele(12,2)=-(13*L)/420+Jz/(10*A*L);mele(2,12)=mele(12,2);

%mele = ...
%[140   0    0    0    0    0    70   0    0    0    0    0;
%  0   156   0    0    0    qq   0    54   0    0    0   -pp;
%  0    0   156   0   -qq   0    54   0    54   0    pp   0;
%  0    0    0    R1   0    0    0    0    0    R2   0    0;
%  0    0   -qq   0    mm   0    0    0   -pp   0    nn   0;
%  0    qq   0    0    0    mm   0    pp   0    0    0    nn;
%  70   0    54   0    0    0   140   0    0    0    0    0;
% 0     54   0    0    0    pp   0   156   0    0    0   -qq;
% 0     0    54   0   -pp   0    0    0   156   0    qq   0;
%  0    0    0    R2   0    0    0    0    0    R1   0    0;
%  0    0    pp   0    nn   0    0    0    qq   0    mm   0;
%  0   -pp   0    0    0    nn   0   -qq   0    0    0    mm];
```

```
%mele=mele*((m/A)*A*L)/420;

mele=mele*(m*L);
%==================================================
%==================================================
%This is the stiffness matrix of a 12 dof element

kele=zeros(12);
%kele(1,1) = A*E/L;kele(7,7)=kele(1,1); .
%kele(7,1) =-kele(1,1);kele(1,7)=kele(7,1);

%kele(4,4) = G*Jx/L;kele(10,10)=kele(4,4);
%kele(10,4)=-kele(4,4);kele(4,10)=kele(10,4);

%kele(2,2) = (12*E*Jz)/(L^3);kele(8,8)=kele(2,2);
%kele(2,8) =-kele(2,2);kele(8,2)=kele(2,8);

%kele(3,3) = (12*E*Jy)/(L^3);kele(9,9)=kele(3,3);
%kele(3,9) =-kele(3,3);kele(9,3)=kele(3,9);

%kele(5,5) = (4*E*Jy)/L;kele(11,11)=kele(5,5);
%kele(5,11)= (2*E*Jy)/L;kele(11,5)=kele(5,11);

%kele(6,6) = (4*E*Jz)/L;kele(12,12)=kele(6,6);
%kele(6,12)= (2*E*Jz)/L;kele(12,6)=kele(6,12);

%kele(3,5) =-(6*E*Jy)/(L^2);kele(5,3)=kele(3,5);
%kele(3,11)= kele(3,5);kele(11,3)=kele(3.11);
%kele(9,5) =-kele(3,5);kele(5,9)=kele(9,5);
%kele(11,9)= kele(9,5);kele(9,11)=kele(11,9);

%kele(2,6) = (6*E*Jz)/(L^2);kele(6,2)=kele(2,6);
%kele(2,12)= kele(2,6);kele(12,2)=kele(2,12);
%kele(12,8)=-kele(2,6);kele(8,12)=kele(12,8);
%kele(8,6) = kele(12,8);kele(6,8)=kele(8,6);

k1 = E*A/L;
k2 = G*Jx/L;
k3 = E*Jz/L;
k4 = E*Jz/(L^2);
k5 = E*Jz/(L^3);
k6 = E*Jy/L;
k7 = E*Jy/(L^2);
k8 = E*Jy/(L^3);
kele=...
[k1     0      0     0     0      0     -k1      0       0      0      0      0;
 0   12*k5     0     0     0    6*k4     0    -12*k5     0      0      0    6*k4;
 0     0    12*k8    0   -6*k7    0      0      0     -12*k8    0    -6*k7    0;
 0     0      0     k2     0      0      0      0       0     -k2     0      0;
 0     0    -6*k7    0   4*k6     0      0      0      6*k7    0     2*k6    0;
 0   6*k4     0      0     0    4*k3     0    -6*k4     0      0      0    2*k3;
-k1    0      0      0     0      0     k1      0       0      0      0      0;
 0  -12*k5    0      0     0   -6*k4     0    12*k5     0      0      0    -6*k4;
 0     0   -12*k8    0   6*k7     0      0      0      12*k8    0    6*k7    0;
 0     0      0    -k2     0      0      0      0       0     k2      0      0;
 0     0    -6*k7    0   2*k6     0      0      0      6*k7    0     4*k6    0;
 0   6*k4     0      0     0    2*k3     0    -6*k4     0      0      0    4*k3];
```

```
function[vectorprod]=cross(vec1,vec2)

% evaluates   the cross product (vec1 x vec2)

vectorprod=[vec1(2)*vec2(3)  - vec1(3)*vec2(2);
            vec1(3)*vec2(1)  - vec1(1)*vec2(3);
            vec1(1)*vec2(2)  - vec1(2)*vec2(1)];
```

```
function [res]=crossdot(vec1,vec2,vec3)

%  evaluates the product vec1_cross_[vec2_dot_vec3]

[pippo1]=dot(vec2,vec3);
[res]=cross(vec1,pippo1);
```

```
function [cross] = crossdyad(r,c)
cross(1,1)=2*r(2)*c(2)+2*r(3)*c(3);
cross(2,2)=2*r(1)*c(1)+2*r(3)*c(3);
cross(3,3)=2*r(2)*c(2)+2*r(1)*c(1);
cross(1,2)=-r(1)*c(2)-r(2)*c(1);cross(2,1)=cross(1,2);
cross(1,3)=-r(1)*c(3)-r(3)*c(1);cross(3,1)=cross(1,3);
cross(3,2)=-r(3)*c(2)-r(2)*c(3);cross(2,3)=cross(3,2);
```

```
function [J1,J2,J3]=cylinder(M,radius,height)
J1 = (M/2)*(radius^2);
J2 = (M/12)*(3*radius^2+height^2);
J3 = J2;
```

```
function [dotprod]=dot(vec1,vec2)

% evaluates the dot product (vec1*vec2)

dotprod = vec1(1)*vec2(1) + vec1(2)*vec2(2) + vec1(3)*vec2(3);
```

```
function[evalues,evectors,s,poles,hertz]=eigenproblem(eval,evec)

evalues = real(diag(eval));[evalues,i]=sort(evalues);
evalues = evalues(i);
evectors=real(evec(:,i));
s = size(evalues);evalues=evalues([s:-1:1]);
evectors=evectors(:,[s:-1:1]);


poles = sqrt(evalues)/(2*pi);       % complex poles in hertz
[poles,i]=sort(poles);              % order poles
hertz = sqrt(-evalues)/(2*pi);      % real frequencies in hertz
[hertz,i]=sort(hertz);              % order frequencies
```

```
function [glob] = fem(element,number)
[totdof,totdof]=size(element);
nn=totdof/2;
g=(number+1)*nn;
glob=0*eye(g);
a=[eye(totdof),0*ones(totdof,g-totdof)];
for c=[1:1:number]
glob=a'*element*a+glob;
a=[0*ones(totdof,nn)  a(:,[1:(g-nn)])];
end
```

```
function [J1,J2,J3]=inertial(M,a,b,c)
J1=(M/12)*(b^2+c^2);
J2=(M/12)*(a^2+c^2);
J3=(M/12)*(a^2+b^2);
```

```
function [J1,J2,J3]=inertia2(M1,M2,J11,J12,J13,J21,J22,J23,r1,r2)
J1=J11+J21+M1*(r1(2)^2+r1(3)^2)+M2*(r2(2)^2+r2(3)^2);
J2=J12+J22+M1*(r1(1)^2+r1(3)^2)+M2*(r2(1)^2+r2(3)^2);
J3=J13+J23+M1*(r1(2)^2+r1(1)^2)+M2*(r2(2)^2+r2(1)^2);
```

```
function [M]=massmx(mass,J1,J2,J3)
M=[mass    0       0     0     0      0;
    0     mass     0     0     0      0;
    0      0      mass   0     0      0;
    0      0       0     J1    0      0;
    0      0       0     0     J2     0;
    0      0       0     0     0     J3];
```

```
function [TP,TTP,TTTP,n3]=ordertrans(ntot,ntotflex)
%==================================================================
% order the state vector
%==================================================================
TP = 0*eye(ntotflex);TTP=0*eye(ntot);TTTP=0*eye(ntot-2);
n3 = ntotflex/6;
TTP(1,1)=1;TTP(2,2)=1;TTP(ntot-1,ntot-1)=1;TTP(ntot,ntot)=1;
for i=1:n3,                % for each of the nodes :
TP(i,6*i-5)      = 1;      %x
TP(i+n3,6*i-4)   = 1;      %y
TP(i+2*n3,6*i-3)= 1;       %z
TP(i+3*n3,6*i-2)= 1;       %angle theta1
TP(i+4*n3,6*i-1)= 1;       %angle theta2
TP(i+5*n3,6*i)   = 1;      %angle theta3
end
TTP(3:ntot-2,3:ntot-2)=TP(1:ntotflex,1:ntotflex);
TTTP(1:ntot-4,1:ntot-4)=TP(1:ntotflex,1:ntotflex);
```

```
function [evec]=ordine(x,y,z,evec)
%=====================================================================
% ordering eigenvectors : x y z tet1 tet2 tet3
%=====================================================================
X = [x y z];
v = evec(:,1:6);
v = v-X*inv(X'*X)*X'*v;
evec=evec-X*inv(X'*X)*X'*evec;
[u,s,v]=svd(v);v=u(:,1);
evec(:,1:4) = [x y z v];
```

```
function [x,y,z]=ordinel(choice,n3)
%=================================================================
%  Take only rotational degrees of freedom for rigid body
%=================================================================
if choice==1,
x = [ones(n3,1) ;0*ones(5*n3,1)];
y = [0*ones(n3,1) ;ones(n3,1) ;0*ones(4*n3,1)];
z = [0*ones(2*n3,1) ;ones(n3,1) ;0*ones(3*n3,1)];
elseif choice==2,
x = [ones(n3,1) ;0*ones(5*n3,1);0*ones(4,1)];
y = [0*ones(n3,1) ;ones(n3,1) ;0*ones(4*n3,1);0*ones(4,1)];
z = [0*ones(2*n3,1) ;ones(n3,1) ;0*ones(3*n3,1);0*ones(4,1)];
elseif choice==3,
x = [ones(n3,1) ;0*ones(5*n3,1);0*ones(2,1)];
y = [0*ones(n3,1) ;ones(n3,1) ;0*ones(4*n3,1);0*ones(2,1)];
z = [0*ones(2*n3,1) ;ones(n3,1) ;0*ones(3*n3,1);0*ones(2,1)];
end
```

```
function [evec]=ordine2(choice,TP,TTP,TTTP,evec,ntot)

if choice==1,
evec = TP*evec;
elseif choice==2,
evec = TTP*evec;
% order last two columns
evec=evec([3:ntot 1:2],:);evec=evec(:,[3:ntot 1:2]);
evec([ntot-3:ntot-2 ntot-1:ntot],:)=...
evec([ntot-1:ntot ntot-3:ntot-2],:);
evec(:,[ntot-3:ntot-2 ntot-1:ntot])=...
evec(:,[ntot-1:ntot ntot-3:ntot-2]);
elseif choice==3,
evec = TTTP*evec;
end
```

```
function [ROT1,ROT2,ROT3,ROT] = rotation(theta1,theta2,theta3)
% rotation matrices from f1-f2-f3 to payload
c1=cos(theta1);c2=cos(theta2);c3=cos(theta3);
s1=sin(theta1);s2=sin(theta2);s3=sin(theta3);
%  1) of theta1 cckwise about +X
ROT1=[1      0      0;
      0      c1     s1;
      0     -s1     c1];
%  2) of theta2 cckwise about +Y
ROT2=[c2     0     -s2;
      0      1      0;
      s2     0      c2];
%  3) of theta3 counterclockwise about +Z
ROT3=[c3     s3     0;
      -s3    c3     0;
      0      0      1];
ROT=ROT3*ROT2*ROT1;  %from f1-f2-f3 to payload or Cpf
```

```
function[TR]=transform(r)
TR=[1      0      0      0      r(3)   -r(2);
    0      1      0     -r(3)   0       r(1);
    0      0      1      r(2)  -r(1)    0;
    0      0      0      1      0       0;
    0      0      0      0      1       0;
    0      0      0      0      0       1];
```

```
function [skewmatrix] = skew(vector)
skewmatrix=[0              -vector(3)     vector(2);
            vector(3)          0         -vector(1);
           -vector(2)      vector(1)         0];
```

```
function[DJ]=recrient(DJprincipal,ROTmat)

DJ=ROTmat'*DJprincipal*ROTmat;
```